3D Reconstruction Using Labeled Image Regions

Eurographics Symposium on Geometry Processing (2003)
L. Kobbelt, P. Schröder, H. Hoppe (Editors)

3D Reconstruction Using Labeled Image Regions

Remo Ziegler,[1], Wojciech Matusik,[2] Hanspeter Pfister[1] and Leonard McMillan[3]

[1] Mitsubishi Electric Research Labs, Cambridge, Massachusetts, USA
[2] Laboratory for Computer Science, MIT, Cambridge, Massachusetts, USA
[3] University of North Carolina at Chapel Hill, Chapel Hill, North Carolina, USA

Abstract

In this paper we present a novel algorithm for reconstructing 3D scenes from a set of images. The user defines a set of polygonal regions with corresponding labels in each image using familiar 2D photo-editing tools. Our reconstruction algorithm computes the 3D model with maximum volume that is consistent with the set of regions in the input images. The algorithm is fast, uses only 2D intersection operations, and directly computes a polygonal model. We implemented a user-assisted system for 3D scene reconstruction and show results on scenes that are difficult or impossible to reconstruct with other methods.

Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Artificial Intelligence]: Vision and Scene Understanding: modeling and recovery of physical attributes; I.3.3 [Computer Graphics]: Picture/Image Generation: display algorithms

1. Introduction

Creating photorealistic 3D models of a scene from multiple photographs is a fundamental problem in computer vision and image-based modeling. The emphasis for most computer vision algorithms is on automatic reconstruction of the scene with little or no user intervention. Consequently, these algorithms make *a priori* assumptions about the geometry or reflectance of the objects in the scene. In contrast, many image-based modeling systems require that the user directs the construction of a 3D model that is used as a scene representation. However, the specification of these 3D models can be difficult, and tools for it are not very flexible, i.e., they allow only for a very simple set of shapes.

In this paper we propose a system for 3D model creation that only requires simple 2D photo-editing operations from the user. The input to our system is a set of calibrated images of a scene taken from arbitrary viewpoints. The user first identifies 2D polygonal regions in each image using simple segmentation tools, such as polylines and intelligent scissors [18]. Each region is assigned an ID, such that regions with corresponding IDs in two or more images are projections of the same object in the world. For example, images of a person could be segmented into head, torso, legs, and hands. Figure 1 shows an example of input images and the corresponding user segmentation.

Our algorithm automatically computes the 3D geometric model that is consistent with the user's segmentation. The solution is unique, in that is it the maximum volume that reproduces the user's input. The algorithm is fast and directly computes one or more watertight polygon meshes of the scene. The model can be immediately displayed using graphics hardware, as shown in Figure 1.

Figure 1: *Left: Example input photograph. Middle: User labeling of image regions. Right: 3D model reconstructed from nine labeled images.*

A fundamental problem in 3D reconstruction is assigning correspondences between points in two or more images that are projections of the same point in three dimensions. Previous work uses pixels or object silhouettes to specify correspondence. Our approach is the first to use correspondence between arbitrary image regions. In this paper we present a new problem formulation for general scene reconstruction

from image regions. We define a necessary condition for the removal of labeling inconsistencies from multiple views. We develop a new algorithm, called *cell carving*, and show how it computes a unique solution to the problem. We also present an efficient implementation of cell carving that uses graphics hardware and standard 2D polygon intersections.

Our user-assisted 3D reconstruction system is able to reconstruct scenes that are very difficult or impossible to reconstruct with traditional methods: scenes with significant occlusion among multiple objects, large variations in geometric scale and detail, and a mix of textured, uniformly colored, specular, and transparent surfaces.

2. Previous Work

The problem of 3D model reconstruction from photographs has received a tremendous amount of attention in the computer vision literature. Excellent overviews can be found in [11, 7]. Here we review the research that is most relevant to our work.

2.1. Multi-View Stereo Reconstruction

Multi-view stereo algorithms reconstruct 3D structure by automatically computing pixel correspondences across images. Stereo correspondence techniques work well when the distance between viewpoints (often called the baseline) is not too big. This is especially true for video sequences [22], where tracking is used to assist in correspondence matching between frames. To deal with large changes in viewpoints, some approaches extract partial 3D shape from a subset of the photographs using multi-baseline stereo algorithms [19, 33, 26]. However, to produce a single 3D model requires complex reconstruction and merge algorithms [32, 5], and there are no guarantees on global consistency with the entire set of photographs for the merged model.

Accurate point correspondences are difficult to compute in regions with homogeneous color and intensity. View-dependent effects, such as specular highlights or reflections, lead to correspondence mismatches. Obtaining dense correspondence for many image points is especially hard. Finally, differences between images due to occlusions are difficult to handle. This is a severe problem for general scene reconstruction where such occlusions happen frequently.

2.2. Shape From Silhouettes

Shape-from-silhouette techniques – for example [15, 30] – compute the 3D model as the intersection of visual rays from the camera centers through all points on the silhouette of the object. Because of their simple and efficient computer vision processing, shape-from-silhouette methods are especially successful for real-time virtual reality applications [14, 17].

Shape-from-silhouette techniques reconstruct a shape known as the *visual hull* [12] – they can never recover concavities. Li et al. [13] improve the visual hull by adding depth from multi-view stereo, subject to some of the drawbacks mentioned above. Shape-from-silhouette methods fail for scenes with multiple, occluding objects, and they only work for outside-looking-in camera arrangements. [†] Our cell carving algorithm is able to reconstruct concavities. It handles arbitrary object occlusions and camera placements.

2.3. Photometric Techniques

Additional photometric constraints can be used to recover a shape that is demonstrably better than the visual hull [11]. Voxel coloring [27] gradually carves out voxels from a 3D volume that are not color-consistent with all images. Recent approaches extend voxel coloring to arbitrary camera placements [4, 11], graphics hardware acceleration [25], and multiple color hypothesis for each voxel [29].

Voxel approaches do not work well for large scenes or objects with big differences in scale. Constructing a surface model for interactive viewing requires a lengthy process that may introduce inconsistencies in the mesh. Fundamentally, all photometric approaches rely on a locally computable analytic model of reflectance. This assumption fails for global illumination effects such as shadows, transparency, or interreflections. Since simultaneous recovery of surface shape, reflectance, and illumination is difficult, all photometric reconstruction techniques assume that surfaces are Lambertian or nearly so. They do not work for objects with homogeneous surface color. Our motivation for developing a user-assisted reconstruction method was to overcome these limitations and to reconstruct highly specular, transparent, and uniformly colored objects.

2.4. Image-based Modeling

Image-based modeling systems split the task of 3D model construction between the user and the computer. Some methods reconstruct a model from a single image using user-defined billboards [9] or depth images [10]. Oh et al. [20] use a single image and provide various tools to paint a depth image, edit the model, and change the illumination in the photograph. Despite the utility and versatility of these systems, specifying depth images for multiple photographs requires an impractical amount of user input. More importantly, there are no guarantees that the resulting 3D model is globally consistent with all the input photographs.

It is of considerable advantage to use geometric constraints when the scene is known to have a certain structure. Some systems use user-guided placement of polyhedral

[†] The scene must lie inside the convex hull of the cameras. The opposite configuration is called inside-looking-out.

primitives to reconstruct architectural scenes [6, 8], others exploit the geometric characteristics of scenes with planes and parallel lines [3, 23]. Similar methods have been successfully adopted by commercial products [2, 21, 24]. However, these approaches are not applicable for reconstructing arbitrary scenes, and they rely on lengthy optimization procedures.

Our method follows the tradition of image-based modeling by relying on the human for the difficult task of assigning region correspondences. Our algorithm is able to automatically reconstruct a globally consistent mesh without making geometric assumptions about the scene. Because cell carving uses geometry for reconstruction, it is able to include geometric constraints (for example, marking planar regions in each image).

3. Cell Carving Algorithm

We first look at the problem of n-view reconstruction using image region correspondence. As a solution to the problem we introduce the cell carving algorithm. In Section 6 we present an efficient implementation of cell carving using graphics hardware.

3.1. Problem Formulation

We are given n images of the same scene taken with calibrated cameras from arbitrary viewpoints. The scene is a static environment of two- and three-dimensional impenetrable objects. In this discussion, we consider each camera to be omni-directional with a spherical image plane at a unit distance from the center of projection (*COP*). All rays entering the camera are parameterized in terms of spherical coordinates (ρ, θ). Later we will use the more familiar pinhole camera model.

Each point in the scene can be projected into any camera by intersecting the image sphere with the directed line from the point to the camera's COP. For any two points, we say that P_1 is visible from P_2 if the open segment P_1P_2 does not intersect any object in the scene. We produce an image by projecting all visible points in the scene onto the image sphere. A depth image is an image that stores the depth along the line of sight to each visible point. A depth discontinuity is an abrupt depth change in a depth image.

Suppose we take n images of a scene where all objects or parts of objects have some diffuse color. Each point in an image will contain the color of the closest visible object surface along the line-of-sight. Ignoring depth or color discontinuities for the time being, we call the set of all image points with the same color an *image region*. We call the color of an image region its *region ID*. An image that contains region IDs for each point is called a *label image*.

We can now pose the following reconstruction problem: Given n label images, what is the scene that produces them? As shown by Kutulakos and Seitz [11], there may be infinitely many such scenes. To get a unique solution we restrict ourselves to finding the scene with maximum volume that produces the label images. ‡

3.2. Consistency and Cells

Consider shooting a ray from a point p in the scene to a camera C. The ray intersects the label image of camera C at an image region with an associated region ID. A point is *consistent* with a set of *visible* cameras if it projects onto the same region IDs in these cameras. If the point is not visible in a camera, it is trivially consistent with that camera. A point is also trivially consistent if it is visible in only one camera. Figure 2 shows a 2D example with consistent and inconsistent points for a set of three cameras.

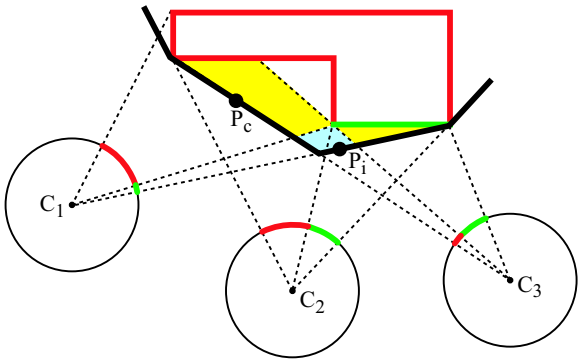

Figure 2: *2D example of consistency. P_i is inconsistent because the region IDs of its projections are not the same. P_c is consistent. Cyan cells are inconsistent, and yellow cells consistent. The thick black outline is the initial volume.*

Now consider shooting rays from a point p to all cameras and identifying all image regions where the rays intersect the image planes. Note that we do this without taking occlusions into account. We list the region IDs of the projections as a vector with one entry per camera. We call this vector the *point ID*. Each 3D point has exactly one point ID, but there may be many points with the same ID. We call the set of points with the same ID a *cell*. The point ID of points in a cell is called the *cell ID*.

A cell is called consistent with a set of cameras if and only if all points of the cell are consistent with these cameras. Because only visible points can be consistent, we can restrict the consistency checks to the boundary of cells. Figure 2 shows examples of consistent and inconsistent cells. Note that some cells are inconsistent inside the visual hull of the object.

Our definition of consistency is very similar to the definition of photo-consistency by Kutulakos and Seitz [11]. The

‡ Some other criteria could be used. For example, one could try to find the smoothest scene that produces the label images.

only difference is that we do not consider background regions because we assume omni-directional cameras for now. Kutulakos and Seitz [11] call the maximum volume that is photo-consistent the *photo hull*. If the label images are produced by projecting a scene as described above, then our reconstruction problem is equivalent to finding the photo hull. Kutulakos and Seitz prove that all points inside the photo hull are consistent, and that all points outside the photo hull are inconsistent. Using this fact, we prove the following theorem in Appendix A:

Theorem (Cell Reconstruction Theorem): Given n label images. The scene with maximum volume V that produces the label images is composed only of consistent cells C. Cells outside of V are inconsistent.

3.3. Necessary Conditions for Inconsistency

This theorem states that removal of all inconsistent cells will yield the maximum volume V that produces the label images. The next important question is how many different colors have to be observed on a cell to make it inconsistent? Kutulakos and Seitz [11] state that any point (cell) with at least two different colors is inconsistent. However, as Figure 3 demonstrates in 2D, two colors are not sufficient to label cells as inconsistent.

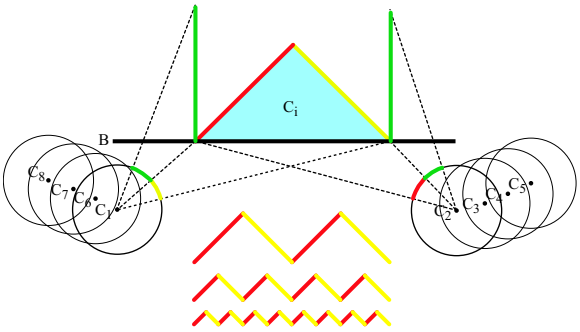

Figure 3: *Two colors are insufficient to determine the inconsistency of cells in 2D.*

The object in the figure contains a V-shaped concave region with two distinct uniform colors (red and yellow). The two cameras C_1 and C_2 are positioned such that each camera sees the projection of exactly one of the two colors. The initial volume for the scene is outlined by the thick black line B (we discuss initial volumes in Section 4).

The cell C_i (cyan) would be removed because of photo-inconsistency (cameras (C_1, C_2) see two different colors). However, this is incorrect, since the maximum volume that is photo-consistent with the images may include large parts of C_i. As shown on the bottom in the figure, the V-shaped concavity can be replaced by infinitely many shapes with two colors that are also not photo-consistent with the two cameras. In the limit, these shapes converge to a plane of micro-facets that project a different color into each camera. As a matter of fact, one could place infinitely many cameras (e.g., C_3 to C_8) without ever being able to remove the cell C_i.

We need to observe at least three different colors to determine the inconsistency of a cell in 2D. The equivalent example in 3D consists of stacked cubes with different colors on each visible side viewed orthographically from the direction of the face normals. That means we need to observe at least four different colors to determine inconsistency in 3D.

Intuitively, the microfacets in Figure 3 separate the cameras into two classes, each of which observes exactly one color on the surface. It is impossible to find a geometric construction in 2D that would yield three classes of cameras, each observing a different color. A similar argument holds true for three dimensions.

The requirement of observing four different colors to determine inconsistency in 3D may be hard to meet in practice, especially with a small number of input photographs. Fortunately, our system enables the user to resolve these situations with additional image segmentation. If the user identifies all interior edges with depth discontinuities, we need only two colors for inconsistency checks.

For example, Figure 4 shows a scene with two cameras C_1 and C_2. The user has introduced additional regions (shown

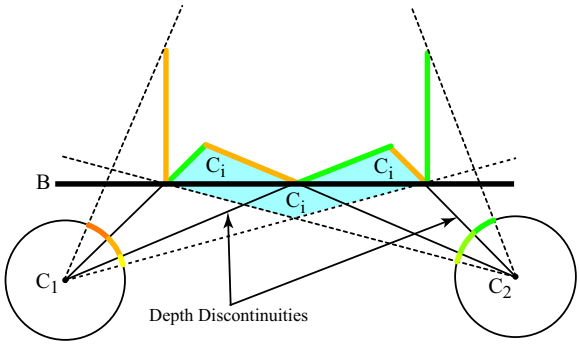

Figure 4: *Segmentation along depth discontinuities allows to correctly determine consistency with fewer than four colors.*

with different colors) along apparent silhouettes to indicated depth discontinuities in the image. As shown in the figure, the inconsistent cells C_i can now correctly be determined.

3.4. Reconstruction by Cell Carving

Now we describe a scene reconstruction algorithm. Given an initial set of cells V_i that contains the scene, the algorithm iteratively removes cells from V_i that are inconsistent in at least four cameras. The remaining cells are the maximum volume that produces the label images. We call this algorithm *cell carving*. Pseudo-code is shown in Figure 5.

```
Initialize the volume containing the true scene.
Cell Carve ( input images, initial volume )
while VolumeConsistent = FALSE do
  Project region IDs onto the current volume.
  Detect a point P on the volume surface that has inconsistent labeling in at least four cameras.
  if there are ID inconsistencies then
    Project inconsistent point P into all input images. It falls into a region with some ID in each image.
    Compute a cell C corresponding to these regions.
    Subtract C from the current volume.
  else
    VolumeConsistent = TRUE
  end if
end while
```

Figure 5: *Cell carving algorithm.*

Cell carving is a generalization of previous scene reconstruction algorithms. Assume the set of image regions is the set of all rectangular image pixels. If we use the scene radiance at each pixel as the pixel's region ID, cell carving reconstructs the photo hull. On the other hand, if the user assigns the same region ID to foreground and background objects, respectively, cell carving reconstructs the visual hull. The quality of the scene reconstruction depends on how much effort the user puts in specifying the image regions: Many small regions will result in more detailed 3D models than a few, large regions. This is also a generalization of voxel carving, in that the fidelity of the scene is not determined by some pre-determined voxel grid, but by actual geometry observed in the images.

The number of image regions determines the number of cells that make up the maximum volume. Assume we have n images and m regions in each image, then there are at most m^n cells. However, the vast majority of these cells are empty. There are configurations where there may be m^3 consistent cells, for example, if the images correspond to three sides of the orthographic cube. In our case, m is typically small (usually less than 20 regions per image). To reduce processing time and memory footprint even further, we construct cells only when needed, as described in Section 6.

4. Additional Constraints

So far, we have described the theoretical foundation of cell carving. We now discuss some additional constraints that make the algorithm practical for implementation.

We first restrict the omni-directional camera assumption to a perspective pinhole camera. Projections of lines in three dimensions now remain lines in the image plane. Conversely, regions defined by polygons lead to polyhedral cells in 3D. This allows us to use a fast method for constructing polyhedra [16] that only uses 2D polygon intersections (see Section 6).

To uniquely identify cells, all points in the scene need to project to a region in all cameras. However, pinhole cameras have a limited viewing frustum. We simulate an omni-directional camera by assuming that the image plane is infinite. 3D points outside the viewing frustum project to a region on either side of the image plane with a unique *unknown ID*. Other image regions that should not be reconstructed (e.g., sky) may not be labeled by the user. We assign these unlabeled regions a unique *background ID*.

As discussed in [11], the maximum consistent volume may not be finite. It may be the volume that fills all free space and reproduces the label images. We restrict the scene with an initial bounding volume that is guaranteed to contain the reconstructed solution. The selection of the initial volume depends on the scene and camera arrangement. If all cameras see the objects in the scene, we compute the visual hull and use it as the initial volume. Otherwise we let the user specify an initial volume as described in the next section.

5. User Input

For scenes where calibration is unknown, we use standard computer vision methods [31] to estimate the cameras' intrinsic and extrinsic parameters. The user selects about 20 feature points in each image and assigns point correspondences, from which the algorithm determines the position and orientation of all cameras. We found this procedure to be very robust.

If the visual hull of the scene cannot be computed because of the camera arrangement, we require that the user specifies the initial volume. To simplify the task, we show orthographic views of the cameras and the 3D feature points computed during calibration. The user then specifies corners of the scene bounding box.

The user segments each image into a set of arbitrary 2D image regions using any of the segmentation tools that are commonly available in photo-editing packages, such as brushes, snakes, region growing, flood fills, watershed algorithms, or intelligent paint. We have built a simple photo editing application (see Figure 6) that provides polylines and intelligent scissors [18]. The boundary of a region is defined by a polygon (possibly with holes) that consists of an outer boundary and zero or more inner boundaries. The intersection of adjacent polygons must be a set of common edges to both. The user also assigns a label L to each polygon corresponding to the different objects in the scene.

Color discontinuities: Labeling purely on the basis of color segmentation is not enough. There are cases where non-adjacent regions with the same label will cause excessive carving, as shown in Figure 7. The red object projects to red regions in the three cameras that are disconnected by the yellow region. We call this a *color discontinuity*. The two cells C_{ryr} (cyan) have the same cell ID: red-yellow-red. Because the frontmost of these two cells is inconsistent with

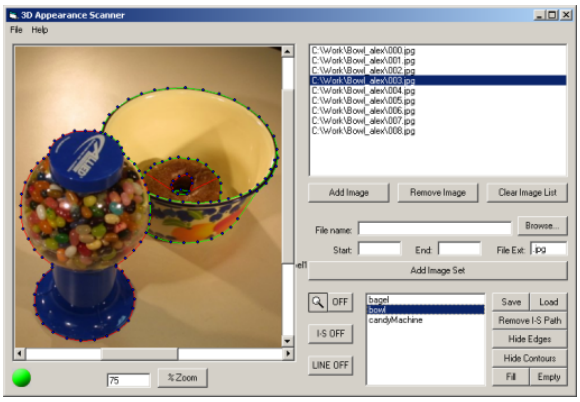

Figure 6: *User segmentation and labeling.*

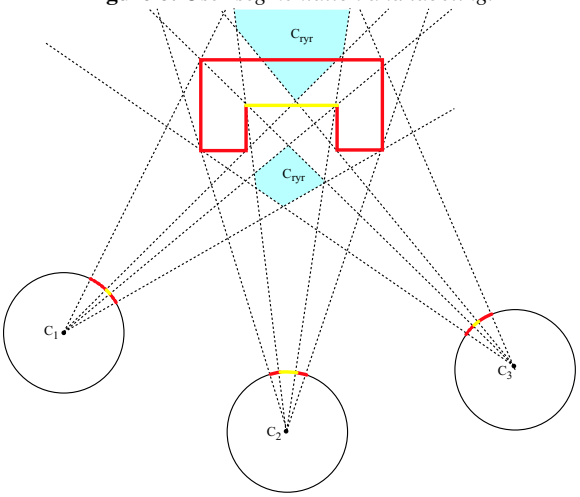

Figure 7: *A 2D example of color discontinuity. The same object (red) projects to disconnected image regions. Without distinct region IDs, both cells C_{ryr} would be wrongly removed.*

the three cameras, it will be removed. But that means the back cell will also be removed because it has the same cell ID. However, the back cell is trivially consistent because it is not visible and should be preserved.

Fortunately, there is an easy remedy to this situation: We automatically assign a unique index I to each disconnected image region. The pair $< L, I >$ is used as the region ID, where L is the user-assigned label. With this approach, the two cells C_{ryr} have different cell IDs, and only the front cell will be removed.

6. Cell Carving Implementation

We now describe an efficient implementation of cell carving that makes use of graphics hardware and fast 2D polygon intersections. Instead of constructing a complete volume of cells, we use lazy evaluation and create cells as needed. The reason is that we are only interested in the boundary representation of the reconstruction. Figure 8 shows the pseudo-code of our implementation.

```
1:  Create volume V containing the true scene
2:  Create a set I of label images
3:  Compute the boundary cells V_c that approximate V
4:  Remove all cells with background IDs
5:  repeat
6:    AllCellsConsistent = TRUE
7:    for Every cell c in V_c do
8:      for Every triangle t in c do
9:        for Every input image i in I do
10:         Project t onto image i with visibility
11:         Record region ID of its projection
12:       end for
13:       k = Number of inconsistent images for t
14:       if k >= 4 then
15:         AllCellsConsistent = FALSE
16:         Remove the inconsistent cell c
17:         Construct the neighbors of c
18:       end if
19:     end for
20:   end for
21: until AllCellsConsistent = TRUE
```

Figure 8: *Cell carving implementation.*

Line 1 and 2 – Pre-processing: As described in Section 5, we are given an initial volume that contains the scene. The user also provides a set of polygons with corresponding region IDs for each image. To quickly check which region ID a point projects to, we scan-convert and flood-fill the polygons with a color corresponding to their region ID.

Line 3 – Construction of the initial boundary cells: We either use the visual hull as the boundary representation of the initial volume, or the user creates a simple bounding box. Because our algorithm works only on cells, we replace this bounding box with a set of cells that encompass it. This way we avoid arbitrary and difficult Constructive Solid Geometry (CSG) operations between cells and the boundary representation.

To construct the initial boundary cells, we pick arbitrary points on the surface of the bounding box that are visible in some of the cameras. We then replace these points with their corresponding cells. To construct a cell, we use a fast algorithm by Matusik et al. [16], which computes the intersection of solids with fixed cross-sections. It reduces this computation to a set of polygon intersections that can be carried out efficiently in 2D. Its run time is nearly quadratic in the number of regions and nearly linear in the number of edges in these regions.

We repeat the construction of cells until no more bounding box is visible. To check visibility, one could use a BSP tree.

Instead, we use graphics hardware and render each cell independently with a different color. If triangles of the bounding box are still visible, we create the corresponding cells until the whole bounding box is covered.

Line 4 – Removing all cells with background IDs: To speed up the cell carving, we first remove all cells that project to a region with background ID. We check all boundary cells to see if they contain the background ID. To remove a cell we delete it from the list of boundary cells and add it to the list of removed cells. To fill holes when removing a boundary cell, we create its neighboring cells if they are not in the lists of boundary cells or removed cells.

To create the neighbors of a cell, an adjacency graph that stores the cell IDs of neighbors for all currently created cells could be used. Because there are typically a small number of cells, we use a less efficient search method to find adjacent cells. Cells outside the initial bounding volume are not created and instead added to the list of removed cells. A cell whose ID is not in the list of boundary cells or removed cells is created as described above.

Lines 7 to 20 – Detecting inconsistent cells: The main loop of the algorithm detects and removes inconsistent cells. We first determine the set of cameras for which a cell is inconsistent. For each cell we keep a list of triangles. For each triangle, we record in which image it is visible by projecting the triangle and recording the region ID of its projection. By construction, projections of cell triangles never cross the boundary of image regions. We then check in how many cameras a triangle has mismatched region IDs. If it is inconsistent with at least k cameras ($k = 4$ in 3D), we tag the corresponding cell as inconsistent. We then remove inconsistent cells and insert the neighboring cells as described above. We repeat this loop until there are no more inconsistent cells.

7. Results

We implemented the user interface to our system in VisualBasic and the cell carving algorithm in C++. All results were computed on a 2 GHz Pentium 4 PC with 1 GB of RAM.

To be able to compare our 3D reconstruction to actual scene geometry, we used a synthetic scene with concavities and objects at different resolutions. We rendered the scene from six viewpoints using 3D StudioMax. Figure 9 shows the original geometry and the six segmented label images. The input images have a resolution of 1024×768.

The upper row in Figure 10 shows the actual geometry for the synthetic scene rendered from novel viewpoints. The middle row in the figure shows the visual hull. Since all cameras are outside-looking-in, we were able to use the visual hull for the construction of the initial boundary cells. The lower row of Figure 10 shows the output of cell carving. Cell carving was able to carve out the teapot from the concavity. Note that the small teapot on the top is represented with more triangles than the rest of the scene.

Images of the following two scenes were taken with a Canon D30 digital camera at 2160×1440 resolution. Figure 11 shows nine images of the "bowl and bagel" scene with corresponding label images. The scene is interesting because it contains an object (the bagel) inside a concavity (the bowl). Figure 12 shows examples of the visual hull and cell carving geometry from novel viewpoints. In contrast to the visual hull, cell carving reconstructs some concavities, even with very few input images. The texture-mapped cell carving reconstruction is displayed with unstructured lumigraph rendering [1].

Figure 13 shows nine images and label images of the "table" scene. The scene contains many specular and transparent objects that are very difficult or impossible to reconstruct with computer vision methods. Figure 14 shows the cell carving geometry of the table objects from novel viewpoints. Most objects are represented with high geometric fidelity and the concavity in the glass is visible.

Figure 15 shows seven images and label images of the "garage" scene. The photographs were taken with a Canon S300 digital consumer camera at 1600×1200 resolution. Note that it is very difficult to provide overhead views in this setting.

Figure 16 shows the visual hull and the output of cell carving from novel viewpoints. The texture-mapped images were rendered using unstructured lumigraph rendering. As expected, cell carving reconstructs the concave geometry of the house corner much better than the visual hull. The difference is also visible in higher texture fidelity when using the cell carving model. It should be noted that seven input images are not enough to yield high quality rendering for some views, as can be seen in the companion animations. However, this scenario is likely to be typical for consumer use.

Table 1 shows quantitative results of our cell carving implementation for different scenes, such as pre-processing time (lines 3 and 4 in Figure 8), run time, number of iterations of the algorithm (lines 7 to 20 in Figure 8), and the number of cells and triangles in the final reconstruction. The

Scene	Preproc.	Time	Iter.	Cells	Tris
Synth	1 min	2 min	8	30	25k
Bowl	2 min	12 min	20	680	62k
Table	4 min	10 min	24	1,010	122k
Garage	2 min	2 min	8	150	16k

Table 1: *Quantitative results for our implementation of cell carving.*

runtime is dependent on the number of edges in all region contours and the total number of images. The number of cells is dependent on the number of regions and the number

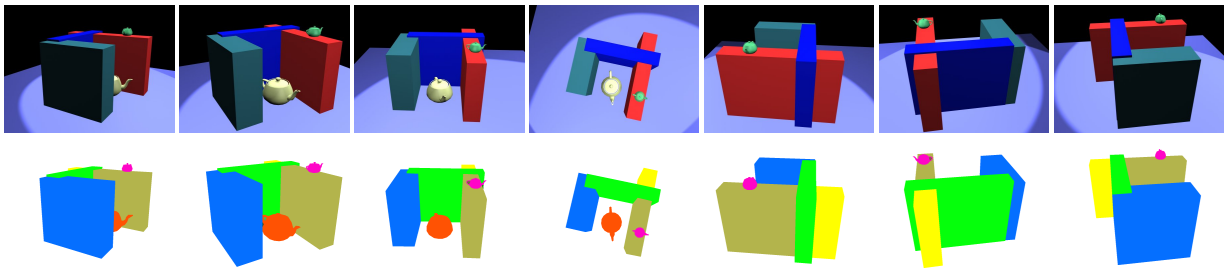

Figure 9: *Seven images of a synthetic scene and the segmented label images.*

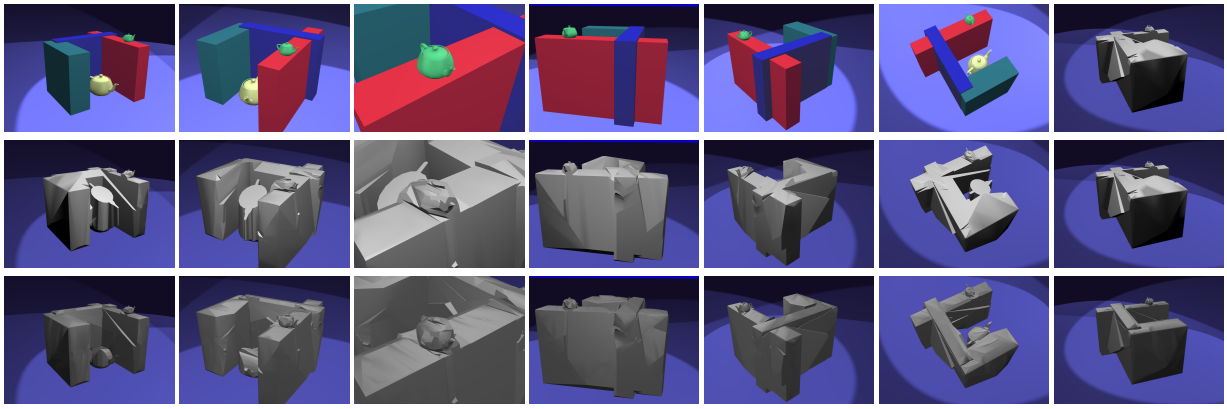

Figure 10: *Upper row: Actual geometry of the synthetic scene. Middle row: Visual hull geometry. Lower row: Cell carving geometry. All images rendered from novel viewpoints that are not part of the input images.*

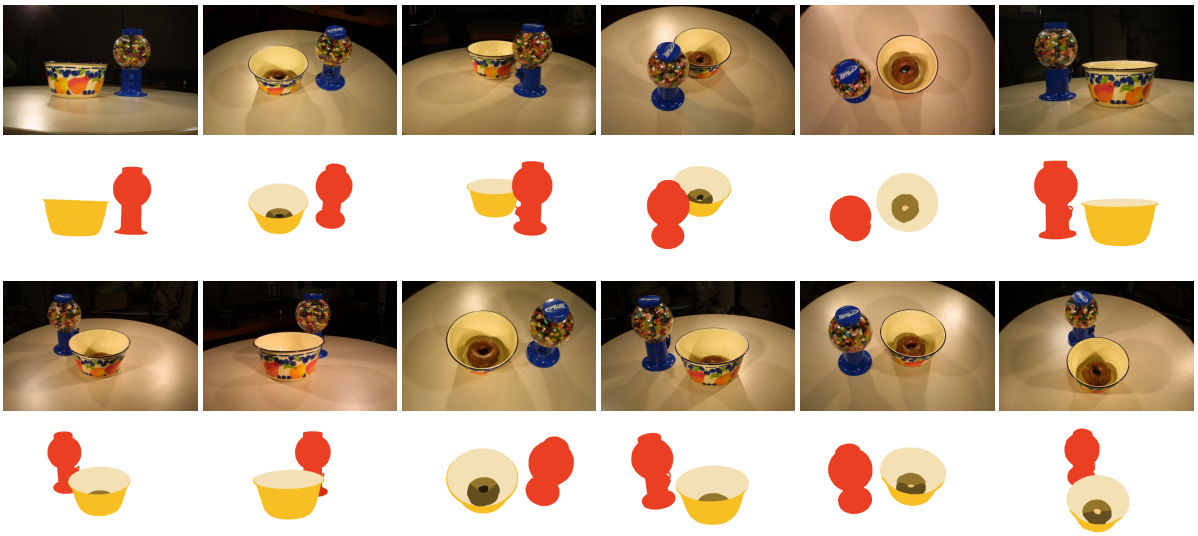

Figure 11: *Twelve images of the bowl and bagel scene with label images.*

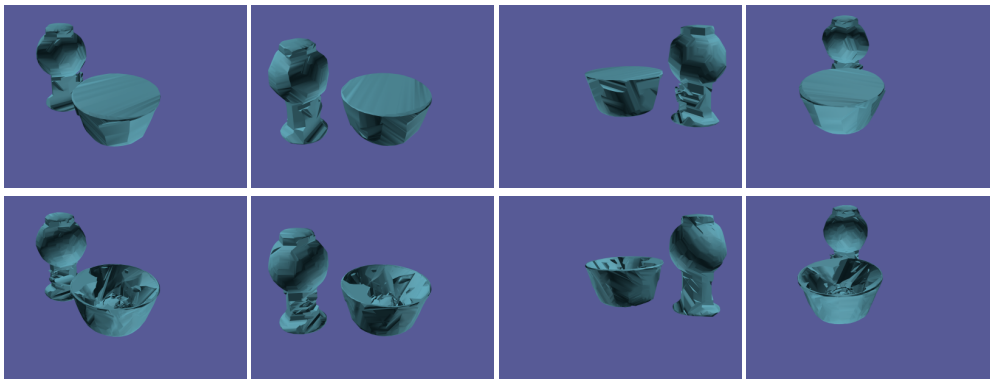

Figure 12: *Upper row: Visual hull geometry. Lower row: Cell carving geometry. All images rendered from novel viewpoints that are not part of the input images.*

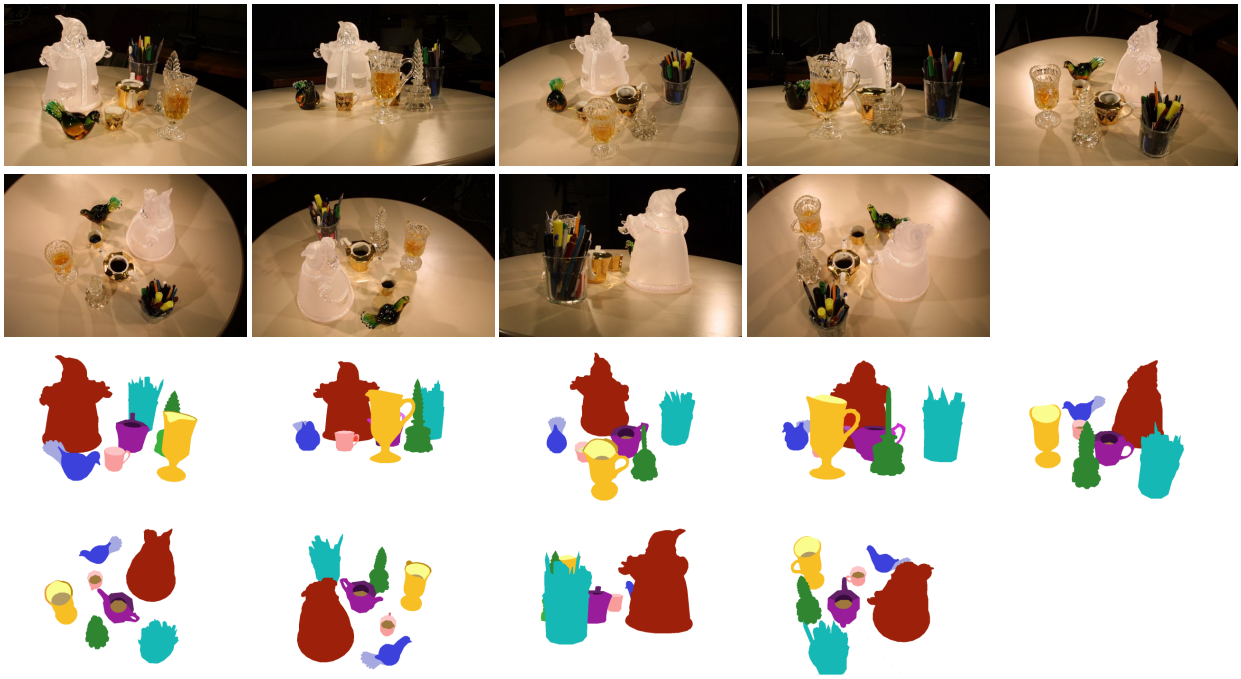

Figure 13: *Images of the table scene with label images.*

of cameras, whereas the number of triangles is dependent on the number of edges in all the region contours.

8. Future Work

We would like to reduce the amount of user input as much as possible. Automatic computation of the initial volume may be achieved using information about camera pose and region correspondences. At some point the user could be eliminated from the loop when unsupervised segmentation algorithms improve and when region correspondences can be assigned automatically.

We could use an analytical visibility solution for more precision in our reconstruction. However, recent advances in graphics hardware – including floating point framebuffers and arithmetics – probably make this unnecessary. To get a smoother reconstruction, one could use other contour representations, such as splines. One could improve the reconstruction using other techniques, for example, multi-view stereo. Per-pixel depth estimates could be used to produce a mesh of the scene with displacement maps.

Furthermore, we would like to add additional geometric constraints. For example, the user could assign a planar constraint to certain image regions, such as walls and ground planes, to improve the reconstruction. We also would like

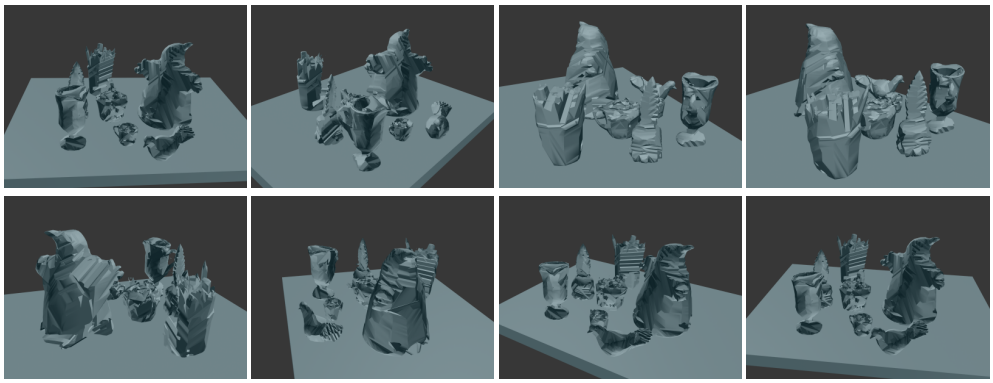

Figure 14: *Cell carving geometry for the table scene.*

Figure 15: *Images of the garage scene with label images.*

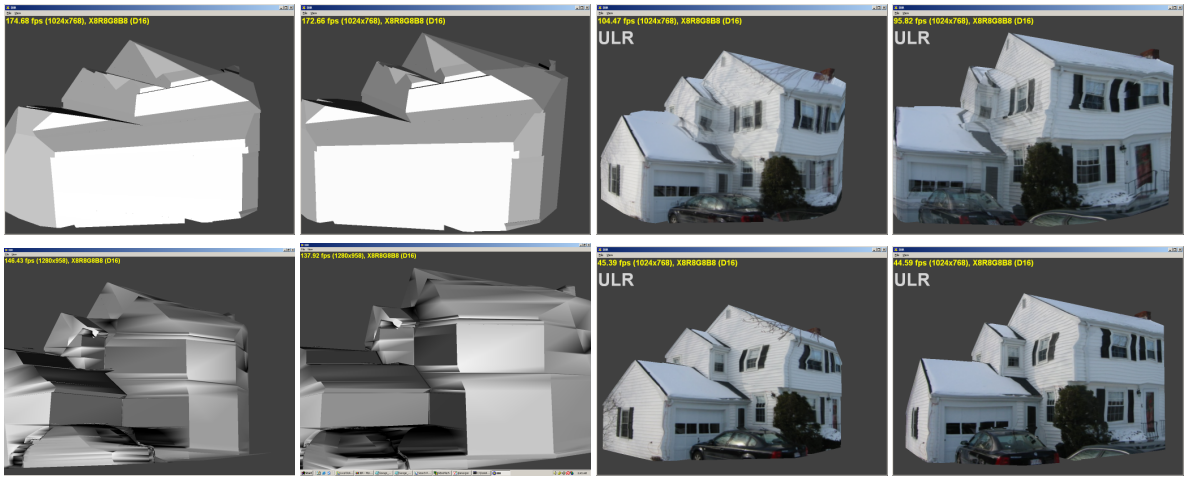

Figure 16: *Upper row: Visual hull geometry of the garage scene. Lower row: Output of cell carving. Note how cell carving reconstructs the concavity. All images rendered from novel viewpoints.*

to introduce interactive scene editing tools as used in Oh et al. [20]. A promising idea is to use plenoptic image editing [28] that propagates edits in one image to all other images using the reconstructed 3D model.

9. Conclusions

We presented a novel method for reconstructing an arbitrary 3D scene from a set of input images. Instead of relying on computer vision algorithms for segmentation and object correspondence, our approach divides the tasks equitably between user and computer: Human beings are good at recognizing and finding corresponding objects in a set of images, whereas computers are well suited to automatically constructing the 3D model consistent with the user's labeling. We have introduced a new formulation of the 3D reconstruction problem for labeled image regions. We have shown necessary conditions for label consistency, and we have proved the correctness of using cells for reconstruction. Our novel cell carving algorithm can be efficiently implemented and produces polygon meshes that can be immediately texture-mapped and displayed.

10. Acknowledgments

We would like to sincerely thank Daniel Vlasic who was involved in the initial stages of this research and who wrote the ULR viewer. We also thank Elena Jakubiak and Alex Wong for designing and implementing the graphical user interface, and Jennifer Roderick Pfister for proof reading an early version of the paper. The anonymous reviewers provided valuable feedback. We especially thank reviewer number two who pointed out a flaw in our initial description of the necessary conditions for photo-inconsistency.

References

1. C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 425–432, Los Angeles, CA, 2001. 7
2. Canoma. http://www.canoma.com/. 3
3. A. Criminisi, I. Reid, and A. Zisserman. Single view metrology. In *Proc. Intl. Conference on Computer Vision*, pages 434–441, 1999. 3
4. W. Culbertson, T. Malzbender, and G. Slabaugh. Generalized voxel coloring. In *Proc. of the ICCV Workshop: Vision Algorithms Theory and Practice*, pages 100–115, September 1999. 2
5. B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 303–312, New Orleans, LS, August 1996. 2
6. P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 11–20, August 1996. 3
7. C. Dyer. *Foundations of Image Understanding*, chapter Volumetric Scene Reconstruction from Multiple Views, pages 469–489. Kluwer, Boston, 2001. 2
8. O. Faugeras, S. Laveau, L. Robert, G. Csurka, and C. Zeller. *Automatic Extraction of Man-Made Objects from Aerial and Space Images*, chapter 3D Reconstruction of Urban Scenes from Sequences of Images. Birkhauser, 1995. 3
9. Y. Horry, K. Anjyo, and K. Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Computer Graphics*, SIGGRAPH 97 Proceedings, pages 225–232, 1997. 2
10. S. Kang. Depth painting for image-based rendering applications. Technical report, Compaq Cambridge Research Lab, 1998. 2
11. K. Kutulakos and S. Seitz. A theory of shape by space carving. *Intl. Journal of Computer Vision*, 38(3):199–218, 2000. 2, 3, 4, 5, 12
12. A. Laurentini. The visual hull concept for silhouette-based image understanding. *PAMI*, 16(2):150–162, February 1994. 2
13. M. Li, H. Schirmacher, M. Magnor, and H.-P. Seidel. Combining stereo and visual hull information for on-line reonstruction and rendering of dynamic scenes. In *Proc. IEEE Workshop on Multimedia and Signal Processing*, 2002. 2
14. B. Lok. Online model reconstruction for interactive virtual environments. In *Symposium on Interactive 3D Graphics*, pages 69–72, March 2001. 2
15. W. Martin and J. K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, March 1983. 2
16. W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of 12th Eurographics Workshop on Rendering*, pages 115–126, 2001. 5, 6
17. W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 369–374, Los Angeles, CA, July 2000. 2
18. E. Mortensen and W. Barrett. Intelligent scissors for image composition. In *Computer Graphics*, SSIGGRAPH 95 Proceedings, pages 191–198, August 1995. 1, 5
19. P. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. Intl. Conference on Computer Vision*, pages 3–10, 1998. 2
20. B. Mok Oh, M. Chen, J. Dorsey, and F. Durand. Image-based modeling and photo editing. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 433–442, August 2001. 2, 11
21. Photomodeler. http://www.photomodeler.com/. 3
22. M. Pollefeys, R. Koch, and L. Van Gool. Self-calibration and metric reconstruction in spite of varying and unkown internal

camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999. 2

23. P. Poulin, M. Ouimet, and M.-C. Frasson. Interactively modeling with photogrammetry. In *Proceedings of Eurographics Workshop on Rendering 98*, pages 93–104, June 1998. 3

24. Realviz image modeler. http://www.realviz.com/. 3

25. M. Sainz, N. Bagherzadeh, and A. Susin. Hardware accelerated voxel carving. In M.P. dos Santos, L. Velho, and X. Pueyo, editors, *Proc. 1st Iberoamerican Symposium in Computer Graphics*, pages 289–297, Portugal, 2002. 2

26. W. Seales and O. Faugeras. Building three-dimensional object models from image sequences. *Computer Vision and Image Understanding*, 61(3):308–324, 1995. 2

27. S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. of Computer Vision and Pattern Recognition*, pages 1067–1073, June 1997. 2

28. S. Seitz and K. Kutulakos. Plenoptic image editing. In *Proc. 6th Intl. Conference on Computer Vision*, pages 17–24, 1998. 11

29. E. Steinbach, B. Girod, P. Eisert, and A. Betz. 3d object reconstruction using spatially extended voxels and multi-hypothesis voxel coloring. In *Proc. of Intl. Conf. on Pattern Recognition*, pages 774–777, 2000. 2

30. R. Szeliski. Rapid octree construction from image sequences. *Computer Vision, Graphics and Image Processing: Image Understanding*, 58(1):23–32, July 1993. 2

31. R. Tsai. A versatile camera calibration technique for high accuracy 3d machine vision metrology using off-the-shelf cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987. 5

32. G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Computer Graphics*, SIGGRAPH 94 Proceedings, pages 311–318, July 1994. 2

33. C. Zhao and R. Mohr. Global three-dimensional surface reconstruction from occluding contours. *Computer Vision and Image Understanding*, 64(1):62–96, 1996. 2

Appendix A: Cell Reconstruction Theorem

Lemma 1: A cell is the maximum volume that projects to the set of regions listed in its cell ID.

Proof (by contradiction): Assume there is another point in the scene outside the cell that projects to the same set of regions. By definition, it is part of the cell. QED.

Lemma 2: The intersection of cells with different cell IDs is empty.

Proof (by contradiction): Assume the intersection is not empty. There must be a point in the intersection of the two cells that has two different point IDs. By construction, every point in the scene has one and only one point ID. QED.

Lemma 3: The scene is completely partitioned by cells C, or $R^3 - \bigcup C_i = \varnothing$.

Proof: This follows directly from Lemma 1 and Lemma 2. QED.

Lemma 4: All points of a cell C have the same visibility in all cameras.

Proof (by contradiction): Suppose there are two points in the cell with different visibility in camera C_k, e.g., P_i is invisible, and P_v is visible. Because region IDs are determined by visible points, P_i and P_v must project to regions with different IDs in camera C_k. That means their point IDs are not the same, which is impossible if they are in the same cell. QED.

Lemma 5: Given n label images. The scene with maximum volume V that produces the label images is composed of cells C such that: $\forall C : V \cap C = C$.

Proof (by contradiction): Suppose there is a cell C that is partially occupied by V. As shown by [11], V contains all consistent points, and $V' = R^3 - V$ contains all inconsistent points. That means C must contain at least one consistent point P_c and one inconsistent point P_i. By definition, consistency is determined by the point ID and the point visibility. Point P_c and P_i have the same point ID by definition of a cell, and they have the same visibility because of lemma 4. That means their consistency is the same. QED.

Theorem (Cell Reconstruction Theorem): Given n label images. The scene with maximum volume V that produces the label images is composed only of consistent cells C. Cells outside of V are inconsistent.

Proof: By Lemma 3 and Lemma 5, R^3 and V are composed completely of cells. Assume there exists a consistent cell C_c that is not part of V. Because C_c is consistent, all points in C_c produce the same region IDs. That means we can add it to V. Assume there is an inconsistent cell C_i that is part of V. That means C_i produces different region IDs, so it can not be part of V. QED.

Corollary: The cell-carving algorithm correctly computes the maximum volume V that produces the label images.

Proof: The cell carving algorithm removes all inconsistent cells from R^3. The remaining cells are in V by the cell reconstruction theorem. QED.