Reconstruction of Textured Models from Multiple Views

Heinz Mayer[1], *Konrad Karner*[2], *and Alexander Bornik*[2]

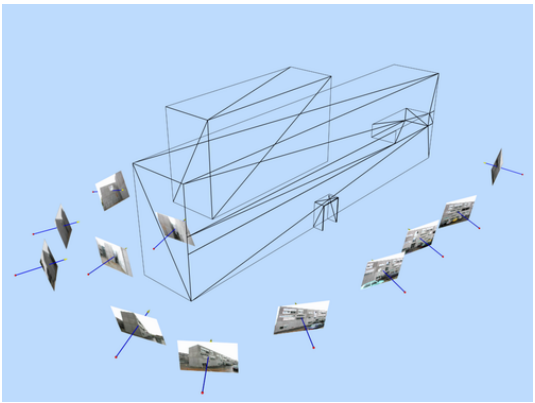
[1] Computer Graphics and Vision, Graz, University of Technology
email: mayer@icg.tu-graz.ac.at

[2] VRVis Research Center for Virtual Reality and Visualization
email: karner@vrvis.at

Abstract:
This paper compares two different methods to reconstruct textured models from real images. View Dependent Texture Mapping (VDTM), one of the investigated methods, executes the texturing during the rendering phase using the real images. In the second method, the multiresolution texture mapping, a texture map is calculated in a preprocessing step. We show, that multiresolution texture mapping outperforms VDTM in all investigated specifications which are geometric complexity, occlusion handling, rendering performance and rendering software support.

1 Introduction

The reconstruction and visualization of real world objects is becoming an increasing research area in both the computer graphics and the computer vision community. In our approach we concentrate on the reconstruction of textured models from multiple real images. Having a set of oriented input images as shown in Figure 1(a) and a geometric description from a photogrammetric modeling step, the goal is to reconstruct a synthetic textured model. A first result of one presented method can be seen in Figure 1(b). In the following chapter we describe two different methods to texture polygons.



(a) Input data consist of multiple views and a geometric representation of the scene.

(b) The goal. Reconstruction of a textured model of the building.

Figure 1: Problem description.

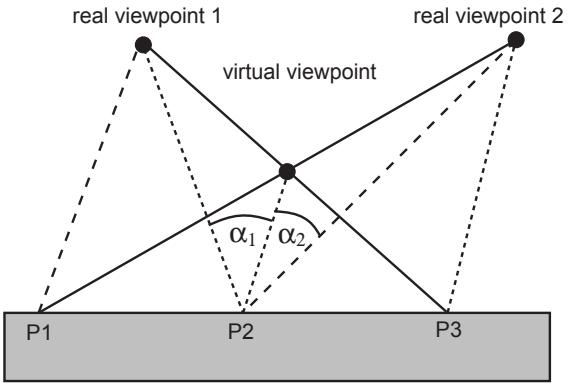


Figure 2: View-dependent fit function.

A comparison between these two methods is given in chapter 3.

2 Textured Models from Multiple Views

Over the years a number of methods to capture shape and surface properties are invented. In this paper we focus on methods using imaging sensors. Furthermore, the approach combines the information from all views according to their geometric resolution. Additionally, the method operates on standard graphics hardware to remain interactive.

Sato et al. [6] presented a method to capture the shape, a parametric reflectance model, and a texture map by using a robotic arm, a CCD camera, and a laser range finder. Although the method delivers good results it assumes that the object can be handled by the robotic arm to reconstruct the geometry and texture from different views. A couple of methods can be found in the literature which focus on the automatic stitching of image sequences [7],[3]. All of them exploit image correspondences to reconstruct the geometric transformation between them. These methods ignore the different geometric resolutions and occlusions within the input images.

2.1 View Dependent Texture Mapping

2.1.1 Method outline

The method presented by Debevec [1] avoids the reconstruction of a unique texture from multiple images by doing this reconstruction during the rendering phase. The scene objects are textured with all the images from the data acquisition. The final texture results from a projective texture mapping of the original images combined by alpha blending. The alpha values are calculated according to the angle between the viewing direction of the synthetic scene and the viewing direction of the image acquisition. The lower this angle the more likely the texture represents the correct surface details of

the object. Due to perspective projection this blending has to be done on a per pixel base. Figure 2 illustrates the geometric relationship. It shows two real viewpoints for which input images exist. The blending function which depends on the ratio between the two angles α_1 and α_2 changes smoothly from point $P1$ where the image from viewpoint 2 is the best view to point $P3$. Furthermore, visibility calculation for every polygon in every view has to be done in a preprocessing step. Doing so prevents artifacts in the resulting texture due to self-occlusion.

Unfortunately, the calculation of the blending factors on a per pixel base for a reasonable number of images turns out to be very time consuming, so interactive rendering in real-time is impossible.

2.1.2 Interactive Rendering

An efficient hardware implementation of view dependent texture mapping can be found in [2]. This solution solves the problem by rendering the whole scene choosing only the best 3 views for each polygon. In addition blending factors are calculated per polygon.

In order to work properly it is crucial to ensure a unique mapping between the polygons and the viewpoints. In most cases there exist some polygons only partly visible from some viewpoint. Thus, the polygons of the scene have to be clipped against the viewing frustum of each viewpoint to solve the problem. In addition occlusion is handled by clipping against shadow volumes formed by occluding polygons and viewpoints.

Uniformly sampling viewpoints across the front face of each polygon results in the best three views to use for rendering. This information is stored in a data structure called view map. As the number of sampling points is constant the blending factors for each polygon can be easily calculated using barycentric coordinates within the triangle of the sampling mesh a virtual viewpoint corresponds to. The barycentric coordinates are very useful in this context, because the always sum up to 1 for any point located within a triangle. This property makes them a good choice for blending factors.

Rendering a scene using the view map is a three stage process, as most of the polygons have to be rendered three times using the appropriate blending factors and texture images.

2.2 Multiresolution Textures

As outlined in Section 1 we can benefit from an existing platform to reconstruct a 3D model with registered views. The overall method to calculate a texture map for a virtual object from multiple views of a real-world object is outlined in Algorithm 1.

Algorithm 1 Method outline.

1: **for all** polygons of the reconstructed model **do**
2: build the quadtree using all views
3: extract a texture with the desired resolution
4: **end for**

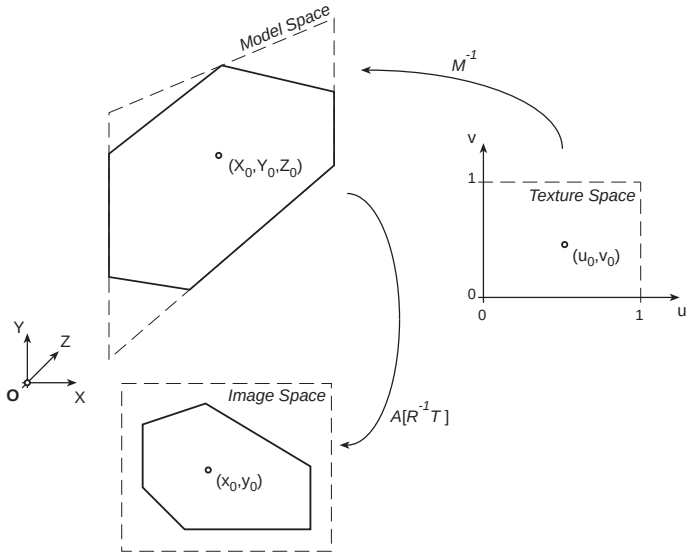


Figure 3: Geometric relation between texture-, model-, and image space.

2.2.1 Geometric Transformations

The geometric relation between an image of an object from a registered view, the 3-dimensional object itself and the texture map are shown in Figure 3. In a first step texture coordinates are mapped to object coordinates using an inverse mapping function M^{-1} which is defined for a bounding rectangle around the object to be textured.

The second transformation is the well known projective transformation. It can be further subdivided into an affine, a perspective and an Euclidean (3D rotation R and translation T) transformation. The resulting transformation

$$\tilde{x} = A \left[R^{-1} T \right] \tilde{X} \tag{1}$$

maps arbitrary 3D points to coordinates within the image. A more detailed description of all these transformations can be found in [4].

2.2.2 Texture Reconstruction

Our method of texture reconstruction is based on an approach initially introduced by Ofek et.al. [5] for elimination of specular highlights. It uses a quadtree as the core data structure. This allows to insert the texture elements (*texel*) at the quadtree levels according to each elements geometric resolution. Insertion into the quadtree is performed by projection of corner vertices of texture space into image space using the following transformation that combines the two transformations from section 2.2.1.

$$\tilde{x} = A \left[R^{-1} T \right] * M^{-1} v \tag{2}$$

After the transformation step the size of the resulting quadriliteral in image space is calculated. Texture space is subdivided recursively until the projection size matches a single pixel. Color values

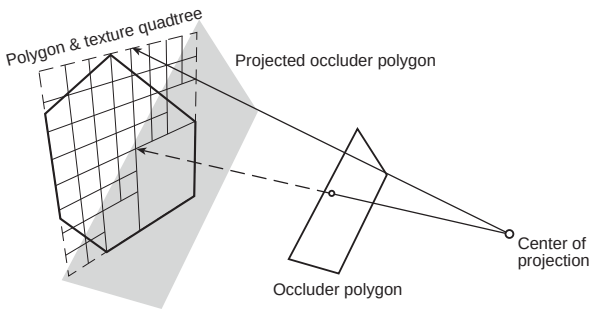


Figure 4: The influence of occluder polygons to the quadtree structure.

are stored within the nodes of the quadtree as well as the size. Multiple entries are combined using the size to weigh the color values. This procedure ensures that parts with higher resolution are more dominant than parts with lower resolution.

After the insertion of images from multiple views some parts of the quadtree might still not contain any values. Therefore, the information is distributed between the levels by propagating the entries up and down the quadtree.

2.2.3 Occlusion Handling

Up to now we assume that the images used for texture reconstruction contain the correct texture information for the surface, which is not true in case of occlusion. Three possible occlusions are:

1. **self occlusion** - parts of the object are in front of others

2. **occlusion by a modeled object** - in this case the occluder object has to be present in the geometric representation

3. **occlusion by a non-modeled object** - small objects, transient objects, or object that are hard to model (e.g. trees)

In case that any type of occlusion occurs, the algorithmic frameworks explained so far fails. Figure 4 shows how an occluding polygon influences the content of the quadtree data structure. Color values from the occluding polygon would enter tree nodes in the region marked grey leading to artifacts in the texture output. Visibility calculations derived from the geometric model in combination with the registration of each particular view are used to selectively insert pixels into the texture quadtree. Thus, recursion is carried out until the highest resolution within the input image is reached or the corresponding part of the quadtree is not visible from that view. Using this method distorting color values can't enter the quadtree.

For the third possible occlusion case, non-modeled occlusion, deterministic visibility calculations are not possible. Instead we employ outlier detection mechanisms. Differences in the pixel color for a particular quadtree node are assumed to relate to differences in resolution. But generally speaking

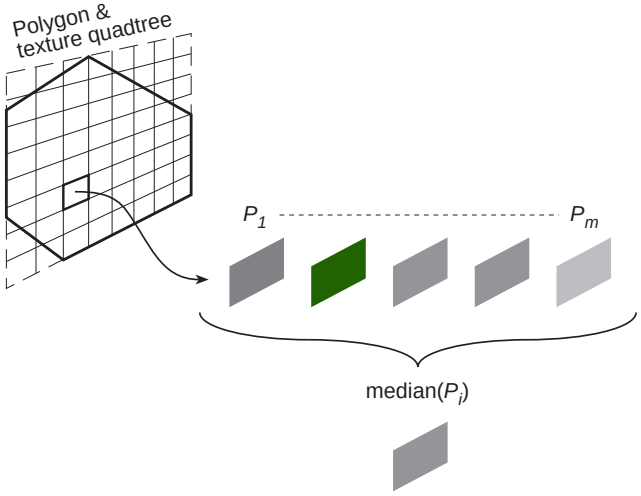


Figure 5: Pixel stack from one quadtree node and occlusion detection in principle.

they are rather small. The usage of weights ensures that higher resolution information is preserved. In case of occlusion the color values are likely to differ from the other values. We employ a median filter to single out undesired values. Therefore, we store the color values and the size for each tree node in lists rather than combining them immediately. After the insertion of all images the median can be calculated for all the nodes of the tree. Color values outside of a user specified range around the median are assumed to be values from occluders. Outliers can be successfully removed having at least three entries per node. Unfortunately, it is hard to predict the number of entries per node in advance, so it is a good idea to use a large number of images for each object to ensure that at least 3 entries per node are present. Figure 5 shows how our median filter works in principle. The result is a multiresolution representation of a texture without unwanted occlusions from modeled or non-modeled objects. By selecting a particular quadtree level a texture with fixed geometric resolution can be extracted. In combination with the geometric description of the scene a virtual model is created for visualization issues.

3 Comparison and Results

Our implementation of multiresolution textures from multiple views poses a powerful framework for texturing complex scenes taking into account modeled and non-modeled occlusions. In addition specular highlights are removed. Due to the fact that we operate on standard VRML-files the output can be viewed using any VRML browser. The outcome is delivered in multiple spatial resolutions, computational complexity depends on the desired spatial resolution and can be specified by the user. This feature and the fact that our method works off-line clearly outplays view-dependent textures, which have the following drawbacks

- *Geometric complexity* increases in contrary to our method due to the intersection with viewing frustums and shadow polygons (see Figure 6).

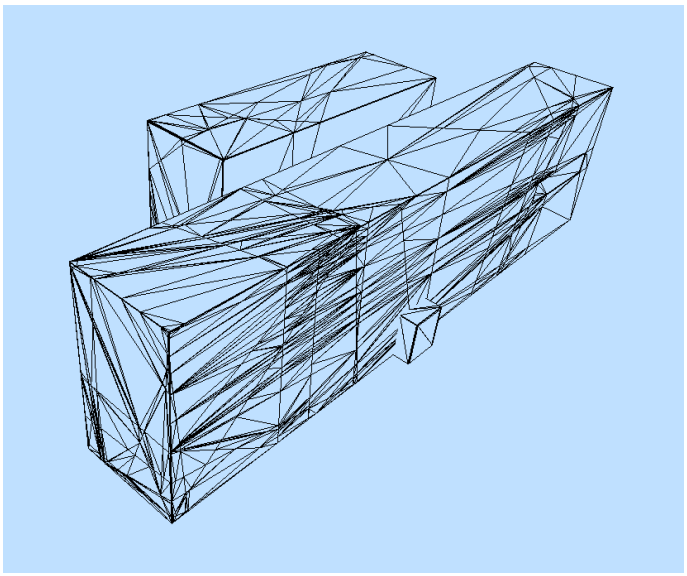
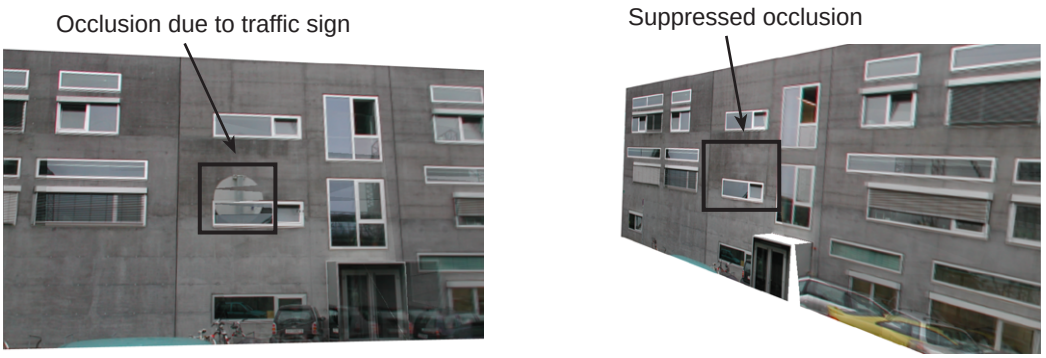


Figure 6: ***VDTM problem 1:*** **The increase of geometric complexity.**



(a) Occlusion conservation.

(b) Occlusion supression.

Figure 7: ***VDTM problem 2:*** **Inappropriate occlusion handling. The angle between orientation of the input views and the virtual viewpoint are used for the blending function. Therefore, results according to occlusion handling vary with viewing direction.**

- *Inappropriate occlusion handling.* Non-modeled occluding objects can't be removed. Even worse, views showing the occluding object in the center of the field of view are preferred (see Figure 7). (Debevec suggests to eliminate by hand masking)

- *Multiple rendering steps* require higher hardware performance.

- *Specialized rendering software* is necessary - we can use any VRML-browser.

References

[1] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, volume 30, pages 11–20, New Orleans, Lousiana, 1996.

[2] Paul E. Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with pro-

jective texture-mapping. In *9th Eurographics Workshop on Rendering '98*, pages 105–116, Vienna, Austria, June 1998. Eurographics Association.

[3] M. Irani, P. Anandan, and S. Hsu. Mosaic based representations of video sequences and their applications. In *IEEE International Conference on Computer Vision*, pages 605–611, Boston, MA, 1995. IEEE Computer Society Press.

[4] Heinz Mayer, Alexander Bornik, Joachim Bauer, Konrad Karner, and Franz Leberl. Multiresolution texture for photorealistic rendering. In *Computer Graphics (SCCG '2001 Proceedings)*, In *Computer Graphics* Proceedings, Annual Conference Series, 2001, pages 174–183, Budmerice, Slovakia, April 2001. Comenius University Bratislava.

[5] Eyal Ofek, Erez Shilat, Ari Rappoport, and Michael Werman. Multiresolution textures from image sequences. *IEEE Computer Graphics and Applications*, 17(2):18–29, March-April 1997.

[6] Yoichi Sato, Mark D. Wheeler, and Katsushi Ikeuchi. Object shape and reflectance modeling from observation. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, In *Computer Graphics* Proceedings, Annual Conference Series, 1997, pages 379–387, Los Angeles, California, August 1997. ACM SIGGRAPH.

[7] R. Szeliski and H. Shum. Creating full view panoramic mosaics and environment maps. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 251–258, Los Angeles, California, 1997.

Parts of this work have been done in the VRVis research center, Graz & Vienna/Austria (http://www.vrvis.at), which is partly funded by the Austrian government research program Kplus.