

# Light Field Rendering for Games

S. Todt, C. Rezk-Salama and A. Kolb

Computer Graphics Group, University of Siegen, Germany

---

## Abstract

*This paper studies the problem of integrating high-quality light field rendering into state-of-the-art real-time computer games. We present a complete production pipeline for generation of light fields from arbitrary complex 3D content using commercial 3D modeling software commonly used in the gaming industry. We show how recent advances in light field rendering techniques can be used to composite light fields with dynamic scene content. The results demonstrate the potential of light fields as a valuable extension to polygonal rendering techniques by employing accurate silhouette reconstruction and correct per-pixel depth values for scene composition. Level of detail techniques are applied to light fields to optimize rendering performance. Dynamic light field rendering is implemented to account for animation, deformation and varying lighting conditions.*

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

---

## 1. Introduction

Today, the visual quality of a game title is one of the main keys to commercial success. Upcoming computer games unleash the power of state-of-the-art graphics hardware to achieve best visual results at maximum performance. Since Blinn and Newell introduced *Texture Mapping* in 1976 [BN76], image-based techniques are frequently employed to achieve sophisticated rendering results in real-time. Texture Mapping is heavily used today and has inspired a set of related image-based rendering techniques such as *Bump Mapping* [Bli78], *Impostors* [Sch95], *View-Dependent Texture Mapping* [DYB98], *Relief Texture Mapping* [OBM00], *Billboard Clouds* [DDSD03], *View-Dependent Displacement Mapping* [WWT\*03] or *Omnidirectional Relief Impostors* [ACB\*07], all of which contribute to the visual experience of today's computer games.

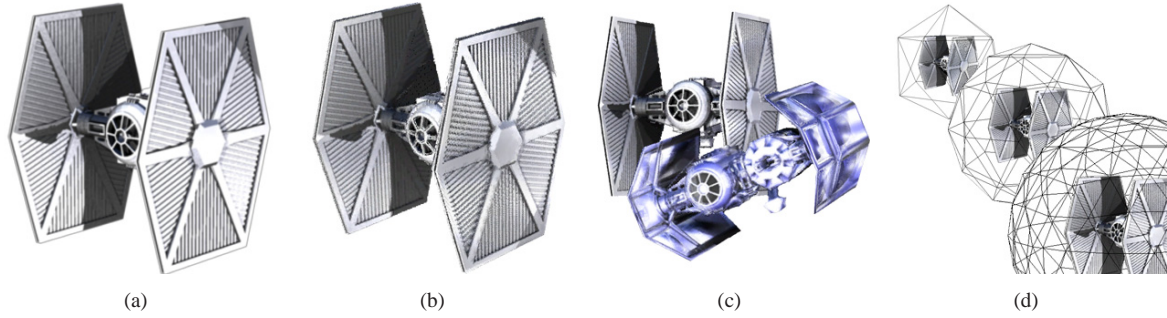
In contrast to *Relief Texture Mapping* and *View-Dependent Displacement Mapping*, light field rendering techniques (LFR) [LH96, GGSC96] focus on the reconstruction of complex lighting and material attributes from a set of input images rather than the exact reconstruction of geometric details. LFR techniques as a powerful alternative to traditional polygonal graphics, however, have not yet been successfully integrated into current real-time games for various reasons. For LFR techniques to be successfully integrated into game engines they must at least provide:

- High quality real-time rendering without visual artifacts
- Accurate compositing with dynamic scene content
- Level of detail (LOD) methods to adjust the render complexity
- Parameterized rendering algorithms to adjust the object to varying conditions
- Continuous 6 degrees of freedom (DOF) for free view point selection

In this paper we present a LFR technique compliant with these requirements and thus ready to be integrated into games. We show how LFR techniques emerging from recent progress on *Spherical Light Field Rendering* [TRSK07] can successfully be integrated into dynamic scenes. We employ a hierarchical light field parametrization to implement an LOD strategy to adjust rendering complexity according to the target performance (see Figure 1 d). We improve compositing capabilities for light fields by providing depth information for per-fragment depth culling techniques (see Figure 1 c). We present a dynamic LFR technique which allows lighting situations to be adjusted and to implement light field animations and -deformations (see Figure 6).

## 2. Spherical Light Fields

Spherical light fields provide a uniform sampling of the observation space by implementing a spherical parametrization on a bounding sphere around the observed object [CLF98].



**Figure 1:** a: Original Tie-Fighter model containing 16428 triangles rendered at  $512 \times 512$  using Mental Ray in 39 sec.. b: Light field reconstructed from 42 samples sampled at  $512 \times 512$  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.

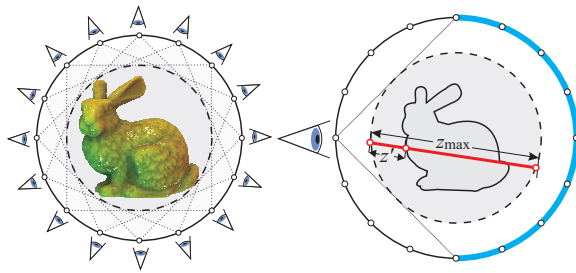
Recently a high-quality spherical light field implementation was presented by Todt et.al. [TRSK07] which facilitates the rendering of light fields without noticeable artifacts in real-time with 6 DOF. This technique is a basis for our LFR approach for use in computer games.

**Representation** The spherical parametrization is based on an icosahedron as a uniform approximation of the sphere. Originally providing 12 uniform distributed sample positions, parameterizations at higher resolutions are achieved by subsequently subdividing the icosahedron yielding 42, 162, 642 and more sample positions (see Figure 1, d). A parabolic parametrization of the opposite hemisphere is associated and stored with each sample camera position (see Figure 2). In addition to the captured color intensities, per-pixel depth information is stored in each sample's alpha channel, providing interleaved *RGBz* data (see Figure 3). Using standard texture compression methods, a compression ratio of 4:1 is achieved. 2.7 MByte are sufficient to store a light field sampled from 42 sample positions at a resolution of  $256 \times 256$  (10.4 MByte for 162 samples).

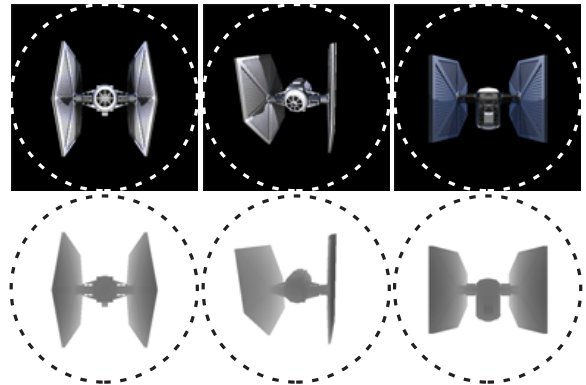
**Generation** Light fields are generated from synthetic objects by rendering the scene from the predefined spheri-

cal sample positions. Per-pixel depth is stored as the ratio of the camera sample position's distance to the object's surface point to the length of the ray secant according to Figure 2.

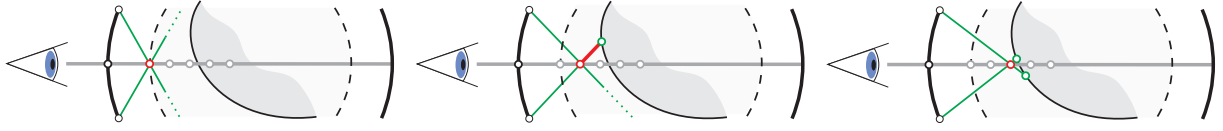
**Rendering** The smooth shaded spherical approximation is rendered for light field reconstruction, with the vertices being equivalent to the sample positions. For each triangle the parabolic *RGBz* texture images and the viewing matrices of the three cameras are bound to their associated vertices. For each triangle of the polygonal sphere, rasterization yields barycentric coordinates per-fragment. For each fragment a ray is cast into the scene. The raycast rendering algorithm described by Todt et.al. [TRSK07] is utilized to establish the intersection point with the surface of the captured object. Starting at the ray intersection with the object's bounding sphere, the ray is iteratively sampled at a fixed step size, as shown in Figure 4. For each sample position the assumption that the current ray position is the



**Figure 2:** For light field generation the synthetic object is rendered from predefined sample positions. The depth value is obtained by dividing the bounding-object distance  $z$  by the bounding sphere's secant length  $z_{max}$ .



**Figure 3:** 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).



**Figure 4:** Ray casting approach employing per-pixel depth to establish ray-object intersection according.

actual object intersection point is validated by projecting this position onto the camera sphere using the adjacent camera positions as center of projection. Depth values are obtained from the corresponding parabolic texture maps and local estimates are calculated for each camera. If one of the local estimates is equal to the ray position within a given tolerance, ray sampling is stopped. This means that we have found at least one camera that reliably observes an object intersection at exactly the ray position. For the final intersection point the fragment color value is determined from the weighted sum of the light field samples. Per sample weights are determined based on distance of the local estimate to the current ray position.

Using the raycasting approach and per-pixel depth correction light fields are rendered without noticeable ghosting artifacts yielding an accurate silhouette reconstruction (see Figure 1 b). In contrast to other image based rendering techniques commonly used in computer games, this spherical light field rendering approach is capable of reconstructing visual effects resulting from complex material attributes, e.g. anisotropic shading, or global illumination effects in real-time. Although techniques like *Impostors* [Sch95], *Billboard Clouds* [DDSD03] or *Omnidirectional Relief Impostors* [ACB\*07] are capable of reconstructing an object's visual appearance for a predefined position and viewing direction they do not provide any technique to reconstruct new virtual views containing the complete range of shading complexity represented in the source images in real-time. As shown by Andujar et.al. [ACB\*07] geometric details are recoverable to a certain degree from relief impostors which than can be used to implement dynamic lighting and global illumination effects. This approach however comes at the price of high computational costs to implement sophisticated shading based on the geometric reconstruction. The spherical light field rendering performs independently of the scene complexity at real-time frame rates.

### 3. Light Field Rendering for Games

We adapted the spherical rendering approach and implemented optimized rendering strategies to suffice the requirements of nowadays real-time render engines to successfully integrate light field rendering in real-time computer games. We take advantage of the spherical parameterization's hierarchical nature and the render algorithm's flexibility presented in Section 2 to implement an LOD strategy. We ex-

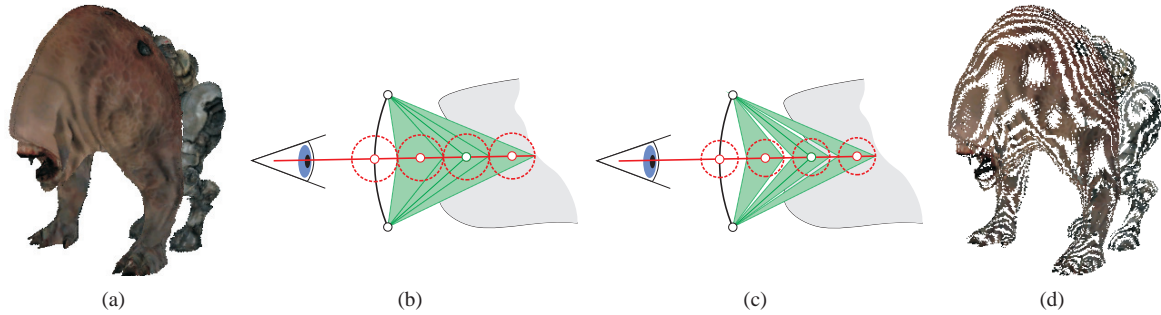
ploit per-pixel depth information used for depth correction of rays to improve light field compositing capabilities and we demonstrate a technique for dynamic LFR. For digital content creation we present a production pipeline including model generation and light field generation.

#### 3.1. Light Field Rendering with LOD

Rendering techniques for complex scenes in computer games are mainly importance driven to optimize performance. Objects in focus are rendered at highest quality whereas distant objects are rendered at lower resolutions. The hierarchical arrangement of sample positions on the spherical approximation resulting from the subdivision of the initial icosahedron (see Section 2) makes the implementation of a discrete LOD strategy available for LFR. Additionally the rendering performance is continuously adapted to the quality needs for distant objects by adjusting the raycaster's step size and intersection evaluation tolerance.

The implemented LOD strategy straightly follows LOD strategies implemented for polygonal 3D models in interactive real-time applications. The LOD is adjusted by coarsening the polygonal approximation being rendered for light field reconstruction (see Section 2). Light fields rendered with a coarser LOD are reconstructed from fewer light field probes according to Chai et.al. [CCST00] which reduces the amount of texture switches in the rendering process and thus reduces the graphics processing unit (GPU) workload (see Figure 1 d). 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 the spherical approximation with 642 sample positions for the most detailed representation. The spherical sample positions are arranged such that all of the vertices of certain LOD are contained in the next finer LOD. No additional spherical approximations nor additional light field samples have to be generated and hosted for this LOD strategy. Coarser LODs are achieved by reducing the sample density in the geometric domain and the light field information in the image domain by reducing the amount of sample positions. Minor popping artifacts, however, can be observed during rendering on LOD switches resulting from image information that becomes recoverable with the higher amount of sample positions, e.g. concavities.

Rendering performance is further adapted to the object distance by adjusting the render settings for the raycasting



**Figure 5:** *a: Light field rendering of the DOOM Pinky character model. b: Showing recoverable area for tolerance chosen to be equal to half of the step size. c: Tolerance smaller than half of the step size result in unrecoverable areas. d: Unrecoverable areas appear as gaps in the light field rendering if tolerance is chosen too small with respect to the step size.*

algorithm to steer the reconstruction quality. As presented in Section 2 the precision of the raycaster is dependent on the chosen step size to sample the ray and the given tolerance at which the ray sampling is stopped. While the LOD strategy based on the reduction of sample positions implements a discrete LOD strategy the render settings can be adjusted continuously with the object distance. A performance gain is achieved by increasing the raycaster's step size. However, discontinuity artifacts are likely to appear due to the incorrect reconstruction of object surface points used to sample the light field probes of the ray's adjacent cameras. Small geometric details are not recoverable with step sizes chosen too large. Notice that the step size is correlated with the chosen tolerance. While the step size can be chosen without theoretical constraints, the tolerance is to be chosen according to the step size. Choosing the tolerance to be less than half of the step size will result in visual discontinuities as local estimates will not be reconstructable from adjacent cameras for all ray positions. With rising angular distance of the adjacent camera with respect to the current ray position the camera's reconstructed local estimate will most likely fail the tolerance test. Tolerance values chosen to be smaller than half of the step size appear as equidistant isosurfaces with a distance equal to the step size and with empty space gaps of size  $StepSize - 2 \times Tolerance$  in the light field rendering (see Figure 5). The combination of both LOD rendering approaches results in a gain of rendering performance and rendering flexibility which allows multiple light fields to be displayed simultaneously by rendering instances of the same light field or different light field objects. Instancing shows higher performance compared to rendering multiple separate light field objects. Rendering multiple distinct light fields results in numbers of texture swaps on the GPU leading to lower rendering performance, whereas with instancing a single light field renders without texture swaps.

### 3.2. Light Field Compositing

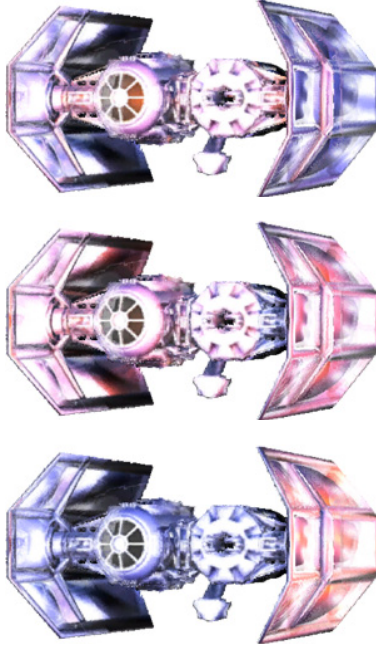
In [TRSK07] ray-object intersections are established per-fragment for accurate silhouette reconstruction and to reduce ghosting artifacts. We use the barycentric fragment coordinates in combination with the ray-object intersection point to determine per-fragment depth values during light field reconstruction at no extra costs in order to take advantage of *OpenGL's z-Buffer* functionality.

The ray-object intersection point yields relative distances to the three adjacent sample positions. Based on the barycentric weights the fragment's position in world coordinates is established according to the known sample positions. By projecting the reconstructed surface point into the current view, per fragment depth values are determined for the surface point according to the current viewing transformation and projection matrices.

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).

### 3.3. Parameterized Light Field Rendering

The low memory footprint of the spherical light field representation in combination with the performance of the LOD rendering algorithm provide the tools for dynamic LFR. Various light field states can be represented by generating a set of light field samples for individual sample positions. The efficient light field representation facilitates to hold multiple light field samples for individual sample positions on the GPU ready to be rendered in real-time. Using the *nVIDIA GeForce 8800 GTX* providing 768 MB of onboard GPU memory, up to 280 distinct light field samples can be stored for each sample position of a single light field acquired at a



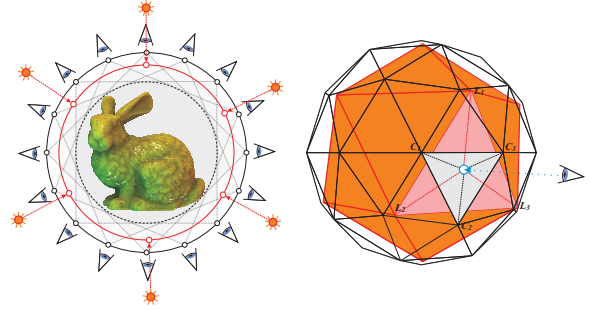
**Figure 6:** Tie-Bomber light field rendered with interchangeable dominant lighting direction (red light source chosen for visualization purposes).

resolution of  $256 \times 256$  from 42 sample positions (990 distinct samples for 12 sample positions). The rendering process is steered by adjustable parameters in real-time to reconstruct a light field object with varying state. For parameterized rendering the light field samples corresponding to the current state are used to reconstruct the light field. This technique opens up many possibilities for various applications, e.g.:

**Deformation** Dynamic deformable objects must show deformations in case of collision. Preparing light field samples for different deformation states provides capabilities to adjust the visual appearance according to the collision. Note that for the changes to the light field objects, we need to store additional images only for those cameras, which actually observe the applied modification. Hence, for minor changes to the object only a small subset of the camera images need to be stored in addition.

**Lighting Conditions** The visual appearance of the light field object must adjust to changing lighting conditions. With parameterized rendering the light field state can be adjusted for e.g. changing environments or dominant lighting directions (see Figure 6).

Changing lighting conditions are reconstructed from a set of pre-acquired light field samples captured for varying lighting conditions. For flexible and continuous reconstruction of varying dominant lighting directions we acquire light field samples for a discrete set of predefined



**Figure 7:** Left: Lighting directions are discretized by a spherical approximation (orange) similar to the approximation used in the spherical camera setup. Right: Barycentric weights are established based on the spherical proxies used for lighting and light field acquisition.

light directions. The lighting direction is discretized based on the spherical approximation also taken for acquisition due to its uniform characteristic. For each light field sample position a sample is acquired for each discrete light direction, defined by the spherical light discretization proxy (see Figure 7, left). The resolution of the spherical proxy used for lighting can be chosen freely. It does not correlate with the camera resolution chosen for light field acquisition.

For rendering the light field the spherical light proxy is applied to identify the light directions ( $L_1, L_2, L_3$ ) from the set of discrete light directions that contribute to the current viewing direction and thus chose the corresponding set of input images for light field reconstruction (see Figure 7, right). According to the intersection point of the current view direction with the spherical light proxy, barycentric blending weights are determined. The light field is then rendered using the light field samples acquired for the discrete incoming light directions as input images. Rendering is performed in three distinct rendering passes according to ( $L_1, L_2, L_3$ ). The render results, being rendered to an off-screen render target are blended in a final render pass using the barycentric weights from the light proxy intersection as blending weights.

This interpolation allows arbitrary lighting directions to be interpolated continuously within the rendering process. Other applications may use this approach for blending light fields acquired for a discrete set of environment maps or other complex lighting situations. Although the total amount of input images is factored by the amount of discrete lighting directions, rendering is still performed at real-time frame rates of 47.5s fps for a light field sampled from 42 positions for 12 discrete light directions at a resolution of  $512 \times 512$ .

**Animation** Animation sequences of single characters have proven to be well suited for storytelling. Interactivity is added if dynamic light field rendering is used instead of



**Figure 8:** Animated light field rendering of the Ray character model, showing an extract of six animation frames.

offline rendered video footage. Animated light field rendering allows the user to adjust viewing direction and position freely while the animation is played. Light field animations are rendered at slightly lower ( $-10\%$ ) frame rates compared to static light field rendering. The light field animation shown in Figure 8 was rendered at real-time frame rates but limited by the target video frame rate of 24 fps. Light field animations are rendered sequentially as defined by the sequence of input images. Theoretically there is no limitation in the length of the animation, as streaming technologies are applicable for light field animation rendering as well. The animation presented in Figure 8 however was generated as a static setup of light field samples, being uploaded on light field instantiation. Here the maximum length of the light field animation is limited by the render target's total GPU memory size.

### 3.4. Light Field Production

For a successful integration of light fields into a game title game developers are in need of a sophisticated production pipeline to generate light fields from content created by commercial 3D modeling software commonly used in the computer game industry, e.g. *Autodesk Maya*. Currently none of the available 3D modeling tools directly supports light field exports of any kind. Using our *Maya* light field generation plug-in, arbitrary complex 3D content can be converted to light field data sets. Any rendering engine available for *Maya* can be used to generate light field samples from 3D content to exhaust *Maya*'s complex shading and rendering capabilities. Current versions of the plug-in support *Maya Software Renderer* and *Mental Ray*. The plug-in allows single light fields to be created automatically or a set of light fields from a predefined animation sequence. In the case of animation, individual light fields are generated for each time step and being merged to a light field collection data set to be used as a light field sequence.

A light field is exported by automatically rendering the scene from pre-defined sample camera positions given by the spherical approximation. The rendering is performed in two steps. The RGB color values are rendered in a high-quality rendering pass employing all of the image quality features defined by the artist. Image improvement techniques available with *Maya*'s renderers, however, also effect per-pixel

depth information stored with the rendering result. Depth information is extracted in a second render pass, rendered at low quality using *Maya*'s software renderer yielding unfiltered per-pixel depth values.

A separate rendering process is employed to convert the complete set of *Maya* samples to the light field representation described in Section 2. For each camera position the rendered high-quality images in combination with the depth information are used to reconstruct a view-dependent depth relief. This is achieved by constructing a highly tessellated polygonal mesh representing the near plane of the viewing frustum of the individual camera. Each vertex is now displaced according to its correct depth value. This geometry is used for light field generation by projecting the vertices to the hemisphere opposing to the current camera position. The projected color and depth information is stored as a parabolic map for each camera.

### 4. Discussion and Results

Comparable light field rendering techniques presented in the past provide 6 DOF for view synthesis at real-time frame rates but do not suffice all of the demands of computer games. *Spherical Light Fields* presented by Ihm [IPL97] employ a comparable parametrization at a significantly higher sample count yielding much larger data sets. Light fields are rendered at high quality showing only minor ghosting artifacts. The data size and high amount of samples considered in the light field reconstruction, however, reduces flexibility and aggravates rendering performance. *Unstructured Lumigraph Rendering* [BBM\*01] and *Free Form Light Fields* [SVSG01] implement a free-form parametrization that provides 6 DOF at varying sampler counts. Both techniques apply a depth correction of rays for high-quality light field rendering but still lack accurate silhouette reconstruction and correct per-fragment depth values necessary for scene composition. For *Unstructured Lumigraph Rendering* a geometric approximation is used for depth correction which has to be generated separately, stored and processed with the light field data. No geometric approximation is stored for *Free Form Light Fields*. Instead, a polygonal mesh is generated from the sample positions freely distributed in space around the captured object. For light field rendering the generated mesh is subsequently subdivided to

yield a fine geometric approximation based on depth values stored with individual samples. The time consuming subdivision process affects rendering performance to a high degree. None of these techniques neither provide a complete production pipeline for automated light field generation from sophisticated 3D scene content nor do they provide parameterizable light field rendering or LOD techniques.

In this paper we demonstrated the usability of our spherical LFR technique for real-time games. Our technique provides a valuable extension to current rendering techniques used in real-time gaming applications. We have shown that spherical light fields provide a high-quality alternative to represent complex models in dynamic scenes. The parameterized rendering of light fields yields the key to dynamic light field content allowing the appearance of the light field to be controlled in real-time. Our rendering technique is optimized for parallel rendering of multiple light fields and the composition of light fields with dynamic scenes. Best rendering results, however, are achieved for scenes including a single high-quality light field of a detailed 3D model included in a polygonal environment to enhance visual quality. With LOD rendering light fields sampled at a resolution of  $256 \times 256$  are rendered at a screen size of  $512 \times 512$  with frame rates up to 85.56 fps for a LOD corresponding to 12 samples. (LOD(42): 79.1 fps, LOD(162): 60.4 fps). Multiple (10) instances of the same light field are rendered at up to 69.58 fps for an LOD(12) and 43.25 fps for an LOD(42).

## Acknowledgements

This work is partially funded by grant KO-2960-6/1 from the German Research Foundation (DFG).

## References

- [ACB\*07] ANDUJAR, C., BOO, J., BRUNET, P., FAIREN, M., NAVAZO, I., VAZQUEZ, P., VINACUA, A.: Omni-directional relief impostors. In *Computer Graphics Forum (Proc. Eurographics EG'07)* (2007), pp. 553–560.
- [BBM\*01] BUEHLER C., BOSSE M., McMILLAN L., S.GORTLER, COHEN M.: Unstructured lumigraph rendering. In *Proc. ACM SIGGRAPH* (2001), pp. 425–432.
- [Bli78] BLINN J. F.: Simulation of wrinkled surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)* (Aug. 1978), pp. 286–292.
- [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 *Proc. Computer graphics and interactive techniques* (2000), pp. 307–318.
- [CLF98] CAMAHORT E., LERIOS A., FUSSELL D.: *Uniformly Sampled Light Fields*. Tech. rep., Univ. of Texas at Austin, 1998.
- [DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard clouds for extreme model simplification. In *Proc. ACM SIGGRAPH* (2003), pp. 689–696.
- [DYB98] DEBEVEC P. E., YU Y., BORSHUKOV G. D.: Efficient view-dependent image-based rendering with projective texture-mapping. In *Proc. Eurographics Rendering Workshop* (1998), pp. 105–116.
- [GGSC96] GORTLER S., GRZESZCZUK R., SZELISKI R., COHEN M.: The lumigraph. In *Proc. ACM SIGGRAPH* (1996), pp. 43–54.
- [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.
- [OBM00] OLIVEIRA M. M., BISHOP G., McALLISTER D.: Relief texture mapping. In *Proc. ACM SIGGRAPH* (2000), pp. 359–368.
- [Sch95] SCHAUFLE G.: Dynamically generated impostors. In *GI Workshop on Modeling, Virtual Worlds* (1995).
- [SVSG01] SCHIRMACHER H., VOGELGSANG C., SEIDEL H.-P., GREINER G.: Efficient Free Form Light Field Rendering. In *Proc. Vision Modeling and Visualization* (2001), pp. 249–256.
- [TRSK07] TODT S., REZK-SALAMA C., KOLB A.: Fast (spherical) light field rendering with per-pixel depth. *Tech. Report, University of Siegen* (2007).
- [WWT\*03] WANG L., WANG X., TONG X., LIN S., HU S., GUO B., SHUM H.-Y.: View-dependent displacement mapping. In *Proc. ACM SIGGRAPH* (2003), pp. 334–339.