Image-based Rendering from Uncalibrated Lightfields with Scalable Geometry

R. Koch[1]*, B. Heigl[2], M. Pollefeys[3]

[1]Institut für Informatik, Christian-Albrechts-Universität Kiel, Germany
[2]Lehrstuhl für Mustererkennung, Universität Erlangen-Nürnberg, Germany
[3]Laboratory for Processing of Speech and Images, PSI-ESAT, K.U. Leuven, Belgium

Abstract. We combine uncalibrated Structure-from-Motion, lightfield rendering and view-dependent texture mapping to model and render scenes from a set of images that are acquired from an uncalibrated hand-held video camera. The camera is simply moved by hand around the 3D scene of interest. The intrinsic camera parameters like focal length and the camera positions are automatically calibrated with a Structure-From-Motion approach. Dense and accurate depth maps for each camera viewpoint are computed with multi-viewpoint stereoscopic matching. The set of images, their calibration parameters and the depth maps are then utilized for depth-compensated image-based rendering. The rendering utilizes a scalable geometric approximation that is tailored to the needs of the rendering hardware.

1 Introduction

This contribution discusses realistic scene reconstruction and visualization from real image streams that are recorded by an uncalibrated, freely moving hand-held camera. This approach allows to easily acquire 3D scene models from real-world scenes with high fidelity and minimum effort on equipment and calibration.

Recently, quite some approaches to this problem have been investigated. Plenoptic modeling [11], lightfield rendering [10] and the lumigraph [5] have received a lot of attention, since they can capture the appearance of a 3D scene from images only, without the explicit use of 3D geometry. Thus one may be able to capture objects with very complex geometry that can not be modeled otherwise. Basically one caches views from many different directions all around the scene and interpolate new views from this large image collection. For realistic rendering, however, very many views are needed to avoid interpolation errors for in-between views.

Structure from motion (SFM) approaches like [13] on the other hand try to model the 3D scene and the camera motion geometrically and capture scene details on polygonal (triangular) surface meshes. A limited set of camera views of the scene are sufficient to reconstruct the 3D scene. Texture mapping adds

* Work was performed during stay at the Laboratory for Processing of Speech and Images, PSI-ESAT, K.U. Leuven, Belgium

the necessary fidelity for photo-realistic rendering of the object surface. Dense and accurate 3D depth estimates are needed for realistic image rendering from the textured 3D surface model. Deviation from the true 3D surface will distort the rendered images.

The problem common to both approaches is the need to calibrate the image sequence. Recently it was proposed to combine a structure from motion approach with plenoptic modeling to generate lightfields from uncalibrated hand-held camera sequences [8]. When generating lightfields from a hand-held camera sequence, one typically generates images with a specific distribution of the camera viewpoints. Since we want to capture the appearance of the object from all sides, we will sample the viewing sphere, thus generating a *mesh of view points*. To fully exploit hand-held sequences, we will therefore have to deviate from the regular lightfield data structure and adopt a more flexible rendering data structure based on the viewpoint mesh. Another important point in combining SFM and lightfield rendering is the use of scene geometry for image interpolation. The geometric reconstruction yields a geometric approximation of the real scene structure that might be insufficient when static texture mapping is used. However, view-dependent texture mapping as in [2] will adapt the texture dynamically to a static, approximate 3D geometry.

In this contribution we will discuss the combination of Structure-from-Motion, lightfield rendering, and dynamic surface texturing. SFM delivers camera calibration and dense depth maps that approximate the scene geometry. Rendering is then performed by depth-compensated image interpolation from a mesh of camera viewpoints as generated by SFM. The novel image-based rendering method takes advantage of the irregular viewpoint mesh generated from hand-held image acquisition. We will first give a brief overview of the calibration and reconstruction techniques by SFM. We will then focus on the depth-compensated image interpolation and show that only a coarse geometric approximation is necessary to guide the rendering process. Experiments on calibration, geometric approximation and image-based rendering verify the approach.

2 Calibration and 3D-Reconstruction

Uncalibrated *Structure From Motion* (SFM) is used to recover camera calibration and scene geometry from images of the scene alone without the need for further scene or camera information. Faugeras and Hartley first demonstrated how to obtain uncalibrated projective reconstructions from image point matches alone [4, 6]. Beardsley et al. [1] proposed a scheme to obtain projective calibration and 3D structure by robustly tracking salient feature points throughout an image sequence. This sparse object representation outlines the object shape, but does not give sufficient surface detail for visual reconstruction. Highly realistic 3D surface models need a dense depth reconstruction and can not rely on few feature points alone.

In [13] the method of Beardsley was extended in two directions. On the one hand the projective reconstruction was updated to metric even for varying internal camera parameters, on the other hand a dense stereo matching technique [3] was applied between two selected images of the sequence to obtain a dense depth map for a single viewpoint. From this depth map a triangular surface wire-frame was constructed and texture mapping from one image was applied to obtain realistic surface models. In [7] the approach was further extended to multi-viewpoint depth analysis. The approach can be summarized in 3 steps:

- Camera self-calibration and metric structure is obtained by robust tracking of salient feature points over the image sequence,
- dense correspondence maps are computed between adjacent image pairs of the sequence,
- all correspondence maps are linked together by multiple view point linking to fuse depth measurements over the sequence.

2.1 Calibration of a mesh of viewpoints

When very long image sequences have to be processed with the above described approach, there is a risk of calibration failure due to several factors. For one, the calibration as described above is built sequentially by adding one view at a time. This may result in accumulation errors that introduce a bias to the calibration. Secondly, if a single image in the sequence is not matched, the complete calibration fails. Finally, sequential calibration does not exploit the specific image acquisition structure used in this approach to sample the viewing sphere.

In [8] a multi-viewpoint calibration algorithm has been described that allows to actually weave the viewpoint sequence into a connected viewpoint mesh. This approach is summarized in the following section.

Image pair matching The basic tool for viewpoint calibration is the two-view matcher. Corresponding image features m_i, m_k have to be matched between the two images of the camera viewpoints P_i, P_k. The image features are projections of a 3D feature point M into the Images I_i, I_k in homogeneous coordinates:

$$m_i = \rho_i P_i M \quad , \quad m_k = \rho_k P_k M \quad , \quad P = K\left[R^T | - R^T c\right] \tag{1}$$

with ρ a non-zero scaling factor, K = camera calibration matrix, R = orientation and c = position of the camera. To solve for P from m_i, m_k we employ a robust computation of the Fundamental matrix F_{ik} with the RANSAC (RANdom SAMpling Consensus) method [14]. Between all image correspondences the fundamental image relation (the epipolar constraint) holds

$$m_i^T F_{i,k} m_k = 0 \tag{2}$$

$F_{i,k(3x3)}$ is a linear rank-2 matrix. A minimum set of 7 feature correspondences is picked from a large list of potential image matches to compute a specific F. For this particular F the support is computed from the other potential matches. This procedure is repeated randomly to obtain the most likely F_{ik} with best support in feature correspondence. From F we can compute the

3×4 camera projection matrices P_i and P_k. The fundamental matrix alone does not suffice to fully compute the projection matrices. In a bootstrap step for the first two images we follow the approach by Beardsley e.a. [1]. Since the camera calibration matrix K is unknown a priori we assume an approximate $\tilde{K}$ to start with. The first camera is then set to $P_0 = \tilde{K}[I|0]$ to coincide with the world coordinate system, and the second camera P_1 can be derived from the epipole e (projection of camera center into the other image) and F as

$$P_1 = \tilde{K}\left[[e]_x F + ea^T|\rho e\right] \ , [e]_x = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{3}$$

P_1 is defined up to global scale ρ and the unknown plane π_{inf}, encoded in a^T (see also [12]). Thus we can only obtain a projective reconstruction. The vector a^T should be chosen such that the left 3×3 matrix of P_i best approximates an orthonormal rotation matrix. The scale ρ is set such that the baseline length between the first two cameras is unity. K and a^T will be determined during camera self-calibration.

Once we have obtained the projection matrices we can triangulate the corresponding image features m_i, m_k with P_i, P_k to obtain the corresponding 3D object features M. The object points are determined such that their reprojection error in the images is minimized. In addition we compute the point uncertainty covariance to keep track of measurement uncertainties. The 3D object points serve as the *memory* for consistent camera tracking, and it is desirable to track the projection of the 3D points through as many images as possible. This process is repeated by adding new viewpoints and correspondences throughout the sequence. Finally constraints are applied to the cameras to obtain a metric reconstruction. A detailed account of this approach can be found in [12, 13].

Estimating the viewpoint topology Since we are collecting a large amount of images from all possible viewpoints distributed over the viewing sphere, it is no longer reasonable to consider a sequential processing along the sequence frame index alone. Instead we would like to evaluate the image collection in order to robustly establish image relationships between all nearby images. We need to define a distance measure that allows to estimate the proximity of two viewpoints from image matches alone. We are interested in finding those camera viewpoints that are near to the current viewpoint and that support calibration. Obvious candidates for these are the preceding and following frames in a sequence, but normally those viewpoints are taken more or less on a linear path due to camera motion. This near-linear motion may lead to degeneracies and problems in the calibration. We are therefore also interested in additional viewpoints that are perpendicular to the current direction of the camera motion. If the camera sweeps back and forth over the viewpoint surface we will likely approach the current viewpoint in previous and future frames. Our goal is now to determine which of all viewpoints are nearest and most evenly distributed around our current view. So far we do not know the position of the cameras, but we can compute the F-Matrix from corresponding image points. For each potential neighbor image I_i

we compute $F_{c,i}$ w.r.t. the current image I_c. To measure *proximity* and *direction* of the matched viewpoint w.r.t. the current one, we can exploit the image epipole as well as the distribution of the correspondence vectors.

Direction: The epipole determines the angular direction α_e of the neighboring camera position w.r.t. the current image coordinates, since it represents the projection of the camera center into the current image. Those viewpoints whose epipoles are most evenly distributed over all image quadrants should be selected for calibration.

Proximity: The distribution of the corresponding matches determines the distance between two viewpoints. Consider a non-planar scene and general motion between both cameras. If both camera viewpoints coincide we can cancel out the camera orientation change between the views with a projective mapping (rectification) and the corresponding points will coincide since no depth parallax is involved. For a general position of the second camera viewpoint, the depth parallax will cause a residual correspondence error e_r after rectification that is proportional to the baseline distance between the viewpoints. We can approximate the projective rectification by a linear affine mapping that is estimated from the image correspondences. We therefore define the residual correspondence error e_r after rectification as proximity measure for nearby viewpoints. The viewpoints with smallest e_r are closest to the current viewpoint.

Weaving the viewpoint mesh With the distance measure at hand we can build a topological network of viewpoints. We start with an arbitrary image of the sequence and compute α_e and e_r for subsequent images. If we choose the starting image as first image of the sequence, we can proceed along the frame index and find the nearest adjacent viewpoints in all directions. From this seed views we proceed recursively, building the viewpoint mesh topology over all views. The mesh builds along the shortest camera distances very much like a wave propagating over the viewpoint surface.

2.2 3D geometry estimation

Once we have retrieved the metric calibration of the cameras we can use image correspondence techniques to estimate scene depth. We rely on stereo matching techniques that were developed for dense and reliable matching between adjacent views. The small baseline paradigm suffices here since we use a rather dense sampling of viewpoints.

For dense correspondence matching an area-based disparity estimator is employed. The matcher searches at each pixel in one image for maximum normalized cross correlation in the other image by shifting a small measurement window (kernel size 7x7) along the corresponding epipolar line. Dynamic programming is used to evaluate extended image neighborhood relationships and a pyramidal estimation scheme allows to reliably deal with very large disparity ranges [3].

The geometry of the viewpoint mesh is especially suited for further improvement with a multi viewpoint refinement [7]. Each viewpoint is matched with all adjacent viewpoints and all corresponding matches are linked together to form a reliable depth estimate. Since the different views are rather similar we will observe every object point in many nearby images. This redundancy is exploited to improve the depth estimation for each object point, and to refine the depth values to high accuracy.

2.3 Experimental results for surface mesh calibration

To evaluate our approach, we recorded a test sequence with known ground truth from a calibrated robot arm. The camera is mounted on the arm of a robot of type SCORBOT-ER VII. The position of its gripper arm is known from the angles of the 5 axes and the dimensions of the arm. The robot sampled a 8×8 spherical viewing grid with a radius of 230 mm. The viewing positions enclosed a maximum angle of 45 degrees which gives an extension of the spherical viewpoint surface patch of 180×180 mm^2. The scene (with size of about 150x150x100 mm^3) consists of a cactus and some metallic parts on a piece of rough white wallpaper.

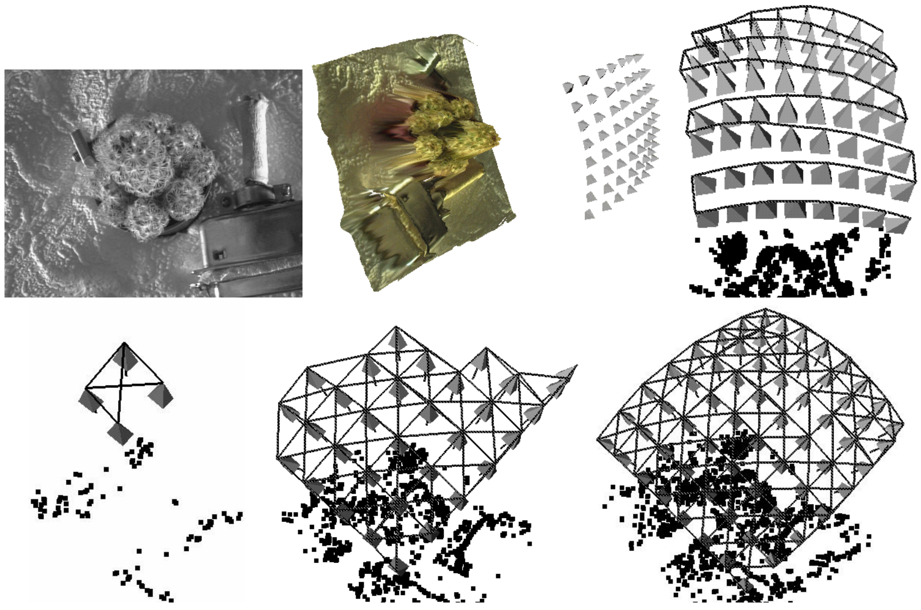

Fig. 1. Top left: one image of the robot sequence. Top middle: The distribution of the camera viewpoints over the 3D scene. Top right: sequential camera path as obtained from tracking along the camera path. Bottom: Intermediate steps of the mesh building after 4, 32, and 64 images. The camera viewpoints are indicated by pyramids that are connected by the viewpoint mesh. The black points in the background are tracked 3D feature points. One can see how the 2D mesh topology is building over the viewpoint surface.

Table 1. Ground truth comparison of 3D camera positional error between the 64 estimated and the known robot positions [in % of the mean object distance of 250 mm].

Camera position	projective		similarity	
Tracking Error [%]	mean	s.dev	mean	s.dev
sequential	1.08	0.69	2.31	1.08
2D viewpoints	0.57	0.37	1.41	0.61

One of the original images is shown in fig. 1(top left) together with the distribution of the camera viewpoints of the robot arm (top middle). Each camera position is visualized as little pyramid. In fig. 1(bottom) calibration using a viewpoint mesh results are shown. The mesh buildup is indicated by the estimated camera viewpoints (pyramids) and their topological relation (mesh connecting the cameras). Each connection indicates that the fundamental matrix between the image pair has been computed.

A quantitative evaluation of the tracking was performed by comparing the estimated metric camera pose with the known Euclidean robot positions. We anticipate two types of errors: 1) a stochastic measurement noise on the camera position, and 2) a systematic error due to a remaining projective skew from imperfect self-calibration. We also compared the simple sequential calibration that estimates $F_{i,k}$ along adjacent images of the recording path only (fig 1, top right), with the novel 2D mesh calibration method (see fig 1, bottom).

For comparison we transform the measured metric camera positions into the Euclidean robot coordinate frame. With a projective transformation we can eliminate the skew and estimate the measurement error. We estimated the projective transform from the 64 corresponding camera positions and computed the residual distance error. The distance error was normalized to relative depth by the mean surface distance of 250 mm. The mean residual error dropped from 1.1% for sequential tracking to 0.58% for viewpoint weaving (see table 1). The position repeatability error of the robot itself is 0.08%.

If we assume that no projective skew is present then a similarity transform will suffice to map the coordinate sets onto each other. A systematic skew however will increase the residual error. To test for skew we estimated the similarity transform from the corresponding data sets and evaluated the residual error. Here the mean error increased to 1.4% for mesh tracking which is still good for pose and structure estimation from fully uncalibrated sequences.

3 Plenoptic Modeling and Rendering

After determining the pose and projection properties of the moving camera we want to use the calibrated cameras to create a scene model for visualization.

One possible method is *lightfield rendering*[10]. To create a lighfield model for real scenes, a large number of views from many different angles are taken. Each view can be considered as bundle of light rays passing through the optical center of the camera. The set of all views contains a discrete sampling of light rays with according color values and hence we get discrete samples of the plenoptic function. The light rays which are not represented have to be interpolated.

The original 4–D lightfield data structure uses a two–plane parameterization. Each light ray passes through two parallel planes with plane coordinates (s,t) and (u,v). Thus the ray is uniquely described by the 4–tuple (u,v,s,t). The (s,t)–plane is the *viewpoint plane* in which all camera focal points are placed on regular grid points. The projection parameters of each camera are constructed such, that the (u,v)–plane is their common image plane and that their optical axes are perpendicular to it.

Often, real objects are supposed to be *Lambertian*, meaning that one surface point has the same radiance value viewed from all possible directions. This implies that two viewing rays have the same color value if they intersect the surface at the same point. If specular effects occur, this is not true any more. The radiance value will change with changing viewing direction, but for a small change of the viewing angle, the color value also will change just a little. Consequently, two viewing rays have similar color values, if their direction is similar and if their point of intersection is near the surface of the scene.

To render a new view we suppose to have a virtual camera pointing to the scene. For each pixel we can determine the position of the corresponding virtual viewing ray. The nearer a recorded ray is to this virtual ray the greater is its support to its color value. So the general task of rendering views from a collection of images will be to determine those viewing rays which are *nearest* to the virtual one and to interpolate between them depending on their proximity.

Linear interpolation between the viewpoints in (s,t) and (u,v) introduces a blurred image with ghosting artifacts. In reality we will always have to choose between high density of stored viewing rays with high data volume and high fidelity, or low density with poor image quality. If we know an approximation of the scene geometry (see fig. 2, left), the rendering result can be improved by an appropriate depth-dependent warping of the nearest viewing rays as described in [5].

Having a sequence of images taken with a hand–held camera, in general the camera positions are not placed at the grid points of the viewpoint plane. In [5] a method is shown for resampling this regular two–plane parameterization from real images recorded from arbitrary positions (*re-binning*). The required regular structure is re-sampled and gaps are filled by applying a multi–resolution approach, considering depth corrections. The disadvantage of this re-binning step is that the interpolated regular structure already contains inconsistencies and ghosting artifacts due to errors in the scantily approximated geometry. To render views, a depth corrected look–up is performed. During this step, the effect of ghosting artifacts is repeated, so duplicate ghosting effects occur.

3.1 Representation with Recorded Images

Our goal is to overcome the problems as described in the last section by relaxing the restrictions imposed by the regular lightfield structure and to render views directly from the calibrated sequence of recorded images using local depth maps. Without loosing performance we directly map the original images onto a surface viewed by a virtual camera.

2–D Mapping: In this paragraph, a formalism for mapping image coordinates onto a plane A is described. The following approaches will use this formalism to map images onto planes and vice versa. We define a local coordinate system on A giving one point $\mathbf{a_0}$ on the plane and two vectors $\mathbf{a_1}$ and $\mathbf{a_2}$ spanning the plane. So each point $\mathbf{p}$ of the plane can be described by the coordinates x_A, y_A: $\mathbf{p} = (\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_0})\,(\mathbf{x_A}, \mathbf{y_A}, \mathbf{1})^{\mathbf{T}}$. The point $\mathbf{p}$ is perspectively projected into a camera which is represented by the 3×3 matrix $\mathbf{Q} = \mathbf{KR^T}$ and the projection center $\mathbf{c}$ (same notations as above). Matrix $\mathbf{R}$ is the orthonormal rotation matrix and $\mathbf{K}$ is an upper triangular calibration matrix. The resulting image coordinates x, y are determined by $\rho(x, y, 1)^T = \mathbf{Qp} - \mathbf{Qc}$. Inserting above equation for $\mathbf{p}$ results in

$$\rho \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \mathbf{Q}(\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_0} - \mathbf{c}) \begin{pmatrix} x_A \\ y_A \\ 1 \end{pmatrix} . \tag{4}$$

The value ρ is an unknown scale factor. Each mapping between a local plane coordinate system and a camera can be described by a single 3×3 matrix $\mathbf{B} = \mathbf{Q}(\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_0} - \mathbf{c})$.

We can extend our mapping procedure to re–project the image of one camera (with center $\mathbf{c_i}$) onto the plane followed by a projection into the other camera (with center $\mathbf{c_V}$). Then the whole mapping is performed by

$$(x_V, y_V, 1)^T = \mathbf{B_V B_i^{-1}}(\mathbf{x_i}, \mathbf{y_i}, \mathbf{1})^{\mathbf{T}} \quad . \tag{5}$$

The 3×3–matrix $\mathbf{B_V B_i^{-1}}$ describes the projective mapping from one camera to another via a given plane. Figure 2(right) shows this situation for two camera positions $\mathbf{c_V}$ and $\mathbf{c_i}$.

Mapping via global plane: We apply the previously described method of mapping an image via a given plane to create a virtual scene view directly from real ones. In a first approach, we approximate the scene geometry by a single plane A. This step seems to be really erroneous but as mentioned before, the lightfield–approach exactly supposes this approximation. In the most simple approach, we follow this method, although at regions, where the scene surface differs much from the plane A, a blurring effect will be visible. But in the next section we will improve our approach for refined geometric scene descriptions.

Following the lightfield approach, we have to interpolate between neighboring views to construct a specific virtual view. Considering the fact mentioned above

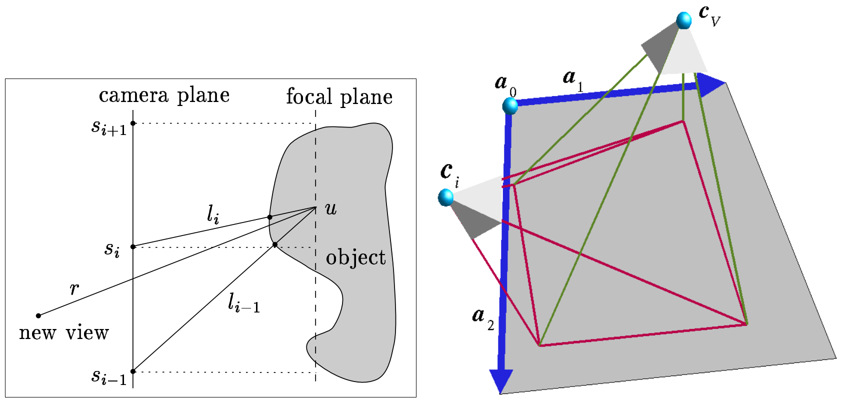

Fig. 2. Left: depth-dependent interpolation errors in the two-plane lightfield approach. The new viewing ray r is interpolated by the weighted sum of l_i and l_{i+1} from the adjacent cameras s_i and s_{i+1}. Since the real surface geometry deviates from the planar intersection point u at the focal plane, ghosting artifacts occur. Right: Projective mapping from one camera into another via a plane.

that the *nearest* rays give the best support to the color value of a given ray, we conclude that those views give the most support to the color value of a particular pixel whose projection center is closest to the viewing ray of this pixel. This is equivalent to the fact that those real views give the most support to a specified pixel of the virtual view whose projected camera centers are close to its image coordinate. We restrict the support to the nearest three cameras (see figure 3). To determine these three neighbors we project all camera centers into the virtual image and perform a 2–D triangulation. Then the neighboring cameras of a pixel are determined by the corners of the triangle which this pixel belongs to. The texture of such a triangle — and consequently a part of the reconstructed image — is drawn as a weighted sum of three textured triangles.

These textures are extracted from the original views by directly mapping the coordinates x_i, y_i of image i into the virtual camera coordinates x_V, y_V by applying equation 5.

To overlay these three textures, we calculate a weighted sum of the color values. Each triangle is weighted with factor 1 at the corner belonging to the projection center of the corresponding real view and with weight 0 at both others. In between, the weights are interpolated linearly similar to Gouraud–Shading, where the weights describe a plane ramp in barycentroic coordinates. Within the triangle, the sum of the three weights is 1 at each point. The total image is built as a mosaic of these triangles. Although this technique assumes a very sparse approximation of geometry, the rendering results show only few ghosting artifacts (see section 4) at those regions where the scene geometry differs much from the approximating plane.

Mapping via local planes: The results can be further improved by considering local depth maps. Spending more time for each view, we can calculate the approximating plane of geometry for each triangle in dependence on the actual view. As the approximation is not done for the whole scene but just for that part of the image which is seen through the actual triangle, we don't need a consistent 3-D model but we can use the — normally erroneous — local depth maps. The depth values are given as functions z_i of the coordinates in the recorded images $z_i((x_i, y_i, 1)^T)$. They describe the distance of a point perpendicular to the image plane. Using this depth function, we calculate the 3–D coordinates of those scene points which have the same 2–D image coordinates in the virtual view as the projected camera centers of the real views. The 3–D point $\mathbf{p_i}$ which corresponds to the real camera i can be calculated as

$$\mathbf{p_i} = \mathbf{z_i}(\mathbf{Q_i}\mathbf{d_i})\mathbf{d_i} + \mathbf{c_i} \quad , \tag{6}$$

where $\mathbf{d_i} = n(\mathbf{c_i} - \mathbf{c_V})$. The function n scales the given 3–D vector such, that its third component equals one. We can interpret the points $\mathbf{p_i}$ as the intersection of the line $\overline{\mathbf{c_V}\mathbf{c_i}}$ with the scene geometry (see figure 3). The 3–D coordinates of triangle corners define a plane which we can use to apply the same rendering technique as described above for one global plane.

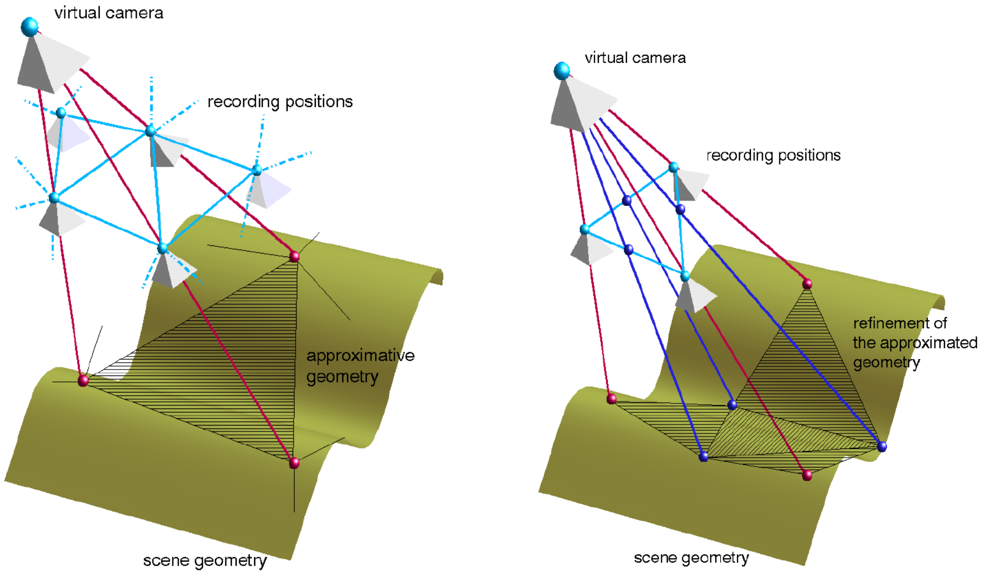

Fig. 3. Left: Drawing triangles of neighboring projected camera centers and approximating scene geometry by one plane for the whole scene or for one camera triple. Right: Refinement of triangulation by inserting new 3-D points corresponding to the midpoints of the triangle sides.

Refinement: Finally, if the triangles exceed a given size, they can be subdivided into four sub–triangles by splitting the three sides into two parts, each. We determine the 3-D points corresponding to the midpoint of each side by applying the same look–up method as used for radiance values to find the corresponding depth value. After that, we reconstruct the 3–D point using equation 6 and project it into the virtual camera resulting in a point near the side of the triangle. It is just in the neighborhood of the side, but this doesn't matter really. Merely, the triangulation structure will be changed slightly. One has to take care of avoiding inconsistencies caused by this look-up. For a given pair of neighboring views, the look-up always has to be done in the same depth map. A simple method is to do this look-up not for each triangle causing several look-ups for the same triangle side, but to determine the 3-D points for each pair of neighboring views in a preprocessing step. This also improves efficiency.

For each of these so created sub–triangles, a separate approximative plane is calculated in the above manner. Of course, further subdivision can be done in the same way to improve accuracy. Especially, if just few triangles contribute to a single virtual view, this subdivision is really necessary. This hierarchical refinement of the geometry can be performed adaptively depending on the required accuracy and available computational resources, hence allowing easy scalability towards scene complexity.

3.2 Scalable Geometry for Interpolation

The approach as discussed above can be used directly for scalable geometric scene approximation. The SFM reconstruction delivers local depth maps that contain the 3D scene geometry for a particular view point. However, sometimes the depth maps are sparse due to a lack of features. In that case, no dense geometric reconstruction is possible, but one can construct an approximate geometry (a mean global plane or a very coarse triangulated scene model). This coarse model is not sufficient for geometric rendering but allows depth compensated interpolation. In fact, the viewpoint-adaptive light field interpolation combined with approximative geometry combines to viewpoint-dependent texture mapping. Since only the nearby images are used for interpolation, the rendered image will be quite good even when only a very coarse geometry is used. The standard lightfield interpolation for example uses only a planar scene approximation that is not even adjusted to the mean scene geometry. Hence our approach is less distorting than standard lightfield rendering. The rendering will also be more realistic than standard texture mapping since we capture the reflectance characteristics of the scene.

The adaptive refinement of the geometry (starting with a mean global planar geometry and adapting to surface detail) can be used to control the amount of geometry that we need for interpolation. For every viewpoint the scene is divided into local planes until a given image quality (measured by image distortion) has been reached. On the other hand, one can select a fixed level of geometric subdivision based on the available rendering power of the texture mapping hardware.

For a given performance one can therefore guarantee that rendering is done in constant time.

This geometric scalability is very useful in realtime environments where a fixed frame rate is required, or in high realism rendering where imaging quality is premium.

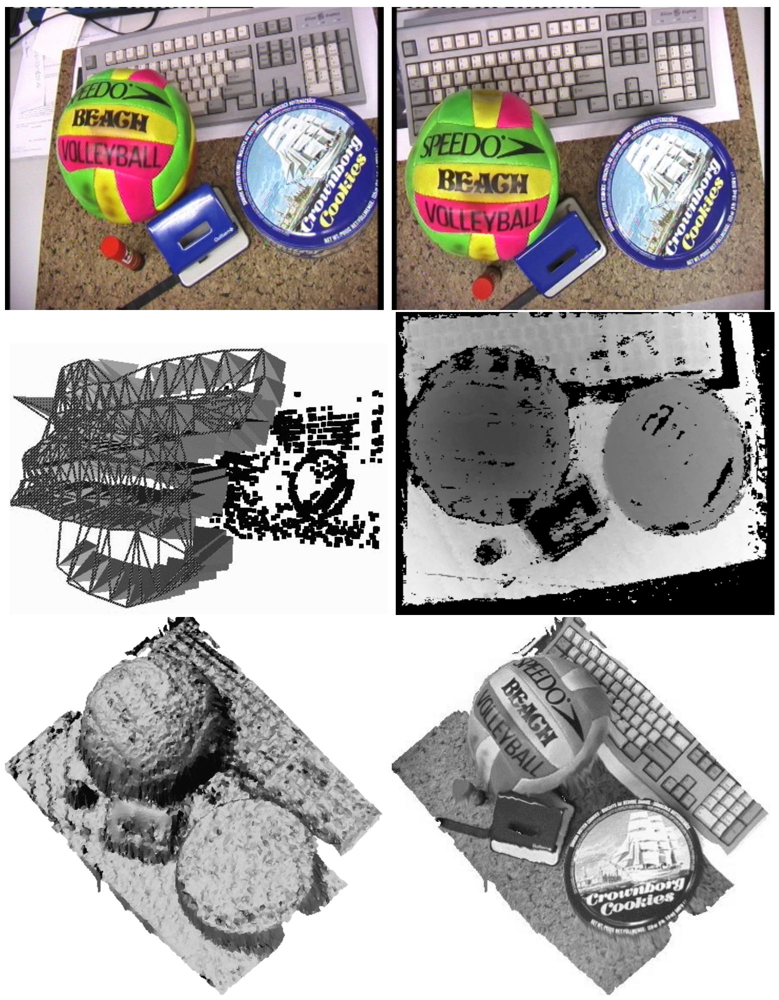
Fig. 4. Top: Two images from hand-held office sequence. Please note the changing surface reflections in the scene. Middle: Camera tracking with viewpoint mesh(left) and depth map from a specific viewpoint (right). Bottom: 3D surface model of scene rendered with shading (left) and texture (right).

4 Experimental results

We tested our approach also with an uncalibrated hand-held sequence. A digital consumer video camera (Sony DCR-TRV900 with progressive scan) was swept freely over a cluttered scene on a desk, covering a viewing surface of about 1 m^2. The resulting video stream was then digitized on an SGI O2 by simply grabbing 187 frames at more or less constant intervals. No care was taken to manually stabilize the camera sweep.

Fig. 4 (top) shows two images of the sequence. Fig. 4 (middle, left) illustrates the zigzag route of the hand movement as the camera scanned the scene. The viewpoint mesh is irregular due to the arbitrary hand movements. The black dots represent the reconstructed 3D scene points. From the calibrated sequence we can compute any geometric or image based scene representation. As an example we show in fig. 4 (bottom) a geometric surface model of the scene with local scene geometry that was generated from the depth map (see fig. 4 middle, right).

Fig. 5 shows different refinement levels of the same geometry as viewed from a particular camera viewpoint. Even with this very rough approximation, very realistic view interpolation can be achieved.

Fig. 6 shows rendering of the same scene without depth compensation (left) and with depth compensation using geometric refinement (mesh 1x subdivided, middle). Even without geometry the rendering looks good, but due to the missing depth compensation some ghosting artifacts occur. This is the result achievable with the standard lightfield approach, but already exploiting the general viewpoint mesh calibration. With geometry compensation the rendering is improved substantially and the ghosting artifacts disappear. Note that we utilized a very coarse geometrical approximation only as displayed in fig 5 (top right) but still achieve high rendering quality.

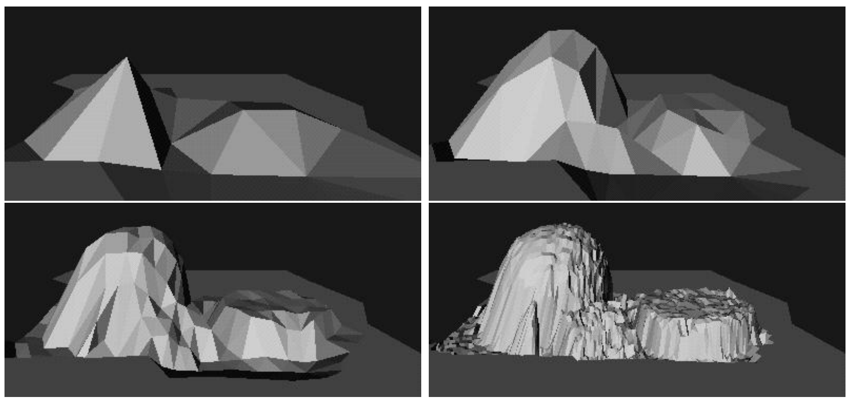

Fig. 5. Viewpoint geometry for depth-compensated interpolation with with different levels of adaptive refinement (level of mesh subdivision: 0,1,2, 4 (top left to bottom right).

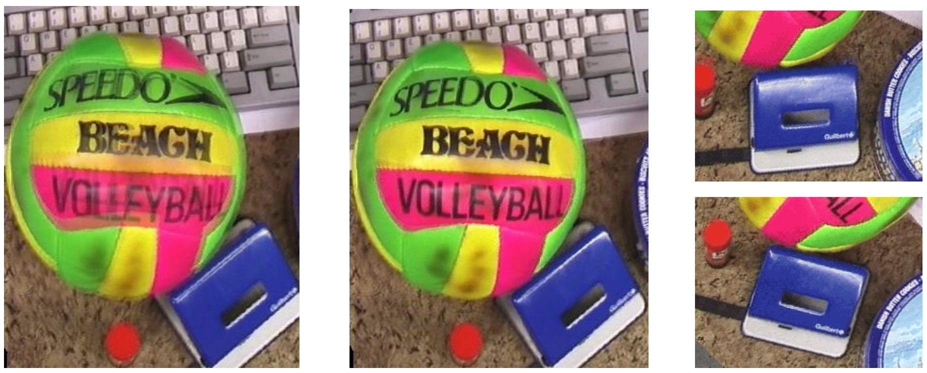

Fig. 6. Top: Novel views rendered from the viewpoint mesh without (left) and with (middle) depth-compensation. Only a coarse geometrical approximation is used (see upper right image of fig. 5). Right: Two views of the scene rendered from different viewpoints with changing reflections. The rendering quality is very high due to the natural appearance of the reflections.

The main advantage of lightfield rendering is that the rendering is *local*, meaning that all depth and color information for a pixel is taken from the three nearest camera images only. This allows changes in surface reflectivity when viewing the scene from different angles. Some rendering results with surface reflections are shown in fig. 6 (right). The same part of the scene is rendered from different viewpoints, demonstrating that the reflections are preserved and the images appear very natural.

5 Conclusions

We have presented a system for calibration, reconstruction, and plenoptic rendering of scenes from an uncalibrated hand-held video camera. The calibration exploits the proximity of viewpoints by building a viewpoint mesh that spans the viewing sphere around a scene. Once calibrated, the viewpoint mesh can be used for image-based rendering or 3D geometric modeling of the scene. The image-based rendering approach was discussed in detail and a new rendering approach was presented that renders directly from the calibrated viewpoint using depth-compensated image interpolation. The level of geometric approximation is scalable, which allows to adapt the rendering to the given rendering hardware. For the rendering only standard planar projective mapping and Gouraud-weighting is employed which is available in most rendering hardware.

Acknowledgments

We acknowledge support from the German Research Foundation (DFG) under the grant number SFB 603. Thanks to Luc Van Gool for his support.

References

1. P. Beardsley, P. Torr and A. Zisserman: 3D Model Acquisition from Extended Image Sequences. *ECCV 96*, LNCS 1064, vol.2, pp.683-695.Springer 1996.
2. P. Debevec, Y. Yu, G. Borshukov: Efficient View-Dependent Image-Based Rendering with Projective Texture Mapping. Proceedings SIGGRAPH '98, ACM Press, New York, 1998.
3. L.Falkenhagen: Hierarchical Block-Based Disparity Estimation Considering Neighborhood Constraints. Intern. Workshop on SNHC and 3D Imaging, Rhodes, Greece, Sept. 1997.
4. O. Faugeras: What can be seen in three dimensions with an uncalibrated stereo rig. *Proc. ECCV'92*, pp.563-578.
5. S. Gortler, R. Grzeszczuk, R. Szeliski, M. F. Cohen: The Lumigraph. Proceedings SIGGRAPH '96, pp 43–54, ACM Press, New York, 1996.
6. R. Hartley: Estimation of relative camera positions for uncalibrated cameras. *ECCV'92*, pp.579-587.
7. R. Koch, M. Pollefeys, and L. Van Gool: Multi Viewpoint Stereo from Uncalibrated Video Sequences. *Proc. ECCV'98*, Freiburg, June 1998.
8. R. Koch, M. Pollefeys, B. Heigl, L. Van Gool, H. Niemann: Calibration of Handheld Camera Sequences for Plenoptic Modeling. Proc. of ICCV'99, Korfu, Greece, Sept. 1999.
9. R. Koch, M. Pollefeys and L. Van Gool: Robust Calibration and 3D Geometric Modeling from Large Collections of Uncalibrated Images. Proceedings DAGM'99, Bonn, Sept. 1999.
10. M. Levoy, P. Hanrahan: Lightfield Rendering. Proceedings SIGGRAPH '96, pp 31–42, ACM Press, New York, 1996.
11. L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system", *Proc. SIGGRAPH'95*, pp. 39-46, 1995.
12. M. Pollefeys, R. Koch, M. Vergauwen and L. Van Gool: Metric 3D Surface Reconstruction from Uncalibrated Image Sequences. In: 3D Structure from Multiple Images of Large Scale Environments. LNCS Series Vol. 1506, pp. 139-154. Springer-Verlag, 1998.
13. M. Pollefeys, R. Koch and L. Van Gool: Self-Calibration and Metric Reconstruction Inspite of Varying and Unknown Intrinsic Camera Parameters. Int. Journal of Computer Vision 32(1), 7-27 (1999), Kluver 1999.
14. P.H.S. Torr: Motion Segmentation and Outlier Detection. PhD thesis, University of Oxford, UK, 1995.