



## NRC Publications Archive Archives des publications du CNRC

### Fully Automatic Texture Mapping for Image-Based Modeling

Wuhrer, Stefanie; Atanossov, R.; Shu, Chang

For the publisher's version, please access the DOI link below./ Pour consulter la version de l'éditeur, utilisez le lien DOI ci-dessous.

<https://doi.org/10.4224/8914149>

#### NRC Publications Record / Notice d'Archives des publications de CNRC:

<https://nrc-publications.canada.ca/eng/view/object/?id=7ebcbbe2-fdb3-4c38-a688-8d520cea3c9b>

<https://publications-cnrc.canada.ca/fra/voir/objet/?id=7ebcbbe2-fdb3-4c38-a688-8d520cea3c9b>

Access and use of this website and the material on it are subject to the Terms and Conditions set forth at

<https://nrc-publications.canada.ca/eng/copyright>

READ THESE TERMS AND CONDITIONS CAREFULLY BEFORE USING THIS WEBSITE.

L'accès à ce site Web et l'utilisation de son contenu sont assujettis aux conditions présentées dans le site

<https://publications-cnrc.canada.ca/fra/droits>

LISEZ CES CONDITIONS ATTENTIVEMENT AVANT D'UTILISER CE SITE WEB.

**Questions?** Contact the NRC Publications Archive team at

PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca. If you wish to email the authors directly, please see the first page of the publication for their contact information.

**Vous avez des questions?** Nous pouvons vous aider. Pour communiquer directement avec un auteur, consultez la première page de la revue dans laquelle son article a été publié afin de trouver ses coordonnées. Si vous n'arrivez pas à les repérer, communiquez avec nous à PublicationsArchive-ArchivesPublications@nrc-cnrc.gc.ca.



National Research  
Council Canada

Conseil national de  
recherches Canada

Canada



National Research  
Council Canada

Institute for  
Information Technology

Conseil national  
de recherches Canada

Institut de technologie  
de l'information

# **NRC - CNRC**

---

## ***Fully Automatic Texture Mapping for Image-Based Modeling \****

Wuhrer, S., Atanossov, R., and Shu, C.  
August 2006

\* published as NRC/ERB-1141. 18 pages. August 2006. NRC 48778.

Copyright 2006 by  
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables  
from this report, provided that the source of such material is fully acknowledged.



---

**NRC · CNRC**

---

***Fully Automatic Texture  
Mapping for Image-Based  
Modeling***

Wuhrer, S., Atanossov, R., and Shu, C.  
August 2006

Copyright 2006 by  
National Research Council of Canada

Permission is granted to quote short excerpts and to reproduce figures and tables from this report,  
provided that the source of such material is fully acknowledged.

# Fully Automatic Texture Mapping for Image-Based Modeling

Stefanie Wuhrer      Rossen Atanassov      Chang Shu

## Abstract

Image based modeling techniques construct digital shape models from 2D images of physical objects. They are used in a wide range of applications, where both virtual and real data is required. Texture information is a crucial part of virtual models used to represent real world objects in simulations, animations, virtual and augmented reality, reverse engineering, and various other applications. In this paper, we describe a fully automated method for creating digital 3D models of real world objects with high quality texture maps from digital images. The camera calibration is obtained automatically by using self-identifying markers. The approach for texture mapping is mainly based on Hernandez-Esteban [5] and Rocchini et al. [10]. Examples of results obtained using the proposed application are presented and discussed.

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
<b>2</b>	<b>Camera calibration</b>	<b>6</b>
<b>3</b>	<b>3D Reconstruction</b>	<b>6</b>
<b>4</b>	<b>Texture Mapping</b>	<b>7</b>
4.1	Triangle-Image correspondence . . . . .	8
4.2	Vertex-Image correspondence . . . . .	8
4.3	Atlas packing . . . . .	9
<b>5</b>	<b>Results</b>	<b>12</b>
<b>6</b>	<b>Conclusion</b>	<b>15</b>
<b>7</b>	<b>Acknowledgments</b>	<b>16</b>

# 1 Introduction

Image based modeling techniques are used to construct three dimensional shape models from multiple two dimensional images of physical objects. They are used in a wide range of applications, where both virtual and real data is required. Examples of these applications include, but are not limited to simulations, animations, virtual and augmented reality, and reverse engineering. Once a mesh representation of the three dimensional model is acquired, it remains to find realistic texture information for the object. Using high quality textures on coarse geometric models often yields better visual results than using poor texture on geometrically accurate models. Hence, texture information is a crucial part of virtual models representing real world objects.

Reconstructing three dimensional digital shape models with texture information from images is a well-studied problem, hence the following overview is not extensive. Rocchini et al. [10] proposed a semi-automated approach for acquiring a three dimensional model with texture information from a set of images. The approach requires user interaction in order to calibrate the camera used to take the input images. After this calibration step, the approach automatically reconstructs the three dimensional geometry of the model by using a local registration step for each subsection of each image and by using a silhouette-based approach. One of the major contributions of Rocchini et al. is to obtain high quality texture maps for the three dimensional model. The texture information is copied into a texture atlas, where special care is taken to minimize artifacts along the edges of the mesh.

Duan [1] presents a fully automatic approach based on optimizing the photo consistency using a system of partial differential equations (PDE). Starting from an initial mesh model, the PDE-based approach adaptively refines both the geometry and the topology of the model. However, only closed surfaces can be modeled. Texture information is found automatically.

Another fully automatic approach for acquiring a three dimensional model with texture information from a set of images was proposed by Lensch et al. [7]. The camera calibration is found using a silhouette-based approach. The silhouettes are then used to reconstruct the three dimensional mesh model. The high quality texture information for the model is found automatically and copied into a texture atlas.

Hernandez-Esteban [5] improved the approach of Lensch et al. [7] to find the camera calibration automatically using a refined silhouette-based approach. The initial silhouettes are used as input to an optimization problem that finds more accurate calibration parameters. Hernandez-Esteban uses a particle-based approach that automatically creates textures for the mesh and filters highlights that may be present in the input images.

In this paper, the following problems are addressed. Given a set of input images, the camera calibration is found using a pattern of self-identifying markers. Once the

calibration is known, a silhouette-based approach is used to obtain a three dimensional mesh representation of the model. Finally, a high quality texture map for the mesh is computed. Once the texture information is found, we copy all the texture information into a small set of images called *atlases* in order to export the texture mapped model efficiently. It is possible to export the texture mapped mesh model along with the texture atlases to VRML or POV-Ray<sup>1</sup> file formats. A visualization of the problem statement is given in Figure 1.



Figure 1: *Visualization of the problem statement*

Unlike previous works reviewed above, we use self-identifying markers to calibrate the cameras, which yields a fully automated approach for reconstructing three dimensional models from a set of images. To obtain high quality texture maps for the model, we implemented an approach strongly based on previous work by Hernandez-Esteban [5] and Rocchini et al. [10] that achieves relatively small and densely packed texture atlases. The approach is particle-based and employs both triangle-to-image bindings and vertex-to-image bindings. Highlights are filtered by the approach and a blending technique ensures that patch boundaries along the edges of triangles are smooth.

The paper is organized as follows. Section 2 discusses how the input images are calibrated automatically using self-identifying markers. Section 3 explains how a three dimensional mesh representation of the model is obtained. Section 4 discusses how a high quality texture map for the three dimensional mesh model is obtained. In particular, we discuss how triangles in 3D are associated with texture triangles in the input images, how the best texture information for each vertex in 3D is found, and what steps are required to create texture atlases. Section 5 gives results obtained by our implementation and finally, Section 6 gives concluding remarks.

<sup>1</sup>Available at <http://www.povray.org>

## 2 Camera calibration

The first step in image-based modeling is to determine camera calibration parameters. The pin-hole camera model, illustrated in Figure 2, is usually used in computer vision and computer graphics to represent the image formation process [13]. Suppose  $\mathbf{X} = [X \ Y \ Z \ 1]^\top$  is a 3D point and  $\mathbf{x} = [x \ y \ 1]^\top$  is its projection in the image plane, both in homogeneous coordinate. The perspective projection can be modeled by

$$\lambda \mathbf{x} = \mathbf{K}[\mathbf{R} \ \mathbf{t}]\mathbf{X},$$

where  $\lambda$  is an arbitrary constant,  $\mathbf{R}$  is a rotation matrix and  $\mathbf{t}$  is a translation vector, together they relate the world coordinate frame to the camera coordinate frame, and  $\mathbf{K}$  is an upper triangular matrix defined as

$$\mathbf{K} = \begin{bmatrix} f_x & s & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Camera calibration amounts to find out two types of parameters: intrinsic and extrinsic [13]. The intrinsic parameters are in the matrix  $\mathbf{K}$ , that is, the focal lengths  $f_x$  and  $f_y$ , the principle point  $(u_0, v_0)$ , and the skew factor  $s$ . For modern digital camera,  $s$  can be safely assumed to be zero. The extrinsic parameters, the rotation matrix  $\mathbf{R}$  and the translation vector  $\mathbf{t}$ , define the position of the camera where each image is taken with respect to a world coordinate frame.

Camera calibration is usually done with the help of a calibration pattern. Traditionally, planar patterns, like chessboard [11], or 3D patterns, like cube [4], have been used. These patterns have known geometry and when imaged, they provide correspondences between 3D points with known coordinates and points in the image plane. In this work, we use a planar pattern that consists of self-identifying markers, called ARTags [2]. These markers can be identified reliably and quickly in the images. The corners of the markers provide the 3D-2D correspondences that are necessary for solving the calibration parameters.

We arrange the markers in an array and print them on a planar panel. The object to be modeled is put on the panel and multiple images are taken from different viewpoints. We use Zhang's algorithm [15] to compute both intrinsic and extrinsic parameters.

## 3 3D Reconstruction

Once a set of images are calibrated, that is, the camera intrinsic parameters found and camera pose for each image computed, we adopt the visual hull method to reconstruct



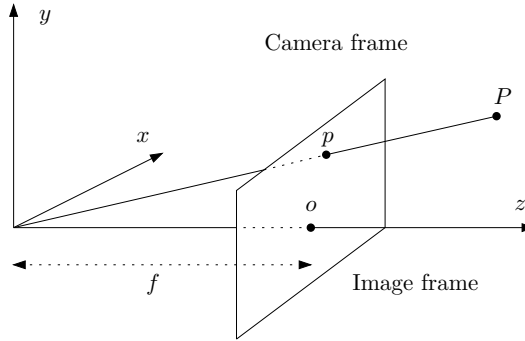


Figure 2: *The pinhole camera model*

a 3D model [6]. The basic idea is that the camera pose together with the silhouette of the object in the image define a cone in 3-space in which the object is contained. By intersecting all such cones, we find an approximation (the visual hull) of the original shape. The more images we take, the closer the visual hull approximates the object. Certain concave part of the object can never be reconstructed. But for most shapes, this method is a simple way to reconstruction 3D shape from images.

The key issue here is to find the silhouette. This is a segmentation problem, which is generally difficult to solve. In practice, controlled background, for instance, blue screen, has been used [12, 9]. In our system, the background is the ARTag array. We use the ARTags as calibration patterns. What is remarkable is that they also help separate the object from the background. This is because we can recreate the image of the panel in each view as if it is imaged at the viewpoint but without the object. Then background subtraction is simply the difference of the two images. Details of this method can be found in Fiala and Shu [3].

Once we find the silhouettes, we use a volumetric method to reconstruct the model. The 3D space is first divided into voxel grids. Each voxel is tested against all the cones. If it is outside a cone, it is carved out. The voxels that have passed the tests form the visual hull. Finally, we run the marching cube algorithm [8] to create the triangle mesh.

## 4 Texture Mapping

In this section, we explain how to obtain a high quality texture map for the three dimensional mesh model. Section 4.1 discusses how triangles in 3D are associated with texture triangles in the input images, Section 4.2 discusses how the best texture information for each vertex in 3D is found, and Section 4.3 explains the steps are required to create texture atlases.

## 4.1 Triangle-Image correspondence

The aim is to find for each triangle  $T$  of the mesh a texture triangle  $\tilde{T}$  located in a single input image that best matches the 3D triangle. Denote the projection of  $T$  on the input image  $I$  using the previously obtained calibration by  $T'$ . The best matching triangle  $\tilde{T}$  is the projection  $T'$  onto an input image  $I$ , such that  $T$  is visible in  $I$  and such that  $\tilde{T}$  maximizes the area of all the projections  $T'$ .

To find the best texture triangle for the 3D triangle  $T$ , we first test for each vertex of the mesh whether it is occluded in any of the input images. This test is done using a hardware-accelerated  $z$ -buffer solution offered by OpenGL. To use this solution, the geometry of the mesh is rendered using the current calibration information as viewing matrix. The vertex  $v$  for which the occlusion test is carried out is pushed to the graphics card and the  $z$ -buffer value is read back. The vertex  $v$  is visible if the  $z$ -buffer value corresponds to the position of  $v$  in the scene. Otherwise,  $v$  is not visible. The results of the occlusion tests are stored in an incidence matrix. For any given triangle, we then only take images into consideration where none of the triangle's vertices are occluded. For each image with this property, we compare the outer normal of the triangle with the viewing direction of the camera. The most fronto-normal camera is chosen, since this camera yields the maximal area in the projection if zoom is neglected.

## 4.2 Vertex-Image correspondence

The method used for texture mapping requires for each vertex  $v$  of the mesh a texture pixel of an input image that best matches  $v$ . Note that the matching of a texture pixel to  $v$  is not determined by the mappings of texture triangles to  $v$ 's incident triangles. The triangles incident to  $v$  may be associated with texture triangles in different input images and conversely, the three vertices of a triangle  $T$  may be associated with pixels in different input images. Therefore, the texture binding process outlined in Section 4.1 needs to be adapted to find the best texture pixel for a given 3D vertex.

The incidence matrix storing occlusion information explained in Section 4.1 is used to find all the images in which a given vertex  $v$  is visible. Among all the images in which  $v$  is visible, we only consider images in which the projection of  $v$  is not a *silhouette vertex*. A pixel in an image is considered silhouette if it is located on the boundary of the object and the background. Such pixels are not used for texture mapping, because slight measurement errors in the camera's calibration parameters result in big color changes of the pixel. The projection of  $v$  is a silhouette vertex if at least one of the triangles adjacent to  $v$  are occluded in the image [10]. Therefore, the matrix storing occlusion information can be used to find silhouette vertices.

Among all the visible projections of  $v$  that are not silhouette vertices, we choose the one located on the most fronto-normal image as the texture pixel with highest quality.

This requires the knowledge of the normal vector for each vertex of the mesh. Clearly, differential geometry implies that since a vertex of the mesh is an discontinuity, the normal vector is not defined. However, various approximation techniques exist to find the normal vector  $\vec{n}$  of a vertex in a triangular mesh. In this project,  $\vec{n}$  is computed as a weighted sum of the normals of  $v$ 's neighboring triangles. This approach is both simple and yields acceptable results. For a more detailed discussion on possible choices of weights, refer to Wilke [14].

### 4.3 Atlas packing

This section discusses the creation of texture atlases used for texture mapping. The approach that was implemented mainly follows the approach published by Hernandez-Esteban [5]. First, the layouts of the texture atlases are created and each triangle of the mesh is associated with a triangle in an atlas by setting texture coordinates for the 3D triangle. Second, the texture atlases are filled with color. Finally, all atlases are exported to JPEG image files and the final representation of the mesh along with texture coordinates is exported as a VRML or a POV-Ray file.

We create one texture triangle per 3D triangle, which has the advantages of introducing zero rectification distortion as well as being easy to implement. To obtain the layout of the texture atlases we assume that all 3D triangles have a good aspect ratio, which allows us to choose texture triangles as equilateral triangles. The layout of the texture atlas is a tiling where most of the triangles are grouped in squares, see Figure 3. The pairing of triangles into squares ensures that the atlas maps are space efficient in practice. Statistics on the space efficiency of this layout can be found in Hernandez-Esteban [5]. Next we describe how to assign a texture tile to each 3D triangle and how to place the texture tile into the global texture atlas.

Each 3D triangle  $T$  is associated with a texture tile size in the following way. The longest edge of the texture triangle  $\tilde{T}$  associated with  $T$  is multiplied by a quality constant. Denote the result of this computation by  $l$ . The tile size is then found by matching  $l$  to the closest integer in the series of tile sizes  $S_i = 2S_{i-1} - 1, i > 0, S_0 = 2$ . Note that all the tile sizes  $S_i, i > 0$  are odd integers. Once the tile sizes are found for each triangle of the mesh, the 3D triangles are associated with texture tiles. To achieve that, tiles in the atlas are paired into squares, called *quads*, sharing a diagonal. The 3D triangles are paired according to the following criteria. The first triangles to be paired are adjacent triangles in the mesh having the same tile size. The next class of triangles to be paired are adjacent triangles in the mesh with different tile sizes. They are paired by increasing the smaller of the two tile sizes. The last triangles to be paired are non-adjacent triangles in the mesh having the same tile size. Since the 3D triangles are not adjacent, the shared diagonal results in a discontinuity in the image. This results in artifacts along the boundary of the 3D texture mapped triangles. To

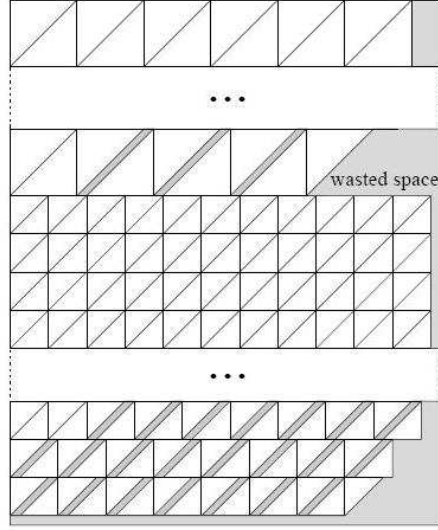


Figure 3: *The layout of the texture atlases. Figure by Hernandez-Esteban [5].*

avoid this artifact, the diagonal of the quad is doubled and bilinear interpolation is used to obtain a smooth transition between the two triangles. Figure 4 shows this effect.

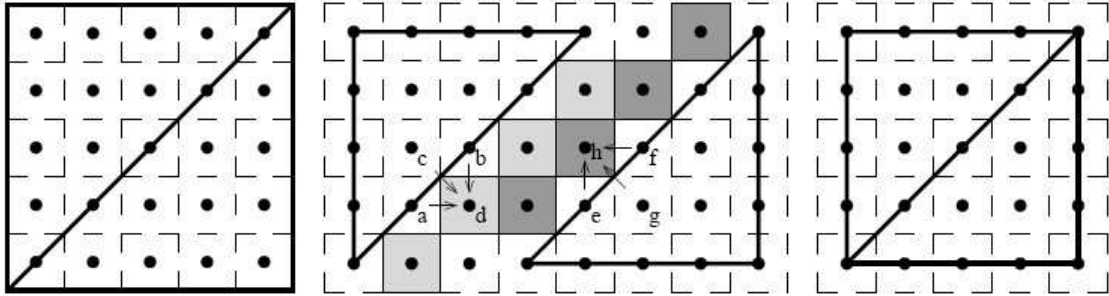


Figure 4: *Bilinear interpolation along texture tiles with doubled diagonals. Figure by Hernandez-Esteban [5].*

After pairing the tiles into quads, each tile is assigned a location in the set of texture atlases. This is achieved by sorting the texture tiles in decreasing order by size. For each tile size, the first tiles to be assigned a location are tiles assigned to adjacent triangles in 3D. The next tiles to be layed out are tiles assigned to non-adjacent triangles in 3D. Finally, isolated tiles are considered. Note that there exists

at most one isolated tile per tile size. The texture tiles are then put into the atlases in decreasing order along rows. If the maximum width of an atlas is reached, a new row is started. If the atlas does not contain sufficient space to hold further tiles, an additional atlas is started. Note that due to the standard of graphics cards, the size the atlas is always a power of 2. Along with laying out the triangular tiles in the atlas, we set the texture coordinates accordingly for the corresponding triangle in 3D.

It remains to fill the texture atlases with color. This is achieved using a particle-based approach, where each triangle of the mesh is treated as particle. To assign colors to the pixels of the atlases, the vertex-image correspondences obtained in Section 4.2 are used. The three vertices of a triangle  $T$  in 3D are associated with at most three input images by vertex-image correspondences. This induces a correspondence of  $T$  to at most three input images. Each 3D triangle  $T$  is projected to each of the corresponding images, which yields at most three projected triangles. The color information of the projections in the images is used to fill the tile in the atlas corresponding to  $T$  with color using barycentric coordinates. Figure 5 illustrates this coloring approach. For each pixel in the tile corresponding to  $T$ , we obtain the color as a weighted sum of the projections of  $T$  in the at most three images associated with  $T$ . The weight used for the coloring corresponds to the barycentric coordinate of the tile pixel.

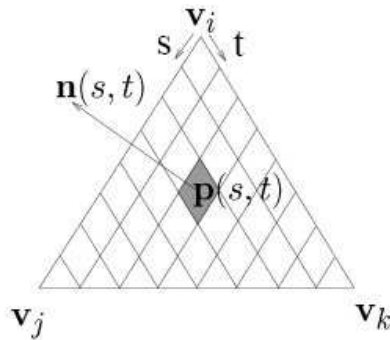


Figure 5: *Particle based approach to blend the colors in the atlas. Figure by Hernandez-Esteban [5].*

For tiles assigned to non-adjacent triangles in 3D, the color of the doubled diagonal is obtained using the bilinear interpolation approach discussed above and shown in Figure 4. This approach blends the colors of the input images associated with vertices of the mesh for each triangle of the mesh. It further offers the advantage that pairing triangles into quads yields shared diagonals, thereby ensuring smooth transitions

along the boundaries of 3D triangles. For non-shared boundaries, the smooth transitions are still achieved by using barycentric weights for the coloring approach. Even adjacent triangles corresponding to tiles of different sizes do not introduce artifacts in the texture map. The weighted blending of color information obtained using input images offers the additional advantage that highlights in the images are filtered. Once the texture atlases are filled with color, a full representation of the textured object is available. The approach used in this project to create a texture atlas is strongly based on the approach proposed by Hernandez-Esteban [5].

Finally, we export all atlases to JPEG image files and the final representation of the mesh along with texture coordinates as a VRML or a POV-Ray file. The exported representation can later be used, hence the proposed algorithm can be viewed as a preprocessing step to various applications. Note that the running time of the algorithm need not be real-time.

## 5 Results

We demonstrate our results using four models. The implementation is carried out in C++ using the OpenGL<sup>2</sup> library for rendering. The models are stored using the trimesh<sup>3</sup> library. This library offers data structures to store, read, write, and manipulate three dimensional triangular meshes. In the following, the running times are only given for finding and exporting the high quality texture information. All running times are measured using a 2.4 GHz Intel Pentium 4 processor with 1 GB RAM. The code was compiled under Linux Fedora 4 using gcc 4.0.2. The images were captured using a Canon Powershot S60 digital camera with 5 MB pixels.

The first model to be discussed is the model of a hat. The 30 input images were automatically calibrated to yield a three dimensional mesh representation of the hat consisting of 711 vertices and 1361 faces. Two of the input images are shown in Figures 6(a) and (b), respectively. The camera positions for the 30 input images are shown in Figure 6(c). Refer to Figure 6(d) to see the reconstructed geometry of the model. All of the 30 calibrated input images were used to obtain the texture map. The results of the textured model are shown in Figures 6(e)-(h). Figure 6(e) shows the exported texture atlas.

Note that the packing of the atlas in Figure 6(e) is dense and space efficient. Figure 6(f) shows that the overall appearance of the object is visually pleasing and clearly compares to the input image shown in 6(a). Figure 6(g) shows a smooth texture map. No artifacts along edges of the mesh are visible. Figure 6(h) illustrates a few artifacts, but the quality is overall acceptable.

---

<sup>2</sup>Available at <http://www.opengl.org>

<sup>3</sup>Available at <http://graphics.stanford.edu/software/trimesh>

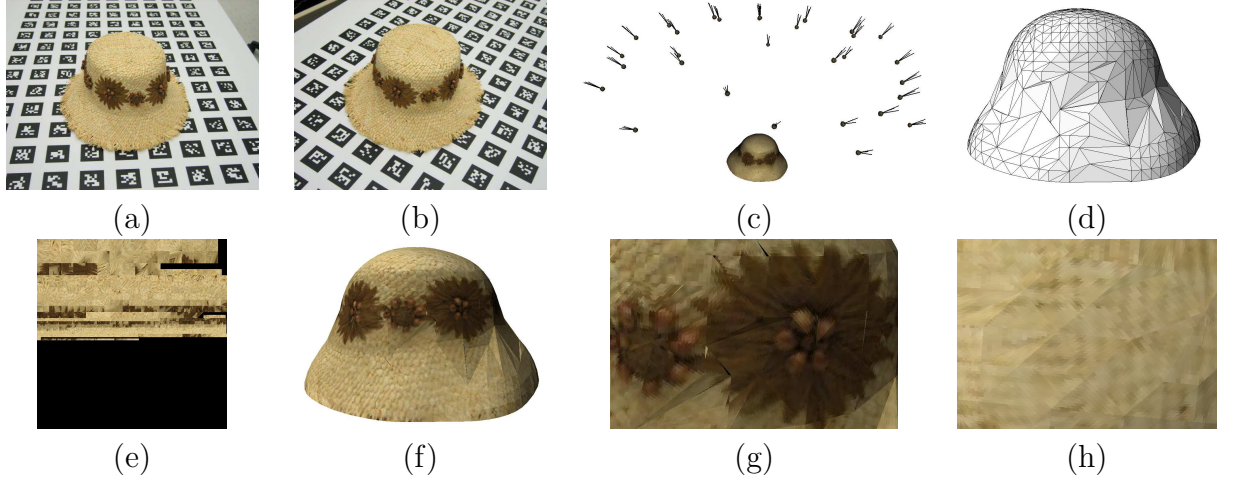


Figure 6: *Texture mapping of a hat model*

The second model to be discussed is the model of a clown. The 20 input images were automatically calibrated to yield a three dimensional mesh representation of the clown consisting of 2346 vertices and 4641 faces. Two of the input images are shown in Figures 7(a) and (b), respectively. The camera positions for the 20 input images are shown in Figure 7(c). Refer to Figure 7(d) to see the reconstructed geometry of the model. All of the 20 calibrated input images were used to obtain the texture map. The results of the textured model are shown in Figures 7(e)-(h). Figure 7(e) shows the exported texture atlas.

The geometric reconstruction of the clown is erroneous, since the hole of the piggy bank at the back of the clown's head does not correspond to a hole in the mesh. The quality of the texture mapped model shown in Figures 7(f)-(h) is of lower quality than in the previous model. This is due to specularities on the surface of the physical object.

The third model to be discussed is the model of a stuffed dog. The 17 input images were automatically calibrated to yield a three dimensional mesh representation of the dog consisting of 3036 vertices and 5856 faces. One input image is shown in Figures 8(a). The camera positions for the 17 input images are shown in Figure 8(b). Refer to Figure 8(c) to see the reconstructed geometry of the model. All of the 17 calibrated input images were used to obtain the texture map. The results of the textured model are shown in Figures 8(d)-(h). Figures 8(d) and (e) show the two exported texture atlases. Note that although the second atlas is almost empty, we can not increase the atlas size slightly, since the size of the atlases needs to be a power of 2.

Figures 8(f)-(h) show that the quality of the texture mapped model is high. Only a few artifacts appear along the dog's face. This implies that the approach yields

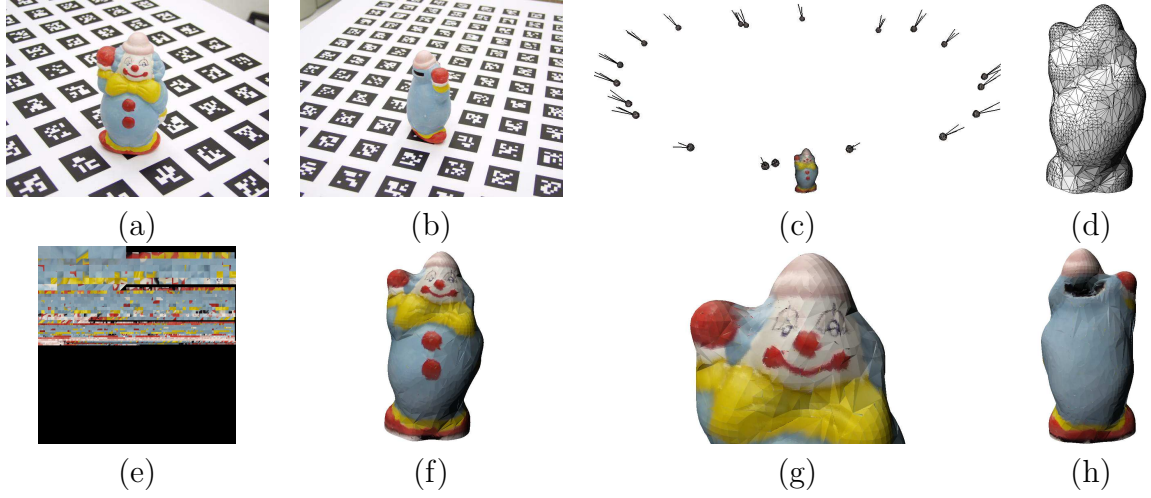


Figure 7: *Texture mapping of a clown model*

high-quality texture maps even when the geometry of the input model is very coarse and does not capture the geometry of the real-world object well.

The last model to be discussed is the model of a bag. The 20 input images were automatically calibrated to yield a three dimensional mesh representation of the bag consisting of 7091 vertices and 13975 faces. One input image is shown in Figure 9(a). The camera positions for the 20 input images are shown in Figure 9(b). Refer to Figure 9(c) to see the reconstructed geometry of the model. All of the 20 calibrated input images were used to obtain the texture map. The results of the textured model are shown in Figures 9(d)-(h). Figures 9(d) and (e) show the two exported texture atlases.

The geometric reconstruction of the bag contains holes. This is due to occlusions in the images. Clearly, it is only possible to reconstruct visible parts of the object. Figure 9(f) has the same appearance as the input image shown in Figure 9(a). Figure 9(h) shows an up close rendering of the model, where no artifacts are visible. This shows that the approach yields high quality for diffuse surfaces when the geometry of the object is approximated well by the input model.

The running times of the approach for each of the input images are given in Table 1 below. Note that the application discussed in this paper does not require to obtain real-time running times. The texture mapping is performed once for a given model and then exported. When using the model later, only a VRML file with corresponding JPEG images needs to be read. Therefore, running times in the order of a few minutes as achieved by the presented application are acceptable.

Finally, we illustrate six different renderings of a scene using POV-Ray and the



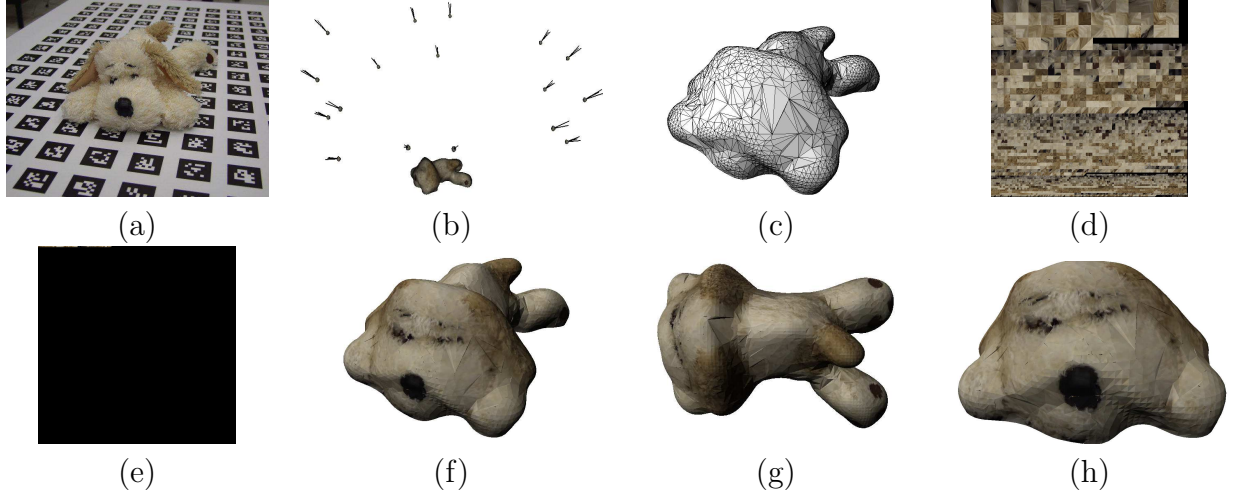


Figure 8: *Texture mapping of a dog model*

Model	# of triangles	# of vertices	# of input images	Building incidence matrix	Building atlas	Exporting JPEG	Exporting VRML	Total time
Hat	1361	711	30	6.1s	4.6s	0.2s	0.1s	12.6s
Clown	4641	2346	20	13.1s	25.0s	0.2s	0.2s	52.8s
Dog	5856	3036	17	15.0s	36.7s	0.3s	0.2s	52.8s
Bag	13975	7091	20	39.5s	200.7s	0.3s	0.4s	242.1s

Table 1: *Running times of the algorithm.*

clown model. The composition was built using the exported clown model, a glass sphere, and a metallic frame. All scenes were rendered using POV-Ray's advanced photon mapping in order to create the caustics effects on the floor and the metallic walls. The left column of images in Figure 10 illustrates the rendering of the scene using three bright white-colored lights and different colored floors respectively. The top two images in the right column were rendered using three different colors for each light source. The last image was rendered using red tinted glass material for the sphere around the clown, all light sources and the floor are white colored.

## 6 Conclusion

We described a fully automated application which creates three dimensional models with high quality texture maps from images of real world objects. The approach uses self-identifying markers to obtain a fully automatic system. The texture map

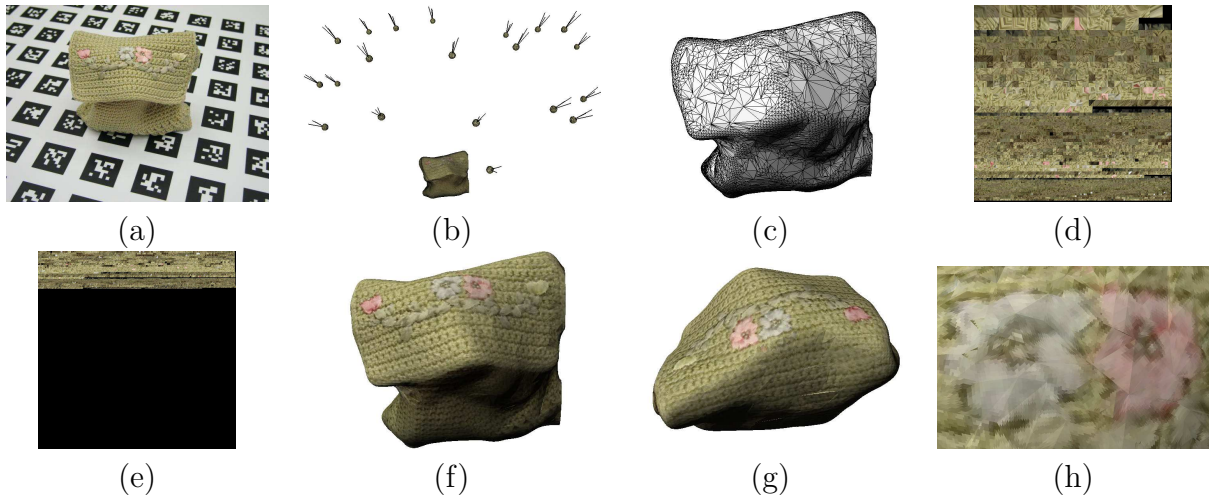


Figure 9: *Texture mapping of a bag model*

of the model offers the advantages of minimizing artifacts along mesh boundaries, of filtering highlights present in the input images, and of space savings. The texture mapped models can be exported by the application and later be loaded by various applications in order to use the digital model of a physical object in simulations, animations, virtual and augmented reality, reverse engineering, and various other applications.

## 7 Acknowledgments

Part of this project was initiated as a course project of a Computer Graphics graduate course lectured by Dr. Jochen Lang at Ottawa University. We wish to thank Dr. Jochen Lang for fruitful discussions on the part of the paper related to texture mapping.

## References

- [1] Y. Duan. *Topology Adaptive Deformable Models for Visual Computing*. PhD thesis, State Univeristy of New York, 2003.
- [2] M. Fiala. ARTag: a fiducial marker system using digital techniques. In *CVPR'05*, volume 1, pages 590 – 596, 2005.

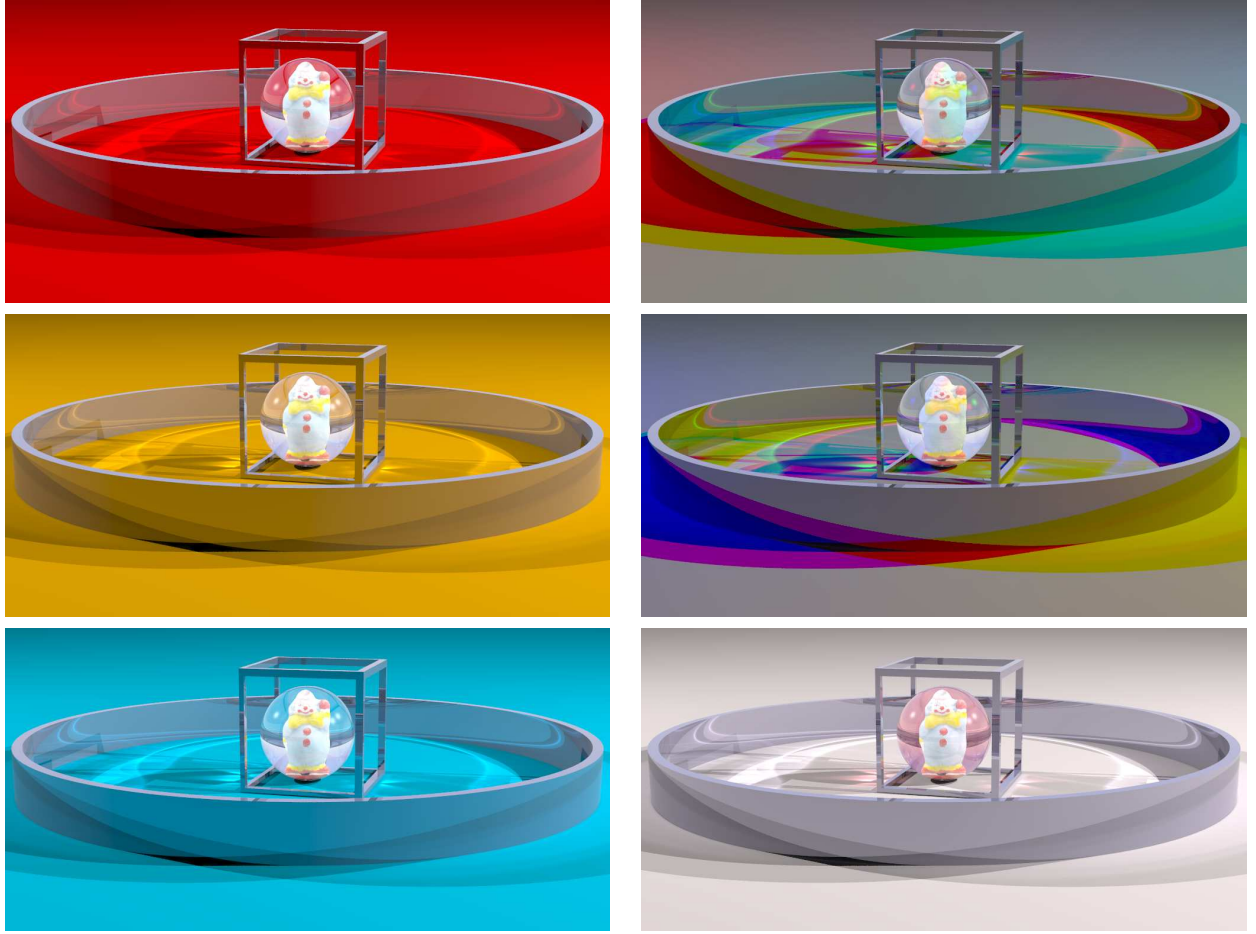


Figure 10: *Scene composition using the clown model. Rendered using POV-Ray.*

- [3] M. Fiala and C. Shu. Background subtraction using self-identifying patterns. In *Proc. of CRV'05 (Canadian Conference on Computer and Robot Vision)*, pages 558–565, May 2005.
- [4] Janne Heikkilä. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, October 2000.
- [5] C. Hernandez-Esteban. *Stereo and Silhouette Fusion for 3D Object Modeling from Uncalibrated Images Under Circular Motion*. PhD thesis, cole Nationale Suprieure des Tlcommunications, Paris, 2004.

- [6] A. Laurentini. The visual hull concept for silhouette based image understanding. *IEEE PAMI*, 16(2):150–162, 1994.
- [7] H. Lensch, W. Heidrich, and H. Seidel. Automated texture registration and stitching for real world models. *Graphical Models*, 63:245–262, 2001.
- [8] W.E. Lorensen and H.E. Cline. Marching Cubes: A High Resolution 3-D Surface Construction Algorithm. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, volume 21, pages 163–169, July 1987.
- [9] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3D photography using opacity hulls. In *SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques*, pages 427–437, 2002.
- [10] Claudio Rocchini, Paolo Cignoni, Claudio Montani, and Roberto Scopigno. Acquiring, stitching and blending diffuse appearance attributes on 3d models. *The Visual Computer*, 18(3):186–204, 2002.
- [11] Chang Shu, Alan Brunton, and Mark Fiala. Automatic grid finding in calibration patterns using Delaunay triangulation. Technical Report NRC-46497/ERB-1104, National Research Council of Canada, Institute for Information Technology, 2003.
- [12] Alvy Ray Smith and James F. Blinn. Blue screen matting. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 259–268, 1996.
- [13] E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice-Hall, 1998.
- [14] W. Wilke. *Segmentierung und Approximation grosser Punktwolken*. Ph.D Dissertation, Technische Universitaet Darmstadt, November 2000.
- [15] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision (ICCV99)*, pages 666–673, Corfu, Greece, 1999.