Shape from Silhouette: Image Pixels for Marching Cubes

Bruno Mercier
SIC Lab, Bât. SP2MI, téléport 2
Bd Marie et Pierre Curie
86962 Futuroscope Chasseneuil, France
mercier@sic.univ-poitiers.fr

Daniel Meneveaux
SIC Lab, Bât. SP2MI, téléport 2
Bd Marie et Pierre Curie
86962 Futuroscope Chasseneuil, France
daniel@sic.univ-poitiers.fr

ABSTRACT

In this paper, we propose to use image pixels for geometry reconstruction with a shape from silhouette approach. We aim at estimating shape and normal for the surface of a single object seen through calibrated images. From the voxel-based shape obtained with the algorithm proposed by R. Szeliski in [18], our main contribution concerns the use of image pixels together with marching cubes for constructing a triangular mesh. We also provide a mean for estimating a normal inside each voxel with two different methods: (i) using marching cubes triangles and (ii) using only voxels. As seen in the results, our method proves accurate even for real objects acquired with a usual camera and an inexpensive acquisition system.

Keywords

Geometry reconstruction from images, shape from silhouette, marching cubes.

1. INTRODUCTION

Since the early years of computer vision, much effort has been dedicated to automatically digitizing shape and reflectance of real objects. For instance, Stanford Digital Michelangelo [11] project aims at digitizing large statues, Callet. et al. [1] digitize statuettes and reconstruct plaster models covered with bronze. Hasenfratz et al. [7] use a digitize shape for placing a real actor in a virtual environment so that shadows and lighting be properly computed.

In most cases, acquisition hardware play a major role for reconstructing objects shape and research efforts have increased a lot during the last decade. Our concern is about objects only described with a set of calibrated photographs. Our final goal is to insert real objects (corresponding to lightfields/lumigraphs or described by a series of images) into virtual environments with proper lighting and global illumination. The whole problem is thus not only geometry estimation but also initial light sources position, reflectance properties for the real object as well as rendering process. This paper only addresses a small part of the whole work: geometry and normal estimation.

The Journal of WSCG, Vol.13, ISSN 1213-6964
WSCG'2005, January 31 -February 4, 2005
Plzen, Czech Republic.
Copyright UNION Agency - Science Press.

The basis of our work is the voxel-based shape from silhouette technique presented by Szeliski in [18]. We propose a new method for combining the marching cubes algorithm with image pixels for precisely recovering a triangular mesh corresponding to the object shape. We also propose two methods for estimating surface normal. As shown in the results, our method has a consequent impact on geometry. Based on this method, we have successfully recovered light sources geometry and object surface reflectance properties [14].

This paper is organized as follows. We firstly describe work most related to our concerns. Section 3 presents the acquisition system we use and work overview. We then detail our reconstruction and normal estimation method. Finally, we provide a series of results before we conclude.

2. RELATED WORK

The literature concerning geometry reconstruction is vast and this section only presents a quick run through the area for most closely related works.

Stereo vision [5, 2] uses two cameras located close one to another so that images of the object be slightly different. Internal and external camera parameters knowledge help to determine corresponding points on images and deduce depth with the help of textures, laser grids or structured light. *Shape from shading* methods aim at recovering objects shape with the assumption that surfaces are lambertian (or almost lambertian) [16, 8].

Shape from silhouette approaches [13, 3] are more adapted to our problem since we do not want to make any assumption about images. Objects can have glossy surfaces, with or without textures and we cannot use active laser grids or test patterns to reconstruct the object geometry. Most shape from silhouette methods rely on a voxel-based datastructure and the approach described by R. Szeliski in 1993 [18] is often used as a basis. For such methods, a well-known drawback concerns cavities. For example a coffee cup handle will be recovered since the hole can be deduced from the visual hull on the image, but the inside will be filled-up with material (unless there is no bottom). Improvements have been proposed for solving this problem with voxel coloring [15], space carving [9] or gradient voxel flux [4] with the assumption that objects surface is mostly lambertian.

From voxels, the marching-cubes algorithm can easily generate a triangular mesh corresponding to the object surface [17]. For example Hasenfratz et al. use such a reconstruction method for interactively integrate a real person into a virtual environment [7]. Our method focuses on such approaches and aims at improving the triangular shape accuracy. To achieve this goal, we propose to use image pixels for guiding the marching cubes algorithm and estimating accurate surface normal.

3. WORK OVERVIEW

Acquisition System

For this work, we used images of both virtual objects and real objects. Virtual objects are convenient for validating the method and providing result quality since camera parameters and object geometry are known. For real objects, we devised the acquisition system described in figure 1. Object and light sources are fixed on a turntable: a camera is located on a tripod with fixed aperture and shutter speed. During acquisition, camera position does not change. Every 5 degrees, two images are acquired with the same viewpoint. The first one is overexposed with an additional light source for separating object from background (a lambertian black cloth) while the second one is used for acquiring the actual object radiance. After one turn, the camera is raised of several centimeters. In practice, only 1 turn (76 viewpoints) is necessary for precisely recovering the object shape, but we also used this system for acquiring complete image-based objects.

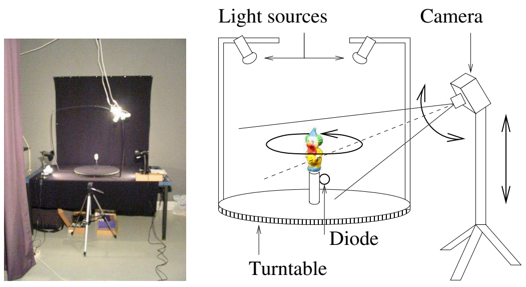


Figure 1: Acquisition system.

Image Processing

For separating background from the object, we use a powerful light source for overexposing images. The object is so bright that it is easy to determine the black background with a seed-fill algorithm even with dark regions on the object. Except background, only 2 connected regions remain: the object and the red diode used for estimating camera orientation. Background pixels are then set as perfectly black on images: (0,0,0) for R,G,B values.

The diode is used to determine the rotation axis of the turntable seen on photographs. Focal length is known a priori (fixed camera parameters) and 3D position is manually estimated. Orientation can thus be deduced from the red diode. We did not use any test pattern for estimating camera parameters.

Reconstruction Process

Our reconstruction method is composed of 3 main steps:

1. the shape from silhouette approach proposed by R. Szeliski in [18] provides a set of voxels;
2. a triangular mesh is generated from marching cubes and image pixels;
3. a normal is estimated for each voxel either with the marching cubes triangles or with a method using only voxels.

4. SHAPE FROM SILHOUETTE

Octree Construction

As a first (rough) approximation of the object geometry, we used the shape from silhouette approach proposed in [18]. With this approach, all the images are used iteratively to hierarchically sculpt the object. The shape initially corresponds to a voxel, recursively refined during the reconstruction process. For each view of the object, the octree voxels obtained so far are projected onto the image plane and compared to image pixels. When a voxel is seen outside the object for one image, it is actually marked as *out* (figure 2(a)); when a voxel is seen inside the object for all the images, it is marked as *in* (figure 2(b)); all the others are often seen inside the object and sometimes on the object boundary, they are marked as *ambiguous* (figure 2(c)) and subdivided into 8 sub-voxels. This process is repeated until no ambiguous voxels exist or a minimum size criterion has been reached. For our method, the algorithm should

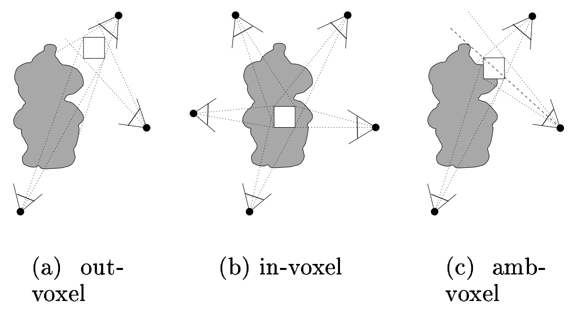


Figure 2: Voxel classification.

stop when ambiguous voxels correspond to a series of 4 (or 9) pixels for more reliable results. Figure 3 shows some results for a clown (real object). Note that for a 256x256 image, pixel resolution corresponds to a depth of 8 in the octree.

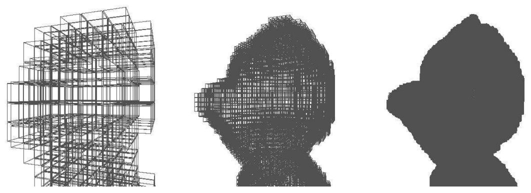


Figure 3: Reconstruction results for levels 5,7 and 9.

Obviously, all ambiguous voxels have the same size, according to the reconstruction process. At the opposite, voxels marked as *in* or *out* are not further subdivided and have various sizes.

Practical Aspects

For our method, voxel projection on the image plane is performed with raytracing. This will be further used with marching cubes in the following section. Each pixel corresponds to a ray originating at the image center-of-projection and going through the pixel (figure 4). These rays are called *pixel-rays* from now on. Pixel-rays corresponding to the object silhouette are called *in-rays* since they hit the object and rays corresponding to background are called *out-rays.*

Surface Thickness

The reconstruction process results in a set of ambiguous voxels called *discrete surface.* For marching cubes, this surface needs to be 6-connected which is not ensured by the previous algorithm. To achieve this goal, we propose to modify the discrete surface with an ad-

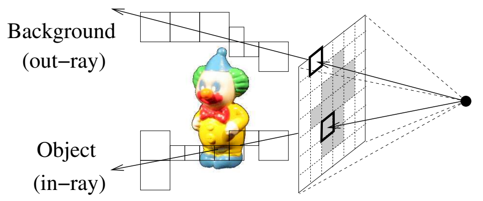


Figure 4: Pixel-rays.

ditional process applied every time ambiguous voxels are subdivided. Voxels classified as in or out can be reclassified as ambiguous when the discrete surface is not 6-connected (as illustrated in figure 5): when two adjacent in and out voxels do not correspond to the same hierarchy level in the octree, we choose to reclassify the smallest one, V_{small}, as ambiguous (since V_{small} parents had been longer classified as ambiguous). In the case of two voxels with similar size, we consider that out-voxels should remain outside the object according to the sculpture method seen above; the in-voxel will consequently be reclassified as ambiguous. A final operation reduces the surface thickness while keeping the 6-connection.

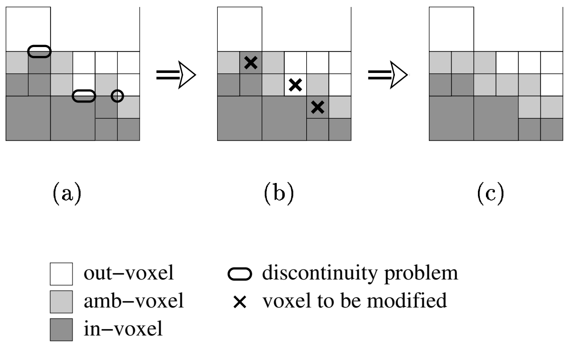


Figure 5: Voxels reclassification; a. "holes" in the surface; b. voxels to be modified; c. modified discrete surface (6-connected in 3D).

5. PIXELS FOR MARCHING CUBES

Original Marching Cubes

For reconstructing a triangular mesh from a set of ambiguous voxels, marching cubes [12] are obviously well-suited to the problem. Originally, the algorithm is dedicated to medical images and uses some density values (weights) associated with each voxel vertex; as a rough simplification, positive weights correspond to points inside the object and negative weights are outside the object. A linear interpolation provides the (estimated) intersection between each voxel edge and the actual object surface. According to these intersections, a triangular mesh can be defined for each voxel with a limited number of configurations (see figure 6).

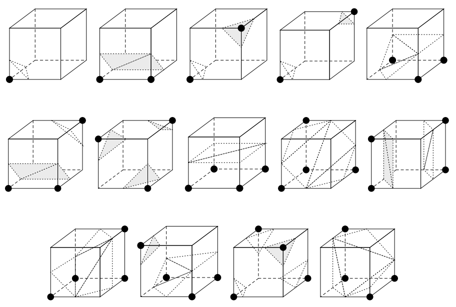


Figure 6: Marching cubes configurations (applied to ambiguous voxels only), black dots indicate vertices located outside the surface.

Since weighting values cannot be computed directly from the discrete surface, edges centers are often used for generating triangles. However, as explained in the following, it is possible to use pixel-rays with marching cubes so that triangles fit the model shape more precisely.

Refining the Method

As a first estimation, voxel vertices are classified according to neighborhood. Each voxel vertex of the 6-connected discrete surface has (at least) either one in-voxel or one out-voxel neighbor. For placing triangles in *ambiguous voxels*, we also use pixel-rays (figure 7).

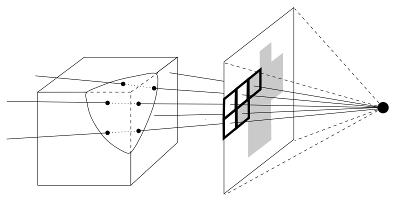


Figure 7: Out-rays should not touch the surface inside an ambiguous voxel; in-rays are not used.

Our algorithm firstly computes the intersection points between out-rays and voxel faces (figure 8). Then, a line corresponding to the object surface on each face is estimated (called *surface line* from now on). This line is placed as close as possible to all intersection points and out vertices. Let us consider the 2D convex hull associated to all these points $\{P_i\}$. It can be shown that for the general case, such a line corresponds to a convex hull line segment L_k. The surface line is thus chosen among $\{L_k\}$ so that the following function be minimized:

$$dist_k = \sum_i dist(P_i, L_k)^2$$

Unfortunately, in some cases, a surface line cannot be computed using the convex hull (see figure 8(d) for example). For such cases, the surface line is defined directly using in-vertices.

Finally, the intersection between surface lines and voxel edge defines the point (*mc-vertex*) used for marching cubes. Note that when the surface line contains a face vertex, our algorithm slightly shift the mc-vertex for avoiding degenerated triangles.

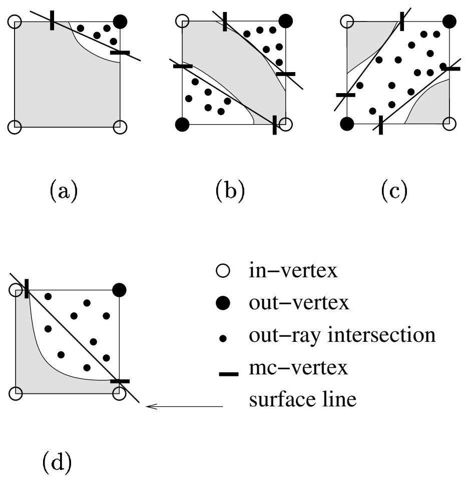


Figure 8: Intersection between out-rays and voxel faces - several cases; a. for usual cases; b. & c. with a disconnected surface; d. voxel vertices used as a boundary.

The defined surface line is bounded by voxel vertices according to classification so that surface continuity be maintained (figure 8(d)). For two adjacent faces, the corresponding surface lines have to be connected (see figure 9(a)). The only adequate choice consists in using the mc-vertex closer to the *in-vertex* since the other one would re-introduce out-rays intersection points inside the object surface. Finally, the same principle applies to adjacent voxels on the discrete surface (figure 9(b)).

6. SURFACE NORMAL

For some application, a normal can be needed for surface voxels (for instance, for estimating light sources properties and BRDF of the object [14]). We propose two methods: the first one uses the triangles generated by our adapted marching cubes algorithm while the second one relies on the discrete surface and its neighborhood.

Normal from Triangles

Inside each ambiguous voxel, several triangles define the object surface. We propose to compute the average triangle normal weighted by area.

Using triangles located in only one voxel leads to a

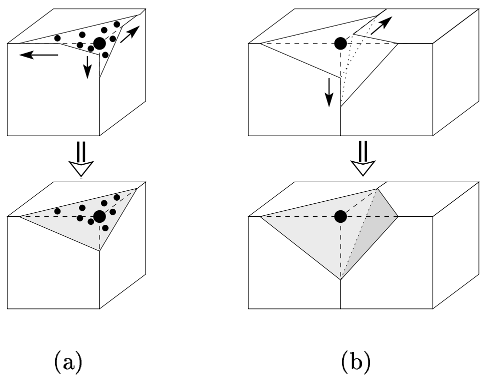


Figure 9: Surface continuity and mc-vertices.

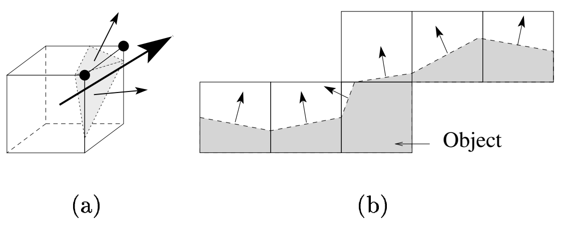


Figure 10: Normal from marching cubes triangles; the surface is *bumpy*.

bumpy normal, introducing artifacts for light sources estimation or during the rendering process (figure 10(b) and 11). This is why we propose to smooth normals according to neighbor voxels triangles. In our application, a parameter called *smoothing distance* is fixed by the user. Practically, with an octree depth equal to 7, our best results have been obtained with a smoothing distance set to 3 or 4 voxels (depending on the object geometry).

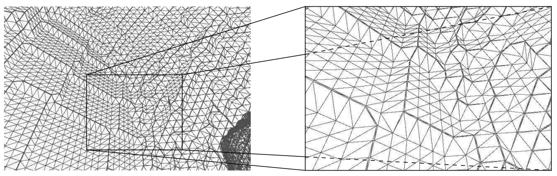


Figure 11: For curved surfaces, due to discrete representation, the marching cubes algorithm does not provide smooth surfaces.

Discrete Normal

Intuitively, a normal estimated from marching cubes should be quite representative of the object surface. However, it is also possible to define a normal in each voxel directly with the discrete surface. Particularly, if a surface mesh is not needed, acceptable results can be obtained. Normal is estimated according to out-voxels in the given neighborhood (figure 12): $\overrightarrow{N} = (\sum_{i=1}^{n} \overrightarrow{V_i})/n$, where $\overrightarrow{V_i}$ corresponds to the unit vector going from the current voxel center to the i^{th} neighbor voxel center; n is the number of voxels used. With this method, normals are defined by a fixed number of directions which could be useful for compression algorithms.

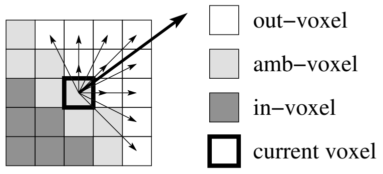


Figure 12: Normal estimation from voxels.

7. RESULTS

Object Shape

Before actually using our method with real objects, validation has been done with known geometric objects (a sphere and a cube). As shown in table 1, using image pixels with marching cubes (MC in the figure) improves consequently shape precision. For the experiments we made, the error is noticeably reduced compared to a marching cubes algorithm with edges centers. Note that when hierarchy depth increases, the two methods tend to provide the same results because the number of pixel-rays becomes lower. Our method will obviously be more accurate with a low-depth hierarchy.

For a 1-meter diameter Sphere			
Octree depth	5	6	7
Edges center MC	*10.8 mm*	*9.3 mm*	*8.9 mm*
Pixel-rays MC	*6.4 mm*	*6.7 mm*	*6.7 mm*

For a 1-meter width Cube			
Octree depth	5	6	7
Edges center MC	*26.4 mm*	*19.4 mm*	*16.7 mm*
Pixel-rays MC	*21.2 mm*	*19.1 mm*	*18.6 mm*

Table 1: Average distance between reconstructed triangles and actual object surface.

Note that a cube is the worst example. A polygon is difficult to reconstruct with a shape from silhouette approach (as any flat surface) since the camera viewpoint is never perfectly located on the polygon plane.

Normal Estimation

This paper describes two different methods for estimating normal inside each voxel. The first one is based on marching cubes triangles while the second one only relies on voxels. For each case, it is possible to smooth normal values according to a user-specified smoothing distance. For estimating normal quality, we compared the estimated normal with the actual (known) surface normal (table 2)

As for surface accuracy, a surface normal obtained with the help of pixel-rays is sensibly more precise (20-25%) than with using edges centers marching cubes or

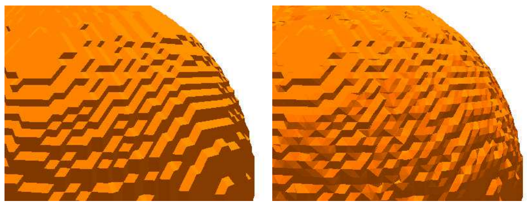
For a 1-meter diameter Sphere			
Octree depth	5	7	
Smoothing distance	1	1	3
Discrete normal	*11.1°*	*12.1°*	*2.3°*
Edges center MC	*5.7°*	*6.3°*	*2.1°*
Pixel-rays MC	*3.3°*	*4.9°*	*1.8°*

Table 2: Average angles difference (degrees) between estimated normal and actual object normal.

the discrete surface. Our method provides a precision with an error less than 5° even without any smoothing. When smoothing, normal precision is about one degree (with a smoothing distance of 5 voxels). Note that with the discrete surface, the smoothing distance should not be less than 3 voxels.

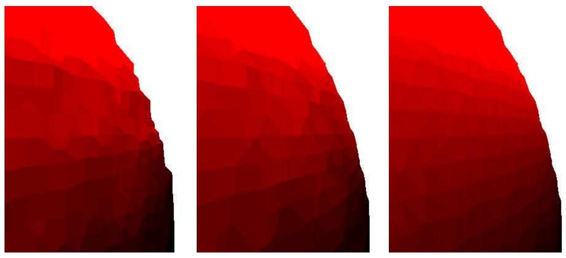
Rendering using Normals

From geometry and normal we have generated new views. Triangles can be directly with Graphics Hardware (OpenGL). For example, figures 13, 15 and 14 show new images.



(a) (b)

Figure 13: Rendering using triangles; a. with *edges-centers* marching cubes; b. with pixel-rays marching cubes.

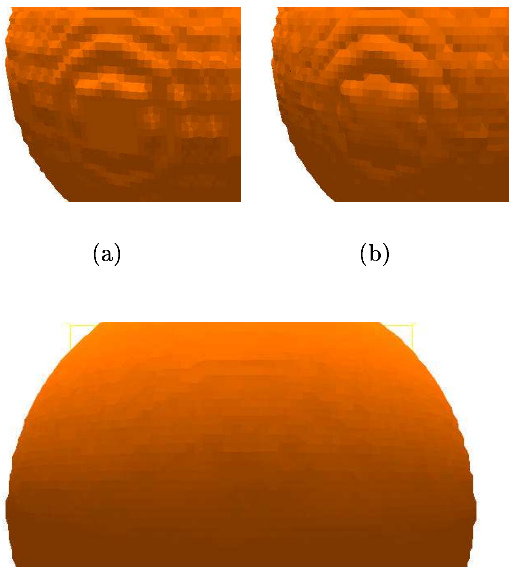


(a) (b) (c)

Figure 14: Rendering using voxel normal with smoothing; a. with *edges-centers* marching cubes note the bumpy surface on object silhouette; b. with pixel-rays, marching cubes contour is smoother; c. with actual sphere normal.

Rendering using Voxel Radiances

It is also possible with pixel-rays (in-rays) to estimate an average radiance emitted by the voxel (figures 16).



(a) (b)

(c)

Figure 15: Rendering using voxel normal: in a voxel, all the triangles normals are replaced by the voxel normal. a. with *edges-centers* Marching cubes; b. with pixel-rays; c. with smoothed normal (distance of 5 voxels).

8. CONCLUSION

This paper presents a method for using image pixels together with marching cubes for a shape from silhouette application. Our work relies on a 6-connected discrete surface obtained with the method proposed by Szeliski [18]. We also propose two methods for estimating the normal inside voxels, using either triangular mesh or discrete surface. As seen in the results, our method proves robust even for a cheap acquisition system with usual camera and turntable.

In the future, we aim at combining the reconstructed information with image-based techniques such as lighfields/lumigraphs [10, 6] for integrating (real) objects into virtual environments with realistic relighting and global illumination. This method has already been used for estimating light sources positions from images [14].

9. ACKNOWLEDGMENTS

We would like to pay a tribute to Alain Fournier (Imager Lab, University of British Columbia, Vancouver) with whom the overall framework around this paper has been initiated in 1999. The quad has been designed by Stewart Cowley who kindly accepted to let us use this model for our work.

10. REFERENCES

[1] P. Callet. Rendering of binary alloys. In *ICCVG 2004*, sep 2004.

[2] Q. Chen and G. Medioni. A volumetric stereo matching method: Application to image-based

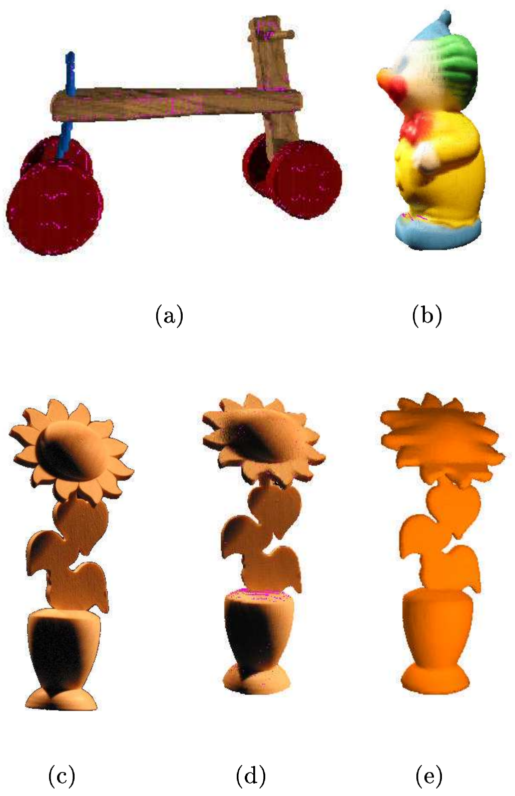


Figure 16: Triangle color corresponds to the average radiance for each voxel; a. for a virtual quad; b. for a real clown (toy); c. Actual photograph; d. for a real wood-flower; e. wood-flower with modified lighting.

modeling. In *CVPR*, pages 29–34. IEEE Computer Society, 1999.

[3] C H Chien and J K Aggarwal. Volume/surface octrees for the representation of three-dimensional objects. *Comput. Vision Graph. Image Process.*, 36(1):100–113, 1986.

[4] C. Hernández Esteban and F. Schmitt. Silhouette and stereo fusion for 3d object modeling. In *3DIM 2003*, pages 46–53, 2003.

[5] Olivier Faugeras and Renaud Keriven. Complete dense stereovision using level set methods. *Lecture Notes in Computer Science*, 1406:379+, 1998.

[6] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *ACM Computer Graphics*, 30(Annual Conference Series):43–54, August 1996.

[7] Jean-Marc Hasenfratz, Marc Lapierre, Jean-Dominique Gascuel, and Edmond Boyer. Real-time capture, reconstruction and insertion into virtual world of human actors. In *Vision, Video and Graphics*, pages 49–56. Eurographics, Elsevier, 2003.

[8] D. R. Hougen and N. Ahuja. Adaptive polynomial modelling of the reflectance map for shape estimation from stereo and shading. In *CVPR*, pages 991–994, 1994.

[9] Kiriakos N. Kutulakos and Steven M. Seitz. A theory of shape by space carving. Technical Report TR692, , 1998.

[10] Marc Levoy and Pat Hanrahan. Lightfield rendering. *Computer Graphics*, 30(Annual Conference Series):31–42, August 1996.

[11] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In Kurt Akeley, editor, *Siggraph 2000, Computer Graphics Proceedings*, pages 131–144. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, 2000.

[12] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(Annual Conference Series):163–169, July 1987.

[13] W. N. 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.

[14] B. Mercier and D. Meneveaux. Joint estimation of multiple light sources and reflectance from images. In *ICCVG 2004*, sep 2004.

[15] S. Seitz and C. Dyer. Photorealistic scene reconstruction by voxel coloring, 1997.

[16] Hemant Singh and Rama Chellappa. An improved shape from shading algorithm. Technical Report CS-TR-3218, Department of Computer Science, University of Maryland Center for Automation Research, College Park, MD, February 1994.

[17] G. Slabaugh, B. Culbertson, T. Malzbender, and R. Schafer. A survey of methods for volumetric scene reconstruction from photographs. In *VG01*, pages 81–100, 2001.

[18] Richard Szeliski. Rapid octree construction from image sequences. In *CVGIP: Image Understanding*, volume 1, pages 23–32, July 1993.