Volume xx (200y), Number z, pp. 1–14

GPU-Based Spherical Light Field Rendering with Per-Fragment Depth Correction

S. Todt[1], C. Rezk-Salama[1], A. Kolb[1], and K.-D. Kuhnert[2]

[1]Computer Graphics Group, University of Siegen, Germany
[2]Real-Time Learning Systems, University of Siegen, Germany

Abstract

Image-based rendering techniques are a powerful alternative to traditional polygon-based computer graphics. This paper presents a novel light field rendering technique which performs per-pixel depth correction of rays for high-quality reconstruction. Our technique stores combined RGB and depth values in a parabolic 2D texture for every light field sample acquired at discrete positions on a uniform spherical setup.Image synthesis is implemented on the GPU as a fragment program which extracts the correct image information from adjacent cameras for each fragment by applying per-pixel depth correction of rays.

We show that the presented image-based rendering technique provides a significant improvement compared to previous approaches. We explain two different rendering implementations which make use of a uniform parametrization to minimize disparity problems and ensure full six degrees of freedom for virtual view synthesis. While one rendering algorithm implements an iterative refinement approach for rendering light fields with per pixel depth correction, the other approach employs a raycaster which provides superior rendering quality at moderate frame rates.

GPU based per-fragment depth correction of rays, used in both implementations, helps reducing ghosting artifacts to a non noticeable amount and provides a rendering technique that performs without exhaustive pre-processing for 3D object reconstruction and without real-time ray-object intersection calculations at rendering time.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Image-Based Rendering

1. Introduction

Research in the field of image-based rendering (IBR) has contributed a variety of algorithms for real-time generation of photorealistic images of complex scenes. Generally, these algorithms synthesize virtual views based on a collection of pre-acquired input images. Unlike the polygonal rendering pipeline, IBR techniques are almost independent of scene complexity, thus providing an effective solution for photorealistic image synthesis at a predictable computational cost.

Light field rendering (LFR) techniques focus on the the reconstruction of complex material properties and lighting environments for an arbitrary 3D scene. The effectiveness of an LFR approach can be evaluated by well-known criteria appointed with the introduction of *Light Field Rendering* by Levoy and Hanrahan [LH96]. Light field techniques are categorized by the parameterisations applied for light field representation, the method of light field generation and the robustness and accuracy of the rendering technique.

We present a light field approach that employs a spherical light field parameterisation and takes advantage of combined per-pixel color and depth information to generate high-quality virtual views in real-time (see Figure 1). The approach described in this paper provides a set of new features that help making LFR more flexible for usage in a wide area of applications. These features are:

- Uniform spherical parameterisation yielding six degrees of freedom (DOF) for virtual viewpoint selection
- High quality light field synthesis without noticeable ghosting artifacts, providing reconstruction of visual features from meso structures and concavities
- Correct per-pixel depth evaluation for reconstructed light field views

(a) (b) (c) (d)

Figure 1: *a: Light field of the Lucy model reconstructed from 42 samples sampled at* 512×512 *rendered at 77.83 fps. b: Light field of Tie-Fighter model reconstructed from 42 samples sampled at* 512×512 *rendered at 78.3 fps. c: Composed light fields of Tie-Fighter and Tie-Bomber. d: Light field rendered at different LOD, with superimposed spherical approximations.*

- Flexible light field compositing capabilities with dynamic scene geometry
- Level of detail to account for varying object distance and visibility
- Synchronous rendering of multiple light fields to generate complex scenes
- Efficient generation of light fields from complex 3D scenes and free-hand acquisition of physical objects

The remainder of the paper is structured as follows. In Section 2 we give an overview of relevant related work, including a survey on existing light field approaches. In Section 3 we introduce our light field representation and show the benefits with respect to combined RGB and depth data. Section 4 describes our novel techniques for GPU-based LFR with per-pixel depth refinement. In Section 5 we explain how light fields can be efficiently generated from 3D geometry. Section 6 evaluates the technique with respect to image quality and rendering performance. In Section 7 we draw conclusions and comment on future work.

2. Related Work

The use of environment maps to capture the incoming light in a texture map [BN76, Gre86] has inspired several successive IBR techniques. As environment maps record the incident light arriving from all different directions at a single point in space, each individual environment map of a scene describes a concrete sample of the *plenoptic function* [MB95] as desribed by Adelson and Bergen [AB91]. Given a set of discrete samples of the plenoptic function, the goal of LFR techniques is to generate a continuous representation of that function. Based on this assumption the plenoptic function can then be expressed as function on the space of oriented lines. The 4D function of position and direction is described as *light field* [LH96]. Different discretisation methods of the 4D function lead to a variety of different parameterisations.

Accounting for the vast number of publications, we restrict our survey to those implementations which are most relevant to our approach, i.e. approaches that provide 6 DOF and employ depth correction of rays to eliminate incoherent light field information during light field synthesis. For a detailed survey of LFR techniques the reader is referred to [SCK07].

2.1. Two-Plane parameterisations

Light field rendering as proposed in the original paper by Levoy and Hanrahan [LH96] restricts objects to lie within a convex cuboid bounded by two planes, the camera plane and the image plane. Rays describing the observation space are parameterized by their intersection points with these two planes leading to the 4D plenoptic function. A set of such two planes is called a *light slab*.

Reconstruction of the light field for a ray described by the intersection points with the two planes $P(u,v,t,s)$ is performed by quadri-linear interpolation of the images taken by the cameras positioned around the intersection point on the camera plane(u,v). Virtual views, however, are limited to a small viewing cone around the dominant direction of rays, parameterized by the two planes. Viewing rays not intersecting one of the planes cannot be reconstructed. With the *Lumigraph*, Gortler et.al. [GGSC96] extend the two plane approach by employing six *light slabs* for light field parameterisation, providing 6 DOF for viewpoint selection. For virtual views reconstructed from light field samples at the edges of the six sided *light slab* setup, however, discontinuity artifacts appear, due to the non uniform sampling of rays in those regions [CLF98].

It has been shown that LFR as described above will provide satisfactory rendering results, if the observed object is positioned exactly on the image plane. In the general case, noticeable ghosting artifacts will appear due to incoherent light field information for adjacent rays. Such incoherency is due to rays hitting the object at a surface point far from the

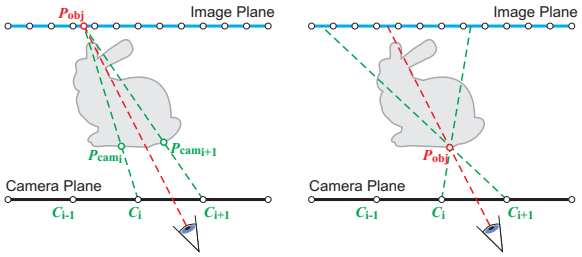

Figure 2: *Depth correction of rays. Without depth correction, the intersection points observed from adjacent cameras do not necessarily correspond to an identical surface point. With depth correction, camera rays passing through a common surface point are used for interpolation.*

image plane, thus resulting in deviating intersection points on the image plane (see Figure 2).

The *Lumigraph* [GGSC96] uses a proximity polygonal representation to overcome incoherency by applying a depth correction of the rays. Discrete depth values are estimated by a raycasting approach for ray-object intersection at rendering time. The discrete depth values are evaluated during image synthesis for depth correction of rays, leading to a significant improvement in image quality.

2.2. Spherical Parameterisations

Spherical LFR techniques overcome the problem of discontinuities by parameterizing rays using a spherical representation. Several flavours of spherical parameterisations have been published in the past.

Spherical light fields [IPL97] use intersection with a positional sphere and a directional sphere placed at the positional intersection point. Using *Sphere-Sphere* parameterisation [CLF98] rays are determined by intersecting the same sphere twice. *Sphere-Plane* parameters are evaluated as the intersection with a plane perpendicular to the ray positioned at the center and its normal direction yielding a position on the surrounding sphere [CLF98].

Spherical light fields avoid discontinuities by uniform parameterisation. The virtual viewpoint can be chosen freely with 6 DOF. For light field reconstruction the sphere is rendered as a smoothly shaded polygonal mesh applying camera images for bilinear interpolation. If an optimal constant depth is stored with each image for the *Sphere-Plane* parameterisation, per sample depth information can be used for depth correction of rays during light field synthesis yielding results comparable to the *Lumigraph* rendering approach [CLF98].

2.3. Unstructured Light Fields

Unstructured LFR techniques address the problem of fixed camera-space parameterisation. During light field acquisition, actual camera parameters are stored with every light field sample to define the final parameterisation space. Image synthesis then is steered by camera parameters to query image data and to establish camera blending weights for interpolation.

Unstructured Lumigraph Rendering [BBM*01] is based on a variation of the light slab setup as used in *two-plane light field rendering*. Each light field sample is represented as a single *light slab*. The complete light field data set consists of a collection of individual *light slabs*. Per fragment blending weights of different sample cameras are calculated based upon a polygonal proxy representation of the scene for depth correction at light field synthesis. Freedom of choice for the virtual viewpoint selection and the quality of the final rendering is steered by the uniformity of sample camera distribution and the quality of the polygonal geometric approximation.

Free Form Light Field Rendering [SVSG01] can be viewed as a generalization of the two-plane rendering approach. Here a connected camera mesh is used for light field representation instead of a set of individual *light slabs*. For acquisition, camera positions can be chosen to lie anywhere outside the convex hull of the observed object. A camera mesh connecting the camera positions used for acquisition is built from the camera sample positions. For image synthesis, the smooth shaded camera mesh is rendered by projecting each of its faces onto the image planes of the n nearest cameras that contribute to that patch.

Schirmacher et al. [SVSG01] apply a depth correction for *Free Form Light Fields* by successively subdividing the camera mesh at rendering time. Subdivision is steered by evaluating per-vertex depth information. The camera mesh is subdivided until a homogenous depth value for all of the triangle's vertices is established or the triangle's area reaches the size of a fragment. Rendering performance is dependent on the method used for per-vertex depth estimation and the object's depth complexity which is the actuating variable for the subdivision algorithm's runtime. Using depth maps have been identified to be an efficient solution to depth correction of rays by means of rendering-performance [SVSG01]. Rendering quality is closely related to the uniformity of the camera mesh and the complete covering of suitable viewing directions needed for reconstruction purposes.

3. Spherical Light Fields with Per-Pixel Depth

The LFR approaches presented above provide a powerful selection of image-based rendering tools. Each of them describes an individual technique to either improve representation, quality or rendering performance of light fields. How-

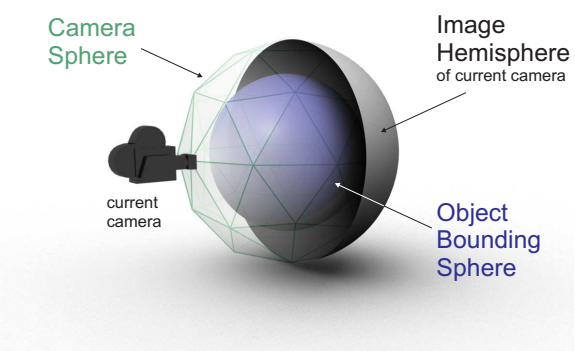

Figure 3: *Sphere-Hemisphere parameterisation of the light field. The object is enclosed in the blue bounding sphere. Virtual cameras are positioned at the vertices of the camera sphere (green). Each camera is recording the opposing hemisphere.*

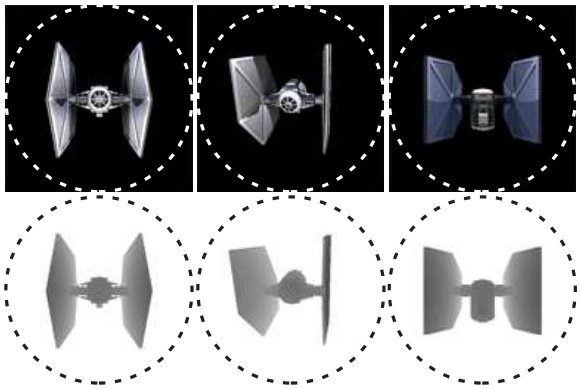

Figure 4: *Three image samples taken for a spherical light field with 42 cameras. Each image represents a parabolic mapping of the hemisphere for color (top row) and depth (bottom row).*

ever, each of these techniques also show some aspects that adversely affect another light field issue.

Our LFR approach applies a uniform spherical parameterisation to overcome discontinuity artifacts as observed for the cubic arrangement of *light slabs* and which potentially occur for unstructured light fields for a non uniform sampled scene. The spherical parameterisation provides full 6 DOF for virtual view point selection and applies a parabolic image space parameterisation to reduce distortions that effect rendering quality in the *sphere-plane* approach. Our rendering method with per-pixel depth correction of rays is inspired by ideas of Schirmacher et al. to use depth maps for ray correction [SVSG01] instead of a constant depth per-view as presented for *sphere-plane* parameterisation to achieve optimized depth correction. The approach directly works on combined RGB and depth (RGBz) image data to realize an efficient light field representation and a high-quality LFR technique. The rendering technique performs without the need for polygonal proxy geometry generation or complex mesh processing as applied by free form light fields.

The parameterisation is outlined in Figure 3. Sample cameras are located at each vertex of a regular, uniform triangulation of the sphere. For each camera an RGBz raster image of the opposing hemisphere is stored using a parabolic parameterisation. The RGBz images for the individual cameras are stored on graphics hardware by writing the depth value in the alpha channel with every pixel. A standard color depth of 8 bit per RGBA channel is sufficient. This data representation is open for commodity texture compression algorithms. Light field reconstruction is performed by rendering the smooth shaded spherical proxy using a customized fragment program which performs effective per-pixel depth correction.

3.1. Spherical Camera Space parameterisation

Uniform camera space parameterisation is achieved by arranging equally spaced sample camera positions on a spherical approximation. Thus a good spherical polygonal approximation has to be generated to map the sample positions to discrete 3D positions for efficient storage and rendering. Platonic solids are known to be well suited for approximation of a sphere [Gas83]. The most complex platonic solid is the icosahedron, a 20-sided polyhedron with identical faces and vertex valences, providing an absolutely uniform distribution of vertices on the unit sphere. The icosahedron is thus a good choice as a generator for uniform spherical approximations [CLF98].

For further refinement we apply a recursive interpolatory subdivision scheme on the solid mesh. With every iteration each triangle is divided into four (nearly) equilateral spherical triangles. We adjust the positions of the vertices resulting from subdivision by projecting the vertices on the unit sphere to maintain a solid spherical approximation. Applying the subdivision process m times, we obtain a spherical approximation with $20 \cdot 4^m$ faces. In practice, we chose $m = 1$ or $m = 2$, yielding 80 or 320 faces and 42 or 162 vertices, respectively.

The spherical approximation is stored explicitly as an indexed face set. With each vertex, a 4×4 viewing matrix is stored which represents the transformation of world to camera coordinates. This matrix is interpreted as extrinsic camera parameters for light field acquisition as well as for reconstruction purposes.

3.2. Parabolic Image Space Parameterisation

For each sample camera defined by the spherical camera space parameterisation we store a 2D RGBz texture of the

Images	Resolution	Uncompressed	S3TC DXT3	S3TC DXT3 & zipped
12	256×256	3.1 MB	0.7 MB	0.2 MB
12	512×512	12.3 MB	3.0 MB	0.5 MB
42	256×256	10.8 MB	2.7 MB	0.6 MB
42	512×512	43.0 MB	10.8 MB	1.5 MB
162	256×256	41.5 MB	10.4 MB	1.9 MB
162	512×512	165.9 MB	41.5 MB	5.9 MB
642	256×256	164.4 MB	41.1 MB	7.5 MB
642	512×512	657.5 MB	164.4 MB	23.5 MB

Table 1: *Sizes of typical light fields with and without compression*

opposite hemisphere. As displayed in Figure 3, the camera sphere must be larger than the bounding sphere of the object's bounding sphere by a factor of $\sqrt{2}$ to ensure that all rays from the current camera through the object boundary sphere will intersect the opposite hemisphere.

Parabolic environment maps [HS99] provide the ideal solution for recording the hemisphere. They consume essentially less storage compared to cube maps [Gre86] and cause less distortion than both cube maps and spherical maps [MH84]. In practice, we chose a resolution of 256×256 or 512×512 pixels for each RGBz texture. Examples of source images for the light field are displayed in Figure 4.

The depth value stored in the alpha channel of individual parabolic texture maps is calculated as the distance z' from the intersection point with the object's bounding sphere to the object divided by the length of the bounding sphere's ray secant $z_{\max}$, according to Figure 5. This results in a depth value between 0 and 1, which can efficiently be stored as 8 bit alpha value in the alpha channel [POJ05]. Storing the depth value as the fractional part of the secant length also allows for an efficient calculation of the object intersection point in the fragment program (See Section 4). In Section 5, we will see that such a representation allows us to easily generate synthetic light fields from a highly tessellated geometry.

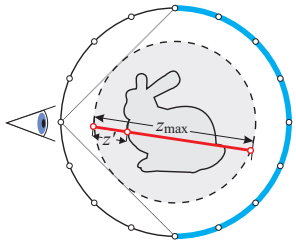

Figure 5: *The depth value is obtained by dividing the bounding-object distance z' by the bounding sphere's secant length z_{max}.*

3.3. Texture Compression

To reduce the amount of graphics memory consumed by the parabolic maps, commodity hardware-accelerated texture compression schemes can be employed. S3TC texture compression [INH99] offers an effective and easy way to compress the light field images. The DXT5 algorithm works on RGBA textures and achieves a compression ratio of 4:1. In our storage scheme, however, the alpha portion contains the depth values, which turn out to be sensitive to compression artifacts especially at object boundaries. To obtain best results for light field reconstruction we used the DXT3 algorithm. DXT3 works on RGBA data as well, but does not compress the alpha portion. DXT3 still offers a total compression ratio of 4:1.

A light field acquired using our spherical parameterisation with 162 images and a resolution of 512×512 pixels consumes 165.9 MB without compression. In comparison, a DXT3 compressed light field of the same dimensions consumes only 41.5 MB of memory. For storing and transmission, the total size of a light field data set can be further reduced significantly by applying standard *ZIP* compression techniques. With both *ZIP* and DXT3 texture compression the same light field is reduced to a average size of about 6 MB (See Table 1).

4. Light Field Rendering

The light field is rendered using a customized fragment program which requires a graphics processor that supports loop and conditional branch instructions according to shader model 3.0. We have implemented two different rendering techniques, both with per-pixel depth correction. The first one is an iterative refinement technique, which is more efficient in terms of runtime performance. The second one is a ray casting technique which is less efficient but more accurate.

In both techniques the light field is rendered by rasterizing the front faces of the camera sphere with respect to the virtual viewpoint. The viewpoint is restricted to be located outside the camera sphere to avoid the vertices of the front faces being culled by frustum culling. The polygonal approxima-

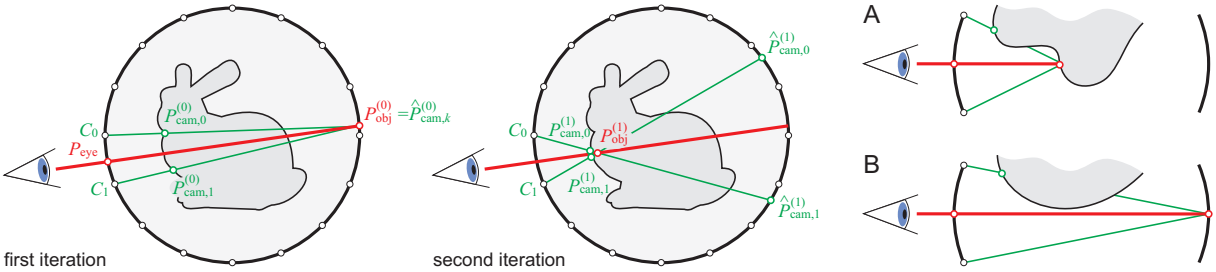

Figure 6: *Left and Middle: First and second step of the iterative depth refinement. Right: Iterative refinement fails at silhouette edges and locations where the correct object point is not visible from all adjacent cameras.*

tion of the camera sphere is rasterzed with each vertex corresponding to a pre-defined camera sample position. For each triangle, the parabolic RGBz texture images and the viewing matrices **M** of the three cameras are bound as uniform parameters to the fragment program. In practice, the RGB and depth information of the parabolic map are bound as separate texture objects. While the RGB texture can be linearly interpolated, noticeable render artifacts at the silhouettes appear when interpolating the depth information. Using the nearest neighbour texture lookup scheme ensures appropriate depth information per pixel.

4.1. Iterative Refinement

LFR techniques presented in the past which apply a depth correction of rays [SVSG01, VG01, VG00] are based on per-vertex depth evaluation for light field synthesis. For each vertex the depth is evaluated based on the depth map stored with individual light field samples. For each fragment to be rendered, color and depth information is linearly interpolated along the edge of a view triangle. This approach, however, is only valid, if the depth values vary linearly between the vertices [VG00]. Vogelgsang and Greiner [VG01] propose a subdivision scheme to adjust the mesh topology according to the vertex depth variance within a triangle. The main problem when sampling the scene by subdividing the view triangle is the possibility of missing geometric features. Thus this recursive subdivision is pursued until a triangle collapses to fragment size or a minimum depth variance is reached. Special care must be taken to avoid the mesh from degenerating.

Our iterative refinement technique evaluates depth per-fragment for ray correction to overcome the computational effort of mesh refinement, which may potentially result in degenerated meshes. The smooth shaded polygonal approximation of the sphere is rendered directly for light field synthesis without the need for additional mesh refinement and exhaustive vertex processing. The iterative refinement technique is outlined in Figure 6. When a triangle is rasterized, each fragment corresponds to a unique position P_{eye} on the camera sphere. In the first step, the fragment program calculates the intersection point of the viewing ray with the camera sphere to obtain a first estimate of the object intersection point $P^{(0)}_{\text{obj}}$. The superscript in this notation refers to the iteration count.

The calculated intersection point is transformed into the viewing space of camera k according to

$$S^{(i)}_k \;=\; \mathbf{M}_k\, P^{(i)}_{\text{obj}} \qquad \text{with} \quad k \in \{0,1,2\}. \tag{1}$$

This is done for each of the three adjacent cameras that correspond to the original triangle's vertices. The sphere intersection points S_k can now be converted to parabolic texture coordinates (u,v), according to

$$\begin{pmatrix} u \\ v \end{pmatrix}_k = \frac{1}{2} \begin{pmatrix} \frac{s_x}{1+s_z}+1 \\ \frac{s_y}{1+s_z}+1 \end{pmatrix}_k \quad \text{with} \quad S_k = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}_k . \tag{2}$$

RGBz samples are obtained from the parabolic texture maps corresponding to the three cameras. The depth value z is extracted form the alpha portion and is used to calculate the camera's local estimate $P^{(i)}_{\text{cam},k}$ for the object intersection point:

$$P^{(i)}_{\text{cam},k} = z \cdot C_k \;+\; (1-z)\, \hat{P}^{(i)}_{\text{cam},k} \tag{3}$$

with $\hat{P}^{(i)}_{\text{cam},k}$ being the object intersection point $P^{(i)}_{\text{obj}}$ projected onto the sphere using C_k as center of projection. Note that Equation 3 is a simple linear interpolation, because z stores the depth value as fractional part of the secant length (see Figure 5).

An improved estimate for the object intersection point P_{obj} can now be found by projecting the three local camera distances onto the viewing ray and calculating the average, according to:

$$P^{(i+1)}_{\text{obj}} = P_{\text{eye}} \;+\; \frac{1}{3} \sum_{k=0}^{2} \left((P^{(i)}_{\text{cam},k} - C_k) \cdot \mathbf{r} \right) \mathbf{r} \tag{4}$$

with $\mathbf{r}$ being the normalized direction of the original viewing ray (red line).

As illustrated in Figure 6 *(middle)*, the iteration can be pursued several times by projecting the updated object point

(a) (b) (c) (d)

Figure 7: *Iterative refinement rendering results. a: Original polygonal Phlegmatic Dragon model (360k polygons) rendered at 275 fps. b: LFR from a light field data set containing 162 input samples at* 512×512 *resolution, rendered at 95.6 fps. c: Light field reconstructed at 72.5 fps from a data set containing 642 samples. d: Light field of the Lucy model, synthesized from 642 samples at 75.3 fps*

$P_{\text{obj}}^{(i+1)}$ onto the sphere using the camera vertices C_k as center of projection. The resulting intersection points are successively transformed into camera coordinates to establish depth values, to determine improved local estimates according to Equation 3 and to eventually calculate an improved intersection point according to Equation 4. The procedure terminates if the desired accuracy is achieved or a maximum number of iterations is reached. In practice a maximum number of iterations of 5 turns out to be sufficient.

The final color of the fragment is calculated as a weighted sum of the RGB values of the different cameras from the final iteration. The weights for each camera correspond to the barycentric coordinates of the fragment with respect to the original triangle.

Although this scheme allows the actual geometry intersection point to be approached quite quickly, unfortunately, the iterative refinement fails in certain situations. Such cases are illustrated in Figure 6 *(right)*.

- Case A shows a situation where the geometry is hit inside a concavity of the object, resulting in an intersection point that is not visible from at least one of the adjacent cameras.
- Case B illustrates a situation where the viewing ray does not hit the object at all, while at least one adjacent camera reports an intersection.

These cases cannot be handled exactly and will inevitably result in visible ghosting artifacts at sharp edges, e.g. silhouettes and concavities. In order to attenuate such artifacts, we can examine the camera estimates $P_{\text{cam},k}^{(i)}$ after a few iterations. If the three estimates are still highly divergent, we discard the point with the closest distance to the camera and proceed with the residual two cameras. If no similar local estimates can be achieved within the next few iterations, the averaging in Equation 4 is replaced by a maximum operation. This procedure, however, cannot completely eliminate the visual artifacts. If the ghosting is still too strong the only effective countermeasure is increasing the number of cameras, or the image resolution.

Figure 7 shows rendering results from traditional polygonal based rendering in comparison to light field rendering results. The *Phlegmatic Dragon* shown in Figure 7a was chosen as original polygonal model from which the light fields were generated from. The model contains detailed micro and meso structures and the chosen material results in clear specular highlights when lit by directional light. Thus, this model is perfectly suited to evaluate the quality of the light field rendering approach, as structures and highlights have been identified to be critical for evaluating the quality of light field rendering approaches [LH96, GGSC96]. Good results are achieved for spherical light fields sampled from 162 or 642 input images at image resolutions of 256×256 or 512×512. The inner region of the object is synthesized at high quality. For this central region of the light field meso (see Figure 7 b,c) and micro structures (see Figure 7 d) are precisely reconstructed. Minor ghosting artifacts, however, resulting from unresolved depth correspondences as shown in Figure 6 *(right)* are visible at the silhouette. Using direct per-fragment depth evaluation within the customized fragment program allows to synthesize light fields at real-time frame rates of up to 110 fps (see Table 2).

4.2. Raycasting Approach

The raycasting approach represents a less efficient, yet a simpler and more accurate solution to depth refinement which allows silhouettes, sharp edges and concavities to be reconstructed precisely without ghosting artifacts. When a triangle is rasterized, the fragment program calculates the viewing ray, which corresponds to the fragment. We then calculate the intersection point of the viewing ray with the bounding sphere of the object.

From this position, we sample the viewing ray iteratively at a fixed step size, as shown in Figure 8. At each step, we

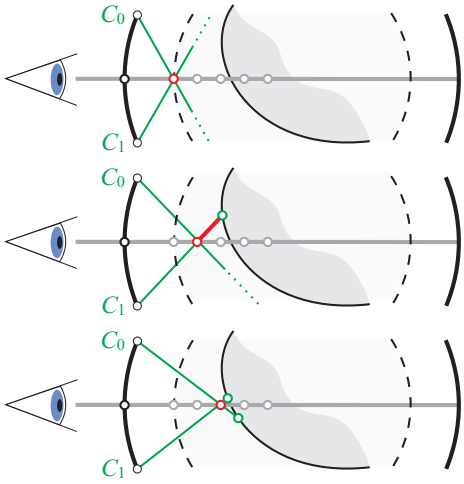

Figure 8: *The raycasting approach samples the viewing ray stepwise at adjacent positions starting at the intersection point with the object's bounding sphere.*

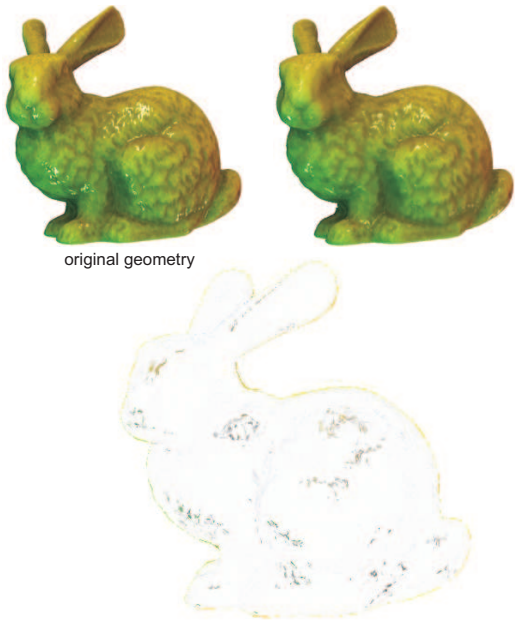

Figure 9: *Top left: Original Stanford Bunny model (70k polygons) rendered at 375 fps. Top right: Light Field rendered from 42 input samples rendered at 79.1 fps. Bottom: RGB difference image showing the pixel difference.*

validate the assumption that the current ray position is the actual object intersection point P_{obj}. We project this point onto the camera sphere using the adjacent camera positions C_k as center of projection. The resulting three intersection points with the sphere are transformed into the viewing coordinate system of the corresponding camera and converted to parabolic coordinates. Depth values are obtained from the corresponding parabolic texture maps and local estimates $P_{\text{cam},k}$ are calculated for each camera according to Equation 3. We then compare these points to the position of the current ray sample. If one of the local estimates is equal to the ray position within a given tolerance, we immediately stop the ray sampling. This means that we have found at least one camera that reliably observes an object intersection at exactly the ray position.

We now compare the intersection points obtained by the other two cameras. If they are also equal to the ray position within the given tolerance, we can assume that all cameras observe the same point and we use the barycentric weights of the fragment to calculate the final color for that pixel as outlined above. However, if the intersection point for one camera is far away from the ray position, we can assume that we are in one of the situations outlined in Figure 6 *(right)*. In this case we discard the color information for the respective camera by setting the corresponding barycentric weight to zero. Afterwards, we normalize the weights again and calculate the final color as a weighted sum.

In contrast to raycasting implementations presented for rendering of displacement maps [WWT*03, WTS*05] and relief textures [OBM00, POJ05, ACB*07], our raycasting approach benefits from multiple input depth maps to handle such ambiguous situations. The raycasting approach presented by Wang et.al. [WWT*03, WTS*05] works on a single depth map and fails for self-occlusion situations and thus cannot handle large variations in depth [BD06]. As *Relief texture mapping* is designed to achieve visually appealing results for small-to-medium scale relief [BD06] it cannot handle concavities and holes [OBM00]. Our approach precisely reconstructs the visual appearance from small structures and large variations in depth from a small amount of input images. Concavities and holes can be reconstructed as depth information can be retrieved from three adjacent depth maps to determine the object intersection. Silhouettes and sharp edges are precisely reconstructed (See Figure 1, 9, and 11). The ray-object intersection point, established by raycasting, can be used to take advantage of *OpenGL's z-Buffer* functionality. Light field renditions can be composed with arbitrary complex (polygonal) scenes by applying the precise silhouette reconstruction in combination with the per-fragment depth values. The compositing capabilities also allow multiple occluding light fields to be accurately displayed. Inter-object occlusions for polygonal objects and light fields are properly handled for dynamic and static scenes (see Figure 1 c, Figure 12).

Figure 9 clearly shows the benefit of raycasting in terms of image quality compared to iterative refinement shown in Figure 7. The micro and meso structures of the polygonal *Stanford Bunny* model are reconstructed precisely from only

42 input samples as depicted in the difference image in Figure 9, bottom. Minor pixel errors occur at the silhouette of the object. Specular highlights are reconstructed quite well from the sparse set of input images. Increasing the number of input samples of course will further improve the reconstruction of specular highlights. Light fields sampled from 162 sample positions can be synthesized at best rendering quality at a frame rate of up to 60.4 fps. Light fields with 42 input images are synthesized at up to 79.1 fps (see Table 2).

4.3. Level of Detail for Light Field Rendering

The number of light field samples, necessary for light field reconstruction is highly related to the distance of the virtual view point [CCST00]. The polygonal spherical approximation rendered for light field synthesis results from a recursive subdivision of an icosahedron as a generator. Thus, the spherical setup provides an hierarchical representation, which can be used to implement an LOD rendering strategy at no extra costs.

The presented LOD strategy directly follows LOD strategies implemented for polygonal 3D models in interactive real-time applications. LOD is adjusted by coarsening the polygonal approximation being rendered for light field reconstruction. Light fields rendered with a coarser LOD are reconstructed from fewer light field samples reducing the amount of texture switches in the rendering process thus reducing the GPU workload. The current LOD is determined in classical sense as a tradeoff of resolution vs. geometric quality. A maximum of 4 LODs is available assuming a highest resolution of 642 sample positions for the most detailed representation. Minor popping artifacts can be observed during rendering on LOD switches resulting from image information that is only recoverable from a higher amount of sample positions. However, such artifacts are less noticeable compared to the well-known popping artifacts observable with most discrete LOD strategies for polygonal models. A light field rendered at different LOD is displayed in Figure 1 d.

5. Light Field Generation

Spherical light fields according to our sphere-hemisphere parameterisation can be generated from synthetic polygonal 3D models using an adapted renderer. Using state-of-the-art 3D sensor technology spherical light fields can be acquired from physical objects.

5.1. Generating Light Fields from Synthetic Objects

For synthetic light field acquisition based on 3D geometry, a given mesh is rendered once for each camera. The viewing matrix is obtained from the spherical parameterisation. It transforms the scene such that the camera is placed at the position $C = (0,0,1)^T$, looking along the negative z-axis. In this setup, the camera sphere is assumed to have unit radius. Hence, the entire scene has to be scaled and translated to fit into the bounding sphere with radius of $\frac{1}{\sqrt{2}}$ centered around the origin.

For generation of synthetic light fields we replace the fixed-function vertex processing step by a customized vertex program, which projects all vertices onto the unit sphere and converts the result to parabolic coordinates. This allows us to efficiently generate synthetic light fields using commodity graphics hardware. Due to the non-linear geometric distortions resulting from the parabolic mapping applied by our vertex program, however, it is mandatory to tessellate coarse geometry into small triangles before light field synthesis. On modern graphics hardware this can efficiently be achieved by a geometry shader program.

The steps performed by our customized vertex program comprise the following.

1. Projection to the hemisphere.
   Each geometry vertex V is transformed with the modelview matrix and then projected from the camera point onto the opposite unit hemisphere. This is computed by casting a ray from the camera at position $C = (0,0,1)^T$ through the vertex and intersecting this ray with the unit sphere. This amounts to solving a simple quadratic equation and results in a projected vertex S.
2. Parabolic mapping of the projected vertices
   As a result of the previous step, the projected vertex S is lying on the hemisphere with $z < 0$. The vertex S is converted to parabolic coordinates, according to Equation 2
3. Depth Adjustment
   The depth value is calculated according to Figure 5.
4. Normal Transformation
   We assume that lighting is calculated in world coordinates. Hence, the normal vectors of the geometry must be transformed with the transposed inverse of the modeling matrix, but not with the viewing matrix.

Arbitrary complex fragment programs can be used in combination with this customized vertex program. The only modification necessary is that the fragment program must write the interpolated depth value generated by the vertex shader into the alpha portion of the final color.

Each parabolic map is stored as an interleaved array of RGBz values. Individual parabolic maps are associated with a transformation matrix defined by the spherical parameterisation.

For usability reasons the light field generation algorithm was integrated into commercially available 3D modeling packages like *Autodesk Maya* which are commonly used in the computer graphics community. Using the plug-in light fields can be generated efficiently from arbitrary complex 3D objects containing sophisticated material and (global) illumination effects. For light field generation any render engine available with the 3D modeling package can be applied. Us-

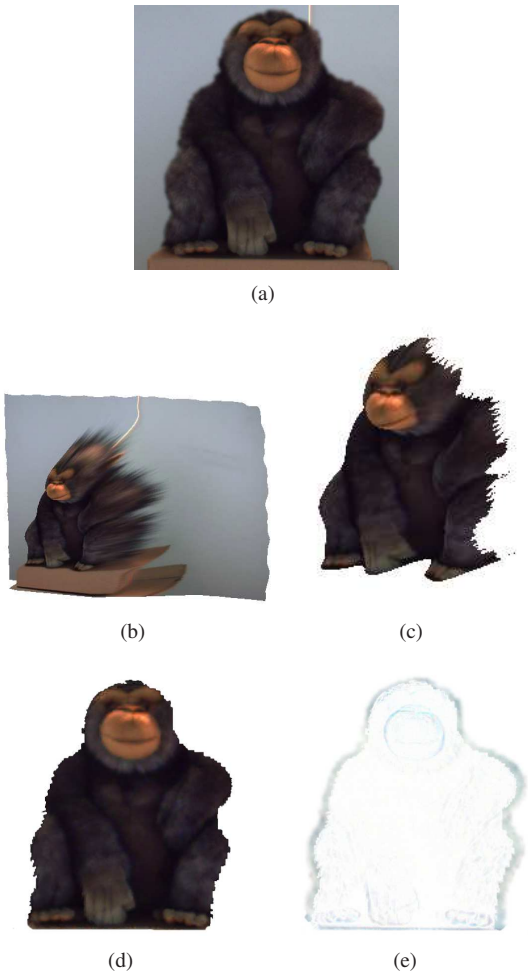

Figure 10: *LFR from physical object. a: Original input image taken from stuffed animal using the 2D3D camera setup. b: Reconstructed polygonal mesh, showing false geometry on silhouettes and background. c: Gradient filtered mesh, background and bottom culled using clipping planes. d: Light field rendering of the stuffed animal. e: Difference image of light field rendering compared to original view showing per-pixel RGB error.*

ing *Mental Ray for Maya*, one of Maya's raytracing rendering engines, high-quality light fields of a Tie-Fighter model have been created for quality evaluation (see Figure 1, Figure 11). Individual light field samples were rendered at a resolution of 512×512 in 39 seconds.

5.2. Acquisition of Light Fields from Physical Objects

We have implemented a processing pipeline for light field acquisition taking advantage of recent progress in sensor technology [PHW*06, YIM06, OBL*05, GYB04] which allow real-time measurement of per-pixel depth information. In combination with RGB sensors these devices can be used for hand-held light field acquisition [PHW*06, TRSK05]. For light field acquisition we use a *PMD* 3D time-of-flight sensor providing depth images at a resolution of 160×120 at 20 fps in combination with a mounted commodity high resolution RGB camera.

PMD image data is processed by applying standard image distortion techniques based on the intrinsic camera parameters for lateral image correction. Per-pixel depth data is filtered using a GPU based processing pipeline for noise reduction, outlier removal and correction of systematic errors according to Lindner et al. [LK06, LLK08]. Depth and RGB data of both cameras is combined by applying extrinsic camera calibration parameters of the fixed 2D3D camera setup for per-pixel projection from *PMD* image space to RGB image space [LK07]. The combined RGBz data from this 2D3D camera setup serves as input data for the spherical light field parameterisation. Figure 10 a shows a sample input image of the 2D3D camera setup.

A dense polygonal mesh representation is reconstructed from each individual input sample (see Figure 10 b). Based on the mesh representation, background information is culled easily by implementing clipping planes to isolate the object of interest (see Figure 10 c). Uncertain per-pixel depth information at the silhouette edges as well as for occluded regions however lead to discontinuity artifacts which are reconstructed as false geometry in the mesh representation 10 b). These artifacts are significantly reduced by gradient filtering based on the input depth values 10 c).

For flexible acquisition of light fields we implemented a rebinning technique to successively insert recorded RGBz images from hand-held acquisition into our light field representation. Taking the optical *DTrack* [Adv] tracking system as a precise solution for real-time camera pose estimation, the camera setup's extrinsic parameters are evaluated in real-time. Based on the extrinsic parameters and the intrinsic parameters, known from calibration, the captured RGBz depth map can be projected to known sample camera coordinates.

For rebinning the camera best approximating the viewing direction of the RGBz camera setup is determined. To identify the most appropriate sample camera within the spherical parameterisation we determine the disparity in major view direction and distance according to [BBM*01] for each of the sample cameras. For the best camera providing the least disparity we perform the steps as described in Section 5.1 to insert the light field sample into our representation. Combined RGBz data is stored in a parabolic map with the per-pixel depth being reparameterized according to the secant length as shown in Figure 5. The light field samples are used directly for light field synthesis without any post processing.

Since combined depth and RGB are evaluated directly by our rendering technique, immediate visual feedback is available at acquisition time even for incomplete light field data.

This allows for interactive quality evaluation and resampling of incorrect light field samples during the acquisition process. Figure 10 d) shows LFR results from a stuffed animal, acquired using the 2D3D setup described above.

The synthesized images clearly resample the object's appearance. The difference image depicted in Figure 10 e shows the per-pixel RGB error based on an original image, of the isolated object, according to 10 a, and the synthesized view generated according to corresponding viewing parameters. Minor ghosting artifacts appear at the inner object, while disparities are more visible at the object's boundary edges.

6. Results and Discussion

We have presented a LFR approach providing two different rendering techniques working on a uniform spherical camera space parameterisation and a parabolic image space parameterisation storing combined RGBz values with each light field sample. The techniques described in this paper have been implemented using OpenGL and Cg under Windows XP. All renderings and performance measurements have been created using an NVidia Geforce 8800 GTX graphics board with 768 MB of local video memory build into an AMD Athlon 64 X2 dual core processor with 2.21 GHz and 3.5 GB main memory.

The iterative refinement technique implements efficient LFR at real-time frame rates of up to 110 fps (see Table 2). Best results are achieved for light fields samples from ≥ 642 sample positions, while ghosting artifacts are obeserved for lower sample counts (see Figure 7). The visual appearance of concavities and small-to-large depth variations are satisfactory reconstructed.

For arbitrary complex 3D objects, the raycasting approach provides best rendering quality at moderate frame rates of up to 79 fps (see Table 2). Light fields can be reconstructed with high quality from 42 input samples (see Figure 1 and 11). For comparable image quality the raycasting approach requires less input images compared to the iterative refinement technique (see Table 1). Concavities, small-to-large depth variations and complex material attributes like highlights or anisotropic reflections are precisely synthesized (see Figure 9). Rendering can be performed more efficiently for scenes containing such complex geometry or material attributes compared to traditional rendering approaches based on polygonal computer graphics. In combination with the presented LOD approach for light fields the raycasting technique provides a powerful tool for a wide area of applications.

Samples	Resolution	Raycasting	Iterative Refine.
42	256×256	79.1 fps	110.4 fps
42	512×512	78.3 fps	107.2 fps
162	256×256	60.4 fps	96.9 fps
162	512×512	57.8 fps	95.6 fps
642	256×256	52.1 fps	78.7 fps
642	512×512	52.5 fps	75.3 fps

Table 2: *Rendering performance for our raycasting rendering algorithm and the iterative refinement approach applied to light fields of varying resolution rendered for a screen size of* 512×512.

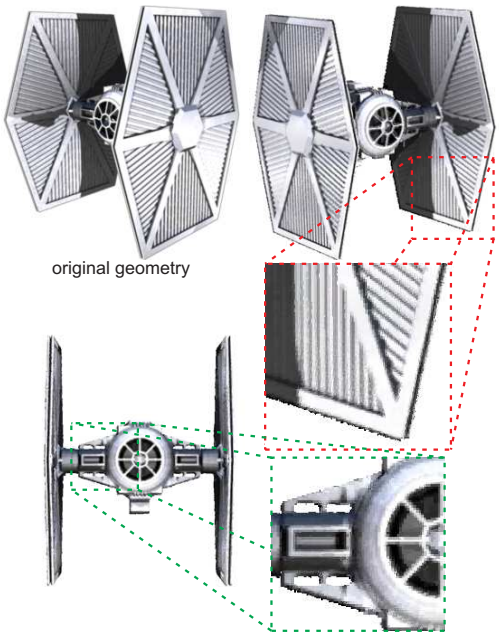

Figure 11: *The top left image shows the original geometry containing 16.5k polygons rendered at* 512×512 *in 39 sec. using Mental Ray for Maya. Renderings (top right, bottom left) are generated from 162 input samples at 57.8 fps. The raycasting approach clearly synthesizes the visual appearance of small structures and precisely reconstructs silhouettes, holes and concavities.*

The light field techniques presented in this paper are applicable to a variety of objects and input sources. Light fields can be acquired from physical objects using state-of-the-art 3D imaging sensors (see Section 5.2), or can be easily generated from synthetic objects using our customized vertex program and the plugin being provided for 3D modeling packages (see Section 5). For elongated objects a set of light fields can be acquired. Due to the flexible compositing capabilities and the high rendering performance, provided by the raycasting approach, multiple light fields can be displayed simultaneously (See Figure 12).

While sparse polygonal objects containing only simple material attributes, as presented for clarity reasons in Figure 7 and Figure 9, do not necessarily profit from the pro-

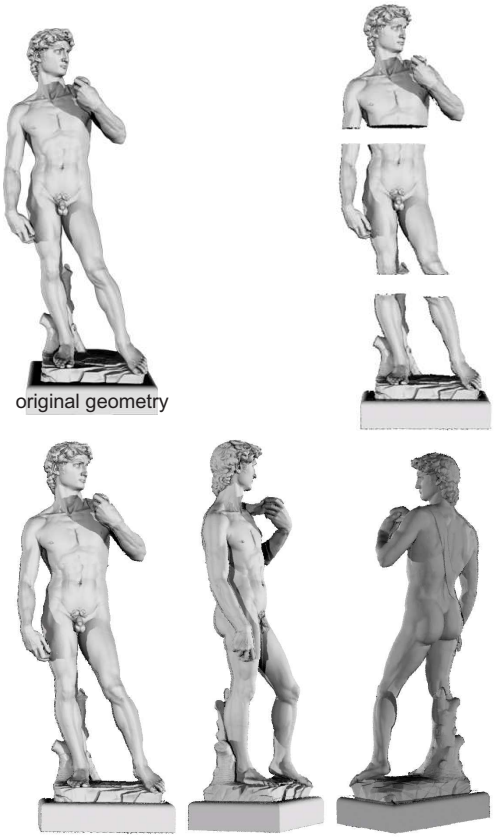

Figure 12: *Light field compositing results. The top left image shows the simplified version (500k polygons) of the original geometry of David obtained from* The Digital Michelangelo Project *[LPC*00] rendered at 47 sec. using Mental Ray for Maya. The top right image displays three separate light fields containing 162 samples at a resolution of* 512×512*. The light fields are rendered simultaneously at a frame rate of 26.5 fps as shown in the bottom row.*

posed light field rendering approaches, arbitrary complex scenes do profit from the light field approach as virtual views are reconstructed at high quality at real-time frame rates. Figure 11 and Figure 12 clearly demonstrate the benefit of our light field rendering approaches. Virtual views are synthesized at a quality comparable to the quality achieved by offline raytracing render engines at a fraction of rendering time. Thus the presented techniques are especially suited to synthesize virtual views of complex scenes containing arbitrary complex materials and lighting conditions. The rendering approaches presented in this work overcome critical issues of related image based rendering approaches presented in the past and provide efficient techniques for high-quality view synthesis at real-time frame rates based on a sparse set of input images.

Free form light fields [VG00] apply a mesh refinement based on a triangle mesh defined by the input sample camera positions (see Section 2.3). Mesh refinement is performed at rendering time thus affecting the rendering performance. For complex scenes a more detailed mesh refinement is applied to achieve good rendering results. Since mesh refinement is steered by the render target's resolution as it defines the minimum triangle size, performance is also limited by the render target resolution. As a consequence performance is limited by both the scene complexity and the render target resolution. As the depth is evaluated per vertex to steer the mesh refinement process, small scale structures as well as local high frequency depth variations are not accounted for in this subdivision process. As a consequence these structures will result in visible ghosting artifacts due to incoherent ray interpolation. Ghosting artifacts are observed at the silhouettes, edges (see Figure 10 b in [VG00]) and concavities (see Figure 9, 2nd image in [SVSG01]). Even for fine meshes uniformity of sampling positions cannot be guaranteed, thus artifacts resulting from discontinuities as observed with the *Lumigraph* rendering using six *light slabs* are likely to appear.

The light field representation presented in this paper guarantees a uniform sampling of rays for the complete observation space, eliminating artifacts resulting from discontinuities. Virtual views can be synthesized at constant quality with 6 DOF. With the depth correction of rays being evaluated per-pixel from three adjacent cameras, small to large scale structures as well as concavities and holes are precisely reconstructed. Thus, ghosting artifacts are not observed at high frequency changes and concavities. For objects with absence of micro and meso structures providing mainly planar regions at large scale structures only, the *free form light field* rendering approach will most likely profit from the selective subdivision scheme, whilst per-pixel depth evaluation will provide comparable quality at decreased frame rates. To overcome the limitations for micro and meso structures, however, the number of sample positions has to be increased for *free form light field* to achieve rendering quality comparable to our raycasting approach.

Unstructured Lumigraph Rendering [BBM*01] uses a polygonal approximation for depth correction for rays. Quality and performance is also limited by the scene complexity. More complex scenes are in need of a more detailed polygonal approximation to reduce ghosting artifacts. Detailed proxies, however, will have negative impact on the performance of the ray-object intersections applied for each fragment. As a consequence, only sparse polygonal proxies are applied in practice, resulting in noticeable ghosting artifacts (see Figure 11 d,e,f in [BBM*01]). Additional discontinuity artifacts appear, if no uniform sampling of the scene is given. The combined RGBz images used in our approaches allow per-pixel correction of rays to be performed independent of the scene complexity. No additional polygonal approximation has to be stored with the light field samples. Very simple

scenes, however, may be represented more efficiently using a sparse geometric representation. In this case, the polygonal proxy will result in less memory consumption compared to our approaches. In the general case, our approaches achieve better performance for arbitrary complex scenes at constant memory costs.

Image based modeling techniques, such as displacement maps [WWT*03, WTS*05] and relief textures [OBM00, POJ05, ACB*07] aim at the reconstruction of geometric detail instead of the full visual appearance. They are used to reconstruct an object's geometry (surface points and normals) from a set of input images, while the illumination and shading is performed afterwards. These approaches are not capable of reconstructing complex view-dependent reflectance and geometric details resulting from self-occlusion, concavities and micro structures [BD06, OBM00]. In comparison, our approach takes three adjacent depth maps into account to evaluate ray-object intersections. It precisely reconstructs both geometry from small structures and large variations in depth as well as complex view-dependent surface reflectance. It is not affected by self-occlusion or concavities (See Figure 1, 9, and 11). Relief- and displacement maps do not target the problem of synthesizing the visual appearance of complex material properties and lighting environments at all.

7. Conclusion

We have presented a novel GPU-based LFR technique based on combined RGB and depth information. We have demonstrated that this technique is capable of reconstructing light fields with high accuracy at interactive frame rates.

Both the iterative refinement and the raycasting approach presented in this article implement LFR techniques that provide full 6 DOF for virtual view point selection. Light field reconstruction is based directly on the RGBz sample images stored with each sample camera. There is no need for complex mesh processing at runtime nor for generation and acquisition of additional polygonal object representations. Thus the light field approach presented in this paper turns out to be more efficient with respect to storage and rendering performance.

For future development our aim is to improve the acquisition process for physical objects. 3D imaging devices applying a different technique for per-pixel depth estimation [YIM06] will be tested in future light field acquisition set-ups. These devices provide per-pixel depth at higher resolutions. From experiments with a simulated 2D3D sensor setup we know that these new sensors will lead to an improved light field acquisition of physical objects.

Acknowledgements

We thank the members of the SFB-603 at the University of Erlangen-Nuremberg for their support. This research project is partially funded by grant KO-2960-6/1 from the German Research Foundation (DFG).

References

[AB91] ADELSON E. H., BERGEN J. R.: The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*. MIT Press, 1991, pp. 3–20.

[ACB*07] ANDUJAR, C., BOO, J., BRUNET, P., FAIREN, M., NAVAZO, I., VAZQUEZ, P., VINACUA, A.: Omni-directional relief impostors. *Computer Graphics Forum 26*, 3 (2007), 553–560.

[Adv] ADVANCED REALTIME TRACKING GMBH: Dtrack tracking system. http://www.ar-tracking.de, last visited 16.06.2008.

[BBM*01] BUEHLER C., BOSSE M., MCMILLAN L., GORTLER S., COHEN M.: Unstructured lumigraph rendering. In *Proc. ACM SIGGRAPH* (2001), pp. 425–432.

[BD06] BABOUD L., DÉCORET X.: Rendering geometry with relief textures. In *GI '06: Proc. Graphics Interface 2006* (2006), Canadian Information Processing Society, pp. 195–201.

[BN76] BLINN J., NEWELL M.: Texture and reflection in computer generated images. *ACM SIGGRAPH Comput. Graph. 10*, 2 (1976), 266–266.

[CCST00] CHAI J.-X., CHAN S.-C., SHUM H.-Y., TONG X.: Plenoptic sampling. In *SIGGRAPH '00: Proc. Conf. on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 307–318.

[CLF98] CAMAHORT E., LERIOS A., FUSSELL D.: *Uniformly Sampled Light Fields*. Tech. rep., University of Texas at Austin, 1998.

[Gas83] GASSON P.: *Geometry of Spatial Forms: Analysis, Synthesis, Concept Formulation and Space Vision for CAD*. Ellis Horwood Series in Mathematics and its Applications. Ellis Horwood, 1983.

[GGSC96] GORTLER S., GRZESZCZUK R., SZELISKI R., COHEN M.: The lumigraph. In *Proc. ACM SIGGRAPH* (1996), pp. 43–54.

[Gre86] GREENE N.: Environment mapping and other applications of world projections. *IEEE Comput. Graph. Appl. 6*, 11 (1986), 21–29.

[GYB04] GOKTURK S., YALCIN H., BAMJI C.: A time-of-flight depth sensor - system description, issues and solutions. In *Proc. IEEE Comp. Vis. and Pat. Rec. Workshop (CVPRW'04)* (2004), p. 35.

[HS99] HEIDRICH W., SEIDEL H.-P.: Realistic, hardware-accelerated shading and lighting. In *Proc. ACM SIGGRAPH* (1999), pp. 171–178.

[INH99] IOURCHA K., NAYAK K., HONG Z.: System and method for fixed-rate block-based image compression with inferred pixel values. United States Patent 5,956,431, 1999. S3 Incorporated (Santa Clara, USA).

[IPL97] IHM I., PARK S., LEE R.: Rendering of spherical light fields. In *Proc. Pacific Graphics* (1997), IEEE Computer Society, p. 59.

[LH96] LEVOY M., HANRAHAN P.: Light field rendering. In *Proc. ACM SIGGRAPH* (1996), pp. 31–42.

[LK06] LINDNER M., KOLB A.: Lateral and Depth Calibration of PMD-Distance Sensors. In *Advances in Visual Computing* (2006), vol. 2, Springer, pp. 524–533.

[LK07] LINDNER M., KOLB A.: Data-Fusion of PMD-Based Distance-Information and High-Resolution RGB-Images. In *Proc. of the Int. IEEE Symp. on Signals, Circuits & Systems (ISSCS)* (2007), vol. 1, pp. 121 – 124.

[LLK08] LINDNER M., LAMBERS M., KOLB A.: Data Fusion and Edge-Enhanced Distance Refinement for 2D RGB and 3D Range Images. *Int. J. on Intell. Systems and Techn. and App. (IJISTA), Issue on Dynamic 3D Imaging* (2008).

[LPC*00] LEVOY M., PULLI K., CURLESS B., RUSINKIEWICZ S., KOLLER D., PEREIRA L., GINZTON M., ANDERSON S., DAVIS J., GINSBERG J., SHADE J., FULK D.: The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proc. Conf. on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 131–144.

[MB95] MCMILLAN L., BISHOP G.: Head-tracked stereoscopic display using image warping. In *Proc. ACM SIGGRAPH* (1995), pp. 39–46.

[MH84] MILLER G., HOFFMAN C.: Illumination and reflection maps: Simulated objects in simulated and real environments. In *ACM SIGGRAPH Course Notes for Advanced Computer Graphics Animation* (1984).

[OBL*05] OGGIER T., BÜTTGEN B., LUSTENBERGER F., BECKER G., RÜEGG B., HODAC A.: Swissranger^{TM} SR3000 and first experiences based on miniaturized 3D-TOF cameras. In *Proc. Range Imaging Research Day* (2005), pp. 97–108.

[OBM00] OLIVEIRA M. M., BISHOP G., MCALLISTER D.: Relief texture mapping. In *SIGGRAPH '00: Proc. Conf. on Computer graphics and interactive techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 359–368.

[PHW*06] PRASAD T., HARTMANN K., WOLFGANG W., GHOBADI S., SLUITER A.: First steps in enhancing 3D vision technique using 2D/3D sensors. In *Computer Vision Winter Workshop 2006* (2006), Czech Society for Cybernetics and Informatics, pp. 82–86.

[POJ05] POLICARPO F., OLIVEIRA M. M., JO A. L. D. C.: Real-time relief mapping on arbitrary polygonal surfaces. In *I3D '05: Proc. Symposium on Interactive 3D graphics and games* (2005), ACM, pp. 155–162.

[SCK07] SHUM H.-Y., CHAN S.-C., KANG S. B.: *Image-Based Rendering*. Springer, 2007.

[SVSG01] SCHIRMACHER H., VOGELGSANG C., SEIDEL H.-P., GREINER G.: Efficient free form light field rendering. In *VMV '01: Proc. Vision Modeling and Visualization* (2001), Aka GmbH, pp. 249–256.

[TRSK05] TODT S. S., REZK-SALAMA C., KOLB A.: Real time fusion of range and light field images. In *ACM SIGGRAPH 2005 Posters* (2005), p. 65.

[VG00] VOGELGSANG C., GREINER G.: Adaptive lumigraph rendering with depth maps. *Technical Report 3, IMMD 9, Universitaet Erlangen-Nuernberg* (2000).

[VG01] VOGELGSANG C., GREINER G.: Ray-tracing in depth-maps for image-based rendering. *Technical Report, IMMD 9, Universitaet Erlangen-Nuernberg* (2001).

[WTS*05] WANG J., TONG X., SNYDER J., CHEN Y., GUO B., SHUM H.-Y.: Capturing and rendering geometry details for btf-mapped surfaces. *The Visual Computer 21*, 8-10 (2005), 559–568.

[WWT*03] WANG L., WANG X., TONG X., LIN S., HU S., GUO B., SHUM H.-Y.: View-dependent displacement mapping. *ACM Trans. Graph. 22*, 3 (2003), 334–339.

[YIM06] YAHAV G., IDDAN G. J., MANDELBOUM D.: 3D imaging camera for gaming application. *Technical Report, 3DV Systems Ltd.* (2006).