3D Reconstruction of Natural Scenes with View-Adaptive Multi-Texturing

K. Mueller, A. Smolic, P. Merkle, B. Kaspar, P. Eisert, and T. Wiegand
Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut
Image Processing Department
Einsteinufer 37, 10587 Berlin, Germany
{smolic/kmueller/merkle/eisert/wiegand}@hhi.de

Abstract

We present a 3D reconstruction and modeling system that operates on a number of input photographs that show a natural scene. Approaches from computer graphics and image processing are combined and performance is shown via experiments. Furthermore, reconstruction quality is analyzed w.r.t. the number and distribution of textures, used for reconstruction. The reconstruction pipeline starts with image acquisition, which consists of a number of photographs of the scene that are sequentially taken at different positions. Since the photographs are not acquired concurrently, they are influenced by different illumination conditions that we mandate to be preserved in the final 3D representation. In the second step, object segmentation is applied and camera calibration provided. This allows the application of shape-from-silhouette approaches, namely a hierarchical voxel approach, where different resolution layers are organized within an octree structure. For applying texture mapping, the voxel model is transformed into a wireframe, which provides smoothing of the object's surface and also reduces the number of surface primitives. Finally, a subset of original images is mapped onto the 3D geometry to provide texture information. Here, view-adaptive multi-texturing is used to preserve natural illumination. Intermediate views are interpolated automatically using adaptive real-time weight calculations for original textures.

1. Introduction

This paper presents a system for 3D reconstruction of natural objects and scenes from a number of photographs. In our scenario, an object of interest (e.g. a building, sight) is captured from a number of different viewpoints, e.g. surrounding the object. Instead of using video cameras as done in related work, we focus on high-resolution high quality images taken with still image cameras. From the images we reconstruct a 3D model of the object that can be rendered from arbitrary viewpoints, i.e. providing the full functionality known from virtual computer graphics objects. Applications of the presented approach include cultural heritage conservation, virtual sightseeing, architecture studies, museums, etc.

In principle, there is a relationship between the number of input images taken and rendering quality. On the other hand, also the complexity and the effort for capturing increases with the number of input images. In our experiments, we therefore investigate the trade-off between rendering quality and the number of input images used for reconstruction and rendering.

Often the process of scene reconstruction is seen as a continuum between two extremes: modeling by classical 3D computer graphics one on side and image-based rendering from a typically very large number of images on the other side [5]. The whole process from image acquisition to rendering of the entire scene includes a number of stages to which different contributions were provided. One important stage is 3D geometry reconstruction from 2D segmentation masks and available camera calibration information [10], [4]. Here a shape-from-silhouette approach is utilized which yields a 3D voxel model. One possibility to limit the complexity and processing time of voxel reconstruction is the construction of a hierarchical octree structure [9]. In the octree construction process, the resolution of the 3D voxel model is refined with each stage by subdividing a relevant voxel into eight new sub-voxels. After creation of the voxel model the obtained geometry is often transformed into a wireframe to better approximate the original surface. Here, a marching-cube algorithm [6] is one possibility to obtain the outer object surface of the voxel model, followed by a smoothing and mesh simplification algorithm, like those provided by DirectX.

Another important aspect of 3D reconstruction is the coloring of the obtained 3D geometry. Here, two different classes have mainly emerged. The first approach assumes the voxel model to be fine enough to give each voxel a single color [8]. In this case, voxels represent color information similar to their 2D pixel counterpart. This form of coloring leads to an object appearance independent of the actual viewpoint in the scene. For the selected hierarchical octree approach, the assumption of a high-resolution voxel model and therefore small voxels is violated. In our approach, rather coarse voxels are also present which would exhibit a single color across the entire surface if voxel coloring would be applied. Therefore we select the second class of coloring approaches, the texture mapping. Here, images are mapped onto the object surface and a number of approaches have been

Proceedings of the 2nd Intl Symp 3D Data Processing, Visualization, and Transmission (3DPVT 2004)
0-7695-2223-8/04 $20.00 © 2004 IEEE

developed, which partly make use of hardware acceleration on graphic cards. Application programming interfaces like DirectX or OpenGL provide easy-to-use functionality for such support. Among the texture mapping algorithms are Lumigraph [1], Lightfield Mapping [2] and view-dependent multi-texturing [3]. The last method stands for texture mapping that depends on the scene viewpoint. On one hand view-dependent multi-texturing describes an initial texturing, where for each surface primitive of the 3D geometric object a texture patch is obtained by interpolation from a subset of nearest views, according to the orientation of the primitive surface normal vector [11]. During Rendering, the appearance remains constant. On the other hand, view-dependent multi-texturing describes the view-dependent interpolation during rendering and thus results in a varying appearance during scene navigation [12]. We suggest selecting the latter approach, since the natural illumination aspects are also visible in the final scene.

The paper is organized as follows: Section 2 gives an overview of the image acquisition and preprocessing steps and Section 3 describes the voxel modeling process. Section 4 presents the geometry model transformation, while view-dependent multi-texturing is shown in Section 5. Finally, Section 6 gives quantitative as well as visual reconstruction results of the proposed 3D reconstruction chain.

2. Image Acquisition

In the first step, camera images were taken around the scene. The location for image acquisition is Neak Pan, a small temple from the 9th century in the north part of Angkor, Cambodia. The temple consists of a cone-shaped tower on a wider cylindrical foundation in the center of a squarish water basin. Along the sides of the basin, 72 camera images were taken within a period of 2.5 hours. Fig. 1 shows the top view of the temple and camera positions along the object.

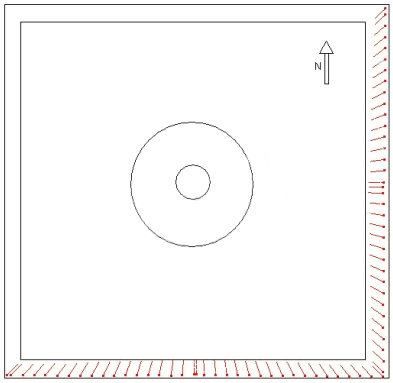

Fig. 1: Camera Positions along the temple, top view

From the original images, camera calibration is carried out to first obtain projection information for the 3D reconstruction process. Therefore, the calibration Software Boujou 1.3 was used to obtain the information. Here, areas are excluded from the calibration process, which are not part of the foreground scene. Intrinsic camera parameters need to be provided, including lense distortion, and the program calculates 3D positions of feature points, camera positions and thus extrinsic camera parameters. The results are presented in Fig. 2 with camera positions as darker arrows (red) and lighter 3D feature points (blue).

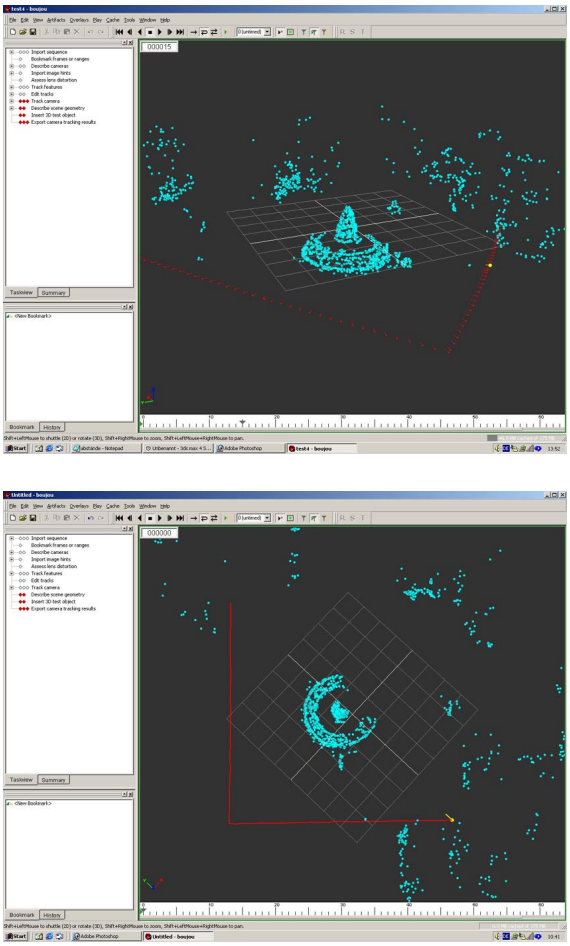

Fig. 2: Camera positions and projected points in 3D-space, side and top view

3. 3D Object Modeling

From the calibration data, projection matrices are calculated between the 3D object and 2D images for the 3D geometry reconstruction process. Among the class of shape-from-silhouette approaches a hierarchical voxel approach was selected which creates an octree structure

of the volume from the camera images. Since the algorithm requires silhouette information from all views, segmentation was carried out first. In this user-assisted approach, color-and-position-based automatic segmentation was applied first to create approximate segmentation information that was refined manually afterwards.

The octree voxel approach starts with one initial cube that is placed in the 3D scene. Its size is chosen big enough to cover the original 3D object completely, i.e. the cube represents the bounding cube of the object. This way, the projections of the cube into all silhouette views cover the scene completely. The next processing step applies recursive subdivision of the cubes starting with the division of the initial cube into eight sub-cubes. Each of the cubes is also projected into all silhouette views and one of the following actions is taken according to 2D silhouette analysis:

1. A cube that is completely inside the silhouettes of all views remains in the octree and is not subdivided further, i.e. it is completely inside the object to be reconstructed
2. A cube that is completely outside of at least one silhouette is omitted.
3. All other cubes are further subdivided to be processed in the next stage.

This process is applied recursively for remaining voxels and a finer model approximation is obtained after each stage.

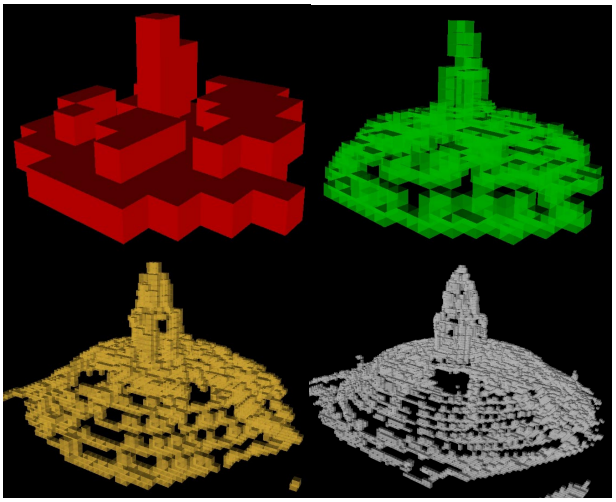

Fig. 3: Stages of different voxel resolutions during the octree generation process

Fig. 3 shows four stages (stage 5-8) from the reconstruction process. Here, only those voxels are shown, which belong to the actual stage. The object at a certain stage is therefore always the sum of all previous stages. The subdivision process is terminated, if a certain resolution is reached.

For better visualization, the final octree voxel model is shown in Fig. 4. Here the coarsest resolution is shown as opaque red cubes in the object center, while finer layers towards the object periphery are shown as transparent white cubes. The entire octree hierarchy is further used for the transformation processing.

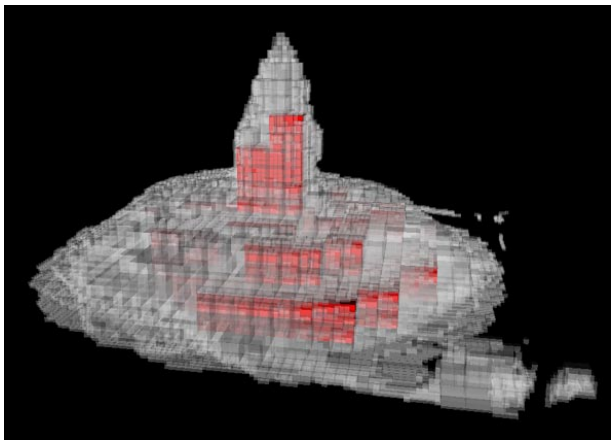

Fig. 4: Reconstructed octree model, coarsest voxel resolution red, finer resolutions transparent

4. Wireframe Transformation

At this stage, the voxel model already represents a 3D-geometry that could also easily be transformed into a wireframe. However, all surface patches are either parallel or perpendicular to each other and if texture mapping is applied at this stage, visible artifacts would occur. Furthermore, the number of faces is rather high in the voxel model, leaving significant redundancy. Hence, a transformation step is applied to the voxel model. The actual transformation consists of the surface extraction from the voxel model, which is carried out using the marching cubes algorithm [6]. This general approach extracts the outer faces of voxel cubes that are part of the surface. The result is a very dense mesh of the object surface. For smoothing and simplification, adjacencies among the extracted faces need to be analyzed. This functionality is already provided by mesh simplification procedures of DirectX. In this process, it is important to neglect the comparison of normal vectors for adjacent faces to allow mesh simplification across perpendicular faces, resulting from the voxel modeling. Finally, a reduced wireframe is obtained that is finally by the renderer in the visualization process. The initial voxel model and obtained wireframe are shown in Fig. 5

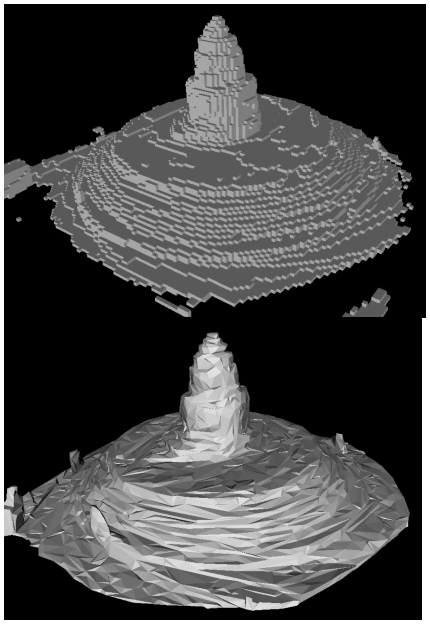

Fig. 5: Transformation of a voxel model into smoothed wireframe

5. View-Dependent Multi-Texturing

In the last step of the 3D reconstruction chain, a subset of original camera images is mapped onto the 3D geometry. The reason for using a multiple texture approach is twofold:

1. The visual quality is better, since individual illumination aspects from the input images are correctly represented. In computer graphics, illumination effects are often modeled using different light sources within the scene, however since the images were recorded at different time instances, lighting conditions change from one view to another. In connection with voxel algorithms, often a constant coloring is applied to each voxel. This leaves the scene with a rather static appearance; again, different illumination conditions are neglected. Furthermore, the constant coloring is not applicable to wireframe models with rather large faces.
2. There is hardware support for the renderer.

The general approach of rendering an object not only includes the texture, but often the normal vectors of visible faces. One example for this illumination is shown in bottom picture of Fig. 5 for the untextured model, where each surface triangle shows different light reflection according to the angle between its normal vector and illumination direction. The resulting illumination is simply derived from the dot product of these two normalized vectors. Here, the problem occurs that the 3D structure is not only visible in the untextured object but also in the finally textured model. To suppress such visible artifacts, lighting based on normal vectors must be avoided. Instead, of exploiting individual normal vectors for each triangular face separately, one camera vector is defined, which is applied to all faces. Thus, illumination remains constant across the whole object as if the object would be a plane. Since we have multiple textures, a camera vector for each view and therefore each texture is required. As a result, each single surface triangle is associated with the same set of camera vectors. In addition, maximum weighting of a certain texture is achieved, if the scene is observed from the associated original camera position. The result of the blending is shown in Fig. 6 where an intermediate viewpoint is shown with main influences from the two nearest views.

Fig. 6: Intermediate view of the textured temple model

When mapping multiple textures onto an object, static interpolation functions are available that initially apply a constant weight to each texture before rendering.

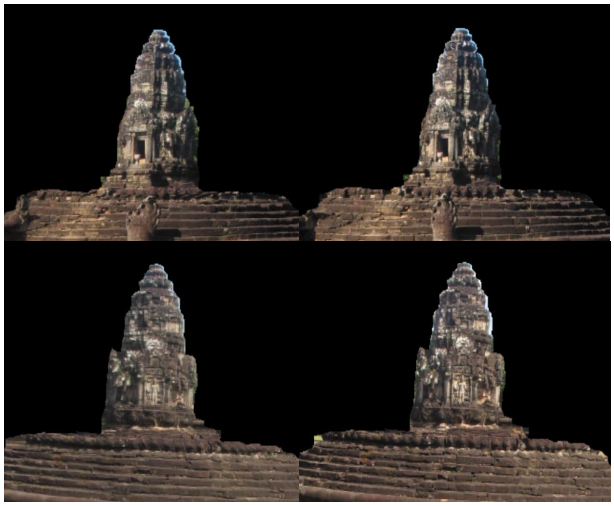

Fig. 7: Interpolated model (*left*) and corresponding original viewpoints (*right*)

To enable view-dependent texture weight calculation, we selected a special form that is carried out for each rendering cycle and hence results in view-dependent texture weights. The visual results for this method are shown in Fig. 7.Here, two intermediate views (left) are shown and for comparison, the original views (right) at the same position. Intermediate views at original positions are not shown, since they are identical to the original views. Detailed reconstruction quality results are presented in the next chapter.

For view-dependent texture interpolation, the unstructured Lumigraph rendering approach was applied [1]. Here, camera vectors are taken to obtain the direction from which an associated texture was taken. Let the camera vectors be denoted by C_i and current viewing direction by v. The angles between v and all C_i are denoted by θ_i. This setup is shown in Fig. 8.

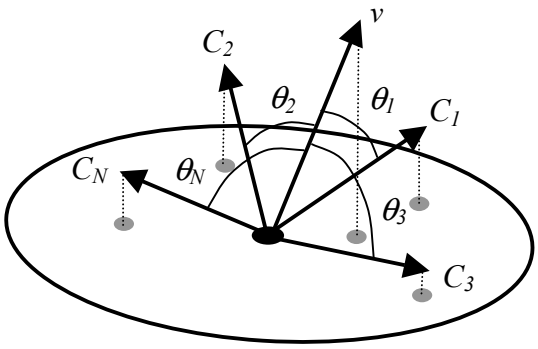

Fig. 8: Unstructured Lumigraph Rendering [8]: Calculate all $\cos\theta$ between viewing direction v and camera vectors C_i

Also note that only camera vectors in the hemisphere around v need to be considered, since textures with $\cos\theta < 0$ are clipped and therefore do not contribute to the final texture weights.

For the texture weighting, $\cos\theta_i$ is calculated for all remaining views first. Then relative weights w_i are obtained by the following function, as described in [12].

$$w_i = \begin{cases} \dfrac{\cos\theta_i}{1-\cos\theta_i}, & \cos\theta_i \neq 1 \\ Float\_Max, & \cos\theta_i = 1 \end{cases} \quad (1)$$

This function causes singularities for relative weights w_i at positions $\cos\theta_i = 0$, i.e. viewing direction and camera vector direction are identical. The weights w_i drop to all sides of the singularity and reach 0 if camera vector and viewing direction have an angle of $\theta_i \geq 90°$. From the relative weights w_i, absolute weights a_i are calculated by normalizing the appropriate values.

$$a_i = \frac{w_i}{\sum_{\forall i} w_i} \quad (2)$$

This approach results in absolute weights that guarantee constant lighting during rendering and smooth interpolation. Moreover, the original texture is shown when an original camera position is reached. The last condition results from the relative weights being nearly infinite at these positions. Due to the normalization, this weight is finally divided by the sum of all weights, which causes an absolute weight a_i of 1.0, while all other absolute weights are neglected. With all weights calculated, multi-view texturing is applied to the object and the scene is rendered. A multistage blending process is applied, in which one texture stage is fed with the actual texture as color argument 1 and the texture factor as color argument 2. This single texture stage is processed in a loop for each applied texture. Source and destination blending are set to 1 to finally guarantee the interpolated final texture $t(u,v)$.

$$t(u,v) = a_1 t_1(u,v) + a_2 t_2(u,v) + \cdots + a_n t_n(u,v) \quad (3)$$

This approach also guarantees texture interpolation, independent of the processing order and is currently being included into MPEG4 AFX [7].

6. Experiments

The final reconstructed object was tested using PSNR measurements for intermediate views vs. original camera views. Furthermore, visual quality is shown and the limits for PSNR-usage are discussed.

6.1. Quality Comparison Results

For qualitative evaluation of the applied view-dependent multi-texturing object reconstruction, a comparison between 3D object and original 2D views was carried out.

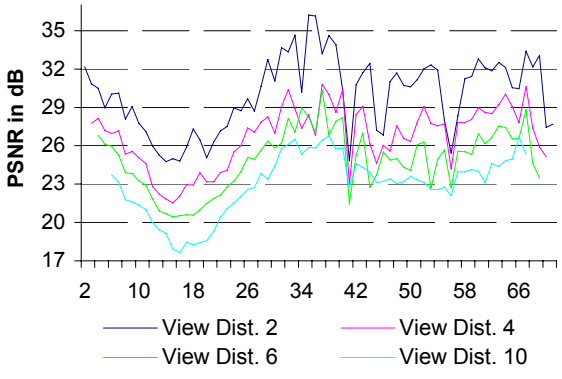

Fig. 9. Intermediate view reconstruction from adjacent cameras with different view distances

Therefore, 2D views are created from the object for all original camera positions. In the first experiment, a benchmark was created to which the algorithms can be compared. Furthermore, this experiment shows, which PSNR values are to be expected, see Fig. 9. For comparison, the object was created with specific textures: If an original camera view n was to be compared with the object, adjacent textures n-1 and n+1 were taken for texturing the object. Here, n stands for the actual camera or original image number. In Fig. 9, this is expressed as a view distance or texture sub-sampling factor of 2. Similarly, view distance 6 means, views n-3 and n+3 were taken, and so forth. These results show, in which range PSNR values are to be expected: First, the reconstruction quality decreases with texture distance from original views, since common surface information is less and less available. Furthermore, the reconstruction quality changes along the camera views due to distorting effects: Shadows from small foreground objects, like the small stone statues in front of the temple have different direction for each view and cannot be modeled correctly at original positions. Moreover, the original images were acquired at different time instances and thus exhibiting different lighting conditions, which influences the reconstruction quality.

After the initial benchmark tests, an object was created using a fixed number of textures and compared to the appropriate reconstruction curve, as shown in Fig. 10. Note that the object geometry was created using the silhouettes of all views.

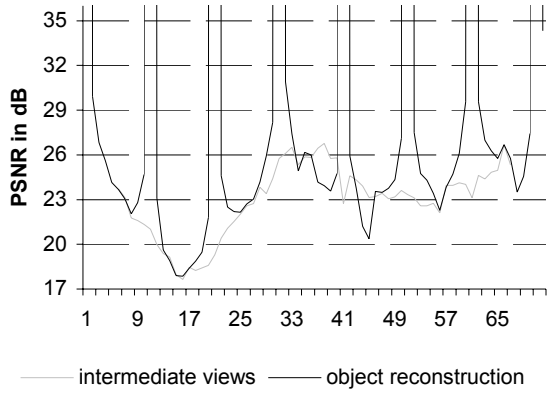

Fig. 10. Comparison of view reconstruction from adjacent views and multi-view object reconstruction

The gray curve in Fig. 10 represents the intermediate view reconstruction with a view distance of 10 views, whereas the black curve shows the results of the fully reconstructed object with 8 fixed views, also using a view distance of 10. The first specific property to observe are the infinite PSNR values at original positions, since the object shows original data at original viewpoints caused by the unstructured Lumigraph rendering, as described above. Beside these, two other quality ranges are present:

1. The full object has the lowest reconstruction quality in the middle between original views. Here, the obtained values are similar to those of the benchmark curve.
2. Between the minima of the quality measure curve and the singularities are the views with higher quality w.r.t. the benchmark, since more information of these views are available in the finally textured object.

To investigate the influence of view distance and similarly number of views in the reconstruction process onto the reconstruction quality, further tests were set up. Here, a statistical method was used, were for each value of "view distance", object reconstruction was carried out with all possible combinations of texture subsets, i.e. if view distance equals 3, original views 1,4,7…70 were taken as textures, followed by 2,5,8…71 and 3,6,9…72. Thus, average PSNR-values could be obtained for all views. In the averaging calculation, singularities were omitted. The results are shown in Fig. 11.

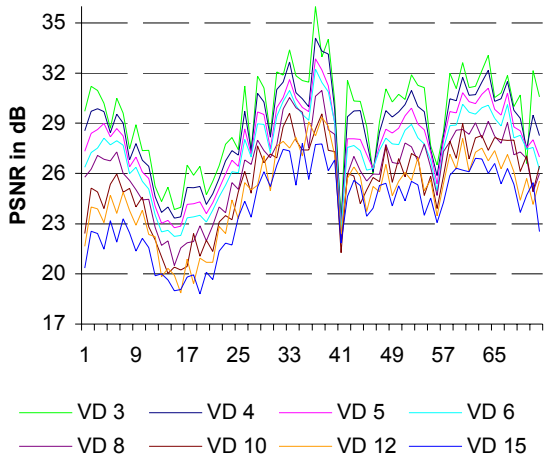

Fig. 11. Object reconstruction with different view distances (VD)

First, the values show similar characteristics in comparison to the benchmark curves, indicating, that the final object appearance is mainly interpolated from adjacent views. The second observation regards the dependence of PSNR from the view distance. Here the expected results show how the reconstruction quality decreases with increasing view distance. For better visibility, the average PSNR-values for each view distance are shown in Fig. 12.

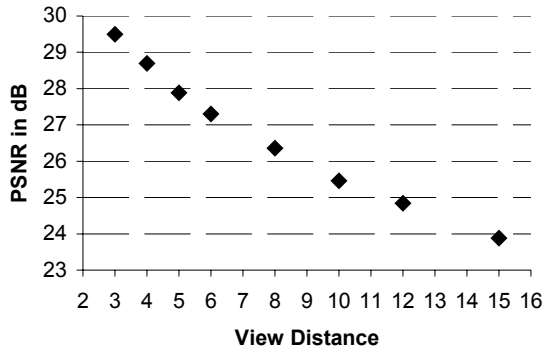

Fig. 12: Average PSNR for different view distances

6.2. Visual Results

After considering objective reconstruction quality, visual comparisons are provided to show some of the artifacts that occur in multi-texture modeling.

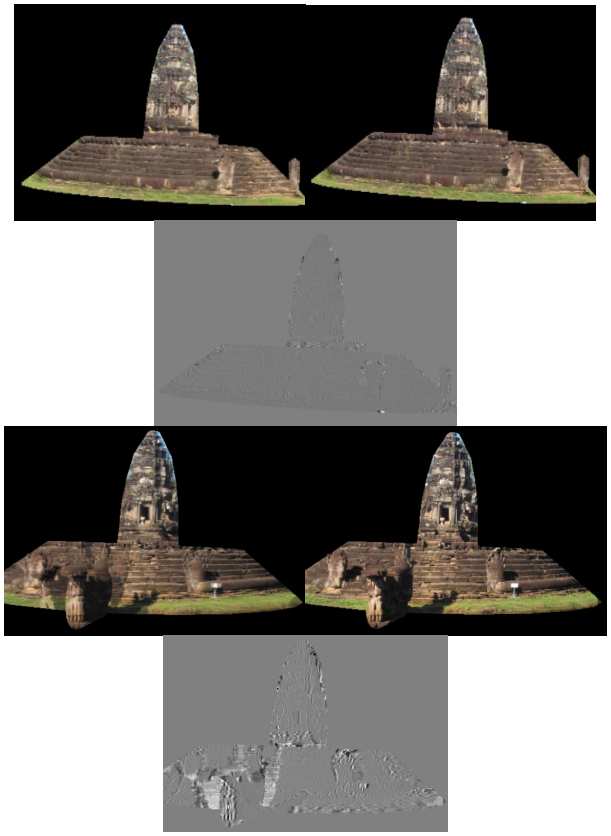

Fig. 13. Visible views (*left*), associated original views (*right*) and difference image (*bottom*)

In the example above, original and reconstructed images are compared. Additionally, the difference image is shown to highlight areas in reconstructed images deviating from the original.

The first comparison in Fig. 13 shows a well-approximated intermediate image on the left. Here, only small errors occur in highly structured areas. Even the small shadow from the small stone statue in front of the temple is represented correctly. The difference image also shows why relatively low PSNR values are obtained for good visual quality: If original and reconstructed textures are shifted by only one pixel, large differences occur which cause such low values. Therefore, the quality comparison had to be carried out against the introduced benchmark results.

The second example shows a worst-case interpolation with serious misalignments in foreground object texturing. Here a denser sampling of the original object would be required, but is not available. There remain a number of improvements for the algorithm, e.g. the obligatory denser object sampling and image acquisition under constant lighting conditions. However to provide a general Multi-texture modeling algorithm, such conditions exist and need to be considered. The advantage of view-dependent texture mapping is its independency of lighting effects, since differently illuminated textures are blended together according to the current viewpoint.

One improvement, currently under investigation, concerns PSNR-adaptive texture selection, where textures for modeling are selected according to the benchmark results from Fig. 9. Here, a denser texture set was selected in areas with low PSNR-values, while the sampling distance in high PSNR-value-areas was increased, to keep the number of used textures constant, as shown in Fig. 14.

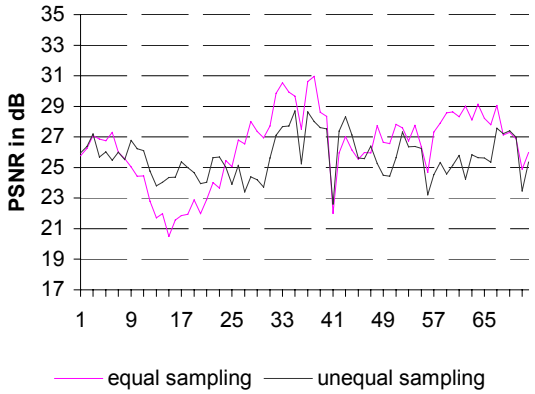

Fig. 14: Comparison of equal and unequal texture sampling using nine views

Although the average PSNR across all views remains the same for both cases, standard deviation could be decreased for the unequal texture sampling from 2.41dB to 1.38dB to achieve a more constant viewing quality during navigation through the scene.

7. Conclusions

In this paper, we have presented a 3D modeling and reconstruction scenario and have shown its usability for a cultural heritage scenario. The modeling process starts with the acquisition of images, which were obtained over a period of 2.5 hours and therefore have different illumination conditions. All images were segmented to extract the foreground data. From the images, calibration information had to be extracted in order to calculate projection matrices between 3D world and 2D images. With this information, an octree-based voxel algorithm was applied to create 3D geometry information. By using such a hierarchical approach, approximation of the synthesized model to all 2D contours is reached by repetitive subdivision of surface voxels only. This algorithm resulted in a 3D object surface with numerous small quadratic patches. For the application of texture mapping, the voxel model was transformed into a triangular wireframe and the number of vertices and faces in the frame was drastically reduced. This approach also provided surface smoothing of the perpendicular surface patches of the initial voxel model. Finally, texture mapping was applied to the 3D object. In our implementation, we selected a texture mapping with multiple views and view-adaptive texture weighting to preserve the original illumination conditions of each separate camera image. Texture weighting was carried out using an unstructured Lumigraph rendering approach to keep lighting conditions constant when navigating through the scene and also to obtain original views whenever the viewpoint reaches such positions.

Reconstruction quality of the presented approach was investigated, considering different number of views for the object reconstruction and providing quality measurements as well as visual results of intermediate views.

Currently, the algorithms are applied to a multi-view camera setting, using movie textures within the view-adaptive Multi-texturing stage. In addition, the 3D model is adapted at each time instance.

References

[1] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen, "*Unstructured Lumigraph Rendering,*" Proceedings of SIGGRAPH 2001, pp. 425-432, 2001.

[2] W. C. Chen, J. Y. Bouguet, M. H. Chu and R. Grzeszczuk, "*Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields*", Proceedings of ACM SIGGRAPH, pp. 447-456, 2002

[3] P. Debevec, C. Taylor, and J. Malik, "*Modeling and rendering architecture from photographs: A hybrid geometry- and image based approach,*" Proceedings of SIGGRAPH 1996, pp. 11-20, 1996.

[4] P. Eisert, E. Steinbach, and B. Girod, "*Multi-hypothesis, Volumetric Reconstruction of 3-D Objects from Multiple Calibrated Camera Views*", ICASSP, pp. 3509-3512, Phoenix, Mar. 1999.

[5] S.B. Kang, R. Szeliski, and P. Anandan, "*The Geometry-Image Representation Tradeoff for Rendering,*" Proc. ICIP2000, IEEE International Conference on Image Processing, Vancouver, Canada, September 2000.

[6] W. E. Lorensen, and H. E. Cline, "*Marching Cubes: A high resolution 3D surface reconstruction algorithm,*" Proceedings of SIGGRAPH, vol. 21, no. 4, pp 163-169, 1987.

[7] K. Mueller, and A. Smolic, "*Study on View-Dependent Multitexturing for MPEG-4 AFX,*" ISO/IEC JTC1/SC29/WG11, MPEG03/M10152, Gold Coast, Australia, October 2003.

[8] S. M. Seitz and C. R. Dyer, "*Photorealistic Scene Reconstruction by Voxel Coloring*", Proc. Computer Vision and Pattern Recognition Conf., pp. 1067-1073, 1997.

[9] R. Szeliski, "*Rapid Octree Construction from Image Sequences,*" CVGIP: Image Understanding, Vol. 58, No. 1, July, pp. 23-32, 1993.

[10] R.Y. Tsai, "*A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV camera and lenses,*" IEEE Journal of Robotics and Automation, Vol. RA-3, No. 4, August 1987.

[11] J. Visnovcova, A. Gruen, L. Zhang, "*Generating a 3D Model of a Bayon Tower Using Non-metric Imagery*", Proc. Intern. Workshop Recreating the Past – Visualization and Animation of Cultural Heritage, pp. 30-39, Ayutthaya, Thailand, 2001

[12] D. Vlasic, H. Pfister, s. Molinov, R. Grzeszczuk and W. Matusik, "*Opacity Light Fields: Interactive Rendering of Surface Light Fields with View-dependent Opacity*", Proc. of 2003 symposium on Interactive 3D graphics, pp. 65-74, 2003.