

# Adventures in Voxel Coloring: CS 559 Term Project

Rob Hess  
Department of Computer Science  
Oregon State University

March 17, 2005

## Abstract

Voxel coloring is a technique used to create a 3D scene reconstruction from a fairly sparse set of images. Perhaps the most useful aspect of this technique is the fact that it can create a photorealistic scene reconstruction without computing image correspondences. In this paper I describe my efforts to implement and test a 3D voxel coloring algorithm.

## 1 Introduction

Seitz and Dyer's technique for scene reconstruction using voxel coloring [3] works by decomposing the scene to be reconstructed into small elements of volume, called voxels. Rather than computing correspondences between images and using triangulation to reconstruct scene geometry, the voxel coloring technique traverses the scene volume voxel by voxel, reprojecting each one back onto the image set and computing a measure of color variance for the pixels onto which it projects. If the variance between those pixels is low, the voxel is considered color invariant, and is col-

ored accordingly. Otherwise, the voxel remains uncolored. After a sweep of the entire scene volume, all voxels corresponding to points in the scene should be colored, while those not corresponding to scene points remain uncolored.

This paper is mainly a discussion of my attempt to implement and test a voxel coloring algorithm, and is not intended to be an in-depth description of the voxel coloring technique. As such, I will discuss specifics of the technique only to a limited extent. The reader interested in a more extensive account of the voxel coloring technique is referred to [3].

The rest of the paper will proceed as follows. In section 2 I will describe the how I collected data, including image data and calibration data, with which to test my algorithm. In section 3 I will discuss my implementation of the voxel coloring algorithm. In section 4 I will present the results of running my voxel coloring algorithm on the test data and will briefly describe some complications that arose during testing.

## 2 Data Acquisition

To test my voxel coloring algorithm, I created sets of twenty images of various objects, including two stuffed animals and a Tabasco sauce bottle. Here I will discuss how I went about acquiring those images, and I will briefly mention how I calibrated my camera.

### 2.1 Image Acquisition

When performing voxel coloring, it is important that the external parameters (i.e. the rotation and translation) of each image’s camera position are known. To facilitate external parameter computation, I fixed a planar camera calibration target to the floor and photographed it from twenty marked locations, as shown in Figure 1. After photographing the calibration target alone, each object was placed on top of it and photographed from the same twenty locations. By photographing each object from the same locations used to photograph the calibration target, I was able to use for scene reconstruction the same external camera parameters computed during camera calibration.

In other reports similar to this [1, 2] authors described acquiring images by rotating the object being photographed while keeping the camera in a fixed location. Since the voxel coloring technique colors voxels whose footprints have high color correlation, I chose instead to rotate the camera about a fixed object with the hope of maintaining constant illumination on each point on the object.

When positioning the camera, I had to be sure to do so in such a way as to satisfy what [3] refers to as the *ordinal visibility constraint*, which requires that a norm  $\|\cdot\|$  exists such that

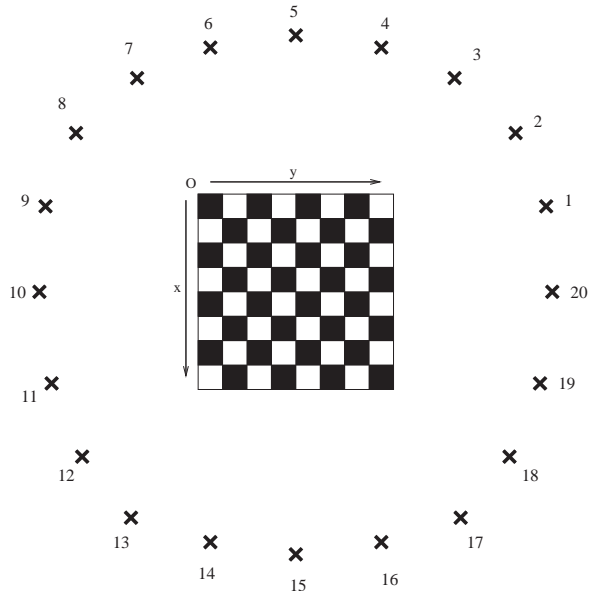


Figure 1: Approximate locations of camera positions for each of twenty images

for all scene points  $\mathcal{P}$  and  $\mathcal{Q}$  and input images  $\mathcal{I}$ ,  $\mathcal{P}$  occludes  $\mathcal{Q}$  in  $\mathcal{I}$  only if  $\|\mathcal{P}\| < \|\mathcal{Q}\|$ . This constraint simplifies resolving visibility relationships between voxels. It is easily satisfied by ensuring that no scene point falls within the convex hull of the camera centers, which I did by positioning the camera slightly above the object being photographed.

To ensure favorable lighting conditions, I photographed in a room well lit by bright sunlight. To obtain more diffuse light, I covered the room’s windows with waxed paper. Also, although I much prefer the aesthetics of natural light, I created one set of images using the camera’s built-in flash for comparison.

After images were collected, I manually segmented the object from the background in each one and colored the background with pure green (0,255,0). This was done so that com-

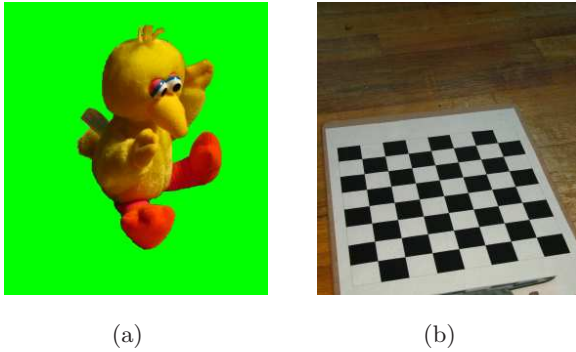


Figure 2: (a) An example image and (b) the corresponding calibration target image

pleted scene reconstructions contained only the object being reconstructed and none of the background. An example image and its the corresponding calibration target photo are shown in Figure 2.

## 2.2 Camera Calibration

Camera calibration was performed using the Camera Calibration Toolbox for Matlab.<sup>1</sup> Camera calibration with a planar target is ill-conditioned if the target is simply rotated in the same plane in each photograph, so to calibrate I used fourteen photos of the calibration target from various locations in addition to the twenty photographs I used to calculate external parameters. The external parameters of all 34 locations are shown in Figure 3.

Calibration with the Camera Calibration Toolbox for Matlab was straightforward, and I will not detail the process here. Using the Toolbox, I recovered the camera's internal parameters, including focal length, aspect ratio,

<sup>1</sup>The Toolbox is available at [http://www.vision.caltech.edu/bouguetj/calib\\_doc/](http://www.vision.caltech.edu/bouguetj/calib_doc/)

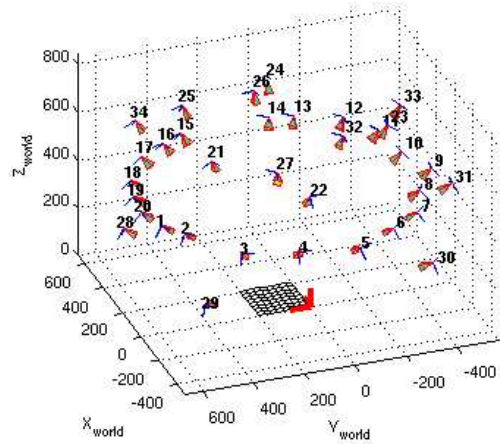


Figure 3: External camera parameters for all 34 calibration images

and principal point, and coefficients for skew and both radial and tangential distortion, as well as the rotation and translation vectors for each of the 34 camera positions.

## 3 Voxel Coloring Algorithm

Here I will describe my implementation of the voxel coloring algorithm. Sacrificing efficiency for ease of implementation, I used Matlab for this program.

As was touched on briefly above, the voxel coloring technique works by performing a voxel-by-voxel sweep of the scene volume, reprojecting each voxel back onto the image set and computing the color consistency of the set of pixels onto which each voxel projects.

The entire scene can be reconstructed in a single pass by making use of the ordinal visibility constraint. Specifically, by decomposing the scene into voxel layers and starting the sweep with the layer closest to the camera vol-

ume, all occlusions can be accounted for by keeping for each image in the image set a binary mask, initially zeros, to mark which pixels have already been used to color some voxel. For example, when performing voxel coloring on an upright Tabasco sauce bottle, the algorithm begins at the lid and proceeds to the base. Image pixels used to color lid voxels are marked and are not considered when coloring the base, even if a voxel in the base happens to project onto some of them.

Each voxel visited is projected onto the image plane of each image in the image set using the camera parameters computed during calibration. In general, a voxel, which is a cube, projects onto some sort of hexagon in the image plane. To simplify the algorithm, I chose instead to project voxel corners onto the image plane and then use the minimum and maximum x and y reprojection coordinates to create a rectangular footprint.

Once a voxel’s footprint is determined, the unmarked pixels within it are found, and their mean red, green and blue values are computed. If the computed mean RGB value is within some user-supplied threshold of the background color, the voxel being examined is considered a background voxel and is not colored. Otherwise, the sum of squared differences is computed between unmarked pixel values and the mean value for each channel and is added to a running total kept for each voxel over the set of input images. If, after its footprint has been computed for each of the input images, a voxel is determined not to be a background voxel, these totals are used to compute each channel’s standard deviation, and the standard deviations for the three channels are averaged and compared to a user-supplied threshold. If the mean standard deviation is

less than the threshold value, the voxel is considered consistent and is colored with the mean color of its footprints, and all pixels inside the voxel’s footprints are marked. This procedure is performed for each voxel within the scene volume, and results are stored in `.vxl` format<sup>2</sup> in a file whose location is user-specified.

## 4 Results

I tested the voxel coloring algorithm extensively using a set of images of a Big Bird doll taken in natural light. Here I present some results of those tests.

I programmed the voxel coloring algorithm to accept as input the voxel size to be used while performing reconstruction. For example, using a voxel size of 1, the scene is broken down into voxels with sides of one unit of length, and using a voxel size of 3, the scene is broken down into voxels with sides of three units of length. The results of running the algorithm on the Big Bird image set with various voxel sizes are summarized in Table 1. Runtimes, although high because of Matlab’s inherent inefficiency, are approximately linear in the number of voxels colored.

Voxel size	Scene size in voxels	Voxels colored	Running time
1	$165 \times 165 \times 175$	92574	13h:38m:27s
3	$55 \times 55 \times 58$	8426	43m:53s
5	$33 \times 33 \times 35$	2403	12m:52s
10	$17 \times 17 \times 18$	327	3m:47s

Table 1: Summary of results for reconstructions of Big Bird at various resolutions

<sup>2</sup>Details of the `.vxl` format can be found at <http://www.cs.princeton.edu/courses/archive/spr02/cs496/assignment4.html>



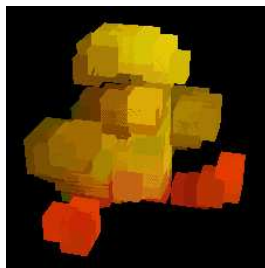
(a) Voxel size = 1



(b) Voxel size = 3



(c) Voxel size = 5

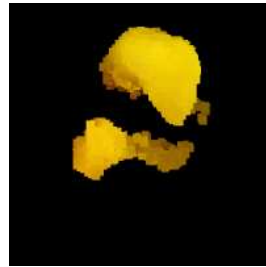


(d) Voxel size = 10

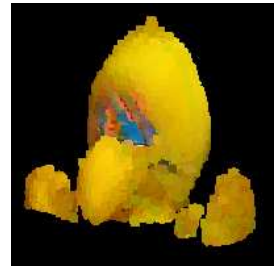
Figure 4: Reconstructions of Big Bird at various resolutions

Reconstructions of Big Bird with various voxel sizes are shown in Figure 4. As you can see, not only does increased voxel size cause an increase in granularity, many voxels remain uncolored because for larger voxels to be deemed color invariant, color consistency must hold over a greater number of pixels. In Figures 4(b), 4(c), and 4(d), voxels around Big Bird’s eyes and head remain uncolored because they contain too wide a range of colors.

Another variable affecting reconstruction quality is the user-selected threshold for color variance, which controls the maximum amount of variance allowable for a voxel to be considered color invariant. Figure 5 illustrates how changes in the color threshold affect re-



(a) Color thresh = 7



(b) Color thresh = 16



(c) Color thresh = 25

Figure 5: Reconstructions of Big Bird’s head using three different color threshold values

construction quality. Each of the three reconstructions in Figure 5 were performed over the same subset of scene volume. You can see that a trade-off exists between reconstruction completeness and the amount of background color included in the reconstruction. Figure 5(a) contains very little of the background color but also contains very little of Big Bird’s head. Figure 5(c) contains most of Big Bird’s head but also contains what I consider an unacceptable amount of background color. Figure 5(b) is somewhere in between.

While not as delicate as the color threshold, background threshold, which determines the maximum distance a voxel can be from

the background color before it is considered a background voxel, also affects reconstruction quality. Figure 6 illustrates how changing the background threshold affects reconstruction quality. We can see that here, too, there is a trade-off between reconstruction completeness and the amount of background color included in the reconstruction. In Figure 6(a) there is a good deal of background color, but Big Bird’s cheek and the base of his neck are mostly intact, whereas in Figure 6(c) there is very little background color, but much of Big Bird’s cheek and the base of his neck are missing. Once again, Figure 6(b) is somewhere in between.

I also encountered some problems during reconstruction that I believe stemmed from imprecision during image acquisition rather than from the reconstruction algorithm. Unfortunately, my image acquisition setup was a low-budget one. Camera positions were marked by putting masking tape on the floor at the feet of the tripod’s legs, and the camera was moved from location to location by lining the tripod’s feet up with different sets of masking tape marks. Obviously, this method is imprecise at best. The muddled eyes and artifacts shown in Figures 7(a) and 7(b) are most likely due to slight inaccuracies in external camera parameters introduced by the imprecise nature of the image acquisition setup.

I was, however, able to get a good reconstruction of Big Bird that was fairly accurate even at novel viewpoints. This is illustrated in Figure 8.

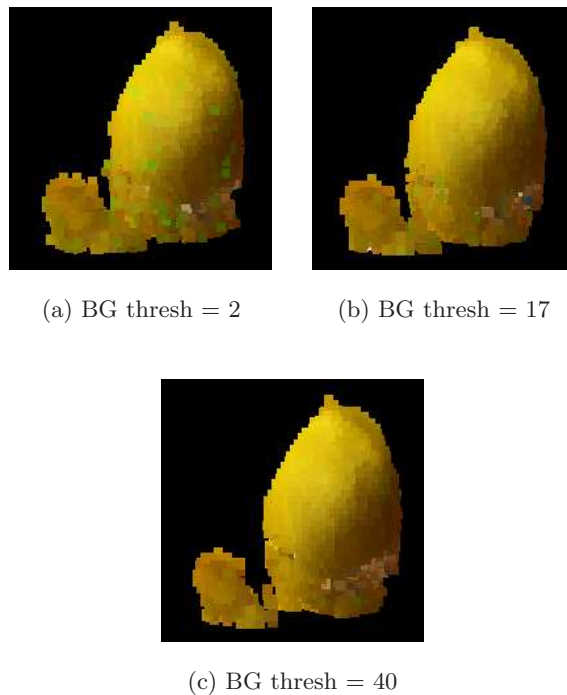


Figure 6: Reconstructions of Big Bird’s head using three different background threshold values

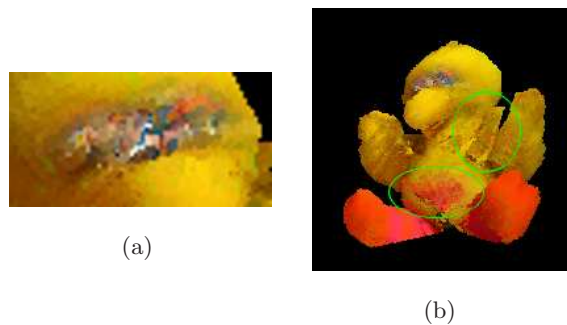


Figure 7: Inconsistencies in reconstructions of Big Bird: (a) muddled eyes and (b) artifacts

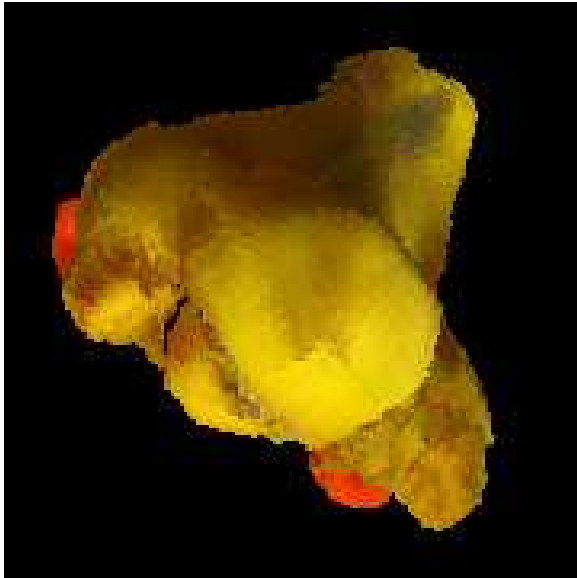


Figure 8: Big Bird from a novel viewpoint

## 5 Conclusions

Considering the low level of sophistication inherent in some of the methodology that went into this project, I was overall quite pleased with the results. There are, however, several improvements I could make to strengthen my results in one way or another.

The first, and simplest, of these improvements would be to change the background color of the images used for reconstruction. This alone would probably have solved the problem of background color being included in the reconstruction. I originally chose green as the background color because I wanted to use a primary color, and the Big Bird doll contained a good amount of red in its legs and some blue in its eyes. Green, however, ended up being incompatible with Big Bird's most prevalent color (yellow). In retrospect, a better choice

for the background color probably would have been black, since most of the colors in the Big Bird doll are fairly distinct from black.

Other problems, such as the reconstruction inaccuracies mentioned above, could have been mitigated simply by using a more sophisticated means of camera positioning to minimize errors in the external camera parameters computed during calibration. Unfortunately, constraints were placed on the level of sophistication appointed to this project due to procrastination.

Finally, I would like very much to port the voxel coloring algorithm from Matlab to C/C++. I found some of the run times for higher resolution reconstructions to be unbearable and suspect that they would have been reduced by magnitudes had the algorithm been implemented in C/C++. Once again, however, I was fairly strapped for time due partially to procrastination, and Matlab allowed me to quickly produce an algorithm that worked well.

## References

- [1] DONTCHEVA, M., AND LI, W. Voxel coloring: Trials and tribulations. <http://www.cs.washington.edu/homes/mirad/voxcol/>.
- [2] LOW, B. Cpsc 514 term project—implementation of photorealistic scene reconstruction by voxel coloring. <http://www.cs.ubc.ca/~low/BYLPaper.pdf>, 2004.
- [3] SEITZ, S., AND DYER, C. Photorealistic scene reconstruction by voxel coloring. *Int. Journal of Computer Vision* 35, 2 (1999).