EUROGRAPHICS 2003 / P. Brunet and D. Fellner
(Guest Editors)

Volume 22 (2003), Number 3

Image Based Interactive Rendering with View Dependent Geometry

J.-F. Evers-Senne and R. Koch

Institute of Computer Science and Applied Mathematics, Christian-Albrechts-University of Kiel, Germany

Abstract

In this paper we present a novel approach for interactive rendering of virtual views from real image sequences. Combining the concepts of light fields, depth-compensated image warping and view dependent texture mapping, this plenoptic modeling approach can handle large and complex scenes. A portable, handheld multi-camera system has been developed that allows to record multiple image streams by simply walking around the scene. These image streams are automatically calibrated and depth maps for all views are generated as input to the rendering stage. For rendering a view dependent warping surface is constructed on the fly and depth-compensated image interpolation is applied with view-dependent texture mapping. Rendering quality is scalable to allow fast preview and to achieve high-end quality with the same approach. The system can handle large and geometrically complex scenes with hundreds of real images at interactive rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Viewing algorithms, I.4.1 [Image Processing and Computer Vision]: Digitization and Image Capture, I.4.8 [Image Processing and Computer Vision]: Scene Analysis

1. Introduction

One of the major goals in computer graphics is to display virtual worlds similar to real ones. For complex scenes, however, it is often not feasible to create them by hand with 3D construction tools. Even worse, those models are most often recognized as synthetic after just a few seconds due to the lack of realistic surface appearance. One well known approach to visualize complex scenes is *Image-Based-Rendering* or IBR. The idea behind it is to capture the appearance of a real scene with images and use this material to generate and display new virtual views of the scene. Modern CCD cameras allow fast and efficient capturing of the visual components of a scene (color, light), while the geometrical components are more difficult to obtain. The same is true for image-based rendering. In most cases it is obvious how to display the images, but geometrical information is needed for a correct synthesis of novel views. View-dependent local geometry information in form of depth maps can be computed from image sequences by either range data scanners or stereoscopic image analysis algorithms. However, due to incorrect camera calibration, difficult lighting conditions and non-static scenes it is often not possible to generate one globally consistent 3D model from hundreds of images and depth maps automatically.

In this work we will present a rendering system which generates view-dependent local geometry on the fly from multiple depth maps. For each new view the depth maps of the surrounding real views are fused in a scalable fashion to obtain a locally consistent 3D model. This geometrical representation is based on triangles and can then be textured with the images corresponding to the depth maps using hardware-accelerated techniques.

The first step in image-based rendering is the acquisition of images of the real scene from many different view points. Here we want to be able to scan the scene by simply walking around the area of interest and to automatically calibrate the cameras from the image data alone. To meet these requirements we have developed a flexible and mobile capturing system for efficient multiview recording in indoor and outdoor environments, using standard laptops and four battery powered synchronised cameras mounted on a rig. The synchronisation in conjunction with the rigid coupling of the cameras supports the calibration even for non-rigid and geometrically very complex scenes with occlusions. In an offline

© The Eurographics Association and Blackwell Publishers 2003. Published by Blackwell Publishers, 108 Cowley Road, Oxford OX4 1JF, UK and 350 Main Street, Malden, MA 02148, USA.

modeling step a set of dense depth maps is then computed from multi-viewpoint stereo analysis. These depth maps are then used as input to the proposed online rendering system for novel view synthesis.

In the next section an overview of related work will be given to help classifying this paper. In section 3 the multi-camera system and the necessary preprocessing of the images will be presented. Then, in section 4 the online rendering system based on images and depth maps is introduced. And finally in section 5 some results and conclusions are given.

2. Previous Work and Motivation

Image-based rendering is closely connected to the plenoptic function introduced by McMillan and Bishop in [12]. This function defines all radiance emitted from one point into every direction; for a dynamic scene the dimension of the plenoptic function is 7. Levoy and Hanrahan proposed in [9] an IBR system called Light Field which interpolated new views using a 4D representation of the plenoptic function (for a static surface). To approximate the plenoptic function a very dense mesh of images from cameras lying in a regular sampled viewpoint plane is used. Less dense sampling of the viewing space results in visual artifacts when interpolating between views. The Lumigraph introduced by Gortler et al in [4] uses a convex 3D shape approximation for depth compensated interpolation. They also suggested an approach to allow the usage of a hand-held camera and used "rebinning" to map the original images. However, this intermediate step of interpolation reduces the quality of the images.

View-dependent texture mapping (VDTM) is an alternative way of rendering visual effects from different views. In [2] Debevec et al. describe a real-time VDTM algorithm which uses hardware-accelerated projective texture mapping. For VDTM a consistent 3D model is required, which is not always easy to obtain. In 1999 Heigl et al. presented in [5] a plenoptic modeling approach based on the images from a hand-held camera. They used depth maps as a local representation of the scene geometry and corrected the interpolation of each ray by using this information. Looking up the color for each ray in the three surrounding cameras is similar to VDTM with thre blended textures. Buehler et al. in [1] proposed their *unstructured lumigraph rendering* which is a hybrid design between VDTM and light field rendering. Unlike VDTM, they do not rely on a high-quality geometric model, but they need a geometrical approximation of the scene. Also Pulli et al. in [15] described *view-based rendering* as a "method between purely model-based and purely image-based methods". Each real view consists of a colored range image, then several partial models are built and blended together.

All mentioned rendering techniques share one issue: They need approximate geometry information. To create 3D models from range images Pulli proposed a volume based approach in [14], but for more complex scenes this could be very hard or even impossible. Point based rendering systems like described in [17] are also useful to render from depth maps and images. But due to holes in the depth maps it is often necessary to fill holes in the local geometry. This problem is better solved by using interpolating surfaces as rendering primitives instead of points.

The other open issue common to most IBR systems is the acquisition of images. In the beginning, static grids with lots of fully calibrated cameras were used. Alternatively, motion control systems scan the viewing space with one single camera. This results in very dense sampling which is a good start for rendering, but which also results in large amounts of data. Levoy and Hanrahan first suggested a method for light field compression, since then this problem has been the focus of many publications[10, 16]. Gortler et al. [4] started using weakly calibrated hand-held cameras and used known markers for pose estimation. At last years SIGGRAPH Matusik et al. presented a system for "Image-Based 3D Photography" in [11]. They use several calibrated cameras, a turntable and rotating light sources as well as known background images displayed on a large plasma screen. With this system they are able to capture the appearance of very detailed objects including specular reflections and fuzzy material.

All these methods do not scale well with the size and complexity of the scene and they are often specialized to sample single objects in controlled environments. Koch and Pollefeys as in ,[8] and [13] used image sequences from uncalibrated hand-held cameras and *Structure From Motion* (SFM) algorithms. This approach scales well despite the fact that scanning a large viewing volume with one single camera is time-consuming even when using video frame rate. In addition, specular reflections, changing lighting condition (clouds), unsteady movement of the camera and dynamic scenes may cause the SFM algorithms to fail.

The rendering method proposed in this paper is similar to the mesh creation from Heigl et al. [5], but the construction of the underlying model is different. The resulting mesh is textured in realtime using VDTM similar to Debevec's approach, but because the geometry information of each surface area may result from up to three cameras, the texture is also chosen from these cameras.

3. Image Acquisition and Offline Geometric Modeling

In this section we describe the image acquisition and preprocessing modules. These modules operate offline and must be performed once for each acquired scene only. The steps are:

- Image capture,
- Camera calibration,
- multi-view depth estimation,
- preparation of depth samples for realtime geometric modeling.

© The Eurographics Association and Blackwell Publishers 2003.

3.1. Multi-Camera image capture

For fast and efficient scene acquisition we introduce a hand-held multi-camera system which is scalable for different scenes. In outdoor environment mobility is given by using standard laptop computers. Their limited performance result in a reduced frame rate. On the other hand in a studio environment the capturing system benefits from the high performance of modern PC systems with higher frame rates.

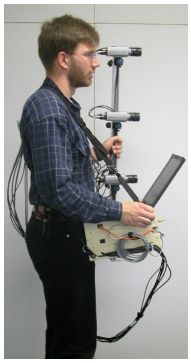

Figure 1: *Multi-camera rig with synchronized image capture*

A prototype system consisting of four digital fire wire (IEEE 1394) cameras and two laptops in a rack, which can be carried by the operator, has been built. The cameras are mounted on a pole with adjustable position and orientation of each camera, so any desired configuration of the camera poses on the pole can be used. The sensor can easily be moved by hand without a tripod, in particular allowing horizontal, vertical and rotational motion to scan the scene by simply walking by. The cameras are time-synchronized so that non-static scenes can be sampled by obtaining "times-sliced", synchronized shots of multiple cameras in which all movements in the scene are frozen for the timeslice. These frozen multiple views of the scene help obtaining 3D information and perform plenoptic rendering of non-cooperative scenes because limited motion in the scene can be tolerated. Different scenes may require a different viewing space. By adjusting the cameras on the pole or by using a pole of different length the viewing space can be adapted to fit the scene. Even more cameras can be added to enhance the quality of the sample mesh. Hence, a 2D scanning of the viewpoint surface is obtained with a single walk-by.

To handle data from four cameras with a mobile system, we decided to use two laptops each connected to two cameras. The laptops are synchronized via standard ethernet with a scalable network protocol [3] which serves two purposes: The distribution of parameters (e.g. exposure time) to all cameras, and capture synchronization in time. Using double-buffering and three threads per node, the resulting frame rate for color images of the size 1024x768 with four cameras and two computers is four fps. This is sufficient for dense view space sampling by a slowly walking operator. In a controlled studio environment, the synchronization protocol allows an arbitrary amount of computers to be connected and the cameras can operate at full frame rate.

3.2. Camera tracking and calibration

The images are acquired with a hand-held multi-camera system with arbitrary camera motion. Since we want to avoid putting markers into the scene, we have no control over the scene content. Therefore, all camera views are uncalibrated and the calibration and the camera track must be estimated from the image sequence itself. Furthermore, the scene is most likely uncooperative, meaning that complex geometry, massive occlusion and moving objects may hinder the calibration. The only control we have is the specific configuration of our camera setup and, to a limited extend, some knowledge about the performed camera motion. Since all cameras operate synchronously we know that ,for one instant, all cameras view a “time-frozen” static scene. We also know that from one recording timestep to the next, the relative configuration between the cameras did not change, hence we can predict possible object positions in the images of the next timestep. Thus the relative orientation (the fundamental geometry between the images) is used to stabilize tracking.

For tracking and calibration we have extended the SFM-approach of Pollefeys and Koch [13, 8] to a multi-camera configuration. Some details can also be found in [6]. In this approach the intrinsic and extrinsic camera parameters for each image are estimated. The standard decription of a projective camera consists of two 3x3 matrices K and R and the 3-dimensional vector C. K contains the intrinsic camera parameters focal length, aspect ratio and image center, R describes the rotation of the camera in space and C is the translation vector of the camera center. The projection matrix M projects the homogeneous 3D point P into a 2D image at image point p with $zp = MP$, z is the projective depth. M is defined as the following projective $3x4$-matrix:

$$M = K[R^T \mid -R^T C]$$

The SFM approach automatically tracks salient 3D features (intensity corners) throughout the images. The calibration results in a projection matrix for each camera of the sequence and a sparse point cloud of the tracked 3D feature points. The approach has been tested extensively with a wide variety of camera configurations and scenes and has proven to be very successful. The camera tracking and calibration may suffer from a projective skew and from error accumulation in the case of very long image sequences. Recording large scenes means that most images do not share a common region of the scene. This afffects the quality of tracking and matching of features. Therefore we cannot guarantee a globally consistent scene model. These problems are greatly reduced with the multi-camera rig since we can exploit the

© The Eurographics Association and Blackwell Publishers 2003.

rigidity constraints of the camera configuration. The problem is even less important in the case of image-based rendering. It is sufficient to render from local geometry that is supported from a local neighborhood only, because during rendering only parts of the scene are visible at once. Depth maps are appropriate local scene descriptions that may be used for this purpose.

3.3. Multi-viewpoint depth estimation

With the calibrated image sequence at hand, one can obtain dense depth maps from multi-viewpoint disparity estimation. From the calibration the epipolar geometry between pairs of images is known and can be used to restrict the correspondence search to a linear search. We extend the method of Koch et al. [7] for multi-viewpoint depth estimation to the multi-camera configuration. This method is ideally suited since we can exploit the 2D grid of linked depth maps for all cameras of the rig. This results in very dense depth maps of the local scene geometry. Only in very homogeneous image regions it might not be possible to extract sufficient scene depth. These regions will be interpolated from neighboring depth values. Using more than one pair of cameras allows us to fill holes from occlusions and to enhance the precision of the depth maps.

To give an idea about the quality of the estimated calibration and the depth maps, a complex real scene was evaluated. 240 images were taken with a four-camera rig at the National History Museum, London, featuring a large dinosaur skeleton in the main entrance hall. The rig was moved alongside the dinosaur looking at the skeleton and the back hall region. One may judge the high quality of the camera calibration and the density of the depth map even with this highly complex scene. In figure 2, left, we see an overview picture of the museum hall with the skeleton. To the right, the tracked camera positions (pyramids) and the 3D feature cloud with tracked points of the dinosaur and the hall back area are displayed. The dinosaur shape shows that even a globally consistent reconstruction of the scene was possible.

Figure 3 shows one of the original camera images and the corresponding depth map. The density and resolution of the depth map is very detailed. The camera is rather near to the skeleton and the depth of the scene is very large, causing displacements of up to 80 pixels between adjacent images. Therefore, image interpolation alone will not suffice to render novel views of the scene and depth compensation is necessary. The scene contains a lot of occluded regions (around the ribs) and one can see that in some of those regions no depth could be estimated. These regions are colored black.

3.4. Sampling the Depth maps

The depth maps serve as geometry input for the view-dependent online modeling that will be described in the next section. Using the depth maps directly with full resolution is not feasible due to the vast amount of data. We do not need such high resolution geometry for view interpolation. Therefore, each depth map is subsampled in a regular grid with a spacing that is parametrized such that it can easily be adapted to the specific needs. This grid is located in the image plane of the real camera. At each grid point a 2D median filter is applied to reduce the effects of outliers and to find the most probable depth. The filtered depth value corresponds to the distance between the point in the scene and the camera center. In other words, it is the length of a ray originating from the camera center through the grid point in the image to the point in the scene which caused the image point. Taking the j-th 2D point p_i^j in the image plane of camera i and the corresponding distance z_i^j from the depth map, the euclidean 3D scene point P_i^j can be calculated as:

$$P_i^j = z_i^j * (K_i R_i^T)^{-1} * p_i^j + C_i$$

The 3D point P_i^j of the 2D point p_j from camera i is called *sample* j of camera i. For each grid point in each camera one sample is created. In untextured regions of the image it is often impossible to determine the depth. This results in holes in the depth map, samples in these regions are discarded. Later on in the rendering stage the geometry in these regions is constructed from the samples from other cameras if possible. The valid samples serve as geometric approximations

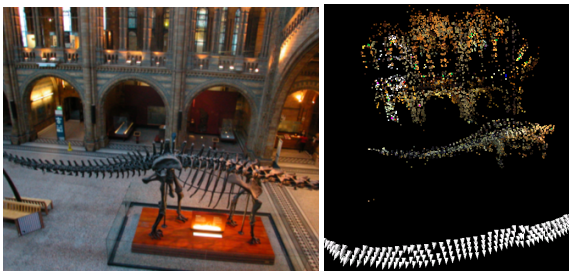

Figure 2: *4-camera acquisition and calibration of the dinosaur scene. Left: Overview of the scene to be captured. Right: SFM-calibration with camera positions (pyramids) and 3D feature points of dinosaur and hall background (snapshot from 3D scene model).*

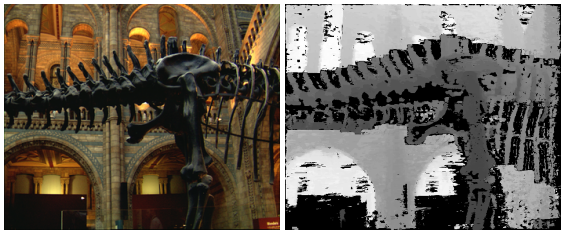

Figure 3: *Left: One of the acquired original camera images. Right: corresponding depth map of camera view (near=dark, far=light, undefined=black).*

© The Eurographics Association and Blackwell Publishers 2003.

that are used to define the view-dependent interpolation surface.

For a better trade-off between performance and quality, a *Level-of-Detail* (LOD) is introduced at this point. When creating the samples, the desired number of levels $L_{\max}$ can be chosen. After their generation all samples belong to level zero L_0 by default. To generate level L_{k+1} each second sample in both direction from level L_k is moved to level L_{k+1}. Thus the number of samples in level L_{k+1} is $1/4$ of the previous number of samples in level L_k which is now reduced by $1/4$. Filtering the samples before subsampling is not required, due to the median filter which was applied when generating the samples.

While rendering with a specific LOD n, the samples of several levels are used in combination. Rendering with the coarsest level only the samples of $L_{\max}$ are used. For the next level $L_{\max-1}$ the samples of $L_{\max}$ and of $L_{\max-1}$ are used. In general, if level L_n is requested, all levels L_x with $x \geq n$ are used.

4. Image-based interactive rendering

The calibrated views and the preprocessed depth maps are used as input to the image-based interactive rendering engine. The user controls a virtual camera which views the scene from novel viewpoints. The novel view is interpolated from the set of real calibrated camera images and their associated depth maps. During rendering it must be decided which camera images are best suited to interpolate the novel view, how to compensate for depth changes and how to blend the texture from the different images. For large and complex scenes hundreds or even thousands of images have to be processed. All these operations must be performed at interactive frame rates of 10 fps or more. We address these issues in the following sections:

- Selection of best real camera views,
- fusion of multiview geometry from the views,
- viewpoint-adaptive mesh generation,
- viewpoint-adaptive texture blending.

4.1. Camera ranking and selection

For each novel view to render, it has to be decided which real cameras to use. Several criterias are relevant for this decision. We have developed a ranking criterion for ordering the real cameras. In the following, N real cameras C_i, $0 \leq i \leq N$ are compared to the virtual camera C_v.

Distance: The first criterion concerns camera proximity. Taking the viewing direction A_v and the center C_v of the virtual camera, the orthogonal distance d_i of each real camera center C_i to this ray can be determined:

$$d_i = ||\frac{(C_i - C_v) \times A_v}{||A_v||}||$$

For better evaluation all distances d_i are normalized with the maximum distance $d_{\max}$ of all cameras. $d_{\max}$ is the maximum of all d_i for the current C_v:

$$d_{\max} = \max\{d_i | 0 \leq i \leq N\}$$

Viewing angle: The second criterion is the angle between A_v and the viewing direction A_i of camera i. Cameras looking into directions different to the virtual camera are penalised because they are less useful for view interpolation. The angular penalty a_i for camera i is defined as:

$$a_i = \frac{\arccos(A_i \cdot A_v)}{a_{\max}}$$

For normalization a maximum threshold angle $a_{\max}$ $=a_{\text{field-of-view}}$ is given. Cameras which are more than their field-of-view "off axis" do not share a common viewing range and can not be used for proper geometry or texture interpolation, these are marked invalid.

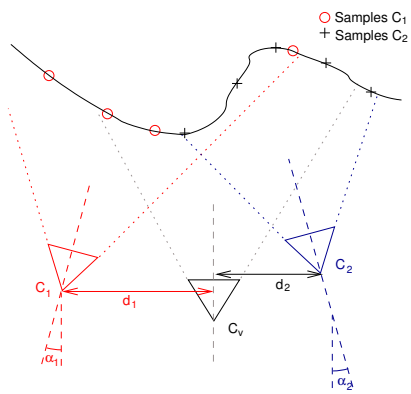

Figure 4: *Criteria evaluation. The ranking criteria are shown for two real cameras C_1 and C_2. C_v sees 3 samples from C_1 and 4 samples from C_2.*

Visibility: The third criterion is a more complex one, we call it the *visibility*. It evaluates the scene volume that a given real camera has as seen in the context of the virtual camera. For this purpose very few ($n_{\max} = 20$) regular samples of the depth maps as described above are chosen such that the whole image of the real camera is covered. These samples are then projected into the virtual camera and checked for visibility. The number of visible samples n_i divided by the number of possible samples $n_{\max}$ gives a rough approximation of the region covered by a real camera. Cameras for which this ratio results to zero are marked invalid. To convert this value into a visibility penalty v_i it is subtracted from one:

$$v_i = 1 - \frac{n_i}{n_{\max}}$$

Figure 4 sketches the different selection criteria. Two real cameras C_1 and C_2 are evaluated w.r.t. the virtual camera C_v.

© The Eurographics Association and Blackwell Publishers 2003.

Distance and viewing angle are given in the figure. For visibility, the depth samples from camera 1 (circles) and camera 2 (crosses) are projected into the virtual camera and evaluated.

All three criteria are weighted and combined into one scalar value q_i which represents the inverse quality of the real camera i to generate the new view:

$$q_i = w_d d_i + w_a a_i + w_v v_i,$$

After calculating q_i for each camera, the list of valid cameras is sorted in ascending order. The interpolation mode finally decides how many of the best suited cameras are selected for view interpolation.

4.2. Multiview depth fusion and mesh creation

The ranked cameras are now used to interpolate novel views. Since the novel view may cover a field of view that is larger than any real camera view, we have to fuse views from different cameras into one locally consistent image. To efficiently warp images from different real views into the novel viewpoint we generate a warping surface approximating the geometry of the scene. Starting from a regular 2D-grid that is placed in the image plane of the virtual camera this warping surface will be updated for each camera motion. The spacing of this grid $S = (s_x, s_y)$ with s_x, s_y in pixels can be scaled to the complexity of the scene. With each point in the grid, a 5-tuple $g = \{P_g, p_g, i_g, b_g, d_g\}$ called *grid point* is associated. p_g is the 2D position in the image plane, P_g the 3D point to be constructed, i_g is the number of the camera responsible for P_g, b_g is a boolean marking this grid point valid or invalid and d_g is the distance from P_g to C_v. The b_g and d_g components of all grid points are set to default values, which are $b_g =$ invalid and $d_g = \infty$.

To fuse 3D information from the first n ranked cameras, for each camera i the following algorithm is used.

- Each valid sample P_i^j for camera i is projected into the virtual camera with $p_i^j = M_v P_i^j$ and the distance $d_i^j = ||P_i^j - C_v||$ is calculated.
- If p_i^j is not in the visible area, the following steps are skipped and the next sample P_i^{j+1} is taken. If it is in the visible area, the nearest grid point g_n is selected. Due to the regularity of the grid, this is easily done with

$$g_n = \mathrm{rnd}(\frac{p_i}{S})$$

- If the current sample p_i^j has a smaller distance d_i^j to C_v then the selected grid points depth d_{g_n}, then the grid points data is updated from this sample, else the update is skipped. When updating a grid point from a sample, P_{g_n} is set to P_i^j, p_{g_n} is adjusted to p_i^j, the cameras number i is stored in i_{g_n} and the new distance value d_i^j is assigned to d_{g_n}.

The algorithm proceeds with the next sample P_i^{j+1}. Updating grid points only with samples which are nearer to the virtual camera ($d_i^j < d_{g_n}$) ensures that occlusion are handled correctly. The samples of the highest ranked cameras are projected first. This ensures that most part of the new view is interpolated from the best cameras. Only in regions that are occluded for the best camera, a lower ranked camera sample is used. Figure 5 depicts this situation.

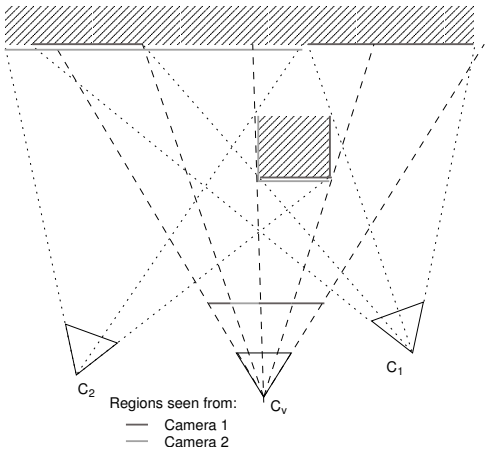

Figure 5: *Geometry fusion from two cameras, with C_1 ranked higher than C_2. Based on the ranking, C_1 will supply most of the information. Only in parts that are occluded for C_1, data from C_2 is filled in.*

After projecting samples and adjusting grid points, most grid points are valid and contain 3D information suitable to represent the part of the scene visible in the virtual camera. But because some grid points could be still invalid and the 2D positions in p_{g_n} are fitted to projected samples, the connectivity of the grid points cannot be given from the grid itself. Using the 2D position of the all valid grid points, a 2D Delauny triangulation of all grid points in the image plane of the virtual camera is performed. Transferring this 2D mesh to the 3D points P_{g_n} gives us a scalable approximation of the 3D scene with triangles. The approximation can be scaled with respect to the sampling density of depth samples and the density of the grid points for triangulation. This surface mesh is recreated after each camera movement as viewpoint-adaptive geometry.

4.3. Texturing

The texturing step effectively maps the real cameras into the virtual view with the help of the viewpoint-adaptive surface mesh. Several slightly different methods for texturing are considered. The most simple one is to chose the best

© The Eurographics Association and Blackwell Publishers 2003.

ranked camera as texture source. If this real camera is not too far away from the virtual camera and both have a similar field of view, the results are good. This is the fastest texturing method since switching between different textures in one render cycle is not necessary and each triangle has to be drawn only once. Problems arise when parts of the mesh are not seen from the selected camera. These parts remain untextured.

To texture all triangles properly it is necessary to select the texture according to the cameras where the geometry originated from. The triangle vertices are depth sample points where the originating real camera is known. However, since each vertex is generated independently, a triangle may have vertices from up to three cameras. Here one may decide to either select the best-ranked camera (single-texture mode) or to blend all associated camera textures on the triangle (multi-texture mode). Proper blending of all textures will result in smoother transition between views but with higher rendering costs for multi-pass rendering.

To test the rendering quality, a synthetic scene was generated by composing a VRML model of the "Arenberg Castle" together with a VRML model of an entrance portal to simulate occlusion, as shown in figure 6 (right). This textured 3D model was then rendered from different views and screenshots along with synthesized depth maps and corresponding projection matrices were saved. Based on this ground truth material, the rendering quality was verified. To measure the image quality, one view (shown in figure 6, right) from the sequence was taken as reference image. The corresponding camera was then removed and this view was interpolated from the remaining images with different texturing methods. The most critical regions are the edges between the background wall and the foreground portal. Due to the variation in depth, most artifacts are located here. For better comparison this region is magnified (figure 6, left) and the reference image is then compared to the different interpolated views by image substraction. The resulting difference image serves as visual error measure and additionally the mean absolute intensity difference (MAD) is given. To emphazise the visual errors a gamma correction of 0.3 was applied on the difference images.

For comparison only, the most simple approximation of the desired view is used. For this purpose the nearest image is taken without depth warping (nearest neighbour), in fact, this is equivalent to standard image interpolation. Figure 7 shows that there are large areas around the portal where this interpolation fails due to the large image displacements.

The difference image shows that the objects are not rendered at the correct depth and the MAD value is 26.5. With our proposed rendering system we can apply depth compensation with different texturing modes.

Figure 6: *An input view of a synthetic scene (right) and a magnified closeup of critical regions (left).*

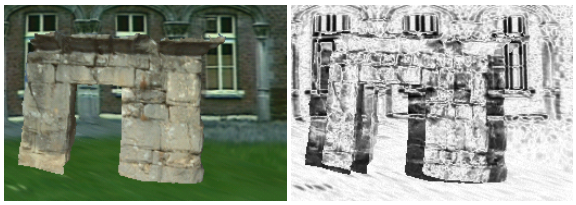

Figure 7: *Rendering (left) and difference image (right) when a nearest neighbor selection without interpolation is used. The difference image shows that large image displacements occur especially in the foreground regions. MAD = 26.5*

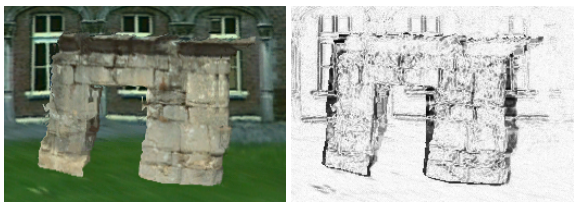

Figure 8: *Rendering with single-camera, single-texture mode: The depth-compensated interpolation removes the image displacements, but edge artefacts remain. MAD = 12.8*

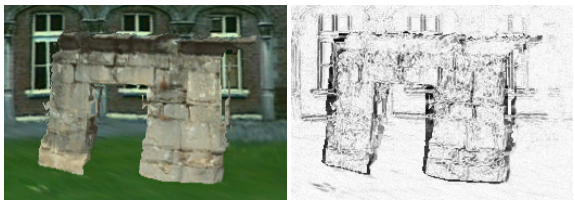

Figure 9: *Rendering with multi-camera, single-texture mode: The view-dependent texturing removes some edge artefacts at the right occluding border of the portal. MAD = 12.4*

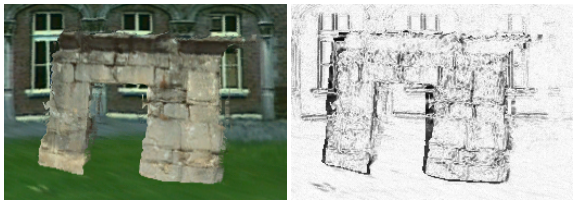

Figure 10: *Rendering with multi-camera, three-texture mode: Texture blending reduces some texture edge artefacts, although the improvement is minor in this scene. MAD = 11.8*

© The Eurographics Association and Blackwell Publishers 2003.

The depth maps were subsampled with 200x160 samples, the size of the grid was also 200x160 points. Figure 8 was rendered using only the highest-ranked camera for texturing (single camera, single texture mode). It is visible that the geometrical approximation removes most of the more serious errors resulting in a MAD of 12.8. Due to the discretized sampling and geometrical construction, fine structures like the windows in the background are not modeled sufficiently which results in the remaining errors. This rendering mode is very fast since all textures are taken from a single texture map.

Using more sophisticated texturing compensates some more errors so that the final visual quality is very close to the original view. Figure 9 is rendered from multiple cameras, but each triangle uses one texture only (multi-camera, single texture). From three possible cameras, again the best ranked can be taken to texture the triangle. Some texture switching is required but recent graphics hardware is fast enough for this. In regions where most triangles are textured from one camera the result is the same as in the first method. In regions textured from many different cameras this can result in texture artefacts because adjacent fragments do not always match. This method gives a MAD of 12.4. Since each triangle has to be drawn only once, the slowdown compared to single camera texturing is small. Sharp edges between textures can be avoided by multi-texturing and blending. Each triangle is drawn three times using the textures associated with the three cameras (multi-camera, multi-texture mode). The previously mentioned edges between textures are blended smoothly, as shown in figure 10 resulting in a MAD fo 11.8. On modern graphics hardware it is also possible to use single-pass multi-texturing. Different texture units are loaded with the three textures and then the triangle is drawn only once. This gives a speed-up of approximatly 30% compared to the multi-pass texturing. The performance gain is not factor 3 as one would expect, because for each triangle the texture units have to be reloaded which is quite expensive.

Figure 11: *Parking lot scene. Top: Two original images. Bottom left: Depth map, right: Novel viewpoint rendered from the parking lot scene in multi-camera, multi-texture mode.*

5. Performance issues

We have implemented the described rendering system in C++ using OpenGL and tested it on a standard PC (Linux) with 1.7 GHz AMD Athlon, 1GB memory and a Geforce4 Ti 4400 with 64 MB memory. The following section discusses some of the performance issues that arise. We will discuss memory and processing speed requirements.

5.1. Memory Handling

Expecting several hundreds of camera images to render from, some care has been taken to handle such amounts of textures and depth maps. After creating the depth samples, the depth maps are released immediately. To avoid sample creation for each start of the program, the samples are stored and reloaded if necessary. Even if modern graphics boards have 128 MB of memory, this is far too little for our purpose, but the AGP interface allows the usage of main memory for textures without significant performance loss. When starting, the program reserves a given amount of memory for textures and can also be told to load as many images as possible in advance. Later on, images are loaded on demand when they are used for the first time. If the upper bound of memory for textures is reached, the least recently used texture is released. Texture compression is used to preserve memory, too. The fixed compression ratio is $1/6$, allowing to handle 6 times more textures than without compression. The only problem is that the compression takes some time. In case the texture is loaded on demand, which means the rendering stops until the texture is ready, it takes around 5 times longer compressing the texture than loading it uncompressed. We have therefore precompressed the images, stored them on disk and are loading them as already compressed textures on demand.

Using these techniques, we are able to handle image sequences of nearly arbitrary size. In an initialization run all images are prepared as compressed textures and all depth maps are sampled in an LOD hierarchy. This results in a space- and time-optimized storage of the plenoptic model. The synthetic castle scene for example consists of 216 images of the size 768x512 and the required space for the original images and depth maps is about 566 MB. The compressed representation uses 132MB for depth sample storage and 109MB for compressed textures only. Tests with scenes of 1057 images (3 GByte raw data) on a PC with 1GByte memory showed that all textures could be preloaded. To handle even larger scenes on-demand-loading is also possible

© The Eurographics Association and Blackwell Publishers 2003.

and does not reduce the frame rate significantly. The bottleneck of loading textures is removed.

Typically, the performance of a render engine is measured in the displayed frames per second which are dependent on the scene complexity and visible area. Our rendering system is different in this respect since the frame rate is roughly independent of the scene complexity but depends on the number and resolution of the real views that are used for interpolation. Each time the viewpoint is changed, the warping surface mesh is reconstructed from the depth samples of the best ranked n cameras. This operation is linear in the number of cameras times samples and performed by the CPU.

The necessary steps are camera ranking, projection of the samples and Delauney triangulation, as described above. Camera ranking is quite fast, the selected ranking criteria can be computed with little cost, even 1000 cameras are ranked in 3.9 ms. The Delauney triangulation is linear in time w.r.t. the number of points. A mesh for 100x75 grid points is created in 25 ms, for 200x150 grid points it took around 101 ms.

5.2. Rendering performance

Projecting samples from real cameras into the virtual camera is also quite expensive. A typical density of 200x150 samples results in 30000 projections per camera, and using 10 real cameras gives 300,000 matrix-vector multiplications. These are calculated in approximatly 77 ms. Adding the time for rendering with 4 ms, a total of 182 ms per geometry update for a grid with 200x150 points and 10 cameras with 200x150 samples each is reached.

For this reason, the level-of-detail and the scalable density of grid points is used to reduce the complexity. Each level-of-detail and also each stepping in the grid density reduces the computation time by factor four. Also a proper selection of the relevant cameras helps to reduce the number of projections needed. All three values can be adjusted by the user to trade between quality and speed. So using only 3 cameras with 100x75 samples and a grid with 100x75 points it took only 31 ms to generate a new view. The loss in quality depends on the complexity of the scene but to enhance the quality significantly, a slowdown of factor 8 has to be tolerated for most scenes.

In general, for closeup views of small parts of the scene only a small number of cameras is required, but the depth granularity should be fine. On the other hand, to obtain an overview over a large scene, many real cameras are required simultaneously but the level-of-detail can be chosen moderately because of the reduction of scale. Finally, texture mapping and blending is performed. The texture mapping is fully hardware accelerated and mostly determined by the display resolution and the texturing engine of the graphics card.

With this scalable approach, interactive frame rates of 5 to 10 fps can always be guaranteed, even when using scenes with 1000 cameras. Using coarse resolution to plan a camera path, the final rendering of a virtual fly by could be done fully automated with highest accuracy in non-interactive frame rates.

Figure 12: *Rendered new view of dinosaur sceleton and museum hall. Main bones and hall background are reconstructed correctly. The peaks of the spinal bones are distorted due to improper depth compensation.*

6. Experiments

The system was tested with a large variety of real footage. We tested scenes of different complexity and very different spatial sample rate. Some scenes were taken with a hand-held still camera and very few images were used. For other scenes we used two or four cameras of the rig to obtain many images. As an example of a complex outdoor scene we recorded a walk over a parking lot with trees, cars, and buildings. 214 images (2x107 stereo pairs) with 1024x768 pixels each were recorded. The covered track was about 35 m, resulting in a spatial sampling between the images of about 30 cm. Scene depth extended between 3 and 75 m, causing image disparities of about 100 pixel between adjacent images. Figure 11 shows two images of the original scene and a depth map of the right original image to document that a dense reconstruction of trees, cars and bushes could be obtained. The rendered novel view (bottom right) shows that even small details like the lamp post and the trees are rendered from new a perspective with high realism and little distortion.

Finally we show rendering results of the dinosaur scene as described in section 3.3. An original image (fig. 3,left) was removed and interpolated from the remaining images (fig. 12, left). Most image regions are rendered with high quality. The peaks of the spinal bones are distorted since a

© The Eurographics Association and Blackwell Publishers 2003.

proper depth reconstruction was not feasible due to occlusions. These regions were marked black in the depth map (fig. 3,right).

7. Conclusions

We have discussed a new image-based rendering approach that can handle uncalibrated multi-camera sequences from a hand-operated camera rig. In a preprocessing step the camera calibration and depth reconstruction is performed automatically from the image sequence itself. These view-dependent data are then used for interactive image-based rendering by depth-compensated warping of view-dependent textures. The system can handle very large data sets of hundreds of images at interactive rates. Experiments have shown that the surface-based rendering approach is successful if dense depth can be computed. However, in regions with many small and occluding objects, depth computation may fail and distortions occur. Also, if the disparity between adjacent images is very high, edge interpolation artefacts occur. We are currently investigating how to overcome these drawbacks. Possible solutions may be to selectively switch from surface to point-based rendering or to use multi-layered depth images to account for the occluding regions. We will investigate further in this direction.

Acknowledgements

The work is being funded by the European project IST-2000-28436 ORIGAMI. The images for the dinosaur scene were supplied by Oliver Grau, BBC Research.

References

1. Chris Buehler, Michael Bosse, Leonard McMillan, Steven J. Gortler, and Michael F. Cohen. Unstructured lumigraph rendering. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 425–432. ACM Press / ACM SIGGRAPH, 2001. 2
2. Paul Debevec, Yizhou Yu, and George Boshokov. Efficient view-dependent image-based rendering with projective texture-mapping. Technical Report CSD-98-1003, 20, 1998. 2
3. Jan-Michael Frahm, Jan-Friso Evers-Senne, and Reinhard Koch. Network protocol for interaction and scalable distributed visualization. In *3DPVT 2002*, 2002. 3
4. Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *Computer Graphics*, 30(Annual Conference Series):43–54, 1996. 2
5. Benno Heigel, Reinhard Koch, and Mark Pollefeys. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Proceedings of DAGM 1999*, 1999. 2
6. R. Koch, J.F. Frahm, J.M.and Evers-Senne, and J. Woetzel. Plenoptic modeling of 3d scenes with a sensor-augmented multi-camera rig. In *Tyrrhenian International Workshop on Digital Communication (IWDC): proceedings*, Sept. 2002. 3
7. R. Koch, Pollefeys M., and L. Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *Proc. ECCV'98*, number 1406 in LNCS. Springer, 1998. 4
8. R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, and H. Niemann. Calibration of handheld camera sequences for plenoptic modeling. In *Proceedings ICCV 99*, Korfu, Greece, 1999. 2, 3
9. Marc Levoy and Pat Hanrahan. Light field rendering. *Computer Graphics*, 30(Annual Conference Series):31–42, 1996. 2
10. M. Magnor and B. Girod. Data compression for light field rendering. *IEEE Trans. Circuits and Systems*, 2000. 2
11. Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3d photography using opacity hulls. 2
12. Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, 29(Annual Conference Series):39–46, 1995. 2
13. Marc Pollefeys, Reinhard Koch, and Luc J. Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999. 2, 3
14. K. Pulli, T. Duchamp, H. Hoppe, J. McDonald, L. Shapiro, and W. Stuetzle. Robust meshes from multiple range maps, 1997. 2
15. Kari Pulli, Michael Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In Julie Dorsey and Phillipp Slusallek, editors, *Rendering Techniques '97 (Proceedings of the Eighth Eurographics Workshop on Rendering)*, pages 23–34, New York, NY, 1997. Springer Wien. 2
16. X. Tong and R.M. Gray. Compression of light fields using disparity compensation and vector quantisation. In *Proc. IASTED Conf. Computer Graphics and Imaging*, pages 300–305, October 1999. 2
17. Matthias Zwicker, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. Surface splatting. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 371–378. ACM Press / ACM SIGGRAPH, 2001. 2

© The Eurographics Association and Blackwell Publishers 2003.