International Journal of Image and Graphics
Vol. 3, No. 2 (2003) 1–29

AN IMAGE-BASED INEXPENSIVE 3D SCANNER

ULAŞ YILMAZ*, ADEM YAŞAR MÜLAYİM† and VOLKAN ATALAY‡

Department of Computer Engineering,
Middle East Technical University,
Ankara, TR-06531, Turkey
**ulas@ceng.metu.edu.tr*
†*adem@ceng.metu.edu.tr*
‡*volkan@ceng.metu.edu.tr*

Received 22 February 2002
Revised 15 August 2002
Accepted 26 September 2002

An image-based model reconstruction system is described in this paper where real images of a rigid object acquired under a simple but controlled environment are used to recover its three dimensional geometry and its surface texture. Based on a multi-image calibration method, an algorithm to extract the rotation axis of a turn-table has been developed. Furthermore, this algorithm can be extended to estimate robustly the initial bounding volume of the object to be modeled. The coarse volume obtained is then carved using a stereo correction method which removes the disadvantages of silhouette-based reconstruction by photoconsistency. The concept of surface particles is adapted in order to extract a texture map for the model. Some existing metrics are used to measure the quality of the reconstructed models.

Keywords: Camera calibration; shape from silhouettes; shape from photoconsistency; surface texture extraction.

1. Introduction

Realistic looking three dimensional (3D) models are important components of virtual reality environments. They are used in many multimedia applications such as 3D virtual simulators, 3D educational software, 3D object databases and encyclopedias, 3D shopping over the Internet and 3D teleconferencing. The virtual environments has enabled the user to interact with the models. The user can therefore

- rotate, scale, or even deform the models;
- observe the model under different lighting conditions;
- change the appearance (color, material, etc.) of the model;
- observe the interaction of the model with the other models in the environment, etc.

The most common way of creating 3D models for virtual reality environments is manual design. This approach is very suitable for the creation of models of non-existing objects. However, it is cost expensive and time consuming. Furthermore, the accuracy of the designed model for a real object may not be satisfying. In fact, the number of vertices of a simple model varies between 10000 and 100000, which is relatively high for a manual design. In a second approach, the geometry of real objects can be acquired using a system that captures 3D data directly. Such systems are constructed using expensive equipments such as laser range scanners, structured light, touch-based 3D scanners, or 3D digitizers. In most of these active scanning systems, the texture of the model is not captured while the geometry of the object is acquired precisely as a set of points in the 3D space. This set can then be converted to polygonal model representations for rendering.[1] In a third approach, the model of a real object can be reconstructed from its two dimensional (2D) images. This technique is known as *image-based modeling*. Using an off-the-shelf camera, considerably realistic looking models with both geometry and texture can be reconstructed.[2–9] However, the reconstruction of a complex rigid object from its 2D images is still a challenging computer vision problem under general imaging conditions. Without *a priori* information about the imaging environment (camera geometry, lighting conditions, object and background surface properties, etc.), it becomes very difficult to deduce the 3D structure of the captured object. For practical purposes, the problem can be simplified by using controlled imaging environments. In such an environment, the camera makes a controlled motion around the object, and the background surface and lighting are selected to reduce the specularities on the acquired images.[3,4,9–17]

The aim of this paper is to investigate the reconstruction of 3D graphical models of real objects in a controlled imaging environment as well as to present the work done in our group for such a reconstruction. Our study towards creating 3D object models fit in the category of image-based modeling. Our image-based modeling system consists of four major steps: image acquisition, camera calibration, geometry reconstruction and texture reconstruction. The overall system diagram is shown in Fig. 1. The acquisition is not performed in a very well controlled environment where the object is placed on a turn-table which rotates approximately 10 degrees at each turn, and the images are captured under daylight. As a result, the reconstructed geometry of the model is not precise and it directly affects the quality of the texture. Our acquisition setup is made up of a rotary table with a fixed camera as shown in Fig. 2.

The rotary table used in our system is custom made and is designed to offer a cheap solution. The rotary table rotates approximately 10 degrees each turn step which has a precision of 1 degree. Our CCD camera is the UNIQ1000 model having a resolution of 1294×1030 pixels. Although our camera is not a cheap one, all off-the-shelf cameras can be plugged into our system easily to produce an inexpensive 3D scanner.

With a fixed camera and a turn-table setup, a basic view plan where the camera turns around the object with respect to the turn-table steps can be handled. The

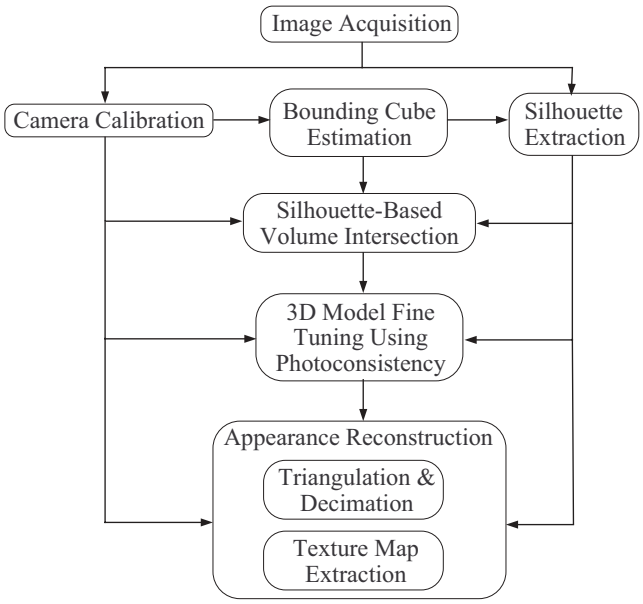


Fig. 1. System diagram.

Fig. 2. System setup.

results obtained by using our setup show that acceptable quality on 3D geometry and texture is attainable by using such a setup up to 20 degrees turn-table rotations. On the other hand the quality of the reconstruction results can be increased by increasing the number of camera viewing positions. More sophisticated camera motion can be obtained by rotating the c-arm around the object where the camera is plugged on and then moving it up and down. This type of setup is a solution for bigger objects and once calibrated it eliminates camera calibration pattern placement at the same scene with the object. Still such equipment can be easily plugged in our system with an increasing hardware cost.

In order to compute the parameters of the camera, we use a multi-image calibration approach.[8] The rotation axis and the distance from the camera center to

this rotation axis remain the same during the turns of the table. Based on this idea, we have developed a vision-based geometrical calibration algorithm for the rotary table.[11] Furthermore, we can easily compute the distance between the rotation axis of the table with respect to the camera center which in fact facilitates the calculation of the bounding box.[7] Once the bounding volume is obtained, a coarse model of the object is computed by carving it out based on its silhouettes. However there are also disadvantages of the silhouette-based reconstruction algorithm. Pertaining to this, we have implemented a multi-image shape from photoconsistency refinement algorithm which removes these disadvantages.[8] The texture for this 3D model is also recovered from the acquired images. The model is considered as a surface composed of particles. The color of each particle is recovered from the images with an algorithm that computes the photoconsistent color for a given particle. Furthermore, if the color cannot be recovered from the images, it will be interpolated. The representation of the model is simple; exporting the model to well known graphical formats such as VRML (Virtual Reality Meta/Modeling Language) is easy. This enables web publishing of the results.

Although many parts of the described system are well-known in various literature and in practice, there are other contributions of this study which can be stated as follows:

(i) developing a vision-based calibration algorithm and its comparison,
(ii) extraction of the rotary axis,
(iii) estimation of the initial bounding box of the acquired object,
(iv) removing the disadvantages of the reconstruction approach based on silhouettes and volume intersection using photoconsistency,
(v) use of particles for a smooth texture recovery.

The rest of the paper follow the logical bottom-up strategy for a 3D reconstruction and it is divided into four main sections. Camera calibration is presented in the next section. Geometry reconstruction is described in Sec. 3. Section 4 gives the detailed description of our texture recovery algorithms. Results obtained in the framework of our study are given in Sec. 5. The paper concludes with Sec. 6 in which we summarize this study and state eventual improvements.

2. Camera Calibration

The purpose of camera calibration is to obtain an estimate of the parameters which determine the transformation of a point in 3D world to a point in a 2D image by a camera. This is necessary in order to be able to associate a 3D world point with a 2D image point. Camera calibration is now a mature field and there exists several methods in this field.[18]

In a setup consisting of a rotary table with a fixed camera, the camera parameters corresponding to the captured images are calculated in three different ways.

In the first approach, the camera calibration pattern is fixed throughout on the table and under the object.[3] In the second approach, the camera calibration pattern is initially mounted on the rotary table. Images for the views to be used in reconstruction are taken, followed by the images of the object. All images are taken from the same positions without the calibration pattern.[4] The third and the last approaches use only but at least three images of the calibraton pattern to determine the rotation axis position with respect to the fixed camera geometry.[11,19] In the first approach, the size of the calibration pattern depends on the size of the object while in the second approach two sets of images are taken for each view and each set of images has to be taken at exactly the same turn angle of the rotary table. These disadvantages have led us to adopt the third approach in which the extracted rotation axis definition with respect to the fixed camera can be used in order to simplify the bounding box estimation. In this calibration approach, it is possible to calculate external camera parameters with respect to the initial position and orientation for any rotation around the rotation axis. In order to realize this third approach, Lavest *et al.*'s multi-image camera calibration method,[20] and the vision-based geometrical calibration method have been adopted.[11,19]

2.1. *Multi-image camera calibration*

One major source of calibration error is the inaccuracy in the 3D to 2D mappings of the n known points. These can be caused either by the 3D part (inaccuracy in the fabrication of the geometrical calibration target) or by the 2D part (inaccuracy in the detection of the projected pattern). Lavest *et al.* proposed a multi-image camera calibration approach to simultaneously estimate the precise 3D coordinates of the geometrical calibration target as well as the intrinsic and extrinsic camera parameters represented by an unknown vector $\mathbf{V}$.[20] The corresponding unknown vector is expressed in Eq. (1) as:

$$\mathbf{V} = \begin{pmatrix} u_0, v_0, d_x, d_y, \mathrm{dist}_x, \mathrm{dist}_y, f\,, \\ X^1, Y^1, Z^1, \ldots, X^n, Y^n, Z^n\,, \\ T_x^1, T_y^1, T_z^1, \alpha^1, \beta^1, \gamma^1, \ldots, T_x^m, T_y^m, T_z^m, \alpha^m, \beta^m, \gamma^m \end{pmatrix}. \tag{1}$$

In Eq. (1), u_0 and v_0 represent the pixel coordinates of the principal point of the image plane, d_x and d_y are the scale factors of the pixel system in x and y directions, respectively, dist_x and dist_y are the lens distortion components, f indicates the focal length, n represents the total number of target points, m is the total number of images taken with different poses of the target, (X^i, Y^i, Z^i) represents the 3D positions of the target points in the calibration pattern, (T_x^i, T_y^i, T_z^i) and $(\alpha^i, \beta^i, \gamma^i)$ represent the translation and rotation from the ith world coordinate frame to the fixed camera coordinate frame. Further details of the method can be found in Ref. 20.

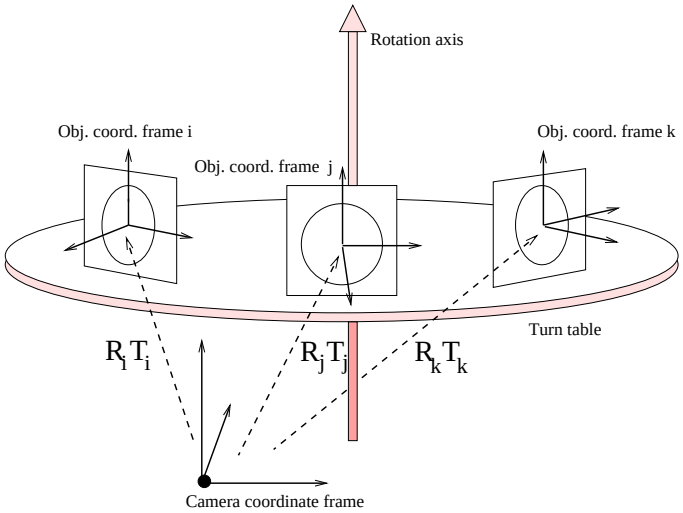


Fig. 3. Relations obtained between the camera coordinate frame and the calibration pattern coordinate frames as a result of the applied camera calibration.

Multi-image calibration process is performed in two phases. In the first phase, only internal parameters are computed. In the second phase, an image sequence is acquired where the geometrical target is fixed on the turn-table whose different poses correspond to different rotation angles of the turn-table. Calibration provides a set of rotation matrices R_i and translation vectors T_i. For the ith position, the rigid transformation $Q_i = [R_i\, T_i]$ relates the geometrical calibration target coordinate frame i to the fixed camera coordinate frame. Figure 3 illustrates the situation with three different calibration target coordinate frame i, j, k corresponding to different rotation angles of the turn-table.

2.2. *Extraction of turn-table rotation axis*

Rotation axis of the turn-table with respect to the fixed camera coordinate frame can be defined by a unit vector u in the direction of the rotation axis and by the translation vector T' perpendicular to u corresponding to the projection of the optical center of the camera onto the rotation axis as illustrated in Fig. 4.

When the rigid transformations Q_i that relates each calibration pattern coordinate frame i to the camera coordinate frame is known, we can then define the transformation Q_{ij} which relates the rigid transformation from target frame i to target frame j with respect to the fixed camera coordinate frame as follows:

$$\begin{aligned} Q_{ij} &= [R_{ij}\; T_{ij}] \\ &= [R_j\; T_j][R_i\; T_i]^{-1} \\ &= [R_j R_i^t \; - R_j R_i^t T_i + T_j]\,. \end{aligned} \tag{2}$$

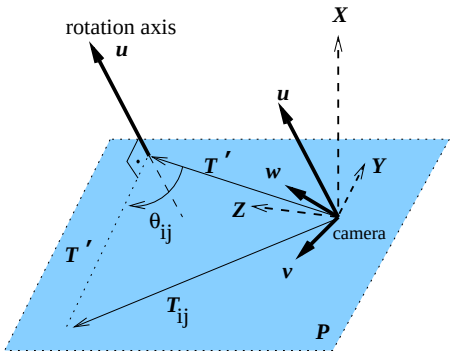


Fig. 4. Rotation axis with respect to the camera geometry.

The corresponding rotation around the axis of the turn-table can be represented by the quaternion q_{ij}:

$$q_{ij} = (\cos(\theta_{ij}/2), \sin(\theta_{ij}/2)u)\,, \tag{3}$$

where θ_{ij} is the angle of the rotation. The unit vector u remains theoretically the same for every R_{ij}. Since the quaternion q_{ij} can be directly calculated from the rotation matrix R_{ij}, the unit vector u can also be easily determined. In order to finalize the determination of the rotation axis of the turn-table the translation vector T' has now to be calculated. For the sake of simplicity and compactness, we consider with the camera coordinate frame (x, y, z) another orthonormal coordinate frame (u, v, w) as shown in Fig. 4, where u is the unit vector associated with the rotation axis and v and w are any two other selected vectors satisfying the orthonormality property of the new coordinate frame with u. T' is the vector pointing to the rotation axis such that T' is perpendicular to u.

As illustrated in Fig. 4, the vectors T_{ij}, T', v, and w are all on the same plane having u as the normal vector. In theory $T_{ij} \cdot u = 0$. However, before proceeding further, the application of a numerical correction on T_{ij} can be useful, since in practice the calculated T_{ij} from Q_{ij} may not be perfectly perpendicular to u:

$$T_{ij} \leftarrow T_{ij} - (T_{ij}.u)u\,. \tag{4}$$

In the newly defined coordinate frame (u, v, w), T' and T_{ij} can be represented as:

$$T_{ij} = V \begin{pmatrix} 0 \\ v \\ w \end{pmatrix}\,, \quad T' = V \begin{pmatrix} 0 \\ v' \\ w' \end{pmatrix}\,, \tag{5}$$

where v, w, v' and w' are the corresponding coordinate values to be computed on the plane defined by v and w and V is the rotation matrix defining the rotation from the (u, v, w) frame to the (x, y, z) frame.

For any point $P = (x, y, z)$ which is defined in the camera coordinate frame and situated on the rotation axis we can write:

$$R_{ij}P + T_{ij} = P\,. \tag{6}$$

Since P is invariant in any transformation Q_{ij}. Equation (6) can also be expressed by:

$$(I - R_{ij})P = T_{ij}\,. \tag{7}$$

Using Eqs. (5) and (7) for $P = T'$, we can write:

$$(I - V^t R_{ij} V) \begin{pmatrix} 0 \\ v' \\ w' \end{pmatrix} = \begin{pmatrix} 0 \\ v \\ w \end{pmatrix}\,, \tag{8}$$

where

$$V^t R_{ij} V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_{ij} & -\sin\theta_{ij} \\ 0 & \sin\theta_{ij} & \cos\theta_{ij} \end{pmatrix}\,. \tag{9}$$

Since it corresponds to the rotation around the axis u by θ_{ij} degree as expressed in the (u, v, w) frame, from Eqs. (8) and (9) the relation between (v, w) and (v', w') can be defined as:

$$\begin{pmatrix} v' \\ w' \end{pmatrix} = H \begin{pmatrix} v \\ w \end{pmatrix}\,, \tag{10}$$

where,

$$H = \begin{pmatrix} 1/2 & \dfrac{1 + \cos\theta_{ij}}{2\sin\theta_{ij}} \\ -\dfrac{1 + \cos\theta_{ij}}{2\sin\theta_{ij}} & 1/2 \end{pmatrix}\,. \tag{11}$$

Since $v = T_{ij} \cdot v$ and $w = T_{ij} \cdot w$, using Eqs. (5) and (10) we obtain finally:

$$T' = (vw)H \begin{pmatrix} T_{ij} \cdot v \\ T_{ij} \cdot w \end{pmatrix} \tag{12}$$

The simultaneous application of the multi-image calibration and the rotation axis extraction provides very accurate parameter values. During the experiments, the obtained precision is about 0.2% for the focal length, the rotation angle θ_{ij}, and the direction u of the rotation axis.[11,19]

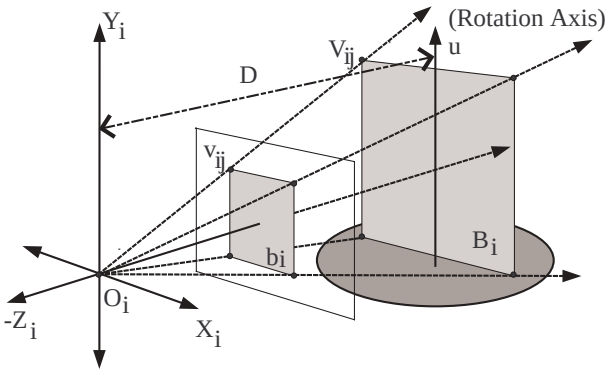


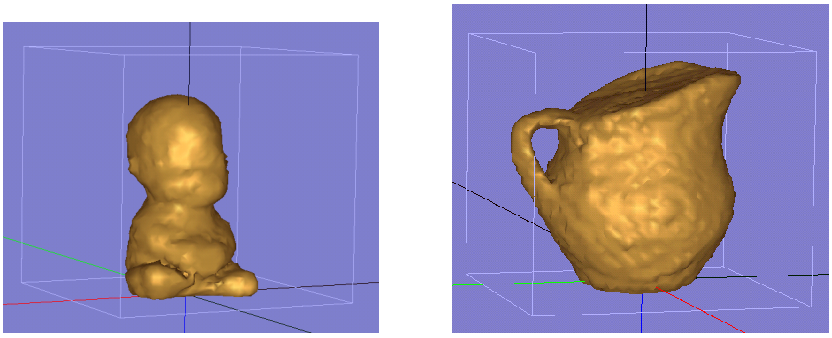
Fig. 5. When each vertex on the bounding rectangle is back projected into the space on a ray passing from the camera optic center and the vertex itself a set of 3D planes is obtained.

3. Geometry Reconstruction

3.1. *Initial bounding box estimation*

Every point of the 3D space occupied by the object volume has projections on the silhouettes of the images. This principle can be used to calculate the coarse object volume which is in fact a convex hull of the object, only if the camera parameters and the object silhouettes on the images are known. The first step is to estimate a bounding box of the object in the space. A bounding box of an object is defined to be the box having the minimum volume and at the same time including the object. This initial volume can be calculated easily if the rotation axis of the rotary table is defined with respect to the camera geometry by the calibration process as described in Sec. 2.[7]

A closed bounding volume can be obtained if the silhouette bounding rectangles from all of the imaging views around an object are backprojected, and then the resultant open volumes are intersected. Assume that the controlled acquisition case with a rotary table and a fixed camera is considered. Then, the position and direction of the table's rotation axis with respect to the fixed camera can be calculated via camera calibration. By using the rotation axis definition and the related camera geometry a 3D counterpart for each silhouette bounding rectangle, which is a plane, can be formed. Each vertex on the bounding rectangle is back projected into the space on a ray passing from the camera optic center and the vertex itself as seen in Fig. 5. 3D counterpart $V_{ij} = (X, Y, Z)$ of vertex j, and $v_{ij} = (x, y)$ of the bounding rectangle on image i, can be calculated by using the parametric equation of the line representing the ray. In this calculation, taking the Z value as the distance between the camera origin and the rotation axis will result in the corresponding X and Y values. If this process is repeated for all of the bounding rectangles, we will obtain a set of 3D planes which are representatives of the 2D rectangles on the images. Like 2D silhouette bounding rectangles which constrain the object in



(a) (b)

Fig. 6. Bounding box for (a) a toy (b) a cup.

2D space, these planes constrain the object in space relative to the viewing camera position from which they are formed. Having these planes in hand, bounding box estimation becomes finding the minimum box container for all of these planes. An example for this configuration is shown in Fig. 6.

The initial bounding box found by the method described here is a minimal one. Although it seems intuitively that the size of the initial bounding box is not important, it does affect the final resolution of the object. Therefore, we achieve a higher model resolution. For the same object, with a greater initial bounding volume, in order to obtain the same detail level on the constructed model, a higher space subdivision parameter should be used as compared to a smaller bounding box.

3.2. *Voxelization and surface triangulation*

Once the bounding box of the object to be modeled is obtained, the next step is to discretize by dividing it into small cubes or voxels. If each side of the cube is divided into n voxels, we will get a voxel space V containing n^3 voxels. The reconstruction algorithm then projects each cube in the voxel space V onto the images by using the related camera parameters. If the projected cube region on any selected image is totally not contained by the silhouette region, it will be removed from the voxel space; otherwise, it is kept in the object voxel space. Algorithm 1 basically gives the idea.[10]

For the sake of simplicity in texture mapping and visualization of the reconstructed object in 3D, the volumetric voxel representation can be transformed to a representation by triangular patches. A simple and practical technique, called the Marching Cubes Algorithm is generally used to perform such a conversion.[21,22] Marching cubes algorithm uses the isolevel information of the vertices of the voxel in order to interpolate an isosurface passing through the voxel. The source of isolevel information in the silhouette-based approach is the sequence of silhouette regions

Algorithm 1 Algoritm to compute the voxels of the voxel space which are not out of the object.

```
reset V_object to voxel space
ForAll voxels in V_object
    ForAll images in the sequence
        project voxel on the image
        If voxel is out of object silhouette
            remove voxel from V_object
        EndIf
    EndFor
EndFor
```

contained in the image sequence. Isolevel values for the vertices of a voxel can be calculated as 0 or 1 when the projection of the vertex on the images is out of the silhouette region or in the silhouette region, respectively. However, this type of approach yields a jagged structured surface. A finer surface can be obtained if the isolevel value of the vertices represents the distance between the projected point position and the silhouette region boundary on the images.[9] Vertices inside the object volume produce negative while the outer ones produce positive isolevel values. In between two vertices having isolevel values with different signs binary search can be applied in order to find the intersection of the surface and the voxel edge where the isolevel value is exactly equal to zero.

3.3. *3D model fine tuning by shape from photoconsistency*

Silhouette-based 3D reconstruction from an image sequence constructs the approximated visual hull of the object, though volume intersection by using images of an object has some important features like:

(i) Approximated (inferred) visual hull of an object is the maximal shape that gives the same silhouette as the actual object for all used camera views in the volume intersection process.[23] Therefore it is guaranteed to enclose the object;
(ii) As the number of images used in volume intersection is increased, better approximation of a visual hull is obtained. That is if the volume of the inferred visual hull decreases monotically as the finite set of used images increases. If an infinite number of different camera viewpoints is used, the exact visual hull of the object can be obtained by volume intersection.

Some excess volume is produced in the resultant visual hull because of the concavities existing on the 3D object and the insufficient camera viewing variety.

In order to improve the 3D reconstruction results of the silhouette-based approach, additional photometric information existing in the acquired image sequence can be used. In the practical turn-table and a fixed camera acquisition environment, if the turn angle between each acquisition step is smaller than 180 degrees, the consecutive images from the sequence would contain the same

object regions as what the object topology permits. The area of these common observation regions for the consecutive camera positions increases as the turn step angle decreases. In the approximated visual hull of the object obtained from a finite set of object images, each surface voxel has a set of images from which the voxel can be seen. If it is assumed that surface voxels of the approximated visual hull represents real object surface, and they are placed at the correct depth positions, stereo theory says that their projections on the images from which they can be seen must be corresponding regions. In other words voxels in the approximated visual hull, having correct depth placement, have photoconsistent projections on the images from which voxels are seen without occlusion.

Using color consistency in volumetric voxel spaces is first introduced by Seitz and Dyer,[24] and the idea, theory and algorithm of using photoconsistency and carving have recently been improved by several researchers.[10,24–27] The algorithm using photoconsistency is generally known as "Voxel coloring". Starting from an initial set of opaque voxels, the voxel coloring algorithm carves or makes transparent the voxels that are not photoconsistent as it iterates on the opaque voxels. The algorithm stops when all the opaque voxels are color consistent.

In order to test the photoconsistency of an individual voxel, the image set from which the voxel is visible must be determined correctly and efficiently. Visibility checking is the elusive part of the algorithms based on photoconsistency since opaque voxels occlude each other in a complex and constantly changing pattern as carving progresses. To simplify visibility checks Seitz and Dyer used the *ordinal visibility constraint* on the camera locations bringing single pass visibility test for all opaque voxels.[24] However the ordinal visibility constraint is a significant limitation since it requires all camera locations to be placed at one side of the scene. Solutions for arbitrary camera placement have been proposed by using multiple plane sweeps along the positive and negative directions of each coordinate axes,[26] and by using special data structures named as item buffers and layered depth images.[25]

Instead of using item buffers or layered depth images which require additional space and complex updates as carving progresses, or using multi sweeps along the coordinate axes, this study proposes traversing a ray from the camera center through a surface voxel is proposed to check the visibility. In fact, a voxel v is invisible from a view i if there exists other opaque voxels in between v and the camera center i, otherwise voxel v is visible. Since it is possible to find the maximum photoconsistent voxels on the ray, it also helps to eliminate threshold checks while carving the voxel from the initial opaque voxel set. Algorithm 2 which is mostly inspired from Matsumoto *et al.* outlines our approach.[10]

In the algorithm, each voxel in the object voxel space V_{object}, starts with a high photoconsistency vote value; that is each voxel on the model generated by the silhouette-based reconstruction is assumed to be on the real object surface. Each view i is then processed in the following manner. For each view i, rays from the camera center c_i through the voxels seen from that view i are traversed voxel by voxel. Each voxel on the ray is projected onto the images $i-1$, i, $i+1$ and

Algorithm 2 Algoritm to compute the photoconsistent voxels.

reset all photoconsistency values of the voxels in V_{object} to max photoconsistency value
ForAll image i in the image sequence
  ForAll visible voxels in image i
    produce a ray from camera optic center
    find max photoconsistent voxel on the ray
    ForAll voxels between the max photoconsistent voxel and camera optic center
      reduce voxel photoconsistency votes
    EndFor
  EndFor
EndFor
ForAll voxel v in voxel space V_{object}
  If the photoconsisency value of v is less than a predetermined threshold
    remove v from V_{object}
  EndIf
EndFor

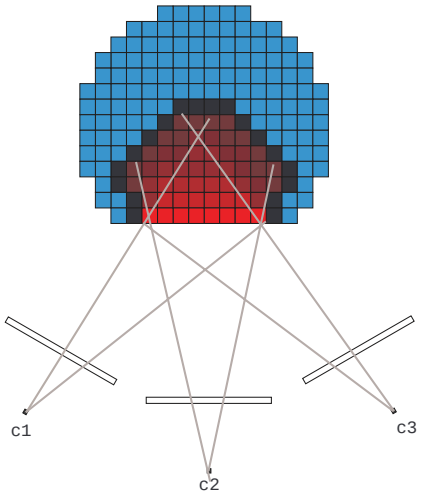


Fig. 7. Darkest colored voxels get the highest voting and the color of voxel shows its vote.

the voxel's photoconsistency value is calculated using texture similarity measures among the projection regions on the images $i-1$, i, $i+1$. Then the voxel with maximum photoconsistency value is found and all the voxels existing between this voxel and the camera center c_i will loose votes in an increasing order as they become closer to c_i. This process is repeated for all the rays which can be generated from the view i. When the process is performed for all the views, excess voxels caused by the silhouette-based reconstruction lose most of their initial photoconsistency votes. Then by thresholding, this excess volume is carved. Figure 7 explains the idea better. In this figure, the darkest colored voxels get the highest voting; i.e. they have

the maximum texture similarity according to the algorithm. The color of the voxel shows its vote.

Silhouette-based 3D reconstruction for the synthetic sphere image sequence shown in Fig. 8(a) generates the object voxel space given in Fig. 8(b). In order to carve these extra voxels, Algorithm 2 is applied. Figure 8(c) illustrates the new object voxel space after the elimination of extra voxels. The carving result becomes clearer after the marching cubes algorithm is applied to the object voxel space. Figure 8(d) gives the triangulated models for the carved object.

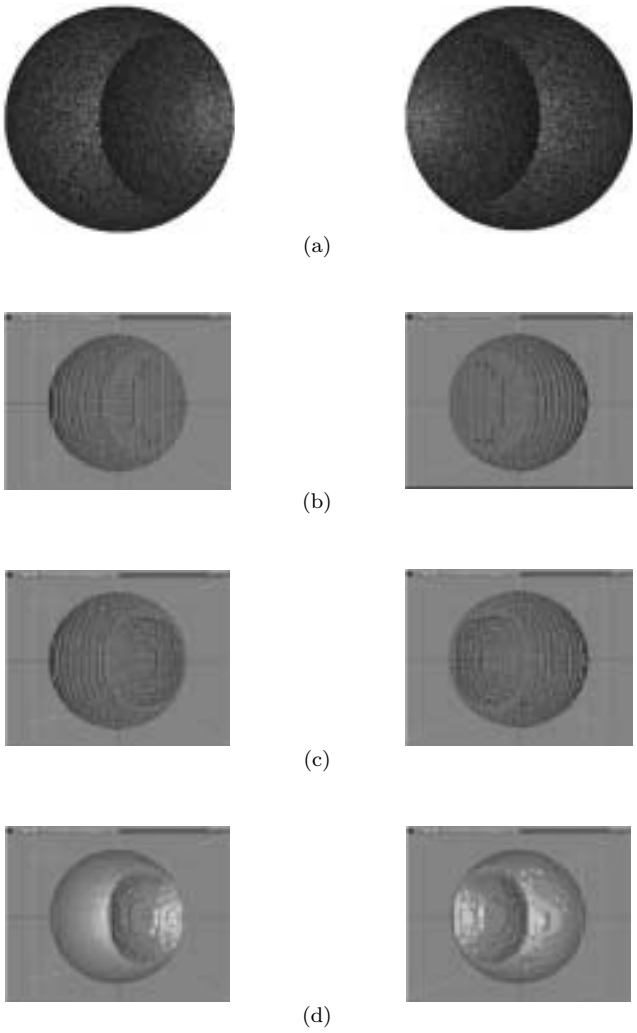


Fig. 8. (a) Synthetic image sequence, (b) object voxel space before carving, (c) object voxel space after carving, (d) triangulated models of the carved object.

4. Texture Reconstruction

There exist several studies for the texture reconstruction of 3D models from real images.[9,28–33] In most of these studies, the model is represented as a triangular wireframe, and each triangle is associated with one of the images for texture extraction. However, this approach causes discontinuities on the triangle boundaries as shown in Fig. 9, since adjacent triangles can be associated with different source images. Furthermore, applying a low pass filter on the boundaries cannot come up with a global solution.

Niem *et al.* constructed the geometry of the object as a wireframe of polygons, and a texture pattern is extracted for each triangle of the model.[3,29] Texture assignment is then reorganized for better and more continuous texture, and a low pass filter is applied on the boundaries. However, this results in blurring the triangle boundaries of the model. Furthermore, if a texture pattern cannot be extracted for a triangle from the image sequence, textures from the adjacent triangles are used. This may cause unexpected results if the texture of the object is not just a repetition of a texture pattern on the surface. Genç and Atalay performed the texture reconstruction at the time of rendering in their dynamic texture mapping approach.[30] In their method, the camera properties, lighting conditions, the geometry of the model and surface properties are assumed to be known. The model is represented as a wireframe of triangles. During rendering, the texture of each triangle is extracted dynamically from the image sequence. The best image is selected as the source of the texture. The selection criterion is defined as a function of the projection area of the triangle on the images and the difference of the image capture angle

Fig. 9. Discontinuities on the triangle boundaries when the texture patches of adjacent triangles are extracted different source images.

and the view angle. Lensch *et al.* represented their model as a triangular wireframe, and each triangle is associated with an image for texture extraction.[31] Similar texture discontinuity problems are also observed in this method. To come up with a solution, Lensch *et al.* blend the textures of triangles which are found to be on the boundaries. However, since blending removes spatially higher frequencies, blurring occurs on the boundary. Furthermore, the discontinuity on the boundary may not necessarily be one triangle wide. Neugebauer and Klein described a texture synthesis method to combine a set of photographic views to a set of textures. In their approach, the color entities of the texture patterns are collected from the images in a weighted manner.[33] The first weight corresponds to a metric for the sampling density of the texture pattern on the image. The second weight is a measure of the visibility and smooth transition, and its definition is based on the 2D distance to the nearest point on the surface that is undefined in the image. The third weight measures how close the color distribution is to the expected distribution which is a uniform distribution with single mode. However, in the case of specular reflection, the distribution has more than one local maximum, which makes the third weight unstable. Finally, the average of all color candidates is taken using these weights. One drawback of this approach is that all color candidates are used in the color recovery: although they are merged in a weighted manner, distortion may occur due to averaging.

Texture mapping is the most popular technique that is used to increase realism in real time virtual reality applications. Although 3D texturing[34] — a very close topic to particles, is also in use, the term texture mapping refers to the conventional 2D texturing.[35] In this paper, 2D texture mapping is used but in order to reduce the drawbacks due to the lack of third dimension information, the concept of surface particles is adapted.[9,36–38] An abstraction is done on the actual representation of the model: the model is considered to be a surface composed of particles with three attributes:

(1) position, $\overrightarrow{P}(x, y, z)$,
(2) normal, $\overrightarrow{N}(x, y, z)$,
(3) color, $\overrightarrow{C}(h, s, v)$.

While reconstructing the texture of the model, instead of associating the triangles to the images, particles are associated with images for texture extraction. This is what makes the proposed method superior to the others: since a triangle is not necessarily textured from a single image, there are no discontinuities on the triangle boundaries due to the fact that they are textured from different images. Each particle on the surface is associated with a pixel on the texture map as shown in Fig. 10. Every particle is assigned a position and a normal vector using a bilinear mapping from the texture space to the model space is given in Eqs. (13) and (14) where s and t are the coordinates of the texture element, $\overrightarrow{v_0}$, $\overrightarrow{v_1}$, $\overrightarrow{v_2}$ are the vertices of the triangle associated with that texture patch, and $\overrightarrow{n_0}$, $\overrightarrow{n_1}$, $\overrightarrow{n_2}$ are the normal

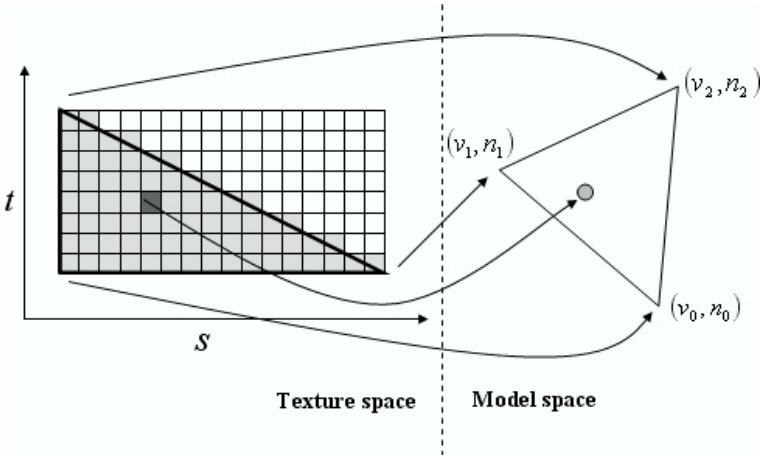


Fig. 10. Association between the texture map and a surface particle.

vectors of the vertices of the triangle.

$$\overrightarrow{P}(x, y, z) = (1 - s - t)\overrightarrow{v_0} + (s)\overrightarrow{v_1} + (t)\overrightarrow{v_2}\,, \tag{13}$$

$$\overrightarrow{N}(x, y, z) = (1 - s - t)\overrightarrow{n_0} + (s)\overrightarrow{n_1} + (t)\overrightarrow{n_2}\,. \tag{14}$$

A particle is not necessarily visible in all of the images in the sequence, it can be occluded or there may be a back face in some of them. Therefore, the extraction takes place in two steps: visibility check and color retrieval. Visibility check is performed using the particle normal and the particle position by simple hidden surface removal and occlusion detection algorithms. If the particle is visible then it is projected on the source image. Generally, this projection is at subpixel level. The neighboring pixels on the image projection points are used in a weighted manner to retrieve the candidate color from the source image, which decreases the possible aliasing effects. The overall color extraction process is illustrated in Fig. 11 where a particle is projected onto the images $S = \{I_0, \ldots, I_{N-1}\}$ in which it is visible, and a set of candidate color values, $C = \{c_0, \ldots, c_{M-1}\}$ are collected.

For each particle, there is a set of source images to extract the color. When the particle is projected onto the images; a set of candidate colors is collected. There are two ways to fuse the color values: merging and selection. Merging is to be used for all of the values in the set. Averaging is an example of this case. However, this method ends up with a distortion in the surface texture as shown in Fig. 12. Selection, on the other hand, is to choose one of the candidate colors. The selection should be done in such a way that the selected color is the closest to the actual view-independent color. One of the alternatives is to select the image whose normal vector produces the minimum angle with the particle normal vector. One drawback of this method is its weakness in removing illumination artifacts. Furthermore, if

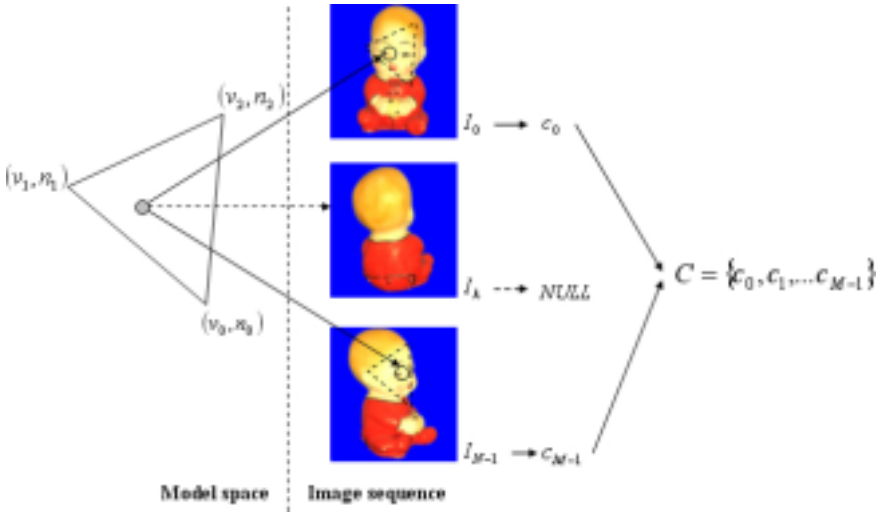


Fig. 11. Color extraction of a particle from the images.

Fig. 12. Example of reconstruction by averaging the candidate color values.

the object has highlights in many of the images, the resultant texture may inherit all of them as shown in Fig. 13.

In a second approach, the candidates are fused in order to produce the most photoconsistent texture. Thus, before assigning a color value to a particle, it should be decided whether the information extracted from the source images is photoconsistent or not. The photoconsistency is defined in Definition 1. The value of Θ is empirical and is set to 60. It is expected that the values of a photoconsistent set concentrates around the view-independent color of the particle.

Definition 1. Let the extracted color, values for a given particle be $C = \{c_0, c_1, \ldots, c_{M-1}\}$ for an image sequence $S = \{I_0, I_1, \ldots, I_{N-1}\}$. The color of the particle for this sequence is photoconsistent if

(1) There exists at least two images in S in which the particle is not occluded.
(2) The particle is not on the background in any of the images in S.
(3) $C_\sigma < \Theta$ where C_σ is the standard deviation of intensity values of the colors in C.

This method is very suitable for removing illumination artifacts as shown in Fig. 14. However if the geometry of the object is not constructed precisely, the photoconsistency criteria will fail for most of the particles, which will cause irregularities on the surface texture.

The above methods can be combined as explained in Algorithm 3: using the first approach, the color of each particle is computed. To eliminate the highlights, particles with highly saturated colors are re-evaluated using the second approach.

Fig. 13. Example of reconstruction by view-dependent selection.

Fig. 14. Example of reconstruction by selecting the median of the color values in a photoconsistent set.

Algorithm 3 Recovering the color of a particle from the images.

ForAll images in the sequence
  calculate the angle between the normal vectors of the particle and the image
EndFor
extract the color from the image that produces the minimum angle
set the color of the particle to the extracted color
IF the extracted color is highly saturated
  reset color set C of particle to empty set
  ForAll images in the sequence
    If particle is visible in the image
      project particle on the image
      insert the extracted color in C
    EndIf
  EndFor
  If C is photoconsistent
    set the color of the particle to the median of C
  Else
    set the color of the particle to $c_{inconsistent}$
  EndIf
EndIf

Fig. 15. Regions whose color cannot be recovered from the images.

Algorithm 4 Interpolating the color of a particle.

ForAll particle P whose color is inconsistent
    get the nearest particle Q whose color is consistent
    set color of particle P to the color of particle Q
EndFor

If a particle is occluded in all of the images or a photoconsistent color cannot be extracted from the sequence, there occurs regions whose texture cannot be recovered on the model as shown in Fig. 15. The colors of the particles in these regions are interpolated using the colors of the adjacent particles as explained in Algorithm 4.

5. Experimental Results and Quality Measurement

In this section, we present the first sample reconstructions and then measure their qualities. The texture of the "box" object is reconstructed with three selection methods: view-dependent selection, photoconsistency check and hybrid suing a sequence of 12 images. All the methods are successful in producing a continuous surface texture. However in the view-dependent selection, highlights on the images are inherited by the model as shown in Fig. 16. Photoconsistency check is quite successful in removing the highlights but produces some irregularities on the model surface as shown in Fig. 17. Results obtained when these two methods are combined are shown in Fig. 18 where highlights are removed and the irregularities are reduced. The corresponding sequence of the original images are given in Fig. 19.

In another experiment, images of three objects, "cup", "toy" and "star" are acquired under daylight with 20 degrees of turn angles between the consecutive

60° 90° 120°

Fig. 16. Sample reconstructions with view-dependency approach.

60° 90° 120°

Fig. 17. Sample reconstructions with photoconsistency approach.

60° 90° 120°

Fig. 18. Sample reconstructions with a hybrid approach.

60° 90° 120°

Fig. 19. Sample images of the "box" object.

Fig. 20. Reconstruction results for a "cup" object, a "toy" object and a "star" object.

images. The texture is again reconstructed with three selection methods. All of the methods are successful in producing a continuous surface texture as shown in Fig. 20. However in this experiment, not all of the acquired images are used in reconstruction. Instead, 12 of the 18 images are used for reconstruction and the remaining 6 are used for quality measurement.

The quality of the reconstructed object model mainly depends on the following parameters: quality of the image acquisition/processing (image resolution, illumination effects, segmentation of the silhouettes, etc.), quality of the calibration, and quality of the reconstructed geometry. In this section, in addition to the visual comparison, the quality of the reconstructed models is measured with a set of methods. These methods are based on the comparison of original images of the object which rendered corresponding images using the reconstructed model. The images that are used for reconstruction (training images) should not be used for quality measurement (test images). If the training images are also used as the test images, the measurements will be biased. Instead, some of the acquired images are not used in reconstruction but are kept as test images to obtain more reliable quantitative analysis. Generally, visual similarity cannot be described by computational similarity. Although there is no universal and accepted method to tell how similar two images are, there still exist in literature the following methods: Hausdorff distance (HD), root-mean-square error (RMSE), direct difference error (DDE), and normalized cross correlation ratio (CO).[39,40]

Hausdorff distance between two images is computed as the maximum distance of the first image to the nearest point in the second image.[39] More formally, directed Hausdorff distance of an image A to another image B can be computed as in Eq. (15), where d is a metric that is defined to calculate the distance from a pixel A_{ij} to the image B. Chess board and city block metrics can be represented as d. Chess board and city block metrics are given in Eqs. (16) and (17), respectively, where M is the image width, N is the image height, and G is the maximum possible intensity value.

$$h(A,B) = \max_{i,j}\{\min_{l,m}\{d(A_{ij}, B_{lm})\}\}\,, \tag{15}$$

$$d_{\text{chess board}} = |i-l|/N + |j-m|/M + |A_{ij} - B_{lm}|/G\,, \tag{16}$$

$$d_{\text{city block}} = \max\{|i-l|/N, |j-m|/M, |A_{ij} - B_{lm}|/G\}\,. \tag{17}$$

Directed Hausdorff distance is not symmetric i.e. $h(A,B) \neq h(B,A)$. In Eq. (18), normalized Hausdorff distance between two images is given as:

$$\text{HD}(A,B) = \max\{h(A,B), h(B,A)\}\,. \tag{18}$$

Normalized cross correlation ratio (NCCR), root mean square error (RMSE) and direct difference error (DDE) are widely used measures in image comparison. Cross correlation is a standard method of estimating the degree to which two images are correlated; root mean square is the mean of the sum of the squares of the difference between the values of pixels in two images; and the direct difference error is an

approximation of the RMSE in which the absolute value of differences is used to approximate the sum of squares. The definitions of each are given in Eqs. (19), (20) and (21), respectively.

$$\mathrm{NCCR}(A,B) = 1 - \frac{\sum_{i,j=0}^{M,N} A_{ij}B_{ij}}{\sqrt{(\sum_{i,j}^{M,N} A_{ij}^2)(\sum_{i,j}^{M,N} B_{ij}^2)}}\,, \tag{19}$$

$$\mathrm{RMSE}(A,B) = \frac{1}{G\sqrt{MN}}\sqrt{\sum_{i,j=0}^{M,N}(A_{ij}-B_{ij})^2}\,, \tag{20}$$

$$\mathrm{DDE}(A,B) = \frac{1}{G\sqrt{MN}}\sum_{i,j=0}^{M,N}|A_{ij}-B_{ij}|\,. \tag{21}$$

The algorithms explained in this paper are implemented by C programming language. Additional libraries are also used for camera calibration, user interface design, graphics rendering and image file operations. While rendering the final model, texture mapping is performed using the routines provided by OpenGL.[41] The images are captured with a 2/3" Color Progressive scan CCD camera (UNIQ1000) at a resolution of 1294×1030. The experiments are performed on a personal computer with 512 MB of RAM, Intel PIII 800Hz CPU and 32 MB frame buffer.

Error analysis is based on the comparison of original images of the object with rendered corresponding images of the reconstructed model. To give an overall idea, the mean and the variance of the differences in terms of five image comparison distances are given in Table 1 for three selection methods for the reconstruction of

Table 1. Results of the error analysis for the "box" object.

	View-dependent (μ, σ^2)	Photoconsistent (μ, σ^2)	Hybrid (μ, σ^2)
$\mathrm{HD}_{\mathrm{chess}}$	(87.19, 4.19)	(85.50, 7.05)	(86.21, 5.14)
$\mathrm{HD}_{\mathrm{city}}$	(77.66, 6.70)	(75.54, 11.23)	(77.25, 6.88)
NCCR	(98.73, 0.14)	(98.83, 0.15)	(98.80, 0.14)
RMSE	(87.08, 3.44)	(87.74, 3.87)	(87.51, 3.63)
DDE	(92.06, 2.88)	(92.62, 3.20)	(92.37, 2.98)

Table 2. Results of the error analysis for the "cup" object.

	View-dependent (μ, σ^2)	Photoconsistent (μ, σ^2)	Hybrid (μ, σ^2)
$\mathrm{HD}_{\mathrm{chess}}$	(78.19, 11.09)	(78.26, 11.09)	(82.47, 29.40)
$\mathrm{HD}_{\mathrm{city}}$	(87.35, 0.62)	(87.46, 0.63)	(90.18, 8.64)
NCCR	(98.38, 0.15)	(98.38, 0.15)	(98.11, 0.22)
RMSE	(91.10, 0.53)	(91.10, 0.53)	(90.72, 0.67)
DDE	(94.48, 0.29)	(94.48, 0.29)	(94.18, 0.38)

the "box". In this particular "box" experiment, the training and test images are the same since the number of images is small. However in order to make a non-biased measurement we have used some of the acquired images in the "cup" experiment as mentioned above. The results of the error analysis of this experiment are given in Table 2. Hausdorff distance, normalized cross correlation ratio, root mean square error and direct difference error are all well-known image comparison techniques in literature. However the results pointed out two facts: quality measures are very high and for a particular object the quality measures with different selection criterion are very close to each other.

6. Conclusions and Future Work

In this paper, we describe an image-based model reconstruction system using real images of rigid and static objects. In fact, it is an end-to-end system for reconstructing 3D object models using a camera and calibrated turn-table. The described system can be used as an inexpensive scanner. The acquisition in this work is not performed in a very well controlled environment: camera makes a controlled motion around the object and the images are acquired in daylight. The improvement contributions described in this study can be stated as follows. A vision-based camera calibration algorithm is developed and it is used in the extraction of the rotation axis. The bounding box of the object is estimated using the extracted rotation axis. To eliminate the drawbacks of the silhouette-based reconstruction, a shape from the photoconsistency algorithm is implemented to correct the disadvantages of the previous steps. The concept of surface particles is adapted in order to extract a texture map for the model. The assessment of the constructed models is performed visually and also by using some metrics to measure the quality of the reconstructed models. Although parts of the system have been described in previous literature, we believe that it is important to put together a complete system. Furthermore, the choice of the method and the way it is implemented is described in detail and could be of interest to the other researchers.

The geometry of the model is represented as a wireframe of triangles. In addition to this actual representation, the model is considered as a surface composed of 3D particles. This abstraction is used to reconstruct a 2D texture map in which the appearance information is embedded. Each particle is associated with an image, and the color of the particle is extracted from that image. This method is very suitable for removing highlights and aliasing effects. Also, in most of the previous studies, there are sharp discontinuities on the surface texture. The discontinuities appear on the boundaries of polygons that form the model. However, in this approach the surface texture is constructed in such a way that the addressed boundary discontinuity problem is out of concern.

In order to measure the quality of the 3D reconstruction results, special 3D test objects could be used. Such objects are designed in computer aided design software and are realized at special cutting machines. They are also painted with

special colors and patterns. The resultant 3D geometry of the object could be compared with its original design using its quantized voxel representation. The special patterns on the object also help in using image comparisons by checking reconstructed pattern features. Using 3D geometry comparisons is a better way as compared to using image comparisons since multiple reconstruction solutions may give acceptable results in image comparison. However to our knowledge, there is no study in the 3D image-based reconstruction literature that uses such a quality measurement. We will also leave this as future work since it is difficult for us to produce such type of objects at the moment.

The final model is stored as a wire-frame composed of triangles, and the appearance is embedded in a 2D texture map. This is one of the most common 3D model representation techniques in computer graphics. The resultant models are converted into VRML (Virtual Reality Meta/Modeling Language) format for Internet publishing.

Acknowledgments

This study is partially supported by TÜBİTAK (Scientific and Technical Research Council of Turkey) under the grant EEEAG-199E024, and Adem Y. Mülayim's study is supported by TÜBİTAK Integrated PhD Program.

References

1. M. Soucy, G. Godin and M. Rioux, "A texture mapping approach for the compression of the colored 3D triangulations," *The Visual Computer* **12**(10), 503–514 (April 1996).
2. P. E. Debevec, C. J. Taylor and J. Malik, "Modeling and rendering architecture from photographs: A hybrid geometry and image-based approach," In *Proc. SIGGRAPH*, 11–20 (August 1996).
3. W. Niem and J. Wingbermũhle, "Automatic reconstruction of 3D objects using a mobile monoscopic camera," In *Proc. International Conference on Recent Advances in 3D Imaging and Modeling*, 173–180 (May 1997).
4. Y. Matsumoto, H. Terasaki, K. Sugimoto and T. Arakawa, "A portable three-dimensional digitizer," In *Proc. International Conference on Recent Advances in 3D Digital Imaging and Modeling*, 197–204 (May 1997).
5. M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool, "Automated reconstruction of 3D scenes from sequences of images," *ISPRS Journal of Photogrammetry and Remote Sensing* **55**(4), 251–267 (2000).
6. P. Poulin, M. Ouimet and M.-C. Frasson, "Interactively modeling with photogrammetry," In *Proc. Eurographics Workshop on Rendering*, 93–104 (June 1998).
7. A. Y. Mülayim, O. Özün, V. Atalay and F. Schmitt, "On the silhouette-based 3D reconstruction and initial bounding cube estimation," In *Proc. Vision Modeling and Visualization*, 11–18 (November 2000).
8. A. Y. Mülayim and V. Atalay, "Multibaseline stereo correction for silhouette-based 3D model reconstruction from multiple images," In *Proc. SPIE, Three-dimensional Image Capture and Applications IV*, 24–25 (January 2001).
9. F. Schmitt and Y. Yemez, "3D color object reconstruction from 2D image sequences," In *Proc. International Conference on Image Processing*, 65–69 (October 1999).

10. Y. Matsumoto, K. Fujimura and T. Kitamura, "Shape-from-silhouette/stereo and its application to 3D digitizer," In *Proc. Discrete Geometry for Computer Imagery Conference, Lecture Notes in Computer Science 1568*, 177–188 (1999).
11. A. Y. Mülayim, Y. Yemez, F. Schmitt and V. Atalay, "Rotation axis extraction of a turn table viewed by a fixed camera," In *Proc. Vision Modeling and Visualization*, 35–42 (November 1999).
12. P. Ramanathan, E. Steinbach and B. Girod, "Silhouette-based multiple-view camera calibration," In *Proc. Vision, Modeling and Visualization*, 3–10 (2000).
13. A. W. Fitzgibbon, G. Cross and A. Zisserman, "Automatic 3D model construction for turn-table sequences," In *Proc. SMILE Workshop on Structure from Multiple Images in Large Scale Environments*, 154–170 (Springer-Verlag, June 1998).
14. M. Potmesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images," *Computer Vision Graphics and Image Processing* **40**(1), 1–29 (October 1987).
15. R. Szelisky, "Rapid octree construction from image sequences," *Computer Vision Graphics and Image Processing* **58**(1), 23–32 (July 1993).
16. J. Y. Zheng, "Acquiring 3D models from a sequence of contours," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(2), 163–178 (February 1994).
17. W. B. Seales and O. D. Faugeras, "Building 3-dimensional object models from image sequences," *Computer Vision and Image Understanding* **61**(3), 308–324 (1995).
18. O. Faugeras, *Three-Dimensional Computer Vision — A Geometric Viewpoint* (Cambridge, Massachusetts, USA, November 1993).
19. A. Y. Mülayim, F. Schmitt and V. Atalay, "Vision-based geometrical calibration of a turn-table for 3D object reconstruction," *Technical Report TR-2000-01*, Middle East Technical University (2000).
20. J. M. Lavest, M. Viala and M. Dhome, "Do we really need an accurate calibration pattern to achieve a reliable camera calibration?," In *Proc. European Conference on Computer Vision*, 158–174 (1998).
21. W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface reconstruction algorithm," *ACM Computer Graphics* **21**(4), 163–169 (1987).
22. C. Montani, R. Scateni and R. Scopigno, "Discretized marching cubes," In *Proc. Visualization*, 281–287 (1994).
23. A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(2), 150–162 (1994).
24. S. Seitz and C. Dyer, "Photorealistic scene reconstruction by voxel coloring," *International Journal of Computer Vision* **35**(2), 151–173 (1999).
25. W. B. Culbertson, T. Malzbender and G. Slabaugh, "Generalized voxel coloring," In *Proc. International Conference on Computer Vision*, 67–74 (1999).
26. K. Kutulakos and S. Seitz, "A theory of shape by space carving," In *Proc. International Conference on Computer Vision*, 307–314 (1999).
27. L. Zhang and S. M. Seitz, "Image-based multiresolution modeling by surface deformation," *Technical Report CMU-RI-TR-00-07*, Robotics Institute Carnegie Mellon University (2000).
28. U. Yılmaz, A. Y. Mülayim and V. Atalay, "Reconstruction of three dimensional models from real images," In *Proc. International Symposium on 3D Data Processing Visualization and Transmission*, 554–557 (June 2002).
29. W. Niem and H. Broszio, "Mapping texture from multiple camera views onto 3D object models for computer animation," In *Proc. International Workshop on Stereoscopic and Three Dimensional Imaging*, 99–105 (September 1995).

30. S. Genç and V. Atalay, "Texture extraction from photographs and rendering with dynamic texture mapping," In *Proc. 10th International Conference on Image Analysis and Processing* (1999).
31. H. P. A. Lensch, W. Heidrich and H.-P. Seidel, "Automated texture registration and stitching for real world models," In *Proc. Pacific Graphics*, 317–337 (October 2000).
32. H. P. A. Lensch, W. Heidrich and H.-P. Seidel, "A silhouette-based algorithm for texture registration and stitching," *Journal of Graphical Models* **63**(4), 245–262 (July 2001).
33. J. P. Neugebauer and K. Klein, "Texturing of 3D models of real world objects from multiple unregistered photographic views," *Computer Graphics Forum* **18**(3), 245–256 (September 1999).
34. J. M. Dischler and D. Ghazanfarpour, "A survey of 3D texturing," *Computers and Graphics* **25**(1), 135–151 (February 2001).
35. F. M. Weinhaus and V. Devarajan, "Texture mapping 3D models of real-world scenes," *ACM Computing Surveys* **29**(4), 325–365 (December 1997).
36. W. T. Reeves, "Particle systems — a technique for modeling a class of fuzzy objects," *ACM Transactions of Graphics* **2**(2), 91–108 (April 1983).
37. R. Szelisky and D. Tonnesen, "Surface modeling with oriented particles," *SIGGRAPH* **26**(2), 185–194 (July 1992).
38. T. Nishita and Y. Dobashi, "Modeling and rendering of various natural phenomena consisting of particles," In *Proc. Computer Graphics International Conference*, 149–156 (July 2001).
39. V. Di Gesu and V. Starovoitov, "Distance-based functions for image comparison," *Pattern Recognition Letters* **20**(2), 207–214 (1999).
40. R. C. Veltkamp and M. Hagedoorn, "Shape similarity measures, properties and constructions," In *Proc. Visual Information and Information Systems*, 467–476 (2000).
41. M. Woo, J. Neider, T. Davis and D. Shreiner, *OpenGL(R) Programming Guide*: *The Official Guide to Learning OpenGL, Version 1.2* (Addison Wesley, 1999).

Ulaş Yılmaz is a doctoral candidate and a research assistant at the Department of Computer Engineering in the Middle East Technical University, Ankara, Turkey. He received his BSc and MSc degrees in computer engineering from the Middle East Technical University in 1999 and 2002, respectively. His research interests include computer vision, computer graphics and pattern recognition. He is a member of the Turkish Pattern Recognition and Image Analysis Society.

Adem Yaşar Mülayim received his PhD in November 2002 from the Department of Computer Engineering at the Middle East Technical University, Ankara, Turkey. He also received his BSc and MSc degrees in computer engineering from the same department in 1993 and 1996, respectively. His research interests include computer vision, modeling, terrain image analysis, perceptual organization, image processing and pattern recognition.

Volkan Atalay is an associate professor of computer engineering at the Middle East Technical University. Previously, he was a visiting scholar at the New Jersey Institute of Technology. He received a BSc and an MSc in electrical engineering from the Middle East Technical University and a PhD in computer science from the Université René Descartes-Paris V, Paris, France. His research interests include computer vision, document analysis and the applications of neural networks. He is a member of the IEEE Computer Society and the Turkish Pattern Recognition and Image Analysis Society.