Improved Voxel Coloring Via Volumetric Optimization

Gregory G. Slabaugh[1], Thomas Malzbender[2], W. Bruce Culbertson[2], and Ronald W. Schafer[1]

[1] School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332
{slabaugh, rws}@ece.gatech.edu
[2] Client and Media Systems Lab
Hewlett-Packard Laboratories
Palo Alto, CA 94306
{tom_malzbender, bruce_culbertson}@hp.com

Abstract

Voxel coloring methods reconstruct a three-dimensional volumetric surface model from a set of calibrated two-dimensional photographs taken of a scene. In this paper, we recast voxel coloring as an optimization problem, the solution of which strives to minimize reprojection error, which measures how well projections of the reconstructed scene reproduce the photographs. The reprojection error, defined in image space, guides the refinement of the scene reconstruction in object space. Unlike previous voxel coloring methods, ours makes better use of all color information from all viewpoints, and thereby produces higher quality reconstructions. In addition, it allows voxels to be added to, not just removed from, the scene at any time during reconstruction. We examine methods to minimize the reprojection error, including greedy and simulated annealing techniques. Reconstructions of both synthetic and real scenes are presented and analyzed.

1 Introduction

Voxel coloring methods [1] [7] [11] reconstruct a three-dimensional volumetric surface model from a set of calibrated, two-dimensional photographs taken of a scene. To evaluate the quality of a reconstructed scene, one might reproject it to the viewpoint of each camera used to photograph the scene. If the reconstruction is good, these images formed by reprojection should closely match the original photographs. In this work, our goal is to reconstruct a surface model that optimally reproduces the photographs. This approach differs from previous techniques, which instead find a surface model that is merely consistent with the photographs. We will demonstrate that the photo-consistency measure used in previous techniques does not use the full color information available, and can lead to surface reconstructions that are biased to be fat. Our methods take as input a surface reconstructed using a voxel coloring method, and then refine the reconstruction to optimize a reprojection error metric.

To give this notion of optimality context, we define a reprojection error function in image space that we endeavor to minimize. This error function measures the dissimilarity of the original photographs of the scene with the reprojection into each camera of scene being reconstructed. The reprojection error guides changes to the surface model in object space. Voxels are added to and removed from the surface model to minimize the reprojection error.

Our approach offers four novel contributions over the state of the art:

1. It makes better use of the color information available, and can therefore remove, or carve, photo-consistent voxels. The reconstructed scene better reproduces the photographs and projects to better images at novel viewpoints.
2. It allows voxels to be added to, and removed from, the surface at any time during reconstruction, while efficiently maintaining visibility and error information.
3. It evaluates surface modifications on a per scene, rather than per voxel, basis.
4. It operates via tentative surface modifications.

The layout of this paper is as follows. First, in section 2 we examine related work. We focus on techniques that modify object space geometry so that its projection matches photographs taken of a scene, as well as analyze how previous voxel coloring methods can be extended. In section 3, volumetric optimization is discussed, and our reprojection error metric is defined. Then, in section 4 we present a general methodology for minimizing the reprojection error, and then describe two different techniques developed to find a reasonable minimum in the

error landscape. In section 5, experimental results are presented for reconstructions of both synthetic and real scenes. Finally, we conclude with a brief description of future work.

2 Related Work

Three-dimensional surface reconstruction from multiple photographs of a scene is an important topic in computer vision and has received considerable interest in the past few years. Rather than survey the vast body of work on this subject, we will mainly focus on related techniques that apply optimization methods to match reconstructed surface models with photographs of a scene, and on recent voxel-based solutions that serve as the foundation of this work.

Adjusting a three-dimensional surface model so that its reprojection matches two-dimensional photographs of the scene is not a new concept. Fua and Leclerc [4], as well as Rockwood and Winget [10], optimize a surface mesh to minimize an objective function consisting of a reprojection error term, surface smoothness constraints, and a few other terms. Our reprojection error function bears close similarity to the multi-image correlation component in [4] and the essential cost functions of [10]. Unlike their approaches however, ours does not rely upon smoothness constraints, which penalize complex surface geometry that often occurs in real scenes. Also, our voxel-based surface representation can very easily accommodate topological changes, unlike a surface mesh.

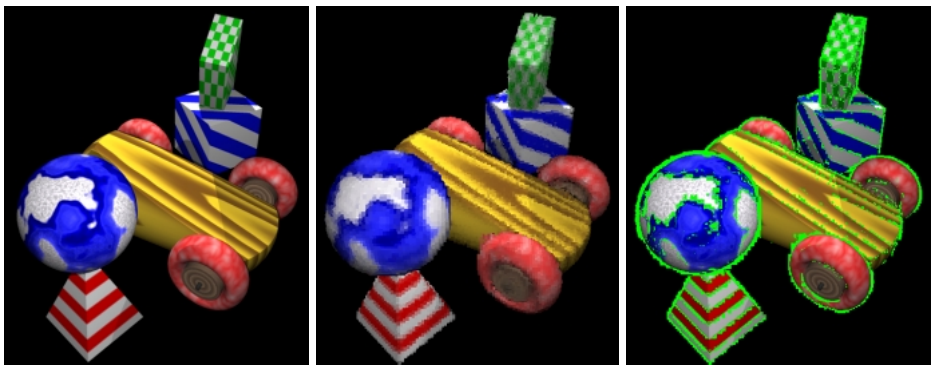
Szeliski and Golland [15], and Fua [3] both initially compute raw disparity maps, and apply optimization methods to refine their representation. Similarly, our work finds an initial volumetric model using a voxel coloring algorithm, and then refines the model using optimization methods. Since the error surface in general contains local minima, starting the optimization with a reasonable representation can help avoid the algorithm from getting caught in a local minimum far from a global minimum. Szeliski and Golland refine the disparity map directly, while Fua refines a set of oriented disks created from the displacement maps. Both methods use a multi-component cost function that consists of a reprojection error term and a smoothness term, and in the case of Szeliski and Golland, a fractional opacity term. These techniques allow for a more arbitrary surface topology, but the smoothness term might not be appropriate for all surfaces. Both of these techniques model occlusion, but not as generally as our methods. However, our methods do not attempt to reconstruct fractional opacity.

Faugeras and Keriven [5] have produced impressive reconstructions by applying variational methods within a level set formulation. The functional optimized by their technique is not a reprojection error, but instead a measure of the similarity of correspondences between images. Their method bears much similarity to ours, in that a surface initially larger than the true scene is refined to a successively more accurate reconstruction. In addition, their technique, like ours, handles occlusion correctly and can deduce arbitrary topologies. It is not clear under what circumstances their method converges. They have not provided runtime and memory statistics, or a detailed implementation of their method, so it is unclear if their methods are practical for reconstructing large scenes.

Recent volumetric solutions to the three-dimensional scene reconstruction problem have been particularly successful in achieving reconstructions that can be projected to form photo-realistic images at novel viewpoints. By exploiting the color correlation of surfaces, Seitz and Dyer's voxel coloring technique [11] finds a set of voxels that are photo-consistent with the photographs. To easily maintain visibility information, voxel coloring restricts the placement of cameras, so that no scene point is contained within the convex hull of the camera centers. Kutulakos and Seitz [7] extend voxel coloring to support arbitrary camera placement by devising a theoretical space carving method. Their implementation of the method, however, relies upon a multi-sweep approach that does not utilize the full visibility of the scene. Culbertson, Malzbender, and Slabaugh [1] present two generalized voxel coloring (GVC) algorithms, which implement space carving using the exact visibility of the scene. One of these algorithms, GVC-LDI, uses layered depth images (LDIs) [13] to keep track of scene visibility. While the GVC-LDI algorithm requires large amounts of memory, the layered depth images permit incremental visibility updates, a feature that we require in this work. We will refer to voxel coloring [11], space carving [7], and generalized voxel coloring [1] collectively as 'voxel coloring algorithms' throughout this paper.

Voxel coloring algorithms determine voxel opacity using a color consistency check. These algorithms carve photo-inconsistent voxels until all visible surface voxels are photo-consistent. Kutulakos and Seitz call this surface the maximal photo-consistent shape, V^*. For a given color threshold, V^* is unique, and represents the union of all possible photo-consistent surfaces. Kutulakos and Seitz show that the maximal photo-consistent shape represents the tightest possible upper bound, for a given a color threshold, on the true scene that can be inferred from the images. Let us denote the true scene as V^T. As an upper bound, V^* tends to be larger, or fatter, [12] than V^T. When V^* is reprojected to the cameras, this fattening artifact often can be readily detected when we compare

with an original photograph, as depicted in figure 1. Due to the fattening artifact, a voxel that occupies V^T often projects to a different set of pixels than a voxel on V^*. Consequently, voxels on a fattened model typically have a less accurate coloring.



(a) (b) (c)

Fig. 1. Voxel coloring algorithms produce fattened reconstructions. The image in (a) was one of the original photographs of our toycar scene. After performing a reconstruction of the scene using the GVC-LDI algorithm, image (b) was formed by projecting the reconstructed scene, V*, to the same viewpoint as in (a). Notice how surfaces in (b) are fatter than in (a), particularly the globe and the checkered box. Next, a difference image was formed by subtracting image (a) from image (b). Pixels for which the absolute difference was more than 50 in any color channel are flagged with a bright green color in image (c). Note that the fattening artifact is quite prominent around edges, especially silhouette edges.

The color consistency check is binary, as its output states that a voxel is either photo-consistent or photo-inconsistent. Using this binary consistency function, voxel coloring algorithms terminate once all visible photo-inconsistent voxels are carved. This approach is simple and efficient, but leads to the fattening artifact. After finding V^*, it may be possible to continue carving voxels so that the reconstructed surface is less inconsistent, as shown in figure 2. To find such an improved surface, the binary consistency function must be changed to yield a multi-valued output, so that photo-consistency can be ranked. The surface can then be altered to decrease its inconsistency. This is the essence of our approach, and we will discuss it in detail in subsequent sections of the paper.

In general, there is not a single color threshold [11] that is ideal for reconstructing all surfaces in a scene using a voxel coloring algorithm. Often, one sweeps the parameter space of color thresholds to identify a global threshold that offers the best reconstruction. Given such a threshold, some surfaces in the scene could likely be more accurately reconstructed with a lower color threshold. But lowering the global threshold can cause other surfaces to become carved that should be in the final reconstruction. Our method can be thought of as seeking the optimal, local color threshold for each voxel in the scene.

In addition to addressing the fattening artifact of voxel coloring, the volumetric optimization methods described in this paper can reduce other voxel coloring artifacts, such as stray, floating voxels and holes in the surface. Floating voxel artifacts occur when the set of visible pixels in a voxel's projection are photo-consistent, even though the voxel should not be included in the scene reconstruction. Holes in the surface occur when the set of visible pixels in a voxel's projection are inconsistent, even though the voxel should be included in the scene reconstruction.

3 Volumetric Optimization

The ultimate goal of voxel coloring algorithms is to find the most accurate reconstruction of the scene possible. One question that comes to mind is if there is an optimal solution to the volumetric scene reconstruction problem. In order for the word 'optimal' to be well defined, it is necessary to state in what sense a solution would be optimal.

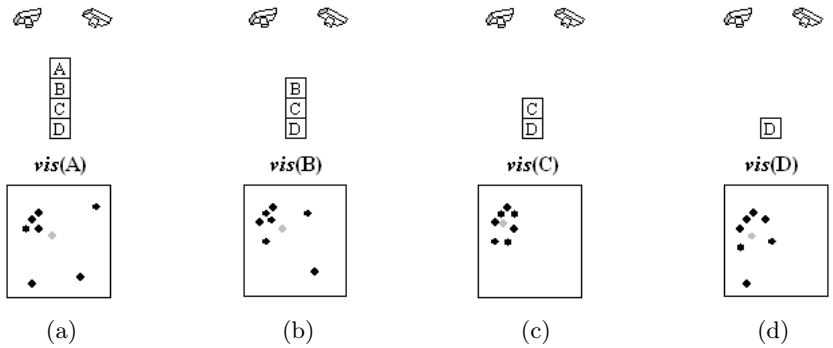


Fig. 2. Voxel coloring algorithms terminate carving too soon. Suppose we are given a hypothetical scene, consisting of four voxels, labeled A through D shown in (a). For simplicity, also suppose that only the top-most voxel is visible to the cameras. Voxel coloring projects voxel A to the cameras, and then determines the set of visible pixels in the original photographs, termed vis(A). We plot in color space the location of each pixel in vis(A). Although color space (r, g, b) is three-dimensional, for simplicity, only two dimensions are drawn in this figure. Photo-consistency is determined by computing the standard deviation about the mean, which represented by a gray dot. If the standard deviation is above a user specified threshold, the voxel is considered photo-inconsistent and carved. Suppose this is the case for (a), and voxel A is carved, exposing voxel B. After computing the mean and standard deviation of vis(B) in (b), we find that voxel B is photo-consistent. Voxel coloring terminates once a consistent surface is found, and therefore voxels C and D remain unprocessed using those algorithms. Alternatively, in attempting to minimize photo-inconsistency, we will tentatively carve voxel B, exposing voxel C, as shown in (c). The pixels in vis(C) have a lower standard deviation, than those of vis(B). We reason that the surface in (c) is therefore a better reconstruction of the scene than that of (b). Carving voxel C increases photo-inconsistency. Because voxel coloring algorithms terminate once a photo-consistent model is found, they produce fattened models. Optimization techniques can continue carving to find better reconstructions.

3.1 Definition of reprojection error

For the purpose of this paper, an optimal voxel coloring algorithm finds a volumetric reconstruction V_o, which, when projected to each camera, minimizes a multi-view image space error function $E(V)$, whose argument is a volumetric reconstruction. Here we assume that the scene is Lambertian. Borrowing from [14], characteristics that $E(V)$ should possess include:

1. the function should incorporate information from all available viewpoints.
2. the function should weight the error so that viewpoints that have more visibility have a greater contribution to the error.
3. the function should be minimized by the true scene, so that $E(V^T) \leq E(V)$ for an arbitrary volumetric surface V.
4. the function should provide a relative ordering of solutions, so as to rank the quality of reconstructed surfaces. If the volumetric surface V_1 is a better solution than V_2, then $E(V_1) < E(V_2)$.
5. the function should relatively simple to compute.

Let the reprojection of the current reconstruction V into the ith camera be denoted as R_i. Also, let the original photograph at the ith camera be denoted as P_i. Since R_i and P_i are color images, they have pixels with r, g, and b components. Let these color components in the jth pixel of R_i be referenced as $R_i(j).r$, $R_i(j).g$, and $R_i(j).b$, respectively. Similarly, we denote the r, g, and b components of the jth pixel of P_i as $P_i(j).r$, $P_i(j).g$, and $P_i(j).b$.

Essentially, we would like to compute the dissimilarity of R_i and P_i. In this work, we define reprojection error using a squared difference measure,

$$E(V) = \frac{\sum_{i=1}^{N} \sum_{j=1}^{M_i} \begin{array}{r} (P_i(j).r - R_i(j).r)^2 + \\ (P_i(j).g - R_i(j).g)^2 + \\ (P_i(j).b - R_i(j).b)^2 \end{array}}{\sum_{i=1}^{N} M_i},$$

where N is the number of images, M_i is the number of pixels used in the comparison for the ith image, and R_i and P_i are images as described above. The reprojection error is the average squared difference between pixels in R_i and P_i, taken over the pixels M_i.

This function possesses the necessary characteristics mentioned above. Specifically, it incorporates information from all views, as the sum over i is over all viewpoints. The variable M_i weights the reprojection error so that viewpoints with greater visibility have a larger contribution. The function reaches its minimal value for the true scene, for which R_i is identical to P_i for the pixels indexed by j. The function does provide a relative ordering of solutions, so that one reconstructed scene can be objectively ranked relative to another. Consequently, the reprojection error can guide changes to the surface during reconstruction. We will show that the difference in reprojection error can be computed incrementally, i.e. for only the pixels in R_i that change. This will yield a very efficient computation of the reprojection error.

3.2 Relating the reprojection error to the consistency measure used in voxel coloring

A comparison of this reprojection error function with that of the standard deviation measure used in voxel coloring is useful in relating the two methods. The main difference between the two is that in voxel coloring, the standard deviation measure is defined on a per voxel basis. In contrast, the reprojection error is defined over the entire projection of the reconstructed scene. A per scene basis for the reprojection error is necessary so that we can globally measure the effect of a surface modification. If however, only one voxel changes visibility, then the reprojection error is simply the square of the standard deviation used in voxel coloring. To see this, recall that during reconstruction, a voxel is projected into each image, and the set of visible pixels in all images, here denoted as X, is found. If the voxel is photo-consistent, its color is set to equal the mean of X. Thus, for a given voxel, each color channel of $R_i(j)$ will be the expected value of the corresponding color channel in P_i, over all images i. We denote this expected value as

$$\begin{aligned} \mu.r &= E[X.r], \\ \mu.g &= E[X.g], \\ \mu.b &= E[X.b]. \end{aligned}$$

Let k index over pixels in X, and suppose the cardinality of the set is L. Then, the reprojection error has the form,

$$E(V) = \frac{\begin{array}{r}(X(k).r - \mu.r)^2 + \\ \sum_{k=1}^{L} (X(k).g - \mu.g)^2 + \\ (X(k).b - \mu.b)^2\end{array}}{L} = \sigma^2.$$

This is simply a formula for the sample variance [8] of X, equivalent to the square of the standard deviation used in voxel coloring, assuming a biased variance measure is used.

3.3 Optimal scene reconstructions exist but are not unique

An optimal scene reconstruction must exist; i.e. there must be some volumetric model that minimizes the reprojection error. However, it is rarely unique, for two reasons. First, there can be many three-dimensional volumetric models that project to the same two-dimensional images. Because of this ambiguity, the problem of three-dimensional scene reconstruction is ill-posed. Second, it is possible for multiple volumetric models to exist that form different images upon projection, but whose reprojection error is identically minimal. Finding an optimal scene reconstruction can be challenging, since the reprojection error function is not differentiable due to occlusions, and can contain numerous local minima far away from a global minimum.

3.4 The search space is large

Performing a combinatorial analysis can shed some light on the size of the search space. The search space for the optimal reconstruction is defined on a six-dimensional (x, y, z, r, g, b) domain for standard color scenes. Voxel opacity accounts for three of these dimensions, as it consists of a bit for the spatial location, defined on a three-dimensional (x, y, z) domain, of each voxel in the reconstruction volume. Each opaque voxel can then have a unique color, defined on another three-dimensional (r, g, b) domain. Consider a reconstruction volume containing v voxels. Since a voxel can either be opaque or transparent, there are 2^v possible voxel opacity configurations.

Now let us consider the number of different colorings of the 2^v possible voxel opacity configurations. Let the variable p be number of opaque voxels in a candidate configuration. Then, there will a total of

$$\binom{v}{p}$$

configurations with p opaque voxels. If the number of possible colors for the voxel is c, then there are c^p possible colorings of the voxel configurations with p opaque voxels. The total number of candidate reconstructions, n, is then the sum of the possible colorings over all possible voxel geometries,

$$n = \sum_{p=0}^{v} \binom{v}{p} c^p.$$

This sum can be evaluated using the binomial theorem, yielding

$$n = \sum_{p=0}^{v} \binom{v}{p} c^p = (c+1)^v.$$

A simple example of this is provided in figure 3.

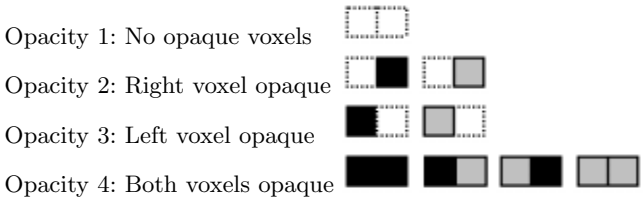


Fig. 3. Example voxel opacity configurations and colorings for v = 2, c = 2. Suppose that we have a reconstruction volume that contains two voxels (v = 2), and each opaque voxel can take on one of two possible colors, (c = 2), which are gray and black. Then, there are 2v = 4 unique voxel opacity configurations, and (c + 1)v = 9 possible reconstructions to check. These nine reconstructions, grouped into the four voxel opacity configurations, are shown above.

When working with real world scenes, both the values of v and c are quite large. If we use 24-bit images, a voxel can have one of $c = 2^{24}$ unique colors, so

$$n = (2^{24} + 1)^v \approx 2^{24v}.$$

Since v is typically large, the number of possible reconstructions defined on this six-dimensional domain is enormous. Thus, we will require effective algorithms to limit our searching in this huge space, and perhaps we can't expect to find a truly optimal reconstruction.

4 Minimizing the reprojection error

We have developed two different approaches to minimizing the reprojection error. In subsections 4.2 and 4.3 we will present each specific approach and discuss its unique characteristics. However, the general methodology for both approaches is the same, and is quite straightforward conceptually. First, our methodology uses a voxel coloring algorithm to find V^*, which can be quickly computed, and is hopefully reasonably close to a global minimum in the reprojection error. Then, V^* is refined by

- tentatively modifying the scene reconstruction by adding, carving, or moving voxels
- incrementally evaluating the reprojection error
- either accepting or undoing the modification

Simply stated, this methodology tentatively modifies the scene reconstruction, and determines if the modification is beneficial. If so, the modification is accepted. Otherwise, the modification is undone, restoring the scene reconstruction to its state prior to the modification.

4.1 An implementation of the general methodology

We present an implementation of this general methodology. An issue that immediately arises is how the scene reconstruction should be modified. While a variety of techniques could be attempted, we choose to implement the carving of surface voxels in the scene reconstruction, and the addition of voxels that are adjacent to surface voxels. We call these adjacent voxels neighbor voxels. By attempting to carve surface voxels, we provide a mechanism for the surfaces to thin, and thereby reduce the fattening artifact, as well as remove floating voxels. We exploit the spatial coherency of surfaces by attempting to add neighbor voxels. This provides a mechanism for holes in the surfaces to be filled. Since moving a voxel from location A to B can be achieved by carving the voxel at A and adding it at B, we do not explicitly implement a move operation.

One of the key aspects underlying our approach is the incremental update of visibility [1], the reprojection error, and the reprojected images. During an iteration of our algorithm, often just a few voxels change visibility. Rather than re-render the entire scene reconstruction to determine visibility from scratch, we use layered depth images (LDIs) to incrementally update visibility for only the voxels whose visibility changes. Unlike a traditional z-buffer, which stores a depth value for each pixel in the rendered image, the LDI stores a linked list of unique voxel identifiers (voxel ids), sorted by depth, at each pixel in the reprojected image. The head of the linked list indicates the voxel that is visible at a given pixel, as shown in figure 4a. When a voxel A is carved from the surface, the LDI can be incrementally updated by removing references to the voxel from the LDI at the pixels in vis(A). This operation is depicted in figure 4b. Similarly, figure 4c shows the incremental addition of a voxel to the scene reconstruction, and the corresponding change to the LDI. From the LDI, our algorithm determines which pixels, in each image, change visibility during a modification. The algorithm efficiently computes the reprojection error only over these pixels. If the modification is accepted, the reprojected images R_i are updated for only those pixels as well. Incremental updates of visibility, reprojection error, and reprojected images provide raw ingredients for optimization: efficient ways to modify the model and compute an error function.

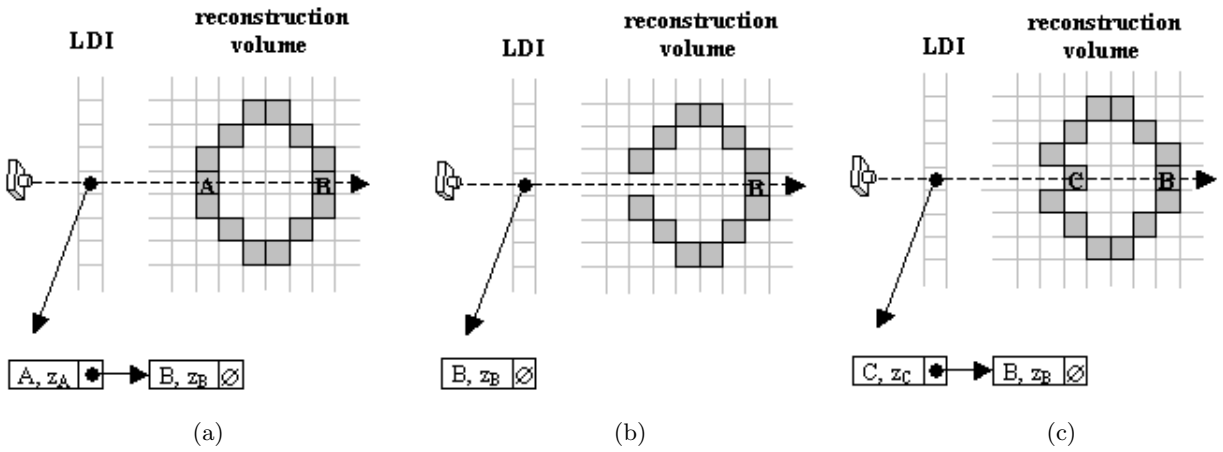


Fig. 4. Incremental update of surface visibility using LDIs. Each LDI pixel is a linked list, storing voxel identifiers sorted by depth, as shown for a particular LDI pixel in (a). In (b), we carve voxel A from the surface. Visibility is updated incrementally (i.e. without re-rendering the entire surface), by removing references to voxel A in the LDI. Doing so exposes voxel B, as it moves to the head of the LDI. In (c), we then add voxel C to the surface, and insert C onto the LDI. Voxel B now becomes invisible, as it moves from the head of the LDI. Whenever the head of the LDI changes, scene visibility has changed.

During reconstruction, a voxel is one of three states:

1. It has been carved and is no longer on a surface.
2. It is uncarved and on a surface.
3. It is surrounded by uncarved voxels, and is invisible from all images.

We identify voxels in the first category by using an array that stores a bit for each voxel in the reconstruction volume. We call this array the voxel carved bits. Like GVC, the bit gets set to the carved state when an uncarved voxel gets carved. Unlike GVC, the bit gets set to the uncarved state when a carved voxel gets added. That is, carved voxels can become uncarved. Voxels in the second category are stored in the surface voxel list (SVL), implemented as a hash table to expedite searching. We call voxels in the third category interior voxels. Interior voxels are not visible from any image, and can therefore be ignored until they become visible.

Once a voxel is evaluated, the algorithm does not re-evaluate it unless its visibility changes. To keep track of which voxels have changed visibility, we maintain a changed visibility surface voxel list (CVSVL). The CVSVL is always a subset of the SVL (CVSVL $\in$ SVL), and is initialized to equal the SVL. CVSVL voxels are the ones that are evaluated, and when there are no more voxels on the CVSVL, the optimization is complete. The tentative voxel list, (TVL), contains a copy of voxels whose visibility changes during modification of the surface. If the modification is accepted, TVL voxels added to the SVL and CVSVL if they are not already on those lists. If TVL voxels are on the lists, then the SVL and CVSVL are updated with new color information only. However, if the modification is undone, TVL voxels are discarded, and no changes are made to the SVL or CVSVL. By storing these voxels temporarily on the TVL, the algorithm avoids making changes to the SVL and CVSVL until it has been determined if the surface modification is accepted. For this reason, we call this data structure the tentative voxel list.

In our methodology, modifications to the surface are made tentatively, for if we determine that a modification results in a less favorable surface, the modification is undone. In our implementation, we make tentative changes to the LDIs. For each pixel that changes on an LDI, we store undo information that is used only if we undo the surface modification. All other changes are stored in temporary data structures, like the TVL, and are copied to permanent data structures upon acceptance of the surface modification.

Figure 5 shows pseudo-code for our approach. After initialization, a voxel C is removed from the CVSVL. If the algorithm is carving voxels, it tentatively removes the voxel C from the LDIs. Then, it tentatively adds voxel C's uncarved neighbors to the LDI if not already on the LDI. This latter step exposes any interior voxels that become visible upon removal of voxel C. If the algorithm is adding voxels, it tentatively adds a neighbor of voxel C to the LDIs, and evaluates the modified surface. Each of voxel C's neighbors is processed in this fashion. After modification, the surface is assessed to determine it is favorable. We have developed a few different methods for determining if the modified surface should be accepted. These methods are discussed in further detail in the next subsections.

One of the nice features of voxel coloring algorithms is their ability reconstruct a scene using either segmented or unsegmented photographs. A photograph P_i is segmented if each pixel is labeled as foreground or background. Foreground pixels correspond to objects that should be reconstructed, while background pixels correspond to objects that should not be reconstructed. If the algorithm is given such information, it computes the reprojection error over the union of the foreground pixels with the pixels in the reprojection of the reconstructed scene, i.e. the pixels in R_i. However, not all scenes have easily segmentable photographs. For these latter scenes, we compute the reprojection error over the pixels that are in the projection of the reconstructed scene only.

We now discuss the function processModification in the pseudo-code. The purpose of this function is to accept, or undo, the surface modification. First, a new coloring is found for each voxel on the TVL, since the visibility has changed for these voxels. Next, the algorithm determines if the surface modification is accepted. The exact nature of this step is described in subsections 4.2 and 4.3. If the surface modification is accepted, the voxel carved bits, SVL, and CVSVL are updated accordingly. We also incrementally update the reprojection images, R_i, of each camera. If the surface modification is undone, the LDIs are restored to their state prior to modification, and the TVL voxels are freed.

4.2 Greedy method

In the greedy method, voxels are added or carved only if they reduce the reprojection error. The reprojection error is computed for the surface before modification, yielding the quantity E_{old}, and after modification, yielding the quantity E_{new}. If the change in reprojection error,

$$\Delta E = E_{new} - E_{old}$$

is negative, the reprojection error has decreased and the surface modification is accepted. If this quantity is non-negative, the surface modification is undone. Since scenes reconstructed by voxel coloring algorithms tend to be

```
load SVL, the output of a voxel coloring algorithm
load voxel carved bits, also output from a voxel coloring algorithm
load original photographs P_i
render SVL to LDIs
render SVL to reprojection images, forming R_i
copy SVL to CVSVL
while (CVSVL is not empty) {
    delete voxel C from CVSVL
    if (carving) {
        remove C from all LDIs; copy changed visibility voxels to TVL
        add uncarved neighbors of C to all LDIs; add changed visibility voxels to TVL
        processModification(C)
    }
    if (adding) {
        for each neighbor N of C {
            if (N ∈ SVL) {
                add N to all LDIs; copy changed visibility voxels to TVL
                processModification(N)
            }
        }
    }
}
save SVL and voxel carved bits

processModification(voxel V) {
    find new colors for all TVL voxels
    determine if surface modification will be accepted
    if (accept) { /* accept modification */
        if (adding)
            set voxel carved bit (V) to uncarved
        if (carving) {
            set voxel carved bit (V) to carved
            delete V from SVL
        }
        incrementally update reprojection images, R_i
        for each voxel T on TVL {
            if (T ∈ SVL)
                update the color for T on the SVL
            else
                add T to the SVL
            if (T ∈ CVSVL)
                add T to the CVSVL
            delete T from TVL
        }
    } else { /* undo modification */
        undo changes to LDIs
        free TVL voxels
    }
}
```

Fig. 5. Pseudo-code implementation of general methodology.

fat, our implementation first performs a greedy carving pass that thins the reconstructed scene, and then performs a greedy adding pass. By executing these two sequential passes, we avoid attempting to add voxels to the fattened model, as such voxels are unlikely to decrease the reprojection error. One application of the greedy method usually results in much improved surfaces. Additional applications of the greedy method can further reduce the reprojection error, with diminishing rewards each time the algorithm is applied. In practice, we typically apply the algorithm once to a reconstruction.

4.3 Simulated annealing method

The reprojection error is a complex function over a six-dimensional domain, and can contain numerous local minima. Since the greedy method only accepts changes to the surface that decrease the reprojection error, it has the disadvantage that it could get caught in a local minimum of the error, possibly far from a global minimum. To address this issue, we developed a simulated annealing method that can accept some changes that increase the reprojection error. This method introduces a few new variables, namely

- k, the Boltzmann constant
- T, the temperature

Surface modifications that lower the reprojection error are always accepted. However, surface modifications that increase the reprojection error are accepted with probability where ΔE is the change in reprojection error, k is the Boltzmann constant, and T is the temperature. Initially, the temperature is high, and it is more likely that unfavorable changes will be accepted. Conceptually, we hope that accepting such an unfavorable change will dislodge the algorithm if it is caught at a local minimum. As the program runs, the temperature is slowly decreased to zero, for which the probability of accepting a surface modification that increases the error diminishes to zero. Thus, at zero temperature, the simulated annealing method defaults to the greedy method.

The annealing schedule guides the temperature, and is based on the number of times we wish to execute the while loop over the CVSVL in the pseudo-code. The entire CVSVL is processed with a temperature T. Then, T is decremented, the SVL is copied again to the CVSVL, and the while loop executes again with a lower temperature. The process terminates after the greedy method is executed with a temperature of zero. In practice, an initial temperature of five seems reasonable. Since this method can make unfavorable surface modifications, it simultaneously attempts to add and carve voxels, unlike the sequential nature (carving pass, then an adding pass) of the greedy algorithm. This provides a mechanism for unfavorable surface modifications to be undone. As in the greedy method, it is possible to apply the simulated annealing approach multiple times, but in practice we only apply the algorithm once per reconstruction.

5 Results and Analysis

We have executed our reprojective optimization algorithms on several data sets, both real and synthetic. For experimental runs we used a dual processor HP J5000 workstation, with 440 MHz processors and 2 GB of RAM. In the tables and figures below, we report time as measured by the CPU.

5.1 Toycar scene

Our toycar scene consists of seventeen photographs of a synthetic scene. This data set is ideal for reconstruction, as the cameras are perfectly calibrated, the surfaces are completely Lambertian, and the textures make the various surfaces relatively easy to distinguish from each other. We first performed a reconstruction in a 168 x 120 x 104 volume using the GVC-LDI algorithm, which took 39 minutes to complete. Then the reprojection error was minimized using greedy and simulated annealing methods. We define the improvement, I, of the reprojection error to be

$$I = \frac{E_{start} - E_{end}}{E_{start}},$$

where E_{start} and E_{end} are the reprojection errors at the start and end of the reprojective error minimization algorithm, respectively. If $Estart = Eend$, then the improvement is zero. However, if the reprojection error diminishes, $0 \leq I \leq 1$.

The execution time, memory usage, and percentage improvement for the two methods are reported in table 1. One would expect the simulated annealing method to run slower than the greedy method, since the simulated annealing method performs surface modifications that increase the error, many of which are later undone. Not surprisingly, the greedy method completed in about half the time of the simulated annealing method. The memory

Method	Execution time (min)	Memory usage (MB)	E_{start}	E_{end}	Improvement
Greedy	134	233	4373	1747	60.1%
Simluated annealing	269	270	4373	1727	60.5%

Table 1. Results for the toycar scene

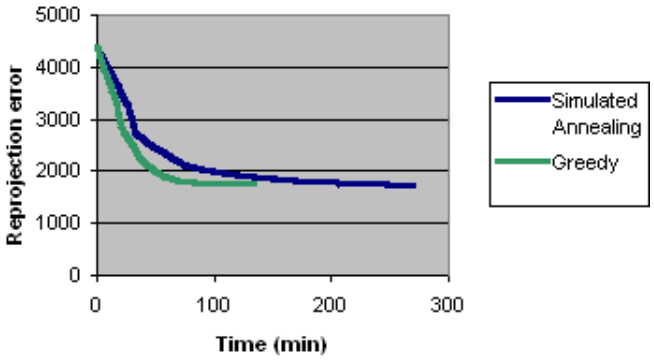


Fig. 6. Reprojection error vs. time, for greedy and simulated annealing methods.

usage of both algorithms was dominated by the LDIs, and was found to be comparable for both algorithms. Both methods significantly reduced the reprojection error for this scene, offering an improvement of about 60reprojection error as a function of time. The simulated annealing approach ultimately gets to a slightly lower reprojection error, but at the expense of taking longer to complete. An input photograph and reprojected images are shown in figure 7. The primary benefit that reprojective optimization offers is the thinning of the fattened reconstructed surface output by GVC-LDI. The surfaces output by greedy and simulated annealing methods more closely match those of the original photograph, compared to the surfaces output by voxel coloring algorithms that are too fat. In particular, notice how the globe in (c) and (d) has a more reasonable size compared with that of (b). This fattening effect is quite apparent when one makes a simple animation that cycles between these images. Since the thinned surface geometry more closely matches that of the true scene, voxels receive a more accurate coloring. For example, in (b) notice how the surface of the car has a noisy appearance, and the small island near the center of the globe's projection is barely visible. In (c) and (d) the surface of the car is considerably less noisy, and the island is much more discernable.

The refined scene is significantly improved, but is not perfect. In (b), on the topmost surface of the blue and white striped cube, there is a hole and indentation that does not get filled by the greedy method. We believe that this demonstrates that our optimization methods, particularly the greedy method, are somewhat sensitive to the quality of the scene reconstruction input into the algorithm. The simulated annealing approach does a better job with this defect, but has a few mis-colored voxels in the red and white striped cone.

5.2 Shoes scene

Our shoes scene consists of twelve photographs, taken indoors, of a cloth toy and two pairs of shoes on multi-colored paper. This data set is more challenging to reconstruct, as it is not segmented and calibration is not perfect. We first performed a reconstruction in a 144 x 128 x 80 volume using the GVC-LDI algorithm, which took 71 minutes to complete. Then the reprojection error was minimized using greedy and simulated annealing methods, as before. Table 2 presents the execution time, memory usage, and percentage improvement for the two methods. As for the toycar scene, the simulated annealing method took considerably more time to finish, but had a slightly better improvement. Figure 8 shows the reprojection error as a function of time for the two methods. An input photograph and reprojected images are shown in figure 9. As before, the scene reconstructed using the voxel coloring algorithm, shown in (b), has surfaces exhibit the fattening artifact when compared with the original photograph in (a). The fattened surface is less detailed, as evinced by the topmost cushion of the toy, where the white speckles in the red cloth are not as bright in (b) as those of the refined scenes in (c) and (d). Also, many of

Method	Execution time (min)	Memory usage (MB)	E_{start}	E_{end}	Improvement
Greedy	491	809	892	595	33.3%
Simluated annealing	709	818	892	557	37.6%

Table 2. Results for the shoes scene

the mis-colored voxels in (b) floating above the multi-colored circles have been cleared up in (c) and (d). Since it is somewhat difficult to see these differences in such small images, we provide a close-up in figure 10.

For carving voxels using non-segmented photographs like that of the shoes scene, we found it useful to add a penalty to the reprojection error to prohibit holes from forming in surfaces. Recall that for non-segmented photographs, the reprojection error is computed over the pixels in the projection of the reconstructed scene. If a voxel that projects to pixels P, which occlude no other pixels, is carved, then these pixels P do not contribute to the reprojection error. As a result, there can be little to no penalty for removing such a voxel, which can result in holes if surfaces that should be reconstructed project to such pixels. Therefore, when carving using non-segmented photographs, we count the number of pixels before and after the surface modification, and assign a penalty based on the difference. For the shoes scene, we assigned a penalty of 50 for each differing pixel. This penalty lowers the probability of a surface hole forming, while allowing for surface modifications over pixels P that significantly lower the reprojection error to be accepted.

We have shown that our methods produce refined scenes that better reproduce the original photographs. To help substantiate the claim that these refined surfaces project to better images at novel viewpoints, consider the results presented in figure 11. In this figure, new views were synthesized for both the toycar and shoes scenes. Figures (a) and (c) show synthesized views using the output of the GVC-LDI algorithm. Figures (b) and (d) show refined surfaces using our simulated annealing optimization method. Notice the thinner, less noisy, and more detailed appearance of surfaces in (b) and (d).

6 Conclusion

In this paper, a new methodology for improved reconstruction quality of volumetric scenes was presented. The methodology extends voxel coloring algorithms by introducing tentative surface modifications, which are performed to minimize a reprojection error measure. The image space reprojection error is used to guide scene modification in object space. We have presented two techniques to minimize reprojection error. Our greedy method only accepts scene modifications that decrease the reprojection error, while our simulated annealing method probabilistically accepts changes that increase the reprojection error; the probability a function of temperature.

We have shown results for synthetic and real scenes that demonstrate the effectiveness of our algorithms. The improved visual quality results mostly from a thinning of the reconstructed surfaces in the scene. The refined scene better reproduces the photographs and projects to better images at novel viewpoints.

7 Future Work

We have demonstrated in this paper that the optimal surface is not unique. It might be interesting to identify desirable surface traits, and then search for volumetric models with such traits among the equivalence class of optimal reconstructions. For example, given an optimal reconstruction, one might try to minimize surface area, or maximize voxel connectedness. Alternatively, a cost function could be defined which includes new terms in addition to the reprojection error. A term that measures the dissimilarity of edges in the reprojected images with those in the photographs might be one worth exploring. A penalty for cusping could help reconstruct planar surfaces. Furthermore, the optimization criterion could be extended to encompass lighting and surface reflectance properties.

We have explored a few methods for arriving at a reasonable minimum in the error surface. Other methods, such as genetic algorithms, might prove fruitful in exploring the huge, multi-modal error space. In addition, we plan applying these approaches to other scenes.

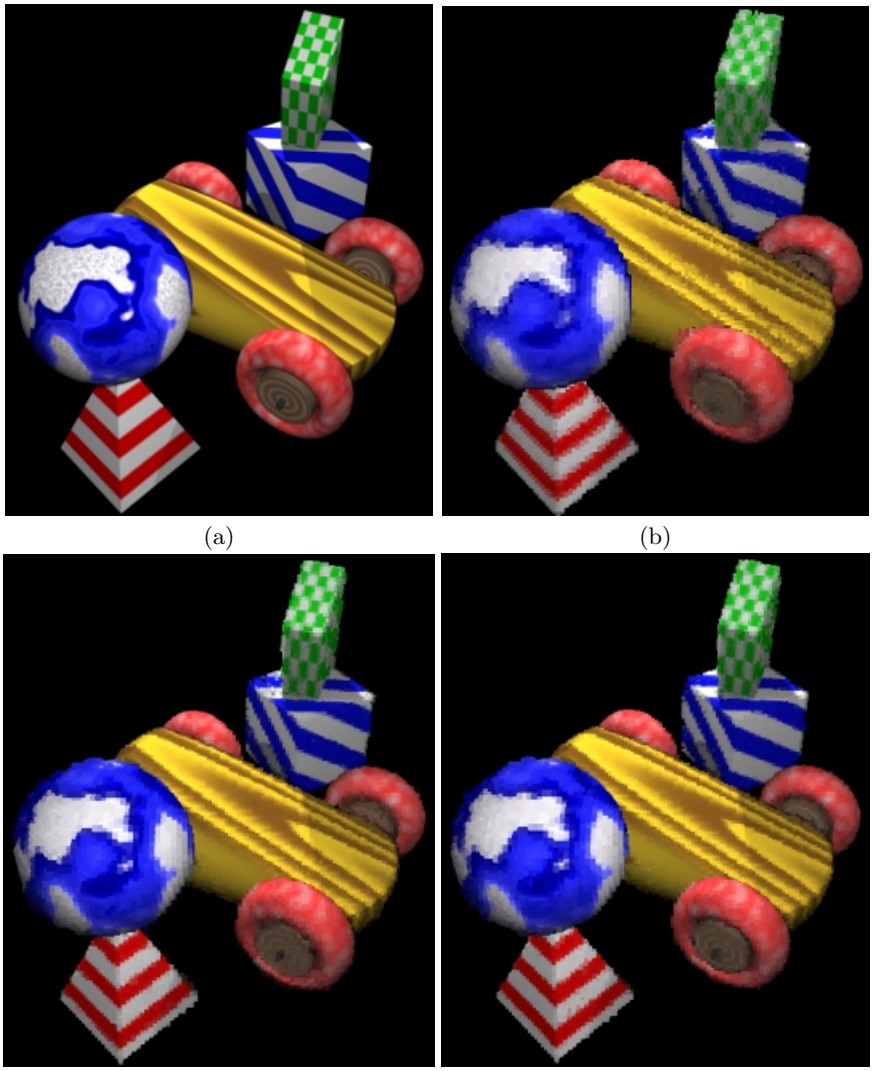
Finally, our code has not yet been optimized, and all rendering is done in software. It would be an interesting exercise to develop software optimizations and hardware implementations to decrease execution time.

Acknowledgments

We would like to thank Steve Seitz and Chuck Dyer for numerous discussions regarding volumetric scene reconstruction. We also would like to express our gratitude to Fred Kitson for his continued support of this research. Finally, we thank Mark Livingston for writing a viewer for volumetric scenes.

References

1. W. B. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized Voxel Coloring," *Proc. of the Vision Algorithms Workshop of ICCV*, pp. 67 - 74, 1999.
2. J. De Bonet and P. Viola, "Roxels: Responsibility Weighted 3D Volume Reconstruction," *Proc. ICCV*, Vol. 1, pp. 418 - 425, 1999.
3. P. Fau, "Reconstructing Complex Surfaces from Multiple Stereo Views," *International Journal of Computer Vision*, 24, pp. 19 - 35, 1997.
4. P. Fau and Y. Leclerc, "Object-Centered Surface Reconstruction: Combining Multi-Image Stereo and Shading," *International Journal of Computer Vision*, 16, pp. 35 - 56, 1995.
5. O. Faugeras and R. Keriven, "Complete Dense Stereovision using Level Set Methods," *Proc. ECCV*, Vol. 1, pp. 379 - 393, 1998.
6. S. Kirkpatrick, C. D. Gelatt, M.P. Vecchi, "Optimization by Simulated Annealing," *Science* 220(4589), pp. 671 - 680, 1983.
7. K. Kutulakos and S. Seitz, "A Theory of Shape by Space Carving," *Proc. ICCV*, Vol. 1, pp. 307 - 314, 1999.
8. A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering*, Addison-Wesley, 1993.
9. A. Prock and C. Dyer, "Towards Real-Time Voxel Coloring," *DARPA Image Understanding Workshop*, pp. 315-321, 1998.
10. A. Rockwood and J. Winget, "Three-Dimensional Object Reconstruction From Two-Dimensional Images," *Computer-Aided Design*, Vol. 29, No. 4, pp. 279 - 285, 1997.
11. S. Seitz and C. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," *Proc. CVPR*, pp. 1067-1073, 1997.
12. S. Seitz, 3D Photography Course Notes, Siggraph, 1999.
13. J. Shade, S. Gortler, L. He, R. Szeliski, "Layered Depth Images," *Proc. SIGGRAPH*, pp. 231-242, 1998.
14. M. Stevens, "Reasoning About Object Appearance in the Context of a Scene," PhD Thesis, Department of Computer Science, Colorado State University, 1999.
15. R. Szeliski and P. Golland, "Stereo Matching with Transparency and Matting," *Proc. ICCV*, pp. 517 - 524, 1998.



(a) (b)

(c) (d)

Fig. 7. Refinement of the toycar reconstruction. One of the seventeen photographs of the scene is shown in (a). A reconstruction of the scene was performed using the GVD-LDI algorithm. The reconstructed scene was reprojected to the same viewpoint as that of (a), resulting in image shown in (b). The reconstructed scene was then refined using the greedy method (c) and the simulated annealing method (d).

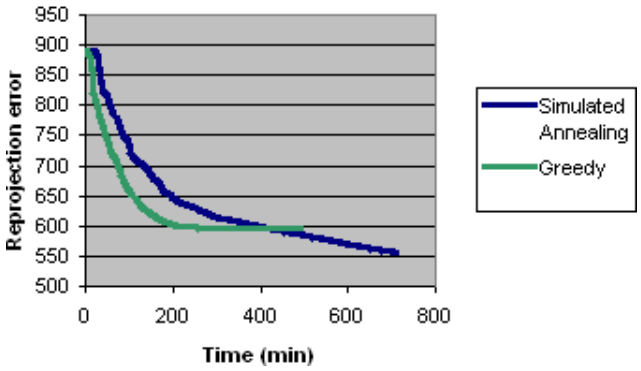


Fig. 8. Reprojection error vs. time, for greedy and simulated annealing methods.

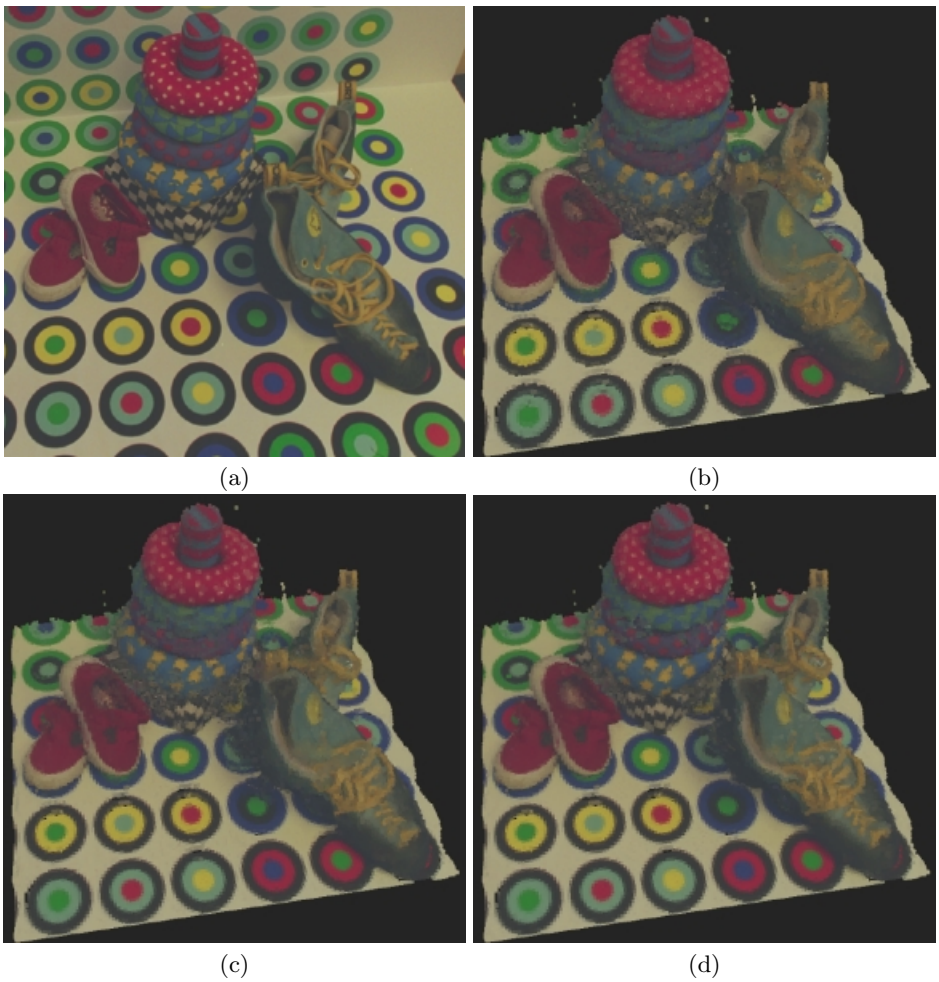
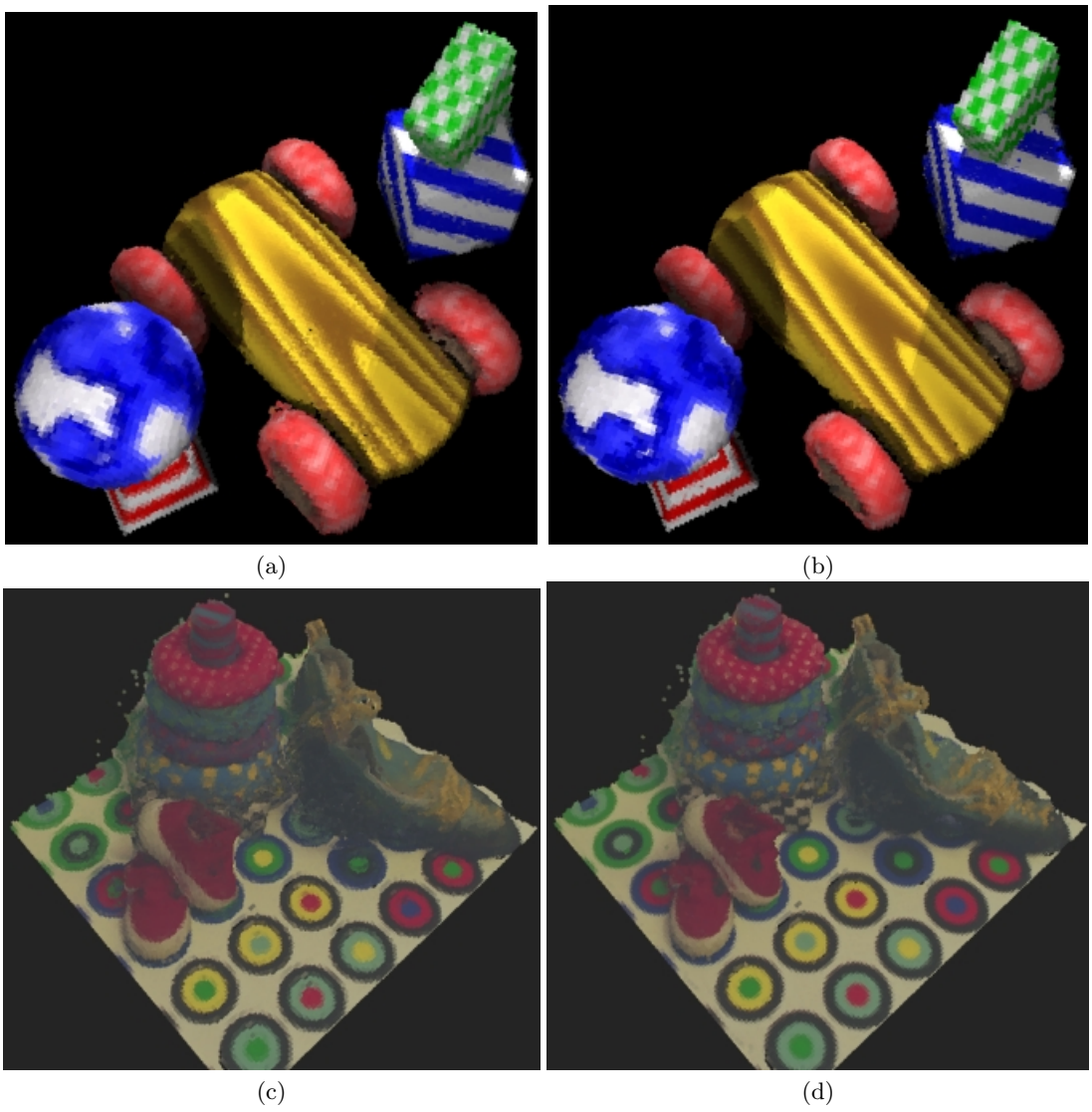


Fig. 9. Refinement of the toycar reconstruction. One of the seventeen photographs of the scene is shown in (a). A reconstruction of the scene was performed using the GVD-LDI algorithm. The reconstructed scene was reprojected to the same viewpoint as that of (a), resulting in image shown in (b). The reconstructed scene was then refined using the greedy method (c) and the simulated annealing method (d).

(a) (b) (c) (d)

Fig. 10. Close-up of cloth toy. As before, (a) is from the original photograph, (b) is a reconstruction using the GVC-LDI algorithm, and (c) and (d) are refined scenes using the greedy and simulated annealing methods, respectively. Notice how more detail is visible in (c) and (d) than in (b); in particular, the white speckles in the topmost cushion, and the black and white checkerboard pattern on the lowest cushion.



(a) (b)

(c) (d)

Fig. 11. Optimized surfaces project to better new views. The scene reconstructed by the GVC-LDI algorithm is shown for a new synthetic view in (a) and (c). The refined scene output by the simulated annealing optimization method is shown in (b) and (d).