High Quality Texture Reconstruction from Multiple Views

Alexander Bornik
Institute for Computer Graphics and Vision
Graz University of Technology
bornik@icg.tu-graz.ac.at

Konrad Karner
VRVis Research Center for Virtual Reality and Visualization
Vienna, AUSTRIA
karner@icg.tu-graz.ac.at

Joachim Bauer
VRVis Research Center for Virtual Reality and Visualization
Vienna, AUSTRIA
bauer@icg.tu-graz.ac.at

Franz Leberl
Institute for Computer Graphics and Vision
Graz University of Technology
leberl@icg.tu-graz.ac.at

Heinz Mayer
SFT Steyr Fahrzeugtechnik
Graz, AUSTRIA
heinz.mayer@sft.steyr.com

Abstract

This paper presents a method to automatically calculate texture maps for a given three-dimensional object out of a sequence of images. It is used in our image-based modeling approach after the registration of the images and the geometric modeling is done. We show that the presented method uses the information from all images by implicitly applying a weighting function. Using this approach the consideration of modeled occlusions as well as the detection and removal of non-modeled occlusions is accomplished. The final resolution of the texture maps can be adjusted on a pixel/cm basis.

1 Introduction

The reconstruction and modeling of real-world objects is becoming an increasing research area in both the computer graphics and the computer vision community. The virtual models, models of our urban environments or cultural heritage, are used for virtual tourism, advertising, entertainment, and simulations. Realistic-looking virtual models strongly depend on the usage of texture maps. In our approach we concentrate on the reconstruction of texture maps from real images instead of using repetitive patterns. The appearance of the 3D models is much more natural and makes the objects distinguishable.

In the next section, we describe related work to our approach. The third section explains the core algorithm used to calculate the texture maps. The results of the introduced method are shown in the fourth section. The conclusions and an outlook into future work are given in the fifth section.

2 Background

The problem of capturing multiresolution textures arises from an initiative called *City scanning*. Thereby an image-based approach to reconstruct the geometry of buildings is used first by means of photogrammetric modeling utilizing multiple views (see Debevec et al. [2] for reference). The main contribution of this work is to create textures out of these multiple views as accurately as possible. Accurate in this sense means without disturbing occlusions or visibility artifacts, and the resulting texture should expose the highest geometric resolution available in any region.

Related research is done in the following areas:

Bidirectional texture function (BTF): A group of

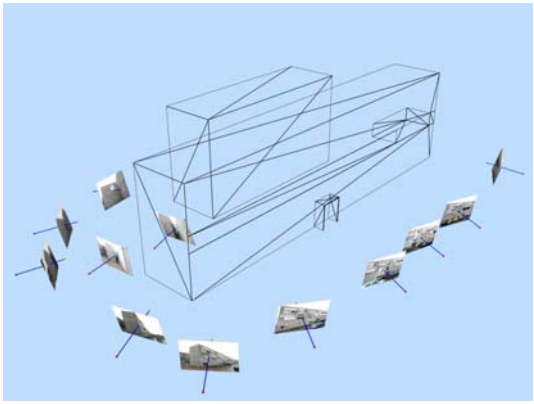
methods to store reflectance characteristics, surface detail, and surface displacements implicit in the data representation. Dana et al. [1] are doing that by capturing images from sample surfaces under many different lighting and viewing conditions. Having a large database of textures from different views under different illumination the BTF is established to represent the surface properties. Since this representation collects a large amount of data from the images and because there is a lot of redundancy in the data Suen and Healey [5] use a more compact representation to hold the BTF. Even if the method captures surface properties at different scales in an elegant way, there is no direct capability to remove occlusions automatically and to preserve high geometric resolution.

Mosaicing. A couple of methods can be found in the literature focusing on the automatic stitching of image sequences [7],[6],[3]. All of them exploit image correspondences to reconstruct the geometric transformation between them. As before there is no occlusion handling and preservation of resolution. The main application is image-based rendering.

View-dependent textures. The method presented by Debevec [2] avoids the reconstruction of an unique texture out of multiple images by doing this reconstruction during the rendering phase. The objects in the scene are textured with all of the images from the data acquisition. To use the best texture available alpha blending is used additionally. The alpha values are calculated according to the angle between the viewing direction of the synthetic scene and the viewing direction of the original image orientation. The smaller this angle the more likely it is that the texture represents the correct surface detail of the object.

The most related work to ours is presented by Ofek et al. [4]. Image sequences from a video stream are used as input for the texture reconstruction. All images are registered by hand. A multi-resolution representation is used to achieve a high geometric resolution at any region of the texture. Multiple entries are used to eliminate artifacts due to highlighting.

Our method uses a similar data structure but differs in the semi-automatic photogrammetric modeling step, the integration of visibility, and the occlusion handling of non-modeled objects.



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

(b) *Goal:* reconstructed textured model of the building.

Figure 1: *Problem description:* input data for our approach versus output data.

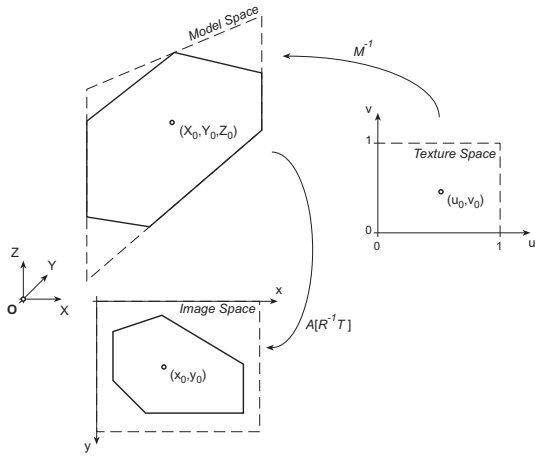


Figure 2: Geometric relation between texture, model, and image space.

3 Image-based Texturing

3.1 Geometric Transformations

The geometric relation between an image of an object, the three-dimensional object itself, and the texture map is described by two transformations. Figure 2 shows these two transformations between texture, model and image space. Texture coordinates are transformed into model coordinates by the inverse texture mapping function $\boldsymbol{M^{-1}}$. This function is often accomplished by setting up a pointtopoint transformation (texture points $[u_i\ v_i]^T$ are assigned to 3D model points $[x_i\ y_i\ z_i]^T$). For example, the VRML standard uses this kind of transformation for indexed face sets. There are, of course, other texture mapping functions available, but for the sake of simplicity and without losing generality, we want to focus on this one in the following discussion.

The second transformation is the well known projective transformation. Using the computer vision notation this transformation can be subdivided into an affine, a perspective and a Euclidean transformation. The affine transformation, also known as the interior orientation of a camera, is calculated during the camera calibration. The extracted parameters are the focal length c, the principal point $\mathbf{c_p} = [x_0\ y_0]^T$, and the coefficients for the lens distortion model. For a more detailed explanation see Zhang [8]. We store the affine transformation together with the perspective transformation in matrix $\boldsymbol{A}$:

$$\boldsymbol{A} = \begin{bmatrix} c & 0 & x_0 & 0 \\ 0 & c & y_0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{1}$$

The exterior orientation is represented by the Euler angles ω, φ, κ and the translation of the origin of the world coordinate system into the center of projection $\mathbf{C} = [X_0\ Y_0\ Z_0]^T$.

$$\boldsymbol{R_{\omega,\phi,\kappa}} = \boldsymbol{R_\omega R_\phi R_\kappa} \tag{2}$$

extending $\boldsymbol{R_{\omega,\phi,\kappa}}$ to be a 3D homogeneous matrix:

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

and

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & X_0 \\ 0 & 1 & 0 & Y_0 \\ 0 & 0 & 1 & Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

A homogeneous point on the image plane $\tilde{\mathbf{x}} = [\alpha x\ \alpha y\ \alpha\ 0]^T$ which is a projection of the homogeneous 3D point $\tilde{\mathbf{X}} = [X\ Y\ Z\ 1]^T$ by a pinhole camera is then given by

$$\tilde{\mathbf{x}} = \boldsymbol{A}[\boldsymbol{R}^{-1}\ \boldsymbol{T}]\tilde{\mathbf{X}} \tag{5}$$

3.2 Reconstructing Textures from Multiple Views

The overall method of modeling a real-world object from multiple views is out lined in Algorithm 1.

Algorithm 1 Method outline.

1: camera calibration
2: capture images from different views
3: registration of all views and reconstruction of the 3D model
4: **for all** polygons of the reconstructed model **do**
5: build a quadtree using all views
6: extract a texture with the desired resolution
7: **end for**

Step one of the method is to calibrate the camera, where we use an algorithm introduced by Zhang [8].

The outcome of the algorithm consists of the intrinsic parameters and parameters describing the lens distortion. This is an off-line process and has to be done only once for a fixed camera lens setting. Next all necessary views to reconstruct the model and the texture have to be taken. Registration and reconstruction are carried out utilizing the images from the calibrated camera and photogrammetric modeling.

Since photogrammetric modeling is not the main topic of this work, we will focus on the interface between model reconstruction and texture reconstruction only. The input data of our approach is the following:

- A geometric representation of the scene.
- For each view:
  - *The camera orientation:* being the intrinsic parameters from the camera calibration (focal length c and principal point $\mathbf{c_p}$) and the exterior orientation of the camera from photogrammetric modeling (rotation $\boldsymbol{R}$ and translation $\boldsymbol{T}$).
  - *The digital image:* as it is taken by the CCD camera and resampled using the lens distortion parameters.

The reconstruction of texture maps from multiple views needs the information concerning the positions of a pixel in texture space in all images. This information is present in the transformations from texture to model and further on to image space. The relation between model space and texture space is simply a coordinate system transformation if polygons are used. Thus, texture coordinates are first transformed by the inverse mapping function ($\boldsymbol{M^{-1}}$) followed by the perspective projection defined in the previous section ($\boldsymbol{A}[\boldsymbol{R}^{-1}\ \boldsymbol{T}]$). For each *image-polygon-texture* relation a transformation t_{ij} between the texture T_i of a polygon and an image P_j can be found. A texture coordinate ν corresponds then to the image coordinate $\mathbf{x}$ by the following relation (see Figure 2):

$$\mathbf{x} = \boldsymbol{A}[\boldsymbol{R}^{-1}\ \boldsymbol{T}]\boldsymbol{M^{-1}}\nu \tag{6}$$

3.2.1 Data Structure

We decided to use a quadtree representation to store our multi-resolution textures instead of image pyramids. Quadtrees offer the advantage of different depths for each subtree, which is useful considering different resolutions in the texture space due to perspective distortion and occlusions.

As outlined in Algorithm 1 for every polygon in the scene a texture will be generated. Taking into account the density considerations mentioned above we start building up a quadtree Q for one of those polygons. The root node is the highest quadtree level and represents the whole texture space. A refinement of a quadtree node splits the covered area of this node into four quarters, which are referred as children of the parent node. As long as there is information available in the images nodes of the quadtree will be refined. The leaves which are the lowest quadtree nodes represent then the highest resolution of the texture.

We start with a quadtree consisting of one node only. Successively each of the n images $P_1, \ldots, P_n$ are integrated into the quadtree using the transformation $t_i : T \rightarrow P_i$ which relates texture coordinates to image coordinates. The resolution for each texture point, which is represented by a node of the quadtree Q is determined applying transformation t_i. The quadtree nodes are expanded according to the texture resolution observed in the image P_i. After gathering all the information from all images this information has to be distributed between the quadtree levels. This enables the extraction of an image texture with a fixed resolution and will suppress noise in the high-resolution portions of the texture.

Node insertion is done by recursively applying Algorithm 2 to all quadtree nodes with a lower texture resolution than the corresponding image resolution. The texture resolution is calculated by projecting the four corners of the quadtree node into the image space and evaluating the area of the projected rectangle in pixels. As long as this area is greater than one the nodes are expanded.

Algorithm 2 Recursive node insertion.

```
1: project corner of node into image (applying transformation t_i)
2: calculate area and estimate resolution
3: if area > 1 then
4:     subdivide node (recursively)
5: else
6:     color ← P_i(x)
7:     certainty ← area
8: end if
```

It is clear that only nodes which can be projected into

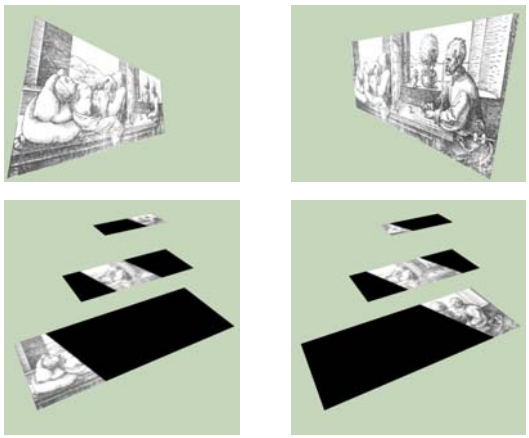
Figure 3: Synthetic scene used for quadtree construction. A simple polygon and two views are used.

a valid model space are further expanded. This is simply performed by an inside-out test for all corners. For calculating the area of the projected rectangle various metrics can be used. Since quadtree generation is an off-line step we calculate the correct area of that quadrilateral but if performance is a critical point other criteria such as the longest diagonal can be used.

Color and *certainty* are the two important fields of the data structure. The first one (*color*) represents straightforwardly the texture point's color and is set by the pixel color of the corresponding image. Due to perspective distortion the scale can slightly vary between texture points when inserting color information to the node. Additionally small shifts between the center point of the texture pixel and the image pixel occur. Therefore a bilinear interpolation in image space is carried out to set the texture point's color. The second field, *certainty*, holds the area of the projected rectangle of one node and is an indicator for the resolution of the used image part. Usually this field is about one during color insertion unless there is no limitation of the quadtree level due to implementation considerations. This field is necessary for the inter-level information distribution later.

To verify the quadtree construction and to understand the relation between views, 3D model and texture we start with a simple scene consisting of one polygon only and two views as presented in Figure 3. Each view is represented by an arrow originating at the center of projection representing the optical axis, and a texured rectangle representing the projection plane. The projection planes are textured with the corresponding image from the camera.

Figure 4 shows levels 6–8 of the quadtree after inserting the image from the left and the right view respectively. As can be seen in the upper pictures different resolutions appear within one view as a result of the perspective distortion of the polygon. This leads to a splitting of the input image into three parts, which



(a) Left view. (b) Right view.

Figure 4: Input image and corresponding quadtree (levels 6–8).

can be seen in the lower pictures for the right and the left view. Parts of higher resolution lie within lower quadtree levels and vice versa. Collecting the information of both quadtrees and distributing the information between all levels yields a multiresolution texture.

A straightforward way to combine the information of the images contained in the quadtree is to average the entries within all levels. Unfortunately entries representing a lower resolution will have the same weight as entries with higher resolutions. Therefore the already stored certainty is used to weight the entries during the color collection. This guarantees that entries of the quadtree that cover a larger area are more dominant than entries that cover smaller areas, which preserves regions of higher resolution. Thus steps 6 and 7 of Algorithm 2 are adapted to the following:

$$\text{color} \leftarrow \frac{\text{color} \cdot \text{certainty} + P_i(\mathbf{x})}{\text{certainty} + \text{area}}$$
$$\text{certainty} \leftarrow \text{certainty} + \text{area}$$

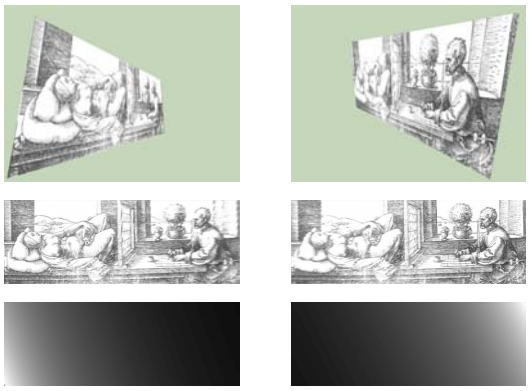
After all images P_i are collected the information is distributed between the levels by propagating the entries up and down. This will fill sparse regions of the quadtree and additionally suppress noise in high-resolution regions. Algorithms 3 and 4 are called with the root node of the quadtree as argument and additionally the value zero for the second algorithm.

Algorithm 3 PropagateUp(node)

1: **for all** children C_i **do**
2: PropagateUp (C_i)
3: color $\leftarrow \dfrac{(\text{color} + \text{color}(C_i)) \cdot \text{certainty}(C_i)}{\text{certainty} + \text{certainty}(C_i)}$
4: certainty $\leftarrow$ certainty + certainty(C_i)
5: **end for**
6: average $\leftarrow \dfrac{\sum_i \text{color}(C_i) \cdot \text{certainty}(C_i)}{\sum_i \text{certainty}(C_i)}$
7: **for all** children i **do**
8: color $\leftarrow$ color - average
9: **end for**

Algorithm 4 PropagateDown(node, parentColor).

1: color $\leftarrow$ color + parentColor
2: **for all** children C_i **do**
3: PropagateDown (C_i, color)
4: **end for**



(a) Left view. (b) Right view.

Figure 5: Input image, corresponding texture, and certainty map.

Reusing the scenario shown in Figure 3, the results of applying the above algorithms after insertion of one input image into the quadtree are presented in Figure 5. Again the upper pictures represent the input images. Since information is propagated through the quadtree we can find a level where every region is occupied by image information. Thus a texture can be extracted which is shown by the pictures in the middle. When comparing the extracted textures for the left and the right view it is clearly visible that the resolution changes from left to right. The lower pictures, which show the corresponding certainty maps of the same quadtree level as the extracted textures, exhibit the same circumstance. At regions with high resolution the corresponding certainty map contains higher values and vice versa. This can be easily verified since higher certainties do have a larger projected area in image space respectively.

Until now image insertion and interlevel distribution are carried out for each image only for demonstration. The results when inserting images from all views first into the same quadtree and propagating the information inserted afterwards are presented in Figure 6. The scenario and the input images are the same as in Figures 3 and 4. Again the extracted texture which represents one quadtree level and the corresponding certainty map are shown. The resulting texture exhibits the benefits from both views. The lower image of Figure 6 shows the resulting certainty map. Higher certainties on both sides of the map are achieved. Also a lower certainty in the middle of the map can be recognized although information from both views is collected. This is due to the nonlinearity of the certainty caused by perspective distortion. Neither the left nor the right image represents the middle of the texture at a high resolution.

3.3 Integrating Visibility

Until now scenarios are selected where neither occlusions from other models nor self-occlusion occur. In other words this means that the whole texture is seen from each viewpoint.

Typically views from real-world scenes expose occlusions, which means that parts of the texture or the whole texture under investigation is not visible from that viewpoint. This can be for different reasons:

1. Self-occlusion:
   some other parts of the same object occlude the interesting surface patch.

2. Occlusion from other modeled objects within the scene:
   a *modeled* object is stored in the scene database from the photogrammetric modeling step.

3. Occlusion from not modeled objects:
   these objects are not modeled since they are geometrically to complex or are unimportant.

Figure 6: Texture and certainty map after information fusion.

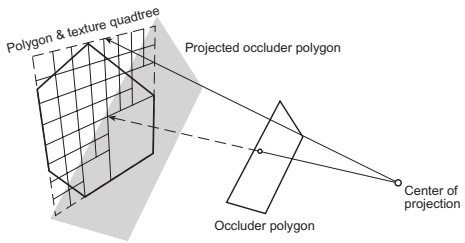


Figure 7: The influence of occluder polygons to the quadtree structure.

Self-occlusion and occlusion from other modeled objects can be eliminated by integration of visibility considerations. The visibility of a quadtree element (i.e., a node) is calculated by shooting rays from every corner of that node $\mathbf{P}$ to the center of projection $\mathbf{C}$.

Step 1 in Algorithm 2 is therefore adapted to additionally perform the visibility test. The second modification concerns the subdivision step 4. This recursion is carried out unless one of four corner points of the node is visible. Starting with the root node of the quadtree this tree will be refined until the highest resolution within the input image is reached or the corresponding part of the quadtree element is not visible from that view.

Figure 7 shows the influence of an occluding polygon on the structure of the quadtree. Two exemplary rays from the corners to the center of projection are shown. One of them hits the occluder polygon and the other one goes through.

3.4 Occlusion Detection

Up to now, we assume that every pixel which is collected in the quadtree represents the correct diffuse reflectance coefficient scaled by the intensity of the ambient illumination. Pixels that pass the visibility calculation step will contribute to the corresponding quadtree node according to their calculated certainty. Differences in pixel color for a particular quadtree node are assumed to relate to differences in resolution. Since quadtree entries are weighted according to the area in the image space it is guaranteed that higherresolution information is preserved.

If artifacts from occlusion due to nonmodeled objects appear, we have to use outlier detection mechanisms. One commonly used method is the employment of a median filter. To be able to apply a median filter we have to store all entries of the quadtree coming from all input images separately in a list first. Thus the color field in step 6 of Algorithm 2 is no longer a scalar value. Also the certainty field used in step 7 must be stored in the same way as colors. Using the list of color entries the median is calculated after all images are collected in the quadtree. This median and the corresponding certainty then represent the particular quadtree node. The interlevel distribution to propagate the information between the sparsely occupied quadtree level can then be performed as outlined in Algorithms 3 and 4.

It is straightforward that an outlier can be eliminated having at least three entries. Since it is hard to predict

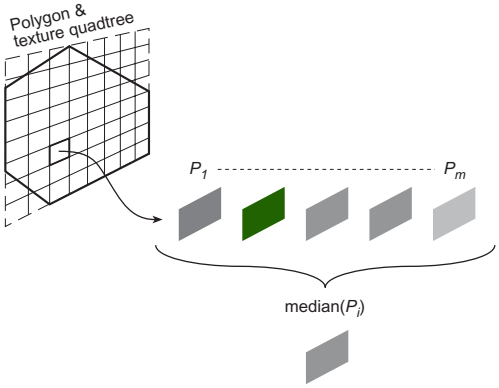


Figure 8: Pixel stack for one quadtree node and occlusion detection principle.

the number of entries and the number of outliers for all quadtree nodes during the image acquisition step, capturing more than three images for part of a texture is a good idea. One of the possible extensions of the method is to use the number of entries per quadtree node to make a plan for the next view during the image acquisition step. However, this requires an on-the-fly reconstruction of the geometric model and registration of all views.

The median is calculated using color information only. Thus, it can happen that an entry with a lower certainty outperforms an entry with nearly the same color and a much higher certainty. To overcome this error the mean of all within a certain distance from some threshold from the median is calculated. Typically we use 10% of the median as threshold. The mean color value is calculated using the stored certainties to weigh the entries. This allows the contribution of entries with high certainty to the final color and certainty field of the quadtree node, even if they are not the actual chosen median.

As introduced in Ofek et al. [4] the use of median filters has the capability to eliminate highlights from textures as well, since highlights can be seen as outliers within the quadtree generation too. Nevertheless outdoor scenes expose diffuse reflectance characteristics in most of the cases. Therefore the highlight elimination capability is more important for indoor scenes. This can be verified in the pictures shown in the 'Results' section.

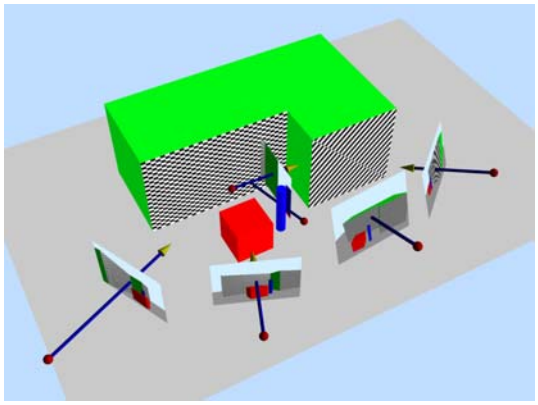
A test scenario is shown in Figure 9. It consists of an L-shaped box with two textured faces. The face with the checkerboard texture is our texture under investigation. In front of this face is a modeled box and a 'non-modeled' occlusion in the form of a cylinder. This test scenario is composed to demonstrate the following algorithmic features:

- *High resolution through multiple views:* From each view the parts which results in a high resolution texture have a higher contribution to the quadtree than the parts with lower resolution. This is achieved by weighting the entries by the projected node area.
- *Correct visibility calculation:* Parts of the texture are occluded by other objects (the box) and by the object itself. The visibility calculation will mask out these regions for the quadtree generation.
- *Occlusion detection:* The cylinder hides parts of the texture at different regions in different views. There should be sufficient viewpoints to eliminate occlusion by the proposed algorithmic extension.

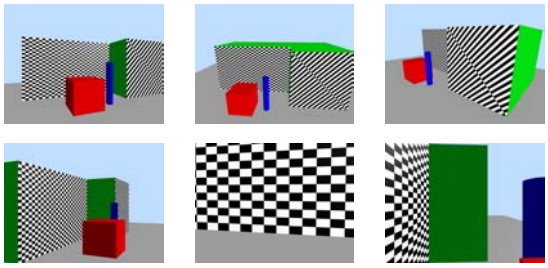
Figure 10 shows the results for the checkerboard texture without and with occlusion detection. The upper picture on the left does not expose artifacts from the box, which is the expected result of a correct visibility calculation. On the other hand the cylinder is mapped into different regions as it is projected at the texture at different viewpoints. In Figure 10 these regions are referred as regions A and B. The lower picture in Figure 10 shows the result of the final algorithm including all items mentioned above. Although occlusions from the cylinder are eliminated in region C still there exists a grayish impression. This indicates insufficient information for the median filter to work correctly. Additionally in region D a higher resolution than in other parts can be noticed, which results from view number five. Looking at the scenario in Figure 9 it can be seen that this viewpoint is closer to the reconstructed texture than any other viewpoint in the scene.

3.4.1 Influence of the Number of Entries per Quadtree Node.

Since the median computations only make sense for sample rates greater than or equal to three, the number of entries actually used for each quadtree node is of particular interest. Therefore in Figure 11 a map of the entry counts is given. The number of entries per quadtree are directly mapped to intensity values. Higher intensity

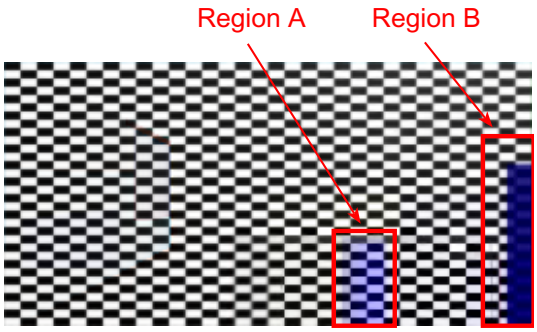


(a) Test Scene.

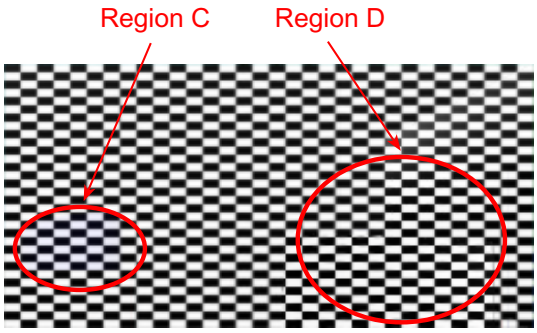


(b) Views from all 6 camera positions.

Figure 9: Scenario for occlusion detection test.



(a) Without occlusion handling.



(b) With occlusion handling.

Figure 10: Reconstructed textures (interesting regions outlined).

Figure 11: Visualized numbers of entries for occlusion detection. Higher intensities represents a higher number of entries.

means higher entry count. This allows easy detection of critical regions. Critical means that if three entries are unavailable the median does not make sense. For that cases only the certainty which represents the resolution of the entry in image space controls the contribution. One extension to the method will be the integration of these maps into the image acquisition step to plan additional views until all texture regions are generated with at least three quadtree entries.

4 Results

4.1 Outdoor Scene

The main motivation of the proposed method is to relight the reconstructed virtual model under different lighting conditions and to generate a synthetic image from a new viewpoint.

Figure 12 shows the relighted model reconstructed from the scenario shown in Figure 1. The building belongs to the new campus area at Graz University of Technology. Twelve views are used to reconstruct all of the textures. Directional illumination is applied to simulate direct sunlight, which can be noticed by the cast shadow casting. The ground floor is an augmentation and modeled by hand for visualization issues. Although the picture in Figure 12 is quite promising some limitations due to the imagecapturing process and the method itself are apparent.

In Figure 13 artifacts from lighting and different sensor sensitivity are visible. Only a few images correspond to the back view of the building. Additionally these views have different viewpoint object distances, which influences the corresponding certainty and causes the sharp intensity transitions of those textures. Some parts of the edges of the building show a high intensity. This is simply a mapping of the sky out of the input image and reflects errors from the geometric modeling.

Figure 12: Relighted virtual building.

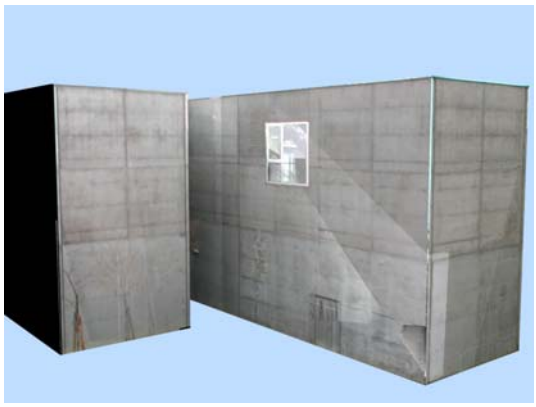


Figure 13: Artifacts due to different lighting and sensor sensitivity.

The limitations discussed so far can be canceled by being more careful in practical issues like improving the registration of images as well as using the same daylight and sensor sensitivity to capture all the images.

Figure 12 on the other hand, expresses the ability to *stitch* texture patches together. Even if the front side is too long to be captured in one image with a reasonable resolution, a highresolution texture can be extracted. Furthermore no artifacts at the borders of the texture patches can be observed. Another important feature of

the method is the occlusion elimination of nonmodeled objects as for example the tree in Figure 17 on the left side or the traffic sign and the vehicles on the right side. The upper picture on each side shows a synthetic view of the reconstructed textured model. The three pictures below the synthetic view represent three out of twelve selected input images.

In every input image on the left side a tree maps on different regions of different faces. Since the geometry of the tree is very complex and occupies only a small portion of the whole scene this is a candidate for the occlusion elimination test. Even if artifacts on the resulting textures are visible the projections of the tree are suppressed.

On the right side multiple occlusions by nonmodeled objects are shown. The traffic sign which is prominent in the first of the three input images is suppressed in the resulting texture. On the other hand, the vehicles occlude wide areas of the lower regions of the front side of the building. Additionally the side walls of the entrance are seldom visible. Therefore too few, entries build up the corresponding quadtree node and further median filtering and averaging delivers the wrong result. Looking at highly specular surfaces like the windows, the collected intensity depends on the viewing direction and thus more different entries for one quadtree node can be expected. Nevertheless due to the weighted averaging it is more likely that input images that represent a higher texture resolution contribute more to the resulting texture. Therefore jitter in those textures is suppressed. Even if the resulting texture does not represent surface properties the windows act like mirrors for of the surrounding real world, which fits also to the virtual model.

4.2 Indoor Scene

Although the presented method is motivated by an outdoor problem the applicability to indoor scenes is of additional interest. Lighting simulation of virtual modified interiors can be used to plan the reconstruction as an example. In addition to the outdoor problems discussed in the previous sections indoor scenes show the following effects:

- *Global lighting:* Closed environments show internal reflections of the surface patches in combination with shadowing effects from directional lighting.

(a) Scenario including reconstructed textures.

(b) Some selected views (2 of 11).

Figure 14: Scenario and input images.

- *Mixed reflection characteristics:* Typically any combination of diffuse, directional-diffuse up to specular objects occurs.

Figure 14(a) is a reconstruction of a part of an office. A pinboard, a black cupboard, and a cardboard box form the scene. The fig tree on top of the black cupboard is only visible on the input images shown in Figure 14(b). Six out of eleven input images are selected to demonstrate the different scales and orientations of the views. As in the case for the outdoor scene the fig tree is geometrically too complex for photogrammetric modeling. The output of the method contains a texture for the pin board and textures to three sides of the cardboard box and can be seen in Figure 18. This experiment should verify the capabilities of the proposed algorithm in dealing with the following problems:

1. Preservation of highresolution texture information. This is of particular interest for the pinboard texture.

2. Correct visibility calculation. No parts from the cardboard box should appear with the pinboard texture.

3. Elimination of occlusions caused by geometrically complex objects, such as the fig tree.

4. Elimination of highlighting effects due to directional-diffuse reflection characteristics. The glossy papers on the pin board are examples for such a surface property.

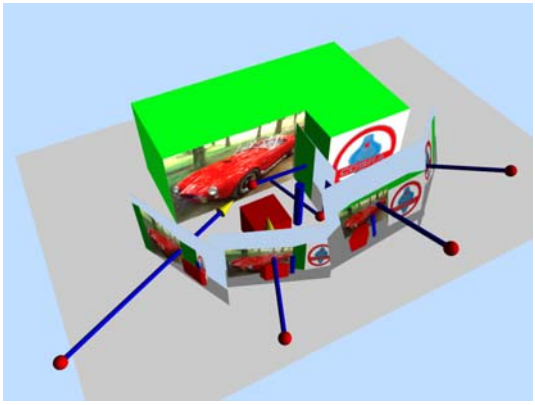
5. Sensitivity to global lighting. Looking at the input images internal reflections from side walls to the pinboard can be observed.

Observing the pinboard texture in Figure 18 preservation of highresolution information can be noticed in the textual parts of the posters. Since there is no influence from the cardboard box visible also the second problem is solved. Elimination of occlusions and highlights are both handled by the median filtering part of the algorithm. Concerning the occlusion elimination due to the fig tree some artifacts at the borders of the leaves can still be observed. Taking a closer look at the input images it is seen that those artifacts are caused by slight shifts of the color planes coming from the CCD sensor. Regarding the highlight elimination it can be seen that especially on the white glossy paper intensity increases due to specularity are suppressed. Still an open problem for further investigations are artifacts due to global illumination which can be observed in the yellow diffuse background of the pinboard.

4.3 Quantitative Evaluation

Up to now we have shown results of our method for two realworld scenes. Looking at them our method seems to perform quite well; however, we still want to obtain more matchable results in the mathematical sense. Unfortunately this is a hard task to perform, as the the original radiometric properties of realworld scenes are always modulated by lightinteraction. Therefore we use a photorealistic texture showing a car for the scene containing the L-shaped box shown in Figure 9 instead of the checkerboard texture. Even though the scene is still artificial it is a good approximation of a realworld scene, and in addition we know about the radiometric properties of the texture, so comparison is possible.

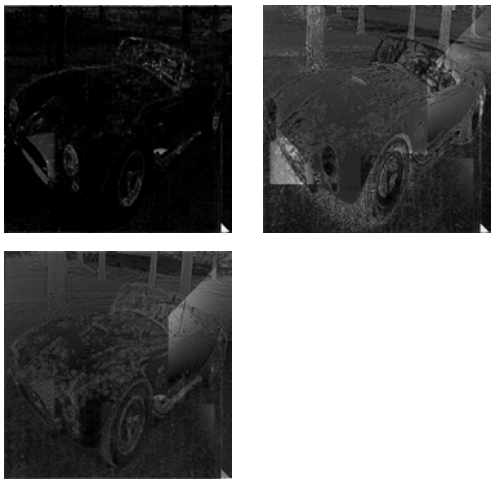
The whole scene used, the original texture and the texture reconstructed by our method are shown in Figure 15.



(a) Scenario for result comparison.

(b) Original texture and reconstructed texture

Figure 15: Scenario, original texture and reconstructed texture.



(a)

Figure 16: Comparison in HSV-space

It can be visually verified that there are no major differences between the two texture images. Figure 16 shows difference images for the H-, S- and V-channels of the images. The differences are scaled to project to the full scale of gray values ranging from 0 to 255, just to show in which parts of the texture our method performs well and where errors occur.

Errors due to an insuffient number of input images are mainly found in the H-channel corresponding to the colour, while intensity variations caused by lighting influence the S- and V-channel. Both cases can be observed looking at the difference images. On the one hand there are major H-differences near the lower right corner due to a small number of images (see also Figure 11). On the other hand the V-channel clearly shows the influence of the shadow thrown onto the the front surface of the box. Significant H-channel differences can also be found near the radiator grille of the car. They are caused by resampling the finegrained dark texture. The visually noticeable differences in this region are negligible. Table 1 gives the mean error of a single pixel and the standard deviation for each color channel: the numbers are based on H-values ranging from 0 to 2π and S- and V-values ranging from 0 to 1.

Table 1: Mean value and variance of pixel-difference.

Channel	$\bar{x}$	σ^2
H	0.01741	0.00456
S	0.05608	0.00379
V	0.03737	0.00179

(a) suppressed. (b) partially suppressed.

Figure 17: Two different situations for occlusion handling.

5 Conclusions

We have presented a method to reconstruct a textured virtual model out of multiple views. Photogrammetric modeling as preprocessing step provides the opportunity to integrate visibility considerations in the texture reconstruction phase. A quadtree representation is used to store the multiresolution information from all views. Preservation of high geometric resolution is performed by inserting the images at the corresponding quadtree levels. Occlusion handling of nonmodeled objects is carried out mainly by median filtering of multiple entries.

The applicability is tested for an outdoor and indoor scene showing the potential of the method. Besides the advantages and limitations discussed so far there are

Figure 18: The texture of the pin board from the scene shown in Figure 14.

some improvements which will be implemented in the future:

- Up to now our method has handled 3-d polygons only. An extension will be the usage of other primitives like cylinders, cones and spheres. The only modification needed is to adapt the transformation between the texture space and the image space to the new primitives.
- Directional illumination artifacts in the resulting texture may occur because of shadowing effects. Knowing the light source position would enable elimination of these artifacts.
- When directional illumination occurs artifacts from shadowing will occur in the resulting texture. One possible solution could be to estimate the light source position from shadow observation. The result of this calculation can then be used to eliminate shadows in the texture information.
- Since more than one pixel intensity exists for most of the texture elements the magnitude of specularity can be estimated and later used to discriminate between occlusion and highlight artifacts.

Additionally the results will be compared with other methods as, for example viewdependent texture mapping.

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

References

[1] K. J. Dana, B. van Ginneken, S. K. Nayar, and J. J. Koenderink. Reflectance and texture of real-world surfaces. *ACM Transactions on Graphics*, 18(1):1–34, January 1999.

[2] P. E. Debevec, C. J. Taylor, and J. 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.

[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] E. Ofek, E. Shilat, A. Rappoport, and M. Werman. Multiresolution textures from image sequences. *IEEE Computer Graphics and Applications*, 17(2):18–29, March-April 1997.

[5] P. Suen and G. Healey. The analysis and reconstruction of real-world textures in three dimensions. *IEEE Transactions of Pattern Analysis and Machine Intelligence*, 22(5):491–503, May 2000.

[6] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, 16(3):22–30, 1996.

[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.

[8] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *International Conference on Computer Vision (ICCV'99)*, pages 666–673, Corfu, Greece, September 1999.