VOLUMETRIC MODEL REFINEMENT BY SHELL CARVING

Y. Kuzu [a], O. Sinram [b]

[a] Yıldız Technical University, Department of Geodesy and Photogrammetry Engineering
34349 Beşiktaş Istanbul, Turkey - kuzu@yildiz.edu.tr
[b] Technical University of Berlin, Department of Photogrammetry and Cartography
Str. des 17 Juni 135, EB 9, D-10623 Berlin, Germany - sinram@fpk.tu-berlin.de

Commission V, WG V/2

KEY WORDS: Calibration, Close Range, Correlation, Reconstruction, Visualization.

ABSTRACT:

In this paper we present a voxel-based object reconstruction technique to compute photo realistic volume models of real world objects from multiple color images. The 3D object acquisition is performed using a CCD video camera, where the images of the object have been captured in a circular camera setup. Before starting the image acquisition, the sensor is calibrated in order to determine the interior and exterior parameters of the camera. Due to the missing of control points on the object, the images undergo a process of relative orientation in a free network using manually measured tie points. The approximate model of the object can be easily acquired by a volume intersection method in a fast and a robust way, which results in the convex hull of the object. The shell of the object, namely the surface voxels are easily obtained from this model. In order to get into the concavities and refine the model we introduced the shell carving method. For the visibility computation, we introduce ray tracing and surface normal vector computation. We use color image matching to search for image correspondences. Furthermore, we make use of the image orientation data for a knowledge based template grabbing. Voxels which are projected into non-corresponding image points are carved away from the shell. And the final set of voxels contains sufficient color and texture information to accurately represent the object.

1. INTRODUCTION

Photogrammetry and computer vision are closely related disciplines which meet on common research areas such as creating 3D object models. While high metric accuracy is more important in photogrammetry, computer vision seeks automation and speed. Therefore it is beneficial to apply techniques developed in both disciplines for faster and more accurate results. In this paper issues from both of these disciplines are referred to compute volumetric models of objects from its color images.

3D object reconstruction from a series of digital images is an important problem both in computer vision and photogrammetry. Optical 3D shape acquisition can be performed either by scanning the object or by taking its images. In this paper the shape of the objects is captured by a CCD camera which is a low-cost alternative to laser scanners.

The shape recovery method described in this paper can be used in many application areas such as in virtual worlds, entertainment, cultural heritage preservation, home shopping and design.

In this paper we use voxels as 3D primitives. Volumetric data was first introduced in the 70's in medical imaging and is now commonly used in scientific visualization, computer vision and graphics. Although voxels consume large amounts of memory they provide more flexible reconstructions of complex objects. Therefore voxel-based reconstruction methods have become an alternative to surface based representations.

In this paper the model is acquired in two steps. First, the approximate model is acquired by a volume intersection (shape from silhouettes) algorithm. The intersection of the silhouette cones from multiple images gives a good estimation of the true model which is called the object's visual hull (Matusik et al, 2000). This algorithm is popular in computer vision due to its fast computation and robustness. However the concavities on an object cannot be recovered with this technique. We refine the model acquired by volume intersection method by our shell carving algorithm. The closest related method to our proposed algorithm is the Voxel coloring algorithm (Seitz and Dyer, 1997). Also see (Kuzu and Sinram, 2002). These algorithms use color consistency to distinguish surface points from the other points in the scene. They use the fact that surface points in a scene project into consistent (similar) colors in the input images. Our algorithm differs from present voxel coloring algorithms in the way that we compute the visibility information. The other basic difference is instead of pooling the pixels of the images a visible voxel projects into, we perform image matching powered by knowledge-based patch distortion to lessen deformation effects.

In the next chapter, the image acquisition setup is described. The image orientation process is also explained. The reconstruction of the model using shape from silhouette technique will be described in chapter 3. Chapter 4 introduces the computation of visibility information. In chapter 5, image matching is given and in chapter 6 the refinement algorithm is explained. Chapter 7 finally summarizes this paper.

2. 3D SHAPE ACQUISITION

We have low-cost system requirements, since we are using a stationary standard CCD video-camera in order to acquire still images. We use a calibration object to compute the interior orientation parameters of the camera. The image acquisition is performed in front of a blue, homogenous background. Multiple views are captured rotating the object resulting in a circular camera setup.

2.1 Camera Calibration and Image Orientation

Prior the image acquisition, the camera should be calibrated. We calibrated the sensor using several images with a calibration object having three perpendicular square planes and 25 control points on each side. Since we use an off-shelf CCD camera, we switch off the auto-focus, so that the focal length remains fixed throughout the whole process.
In a second step, the object is placed inside the calibration frame in order to define some natural control points accurately. We performed a bundle block adjustment with all the images, which delivered the interior camera parameters as well as the coordinates of the control points, which were initially introduced as new points.
In order to compute the object's model, the images should be oriented, the rotations and the position of the cameras should be known. In many cases we cannot mark control points on the objects, therefore natural textures can be used.
The images were adjusted in a bundle block adjustment process. We used enough tie points in all images in the circular camera setup to perform a bundle block adjustment, covering all images. We achieved very accurate results for the image orientations, using the previously calibrated camera. The image projection centers had accuracies of 1-2 *mm*, the rotation were determined with 0.05-0.1 *gon*.

3. APPROXIMATE MODEL

One of the well-known approaches to acquire 3D models of objects is voxel-based visual hull reconstruction, which recovers the shape of the objects from their contours.
A silhouette image is a binary image and easily obtained by image segmentation algorithms. Image pixels indicate if they represent an object point or background point. Since the blue background that we use is sufficiently homogeneous, we can easily define a hue domain, which is considered background. A pixel's position in the IHS-colorspace is examined in order to decide if it represents background or object. We performed the image segmentation using the academic software HsbVis.

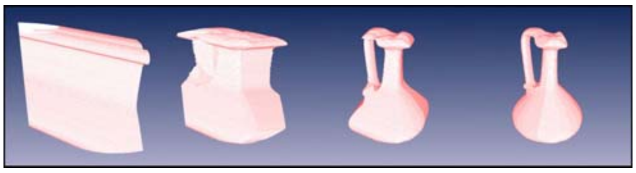


Figure 1: Intersection of silhouette cones

To start with, we define a 3D discrete space which contains only *opaque* voxels with the value "255" representing object points. In order to compute the silhouette cone, we projected all the cube's voxels into every image. If the image coordinate defines a background pixel; the voxel is labeled *transparent* by giving it the value "0" which means the voxel of interest now represents empty regions in the voxel cube. The volume intersection algorithm intersects all silhouette cones from multiple images to achieve the estimate geometry of the object, which is called the object's visual hull. See (Kuzu and Rodehorst, 2000) for more details.
In Figure 1 you see the intersection of the silhouette cones acquired, using 1, 3, 5 and 9 images. As you will notice with increasing number of images this method obtains better approximations to the objects true shape.
As shown in Figure 2, concavities cannot be recovered with this method since the viewing region doesn't completely surround the object. The accuracy of the visual hull depends on the number of the images and the complexity of the object.

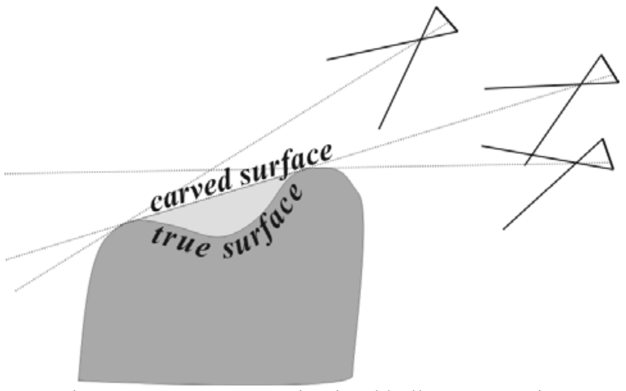


Figure 2: Concave areas in visual hull reconstruction

However, since the result encloses the largest possible volume where the true shape lies and the implementation is straightforward and easy, it is an attractive method for applications where the approximate shape is required. We use the visual hull as the first step of our reconstruction algorithm and we consider the shell carving algorithm as a refinement method to carve away the necessary voxels in the concave areas of the visual hull for a more precise reconstruction.

4. COMPUTATION OF VISIBILITY INFORMATION

It is crucial to find out which voxel is visible in which image. We will use a line tracing algorithm to check each voxel along the line, whether it is background or object voxel. As soon as an opaque voxel is encountered, the initial voxel can be considered occluded. When the line exits the defined voxel cube, it can be stopped, assuming that the voxel is visible. Whether lying on the backside or occluded by another voxel, the algorithm will correctly tell if the voxel is visible or not.

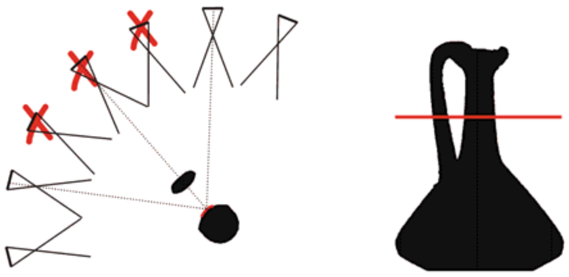


Figure 3: Considering occluded areas

In Figure 3, we show why the knowledge of visibility can be crucial. If we take a closer look at the vase, we will see that the handle is occluding some voxels for some specific images. Hence these images, which theoretically have the best view to the occluded voxels, concerning the viewing angle, cannot see the voxels and therefore should not be considered. From the set of remaining images, the best candidate needs to be chosen.

4.1 Creating the Surface Voxel List

The surface voxel list (SVL) is designed to contain all voxels, which lie on the object's surface. Compared to the total amount of object voxels, the SVL contains only a small percentage, therefore creating the SVL might speed up some operations tremendously. Depending on the neighborhood order fewer or more voxels are considered to be surface voxels and the whole set becomes more or less dense. If noise is likely to be the case but not desired a surface voxel can be defined as follows:

1. The voxel itself is and object voxel
2. At least one neighbor is not an object voxel
3. At least one other neighbor is again an object voxel

Nevertheless, the set of all surface voxels forms the surface voxel list (SVL). The simplest solution constructs a dynamic double-linked list where each item contains the voxels coordinates and each item has a reference to its predecessor and its successor.

4.2 Surface Normal Vector

The surface normal vector gives information of the direction where a voxel is facing (Figure 4). So it can serve as a quality measure for visibility related to a certain viewpoint. When we perform image matching, we would like to decide, whether a certain voxel is a sensible candidate. If it looks away from the camera, we can expect a highly distorted pattern, but when it looks in the direction of the camera it might give a good result.

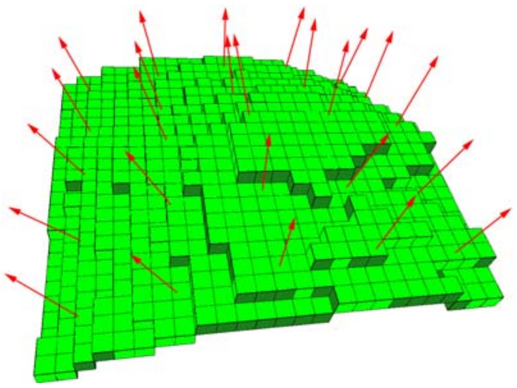


Figure 4: Surface normal vectors on a large voxel surface

Our approach for derivation of surface normal vector calculates a least-squares adjustment on a specific subset of surface voxels for the plane equation $\vec{n}.(\vec{x}-\vec{p})=0$ or:

$$a \cdot x + b \cdot y + c \cdot z + d = 0 \tag{1}$$

We define a certain radius and we take all the surface voxels in this radius and write them into the matrix *A*, with:

$$A = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{2}$$

and the unknown vector:

$$X = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} n_x \\ n_y \\ n_z \end{bmatrix} \tag{3}$$

Since we are only interested in the direction of the tangential surface, the unknown vector contains only the three elements ***a***, ***b*** and ***c***, which directly correspond to the elements of the surface normal vector.

We have to solve the equation $\boldsymbol{A \cdot X = 0}$. We will use the singular value decomposition (SVD) to compute the best solution, with $\boldsymbol{X \neq 0}$ (Hartley, Zisserman, 2000). The SVD will be used to split the design matrix ***A*** into three new matrices ***U***, ***D*** and $\boldsymbol{V^T}$, such that $\boldsymbol{A = U \cdot D \cdot V^T}$, where ***U*** and ***V*** are orthogonal matrices and ***D*** is a diagonal matrix with non-negative entries. The solution of the above equation can be written down as the following steps:

- Perform the SVD for the ***A*** matrix.
- Let *i* be the index of the smallest value in the ***D*** matrix.
- The solution vector ***X*** corresponds to the ***i***.th column in the ***V*** matrix which are the elements of the desired normal vector.

When we calculate the angle between the surface normal and the ray of sight, it can tell us whether the voxel is 'looking in our direction' or not. Hence, if the angle is small, it is facing the image, and if it exceeds 90° it can be considered hidden.

Now, let $\vec{P}$ be the vector of projection, along which the voxel of interest is projected onto the image plane.

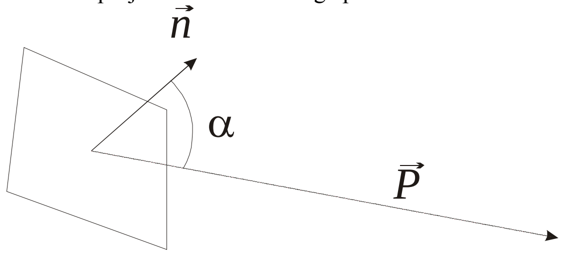
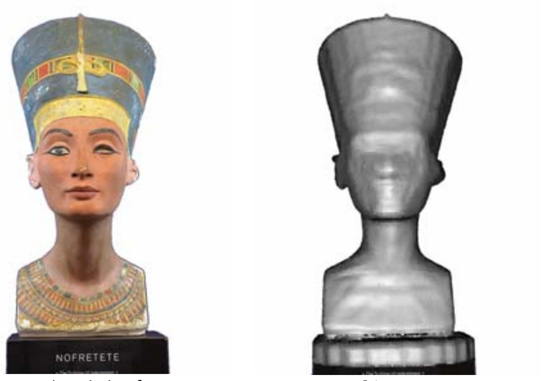


Figure 5: Angle α between the surface normal $\vec{n}$ and the projection vector $\vec{P}$

We can now compute the angle between the surface normal vector $\vec{n}$ and the projection vector $\vec{P}$ (see Figure 5) according to the scalar product (or dot product):

$$\cos\alpha = \frac{\vec{n}.\vec{P}}{|\vec{n}|.|\vec{P}|} \tag{4}$$



a) original *b) gray (light pixels = small angle)*

Figure 6: Visualization of the angle between surface normal and projection vector

Figure 6 is a visual validation of the surface normal vector computation. The left picture is an original digital image, captured by the camera. The right picture shows the angle according to the approximated volume intersection model where the lighter the pixel, the smaller the angle.

4.3 Line Tracing

When computing a voxel's visibility we perform line tracing. The voxel line is defined by the image projection center (X_0, Y_0, Z_0) and either a voxel (V_X, V_Y, V_Z) or a pixel (i, k, $-c$). In our case there is no information outside the defined voxel cube. Since line tracing a time consuming process, we should define a sensible geometric limitation with two preconditions:

- Each voxel must be covered by the line.
- Not too much empty space should be swept by the line.

With the help of the equation below several approaches can be derived.

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix}_{voxel} = R^{-1}\lambda \begin{pmatrix} x_i - x_0 \\ y_i - y_0 \\ -c \end{pmatrix}_{pixel} + \begin{pmatrix} X_0 \\ Y_0 \\ Z_0 \end{pmatrix} \tag{5}$$

For a simple approach, we will choose two values for λ, to define a start- and an end-point for the line. It is based on the assumption, that a λ-factor according to the farthest and the nearest corner of the voxel cube will completely enclose the whole cube. For this definition, all we have to do is to calculate the distance to each of the eight corners, and determine the minimum and maximum of these values. Those two λ-factors will be globally valid for all pixels of one image.

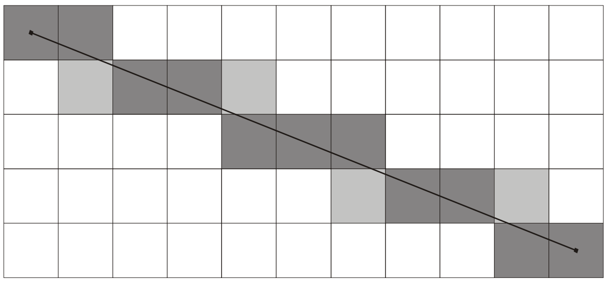


Figure 7: 18-connected (dark pixels) and 6-connected line (light+dark pixels) in 2D

Connectivity is an important issue in line tracing. By choosing the connectivity of the line, we traverse fewer or more voxels. There are different degrees of neighborhoods, which will affect the thickness of the line. In three dimensions we can define 6-, 18-, or 24-connected lines. Figure 7 shows the difference between a 6-connected and an 18-connected line (for simplicity in 2D). An algorithm for a 6-connected line can be found in (Amanatides and Woo, 1987).

5. IMAGE MATCHING

We use color image matching to search for image correspondences since an RGB-triplet contains more information than a single gray value. When we perform image matching, the consideration of red, green and blue channels separately might reveal texture information more clearly.

In general we might classify the possible color image matching approaches into two groups. First, we can throw all color channels into one equation and get one correlation factor as a result. Second, we calculate a correlation factor for each channel separately. Here we will only present the single vector correlation and the difference correlation as our color image matching algorithms.

5.1 Single Vector Correlation

Several tests have shown us that the simplest and at the same time the most reliable solution is the correlation of all the input data in one vector. Assuming the normal case of having three channels of color (RGB, CMY, IHS), we will now have three times as many observations as in gray images:

$$n = width\,.\,height\,.\,3$$

The idea is to put all these observation in one vector, resulting in one single correlation coefficient:

$$r = \frac{\sum_{ch}\sum_{x}\sum_{y}(g_1-\bar{g}_1).(g_2-\bar{g}_2)}{\sqrt{\sum_{ch}\sum_{x}\sum_{y}(g_1-\bar{g}_1)^2.\sum_{ch}\sum_{x}\sum_{y}(g_2-\bar{g}_2)^2}} \tag{6}$$

where: g is the density (gray value) and $\bar{g}$ is the arithmetic mean of densities and ch denotes the color channels.

5.2 Difference Correlation

For this method, we can apply the same approaches, as for the normalized cross correlation. Hence, we can calculate one value by summing up all the differences over the three channels, or we can derive three separate values and calculate a weighted and a non-weighted mean value. We only consider the single vector variant for this approach:

$$r_D = 1 - \frac{\sum_{ch}\sum_{x}\sum_{y} Abs(g_1 - g_2)}{3.\,width.\,height.\,(g_{\max} - g_{\min})} \tag{7}$$

5.3 Introducing Knowledge about the Approximate Shape

In area-based image matching, large base line would cause high patch deformations due to perspective distortions; as a result image matching would fail. However, in our proposed knowledge based patch distortion, this effect is reduced since these deformations are considered and accordingly transformed patches are grabbed for image matching.

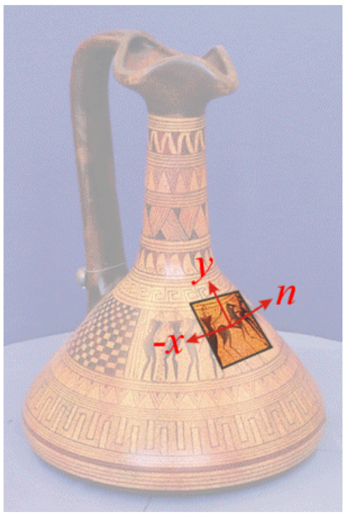


Figure 8: Creating a tangential surface patch

Additionally to the image orientation, we can offer the knowledge of the approximate shape of the object. This is especially the case, after performing volume intersection. The idea is to approximate the image patch on the surface of the object itself. This patch will then be projected into all candidates for the image matching. Hence, we need to construct the tangential surface at the voxel of interest. The computation of surface normal vector has already been described as: $\vec{n}.\left(\vec{x} - \vec{p}\right) = 0$

What we would like to do now is to create a rectangular patch on that surface. It follows that we need a local 2D coordinate system, as seen in Figure 8.

We will now create the vector *x*, with the following two conditions: it is perpendicular to *n* and parallel to the *XY* plane.

It follows that:

$$\vec{x} \| XY \Rightarrow \vec{x} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \qquad (8)$$

and:

$$\vec{n}.\vec{x} = 0 \Rightarrow x_n.x_x + y_n.y_x + z_n.z_x = x_n.x_x + y_n.y_x = 0 \qquad (9)$$

Choosing $x_x = n_y$ and $y_x = -n_x$ delivers the desired vector. The construction of the vector *y* can be derived by taking the cross product of *n* and *x*. Figure 9 shows the improvement to the patches when considering the object shape. We can clearly see a difference between the original slave patches while the knowledge based patches show less variety.

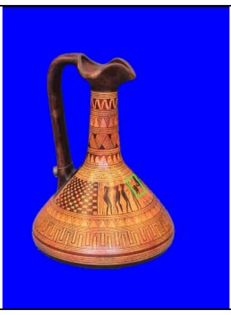
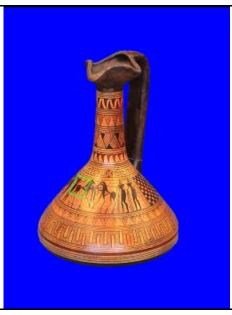
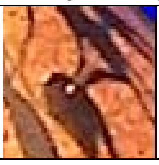
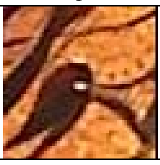
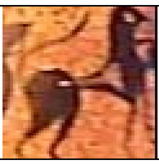
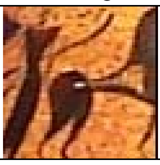
	
Original images with marked surface patch	
	
Grabbed slave patches without knowledge	
	
Grabbed slave patches with considered distortion	

Figure 9: The improvement of taking the object shape into account

6. SHELL CARVING

As the name suggests, in this approach we want to carve away the false voxels on the outer shell of the visual hull. First we create the SVL. All these surface voxels should be processed one by one. For each voxel, we will check if it is part of the true surface. The algorithm will stop, if in a shell no voxels are considered false or if a maximum carving depth has been reached.

Very briefly, the algorithm can be separated into the following steps:

- Create surface voxel list.
- For each voxel:
  - Select three, at least two images with the best view.
  - Perform image matching in the selected images.
  - Carve the voxel, if considered different; Set a fixed-flag, if considered equal.

Now, we want to check each voxel, if it is part of the true surface, hence if it is projected into corresponding image templates. Therefore, we need to find at least two images, in which this voxel is visible. From the set of images, we can first exclude those, where the voxel is occluded (see Figure 3). From the remaining set of images, we will have to choose two or three images, where we can assume the best visibility. This assumption is based on the surface normal vector.

If we have two images, we can perform an image matching to see whether the voxel projects into corresponding image points or not. When three images are available, we can crosscheck the similarity with another pair of templates to improve reliability. At this point, the knowledge based patch distortion can be applied for a more reliable template matching.

Once a voxel has been found true, a flag is set, stating that this voxel is true and should not be considered again. Voxels, which are projected to different image patches, will be carved. So shell by shell, we will normally have an increasing number of fixed voxels, and a decreasing number of carved voxels, until this number drops below a certain threshold or until a maximum carving depth has been reached.

The alternate approach is voting-based carving which evaluates every sensible combination of image matching before deciding to carve the voxel or not. For this purpose, a second cube is introduced, which will store the votes of the single decisions. Here, we introduce two counters. The first one keeps track of all comparisons we made. The second counts the number of successful comparisons. The comparison can be made with any of the previously introduced matching approaches, with or without knowledge based patch distortion. However, the two counter numbers tell us now the percentage of successful comparisons. Applying a threshold to this percentage will either carve or leave the voxel.

Nevertheless, we are storing the actual voting value inside a new cube. For visualization purposes, this value is scaled to fit the value range of 0…255, hence percentage*2.55 is actually stored into the cube. When we now visualize the cube, we can clearly see how the single surface voxels were considered. The lighter the value, the more certainly it is a surface voxel, and the darker, the safer it is to carve. Figure 10 illustrates the results of the voting.

7. SUMMARY

In this paper we presented shell carving as a refinement algorithm to the approximate model acquired by volume intersection method. An experimental image acquisition setup was explained on which the introduced algorithms were tested. Visibility information is recovered using line tracing. As an extension to line tracing, we introduced surface normal vector derived from a regional section of surface voxels. Two color image matching algorithms are introduced to search for image correspondences. Since RGB-triplets contain more information than a single gray value, these algorithms turned out to deliver more accurate results. We made use of the image orientation data for a knowledge based template grabbing. This takes perspective distortion into consideration and makes cross correlation insensitive to rotated images. The image matching was significantly improved by the knowledge-based patch distortion. The approximate model from volume intersection is improved successfully by combining all these tools and we presented some results.

REFERENCES

Amanatides, J., Woo A., 1987. A Fast Voxel Traversal Algorithm for Ray Tracing. Proc. Eurographics '87, pp 1-10.

Hartley R., Zisserman A., 2000. Multiple View Geometry in Computer Vision. Cambridge University Press.

Kuzu, Y. Rodehorst, V., 2001. Volumetric Modelling using Shape from Silhouette. Fourth Turkish-German Joint Geodetic Days., pp. 469-476.

Kuzu, Y. Sinram, O., 2002. Photorealistic Object Reconstruction using Voxel Coloring and Adjusted Image Orientations. ACSM/ASPRS Annual Conference, Washington DC, Proceedings CD-ROM, Proceed\00437.pdf.

Matusik, W. Buehler, C. Raskar, R. Gortler, S. J. and McMillan, L., 2000. Image-Based Visual Hulls. SIGGRAPH 2000 , Computer Graphics Proceedings, Annual Conference Series, pp. 369-374.

Seitz, M. Dyer, R., 1997. Photorealistic Scene Reconstruction by Voxel Coloring. Proceeding of Computer Vision and Pattern Recognition Conference, pp. 1067-1073.

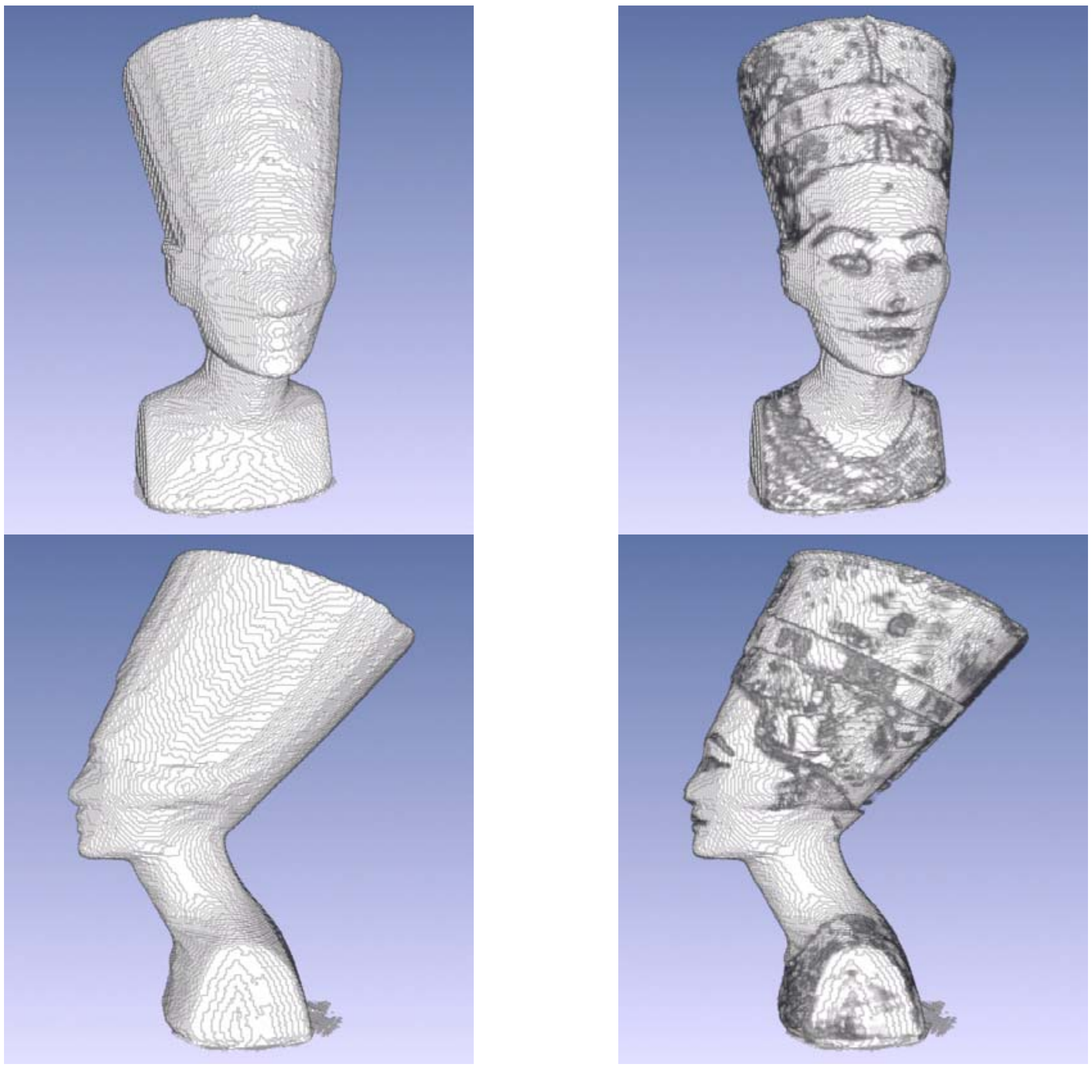


Figure 10: Different viewpoints of the Nefertiti cube. The result of shape from silhouette (left), refinement by voting based carving (right)