Real-Time Rendering and Acquisition of Spherical Light Fields

---

Echtzeit Rendering und Akquisition Sphärischer Lichtfelder

Dem Fachbereich Elektrotechnik und Informatik der Universität Siegen

zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften (Dr.-Ing.)

zur Genehmigung vorgelegte Dissertation

von

Dipl.Inf. Severin Sönke Todt

Siegen – Dezember 2008

1. Gutachter:	Prof. Dr. A. Kolb
2. Gutachter:	Prof. Dr. G. Greiner
Vorsitzender:	Prof. Dr. V. Blanz

Abstract

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

This dissertation demonstrates that the presented image-based rendering technique provides a significant improvement compared to previous approaches. Two different rendering implementations are explained which make use of the 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.

The presented light field techniques open up for the implementation of efficient and flexible rendering approaches. This work presents an efficient level of detail rendering approach for light fields and introduces a flexible rendering technique for remote access to light field representation in a web-based client server application.

For the acquisition of spherical light fields with per-pixel depth a new acquisition system is presented which makes use of recent advances in 3D sensor technology to acquire combined RGB and depth images directly.

Zusammenfassung

Bildbasierte Renderingmethoden haben sich in der Vergangenheit als effiziente Alternative zu traditionellen Renderingmethoden auf Polygonbasis erwiesen. Diese Doktorarbeit präsentiert eine neue Lichtfeld-Renderingmethode die unter Ausnutzung von pro Pixel Tiefenwerten eine hochqualitative Bildsynthese in Echtzeit ermöglicht. Für einen diskreten Satz an gleichmässig auf der Oberfläche einer Kugelrepräsentation angeordneter Samplepositionen speichert die Technik kombinierte RGB und Tiefenwerte in einer gemeinsamen parabolischen Textur. Die Bildsynthese erfolgt auf der GPU und ist in einem angepassten Fragment Program umgesetzt, das zur Bestimmung der Fragment Farbe korrekte Bildinformationen aus benachbarten Samplepositionen extrahiert.

Im Vergleich zu denen in der Vergangenheit vorgestellten Lichtfeldverfahren stellt die in dieser Arbeit präsentierte Technik eine signifikante Verbesserung dar. Es werden zwei unterschiedliche Renderingverfahren dargestellt, die beide die gleichmässige Samplingstruktur nutzen, um Disparitätsprobleme zu vermeiden und virtuelle Ansichten mit sechs Freiheitsgraden zu ermöglichen. Während das eine der Verfahren einen iterativen Ansatz zum Rendering von Lichtfeldern mit Tiefenkorrektur implementiert, verfolgt das andere einen Raycasting Ansatz und erzielt im Vergleich zum erst genannten Verfahren deutlich bessere Ergebnisse. Mit Hilfe der in beiden Verfahren zur Anwendung kommenden pro Pixel Tiefenkorrektur werden bei gleichzeitigem Verzicht auf umfangreiche Geometrieverarbeitung Ghosting Artefakte signifikant reduziert.

Die präsentierten Lichtfeld-Renderingverfahren ermöglichen die Umsetzung weiterer effizienter und flexibler Renderingmethoden. Diese Doktorarbeit demonstriert einen effizienten Level of Detail Ansatz für die Synthese von Lichtfeldern und stellt ein neues flexibles Verfahren für den Remote Zugriff auf Lichtfeldrepräsentationen auf Basis einer Web basierten Client-Server Anwendung dar.

Für die Akquisition sphärischer Lichtfelder wird im Rahmen dieser Arbeit ein Verfahren vorgestellt, dass neueste 3D Sensorsysteme nutz, um kombinierte Farb- und Tiefendaten direkt zu akquirieren.

Contents

List of Figures

List of Tables

List of Code Samples

1 Introduction

With the invention of the *SKTECHPAD*, the first man-machine graphical communication system, Sutherland fathered interactive computer graphics and graphical user interfaces in 1963 [110]. Since then research in the field of computer graphics focused on achieving photo realistic rendering results from statically growing geometric input data at real-time frame rates. A decade after the birth of interactive computer graphics, Blinn's and Newell's invention of *Texture Maps* [10] and *Bump Maps* [11] revolutionized the visual quality of computer graphics. Their concept of images being used as input data to render compelling scenes is basis of today's image-based rendering techniques [53].

Image-based rendering (IBR) describes a set of techniques that allow three-dimensional graphical interaction with objects and scenes whose original specification began as images or photographs [72]. IBR approaches solve three-dimensional graphics problems by designing data structures that can be robustly computed from images and can subsequently be used to create high quality images at minimal computational cost [47].

Light field techniques adapt the idea of using image based representations of a scene to generate new virtual views by sampling the amount of light traveling through space. The idea behind light fields dates back to the year 1846 when Faraday was the first to propose that light should be interpreted as a field [26]. Faraday's idea combined with discoveries about the properties of light, including the transport and scattering of light, led to pregnant insight in theoretically photometry. With the idea of surface illumination by artificial lighting in mind, Gershun defined the light field concept, which defines the amount of light traveling through every point in space in every direction [32]. Gershun recognized that the amount of light arriving at points in space varies smoothly from place to place and could therefore be characterized using calculus and analytic geometry [57].

This thesis present a sophisticated solution to capture, store and display Gershun's light field using state-of-the-art digital imaging devices and computer graphics technology. The thesis provides a powerful technique to generate artificial photo-realistic renderings of arbitrary complex scenes in real-time.

1.1 Light Field Rendering

Light field techniques sample the amount of light traveling along rays in space (radiance) from pre-defined sample positions. If a bounded 3D observation space is free of occluders, the radiance along rays through space can be assumed to be constant. Under this assumption, and for an object being placed within that region and being illuminated by an unchanging arrangement of lights, the radiance along all rays in space can be measured for locations outside the convex hull of the object. Thus, light field techniques sample a discrete subset of the radiance along rays in observation space.

The positions from which the radiance is sampled is defined by the light field parametrization of the 3D observation space. The parametrization is one of the individual features of light field rendering techniques. Data representation and light field synthesis algorithms are steered by the definition of the parametrization. If the set of defined sample positions is chosen to be dense enough, new perspectively correct images can be constructed for virtual viewing positions which have not been acquired before. This idea is called *light field rendering* [61].

1.2 Scientific and Technological Challenges

The optimal tradeoff between rendering quality and storage efficiency has been in focus of light field research in the past [104] and still is today. Three major challenges have been identified to be crucial for the successful implementation of light field techniques. The storage efficiency which is steered by the light field parametrization, the rendering performance and quality as well as the complexity and accuracy of the light field acquisition process define these major challenges.

Parametrization and Representation The parametrization of the observation space defines the light field representation and thus has high impact on storage efficiency, rendering performance and -quality as well as the techniques applicable for light field acquisition. The parametrization is to be chosen dense enough to allow arbitrary virtual views to be reconstructed without noticeable artifacts but sparse enough to open up for efficient storage schemes.

Rendering Performance and Quality The tradeoff between quality and rendering speed is the key factor to the overall performance of a light field rendering technique. To some degree it is steered by the parametrization of rays. Interpolation schemes applied within the synthesis, however, are the most important aspects of quality and performance. Given a set of light field samples, virtual views are resampled by interpolating the rays from nearby samples. The key to correct interpolation and therefore for best rendering results, is to extract corresponding rays from the nearby light field samples. Non corresponding rays lead to ghosting artifacts in the synthesized view.

Acquisition and Generation Given a defined parametrization, the acquisition of light field samples using digital imaging devices is a challenging task. The demands on acquisition techniques are steered by the factors precision, usability, and availability. Precision in the placement of a capturing device is essential to the successful acquisition of light field samples from positions being defined by the chosen parametrization. To sample a complete light field, either the digital imaging device can be moved to varying position or multiple cameras can be setup in an arrangement according to the parametrization.

1.3 Contributions

The major challenges of light field rendering techniques from representation over rendering to acquisition are in focus of this thesis. This thesis presents an alternative representation for light fields and techniques to exploit this representation for both, efficient rendering and acquisition. In this work these different parts are discussed in detail. Parts of this work have been published by the author in several scientific articles [91, 111, 112, 114–117]. The main contributions of this thesis are:

Spherical Light Field Parametrization with Per-Pixel Depth This thesis presents a new light field parametrization which implements a uniform sampling of the observation space and allow for virtual view synthesis with full six degrees of freedom (DOF). For each light field sample a combined RGB and depth image is stored which represents the light field information as well as an implicit geometry representation of the captured scene in a single image. This new representation provides a storage efficient parametrization which makes high quality rendering approaches applicable.

High Quality Rendering of Spherical Light Fields Two light field rendering approaches are presented in this thesis, both of which exploit per-pixel depth values to achieve high quality light field rendering at real-time frame rates. The one rendering method implements an iterative rendering technique which implements a runtime efficient approach. The second approach is based on raycasting techniques and implements a less runtime efficient approach which provides best rendering quality at moderate frame rates.

Level of Detail for Light Field Rendering This thesis presents an innovative rendering concept which carry traditional level of detail (LOD) techniques over to light field rendering approaches. With the LOD light field rendering a powerful solution for rendering performance management is presented in this work.

Progressive Light Field Rendering For remote network based access to light field representations a progressive light field rendering technique has been developed. This technique implements a rendering approach which is focused on minimizing data transmission and successive refinement of light field renditions. From a sparse light field representation a render client progressively refines local data in order to provide high quality results for selective areas of interest, on user demand.

Acquisition of Spherical Light Fields A complete prototype system that is capable of sampling a spherical light field with per-pixel depth is presented in this work. The proposed system acquires depth and RGB images synchronously, without the need for additional geometry processing, and provides immediate visual feedback even for incomplete light field representations.

1.4 Outline

The remainder of this thesis is organized as follows. Chapter 2 reviews previous light field rendering approaches in detail and discusses the techniques with respect to the three main technological challenges. Chapter 3 describes the new spherical light field parametrization with per-pixel depth in detail. In Chapter 4 the developed light field rendering techniques are presented which exploit per-pixel depth information in order to achieve sophisticated rendering results. Chapter 5 describes the prototype system setup and the data processing pipeline for the acquisition of spherical light fields from physical objects. It explains how spherical light fields can be generated from synthetic objects efficiently. This thesis is concluded in Chapter 6.

2 Related Work

This chapter introduces the concept of light field rendering. It reviews light field rendering in general and related light field rendering techniques presented in the past in detail. 4D light field approaches aiming at the virtual view synthesis with six DOF are discussed and evaluated based on critical light field issues.

2.1 The Plenoptic Function

The problem of how light interacts with surfaces in a volume of space can be interpreted as a transport problem, the transport of photons along paths through space. The *radiant energy* or *flux* in a volume V is denoted as Φ. The flux describes the amount of energy flowing through a surface per unit time and is measured in watts. The energy is proportional to the wavelength λ. The flux therefore should be notated as ϕ_λ, the radiant energy ϕ at wavelength λ. For the rest of this thesis, however, λ is dropped and ϕ is considered for a specific wavelength λ.

The flux that leaves a surface is described by the *radiance*, denoted by L and measured in watts (W) per steridian (sr) per meter squared (m^2). Steriadians measure a solid angle of direction, and meters squared are used here as a measure of projected area of surface.

Let dA be the surface area from which the radiant energy is emitted in direction θ relative to the normal $\vec{n}$ of dA through a differential solid angle $d\omega$ (see Figure 2.1, left). The radiance (L) leaving the area dA then is expressed by:

$$d^2\phi = L dA \cos\theta d\omega \tag{2.1}$$

Considering dA and $d\omega$ to be vanishing small $d^2\phi$ represents the *flux along a ray* in direction θ.

The radiance along all rays in a region of 3D space has been dubbed the *plenoptic function* [1]. The plenoptic function is defined as a 7D function of

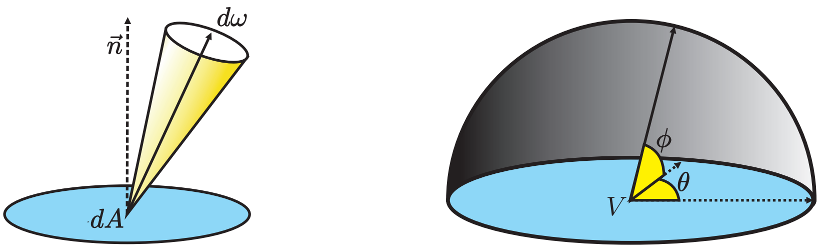


Figure 2.1: Left: Radiance is flux per unit projected area per unit solid angle. Right: Reduced 5D representation of the plenoptic function based on position and direction.

the radiant energy passing through a point in space $V(V_x, V_y, V_z)^T$ from angle (θ, ϕ), for wavelength λ, at a certain moment in time t:

$$P_7 = P(V_x, V_y, V_z, \theta, \phi, \lambda, t) \tag{2.2}$$

Considering the radiance for a certain wavelength only, eliminating λ and limiting observation to static scenes, erasing time factor t, the plenoptic function is reduced to a 5D function of position and direction [41] (see Figure 2.1, right):

$$P_5 = P(V_x, V_y, V_z, \theta, \phi) \tag{2.3}$$

The plenoptic function reduces to a 2D function of direction for the evaluation of radiance for a fixed location:

$$P_2 = P(\theta, \phi) \tag{2.4}$$

Thus, a regular image taken from a certain position with a limited field of view, can be regarded as an incomplete plenoptic sample at a fixed viewpoint. A complete sample is captured by taking multiple images from the same viewpoint for all possible directions [18].

2.2 The Light Field

Under the assumption of the air being transparent, the radiance along a ray in space remains constant. If we further restrict our interest to light leaving

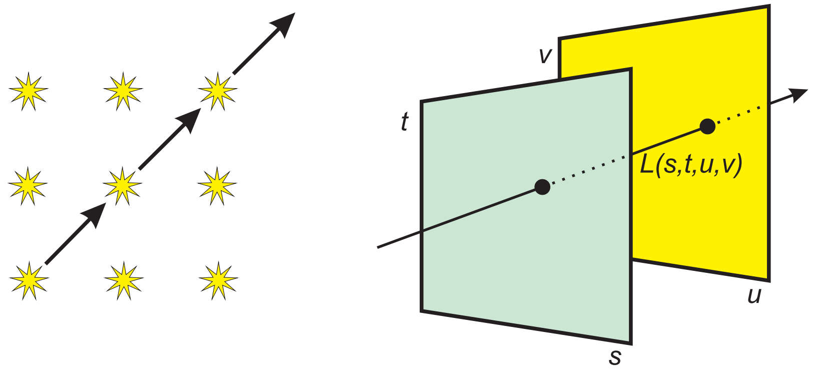


Figure 2.2: Left: Radiance along a ray in space remains constant from point to point for regions free of occluders. Right: Two plane 4D light field parametrization of oriented lines in space.

an object's surface, the plenoptic function can be measured along some surface surrounding the object of interest. Thus, for locations outside the convex hull of an object the plenoptic function can be measured easily using a digital imaging device. With the radiance being constant along rays from point to point (see Figure 2.2, left), the plenoptic function contains redundant information. At any point in space, the radiance along a ray in any direction is determinable by tracing backwards along that ray through empty space to the surface of the convex hull. This redundancy is exactly one dimension, leading to a 4D function to parameterize the surface points and directions of the convex hull. The reduction of dimensions to a 4D function has been used before to simplify the representation of radiance emitted by luminaires [5, 55]. In 1981 Moon called this 4D function the *photic field* [74]. In 1996, however, Levoy and Hanrahan introduced this function of radiance along rays in empty space as *light field* to the computer graphics community [61]. In the remainder of this work only this 4D light field is considered.

For bounded geometric objects being placed within a 3D space free of occluders, all views of an object from outside its convex hull may be generated from a previously captured 4D light field. Virtual views are generated from the representation of the 4D light field as parameterized space of oriented lines. As originally presented by Levoy and Hanrahan the parametrization of oriented lines is defined by the lines' intersections with two planes in arbitrary positions. Two local coordinate systems are defined for theses two planes: (s, t) for the first plane and (u, v) for the second plane. An oriented line L passing through the bounded 3D space is then parameterized by the intersection with these two planes $(L(s, t, u, v))$ (see Figure 2.2, right). The two plane representation of the 4D light field is called a *light slab* [61].

To display an arbitrary view of the captured object which has not previously been acquired, a 2D slice of the 4D light field is resampled. For each image ray passing through a pixel center on the 2D slice the radiance is then approximated by interpolating the 4D plenoptic function from the nearest samples on both, the camera and the image plane.

2.3 Light Field Acquisition

A major burden in the use of light fields is the acquisition of dense samples of the plenoptic function in order to approximate the continuous 4D light field. First approaches aiming at the acquisition of light field samples made use of mechanical gantries. With this acqusition technique a camera is attached to the end of a gantry arm and the arm is subsequently moved to multiple positions. Two useful gantry configurations are the planar and spherical ones. The planar configuration allows the end effector of the gantry arm to move within a planar working space in order to acquire light field samples for a two-plane parameterized light field as described in Section 2.2. A famous example of such a configuration was also used to acquire a light field of Michelangelo's David statue [62]. In a spherical configuration, the end-effector can travel on the surface of a sphere, allowing multiple light slabs and spherical parameterized light fields to be acquired. While these gantries can capture a dense sampling of a light field very precisely, they assume a static scene, are bulky and extremely costly [17].

Dynamic scenes may be captured using arrays of cameras. With the ability to capture dynamic scenes, light fields of complex dynamic objects can be acquired. Available camera arrays in the research area of light field acquisition include the video camera array in the *Virtualized Reality Project* at *CMU* [46], the 8x8 webcam array at *MIT* [133], the 48 pantranslation camera array [135], and the *Stanford Multi-Camera Array* [59, 127, 128]. Such camera array systems, however, are costly and include a huge amount of equipment. Thus, these systems are hard to move and are mainly useful in a laboratory setting.

Recently, research has been focused on exploiting optics to trade of the spatial resolution of a single camera for multiple viewports in order to build mobile devices for light field acquisition. These techniques mount a planar arrangement of microlenses in a camera body [78] or construct a multiple lens gadget that is mounted to a conventional SLR camera [31] to capture a scene from many slightly varying viewpoints. Using such a system, a light field is captured at a single exposure. However, these devices are capable of acquiring two plane parameterized light fields, only. Until now, none of these devices is publicly available.

2.4 Light Field Classification

Since light field rendering was introduced to the computer graphics community, research in this field has brought up various parameterizations and rendering techniques. With the evolution of light field techniques several taxonomies have been introduced for categorization. No common classification, however, has been standardized. Kang [47] classified light field rendering approaches by the technique being applied for ray interpolation. The intermediate data representation of the light field is chosen as categorization taxonomy by McMillan and Gortler [72]. Most prominently the amount of geometric data applied to assist the view synthesis has been proven to be well suited as a criteria for the classification of light field techniques. The *IBR–Continuum* [103] categorizes existing approaches based on the amount of geometry.

In this thesis the idea of the IBR–Continuum and the intermediate data representation are picked up to formalize a catalogue for categorization. Additionally the aspect of sampling uniformity is taken into account, as it significantly influences overall quality of synthesis techniques. The categorization criteria, namely *Geometry*, *Intermediate Data Representation*, and *Uniformity* are described in detail in the following paragraphs.

Geometry Light field rendering techniques differ in the amount of geometric data being applied to assist the image synthesis process in order to determine correspondences for rays being captured from different sample positions. The more detailed a geometric representation, the more precisely correspondences are established. A more precise geometry representation, however, effects storage costs to a high degree. The light field acquisition process is driven by the type and level of detail of geometric data which is required by a certain light field technique. Acquisition devices and the acquisition process have to be designed with respect to the geometric requests.

Light field approaches are differentiated in three catagories, according to the type of geometry being accounted for within the light field rendering technique:

- **No Geometry:** Approaches that do not rely on any geometric data.

- **Implicit Geometry:** Geometry assisted light field approaches relying on implicit geometry data expressed as image correspondences, binary volumes, or depth information per pixel.
- **Explicit Geometry:** Image based synthesis techniques which exploit explicit geometry representations such as polygonal geometry descriptions for image synthesis.

These three distinct characteristics define the first dimension of the classification taxonomy, shown in Figure 2.3.

Intermediate Data Representation Approximate polygonal scene geometry, depth images, plenoptic samples captured as digital input images, and images as reference scene models have been identified as intermediate data representations by McMillan and Gortler [72]. The type of intermediate data representation is the steering component of an image based rendering approach's capability to interactively synthesize new virtual views.

A technique's intermediate representation is crucial to the synthesis' ability to generate new virtual views without costly pre- or post-processing operations being applied to the input data. The availability of direct data visualization methods is one of the main features requested by the computer graphics community for real-time rendering techniques. Direct access to light field data, however, requires the data to be efficiently stored. Thus, the light field representation must provide efficient storage schemes which grant immediate access to light field data and, if present, geometric details. Ideally, the acquisition technique directly supports the intermediate data format such that captured data can be visualized directly.

Light field rendering approaches are categorized in three distinct characteristics according to the complexity and the runtime efficiency of the data processing being required for view synthesis:

- **Direct Rendering:** Image based rendering approaches that take advantage of efficient data representations which allow virtual view synthesis based on the data representation directly, without the need for further data processing.
- **Data Processing on Rendering:** Rendering methods that apply on-the-fly processing of the input data in order to optimize data structures or adjust rendering parameters according to analysis results of the captured scene.
- **Data Pre-Processing:** Light field techniques which are in need of extensive pre-processing steps to extract additional data components from

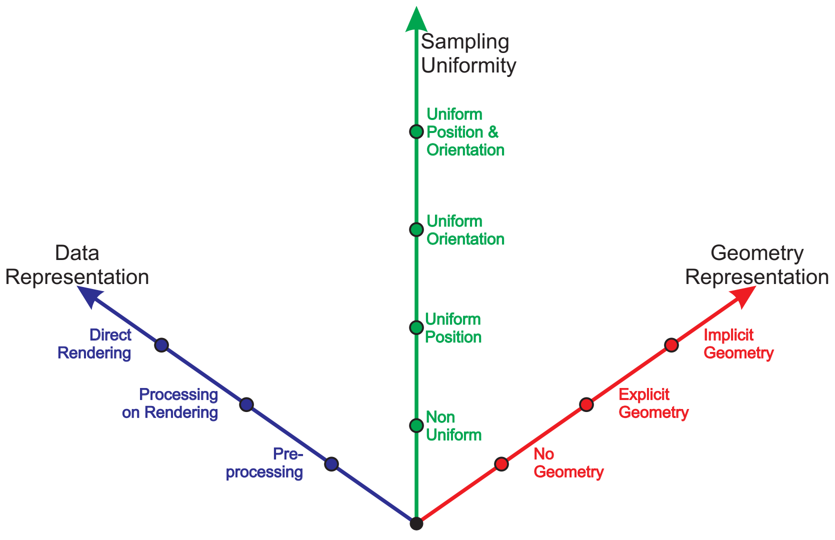


Figure 2.3: Three dimensional categorization of light fields defined by the geometry representation, intermediate data representation, and uniformity of sampling positions and direction.

the input data such as scene reference models, parametrization characteristics or camera- and image parameters.

The intermediate data representation is chosen as the second dimension of the classification taxonomy illustrated in Figure 2.3.

Uniformity Light field rendering techniques should not restrict the virtual viewpoint selection by limiting the viewing direction or viewpoint positions. Rather, they should allow to synthesize arbitrary views without noticeable artifacts or resolution changes for freely chosen viewpoints around an object. This requires the representation to be invariant under both rotations and translations. Such a representation uniformly parameterizes the set of lines intersecting the object's hull [13, 14]. With the set of lines and thus the light field being uniformly sampled, disparity problems can be avoided within light field synthesis. Uniform parameterizations open up for efficient storage and compression schemes to be applied. Compact storage schemes commonly assume a regular data structure to exploit compression techniques for data reduction and intelligent accessing strategies. With acquisition techniques being applied which support a uniform sampling of light fields, uniform representations can efficiently be generated.

Different categories of light field representations are identified by the degree of uniformity:

- **Non Uniform Sampling:** The parametrization does not represent a uniform sampling structure, neither in position nor in direction.
- **Uniform in Position:** Uniform sampling is provided for the position domain.
- **Uniform in Orientation:** Uniform sampling of the direction is achieved by the representation.
- **Uniform in Position and Orientation:** The representation is invariant under both, rotations and translations.

The four specificities of uniformity define the third dimension of our taxonomy being applied to categorize light field approaches (see Figure 2.3).

2.5 Survey of Light Field Rendering Approaches

This section surveys existing light field techniques according to the scientific and technological challenges depicted in Section 1.2 and categorizes these approaches with respect to the categorization taxonomy presented in Section 2.4. This survey, however, cannot justice the large body of light field techniques that have been presented in the past. Notice, that this review is limited to those light fields that are based on the 4D plenoptic function. For a complete review of light field techniques the reader is referred to [47, 101, 103].

2.5.1 Two Plane Light Field Rendering

Light field rendering as proposed in the original paper by Levoy and Hanrahan [61] restricts objects to lie within a convex cuboid bounded by two planes, the camera plane and the image plane. Rays passing through the observation space are parameterized by their intersection points with these two planes. The two plane light field technique captures a subset of such rays in order to synthesize new virtual views by interpolating the radiance being captured from discrete sample positions.

Parametrization

For a discrete set of uniformly distributed sample positions on the camera plane, the radiance along rays is captured for each sample position. The rays passing

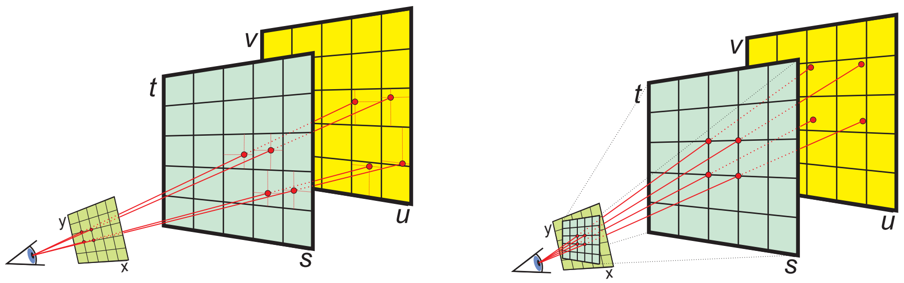


Figure 2.4: Left: A 2D slice of the 4D light field is resampled to generate new virtual views by interpolating the radiance from nearby samples for each pixel. Right: Texture based light field synthesis reduces the interpolation scheme to a simple determination of texture coordinates on a per fetch basis.

through the bounding region and converging in the sample position are captured. As each ray is parameterized by its intersection point with both planes, the amount of rays and the angular distance between rays is determined by the the image plane's sampling resolution. In general, a uniform sampling is chosen for the image plane with a significant higher sampling resolution, compared to the sample positions on the camera plane. Thus, this approach provides a representation that is invariant under both rotations and translations. Obviously, data volume is steered by both, the amount of sample positions and the image plane resolution.

Geometric Representation

No additional geometric data is being included in the two plane light field representation.

Data Representation

Without any geometric data being integrated, the complete light field sampling can be regarded as a collection of 2D digital images which are accessible directly without the need for further data processing.

Synthesis

Light field synthesis is performed on a per-pixel basis by interpolating the light field samples of adjacent sample positions. For each virtual viewing ray being defined by its intersection with the two planes $L(s,t,u,v)$ adjacent sample positions are identified from the camera plane intersection at coordinates (s,t). To determine the final color, light field sampling data is extracted and interpolated for the image plane intersection point at (u,v) for each of the adjacent sample positions (see Figure 2.4, left). Here, quadralinear interpolation generates virtual views with only a few aliasing artifacts, whereas nearest neighbor

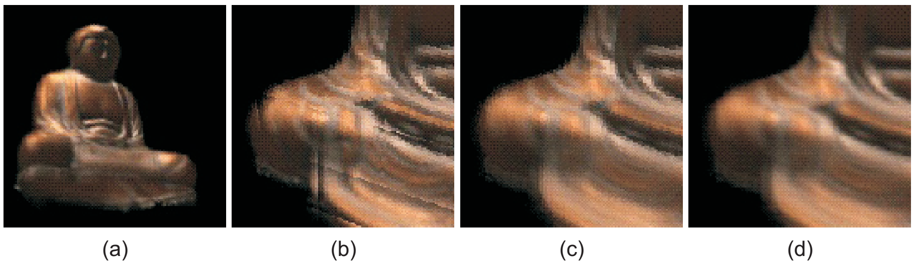


Figure 2.5: The effects of interpolation on ray synthesis. a: Light field rendering of the Happy Buddha model with quadralinear interpolation. b: Closeup rendered with no interpolation. c: Linear interpoaltion on the (s,t) plane. d: Quadralinear interpolation in (s,t,u,v). All images courtesy of Levoy and Hanrahan [61].

and linear interpolation result in noticeable artifacts (see Figure 2.5).

The overall visual quality, however, is effected by the amount and density of light field samples being available for view interpolation. With the increase of samples, the rendering quality enhances whilst the storage efficiency suffers from dense sampling patterns. The rendering performance of this direct interpolation scheme based on per-ray interpolation is limited by the virtual view's target resolution as a ray is being evaluated for each target pixel.

Using texture mapping techniques the interpolation can be implemented more efficiently [104, 121]. For this rendering approach a polygonal representation of the camera plane quadrilateral is drawn with the virtual viewpoint's viewing transformation being applied. The quadrilateral is defined by multiple fetches such that a single fetch $(F(C_0, C_1, C_2, C_3))$ is defined by four sample positions on the camera plane $C_n = (s_n, t_n); n = [0,3]$. For each fetch, texture coordinates $T_n = (u_n, v_n); n = [0,3]$ are determined by intersecting the rays from the virtual viewpoint through the sample positions with the image plane. The texture coordinates are then being applied to texture map extracts of light field samples to the rendered camera plane fetches (see Figure 2.4, right).

While increased rendering performance is achieved using the fetch based interpolation technique, image synthesis quality suffers. Visible seams are visible at the fetch boundaries. If overlapping fetches are rendered and blended towards the edges, partial synthesized views are smoothly blended and thus, noticeable visible edges in the final composed image are avoided. At the borders of the fetches, however, the blending results in slightly ghosting artifacts due to incoherent visual information being blended in these regions.

Although full 6 DOF are available for virtual viewpoint selection, the virtual

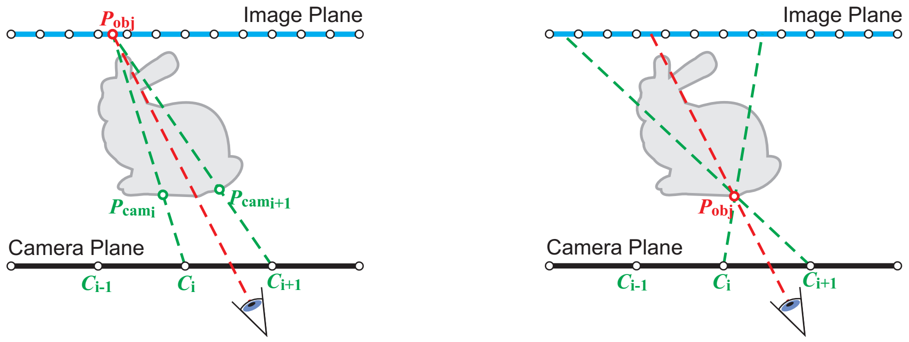


Figure 2.6: Depth correction of rays. Without depth correction, the intersection points observed from adjacent cameras do not necessarily correspond to an identical surface point. With depth correction, camera rays passing through a common surface point are used for interpolation.

viewing position is limited to lie within a viewing cone, defined by the size and the arrangement of the camera- and the image plane. Thus, it takes multiple light slabs to represent all possible views of an object. Therefore line space may be tiled using a collection of light slabs.

It has been shown that light field rendering 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 image plane, resulting in deviating intersection points on the image plane (see Figure 2.6, left).

Acquisition

For the acquisition of light field samples a variety of acquisition devices is applicable. Gantries as well as camera arrays can be used to acquire two plane light fields. With multi-lense and multi-camera setups dynamic light fields are acquirable.

Categorization

The two plane light field rendering technique represents a light field approach which implements image synthesis based on previously acquired samples of the plenoptic function without the need for additional geometric data. It provides a uniform sampling pattern which is invariant in both rotations and translations. Using state-of-the-art acquisition techniques, light field samples can be exploited for image synthesis directly without further processing.

2.5.2 The Lumigraph

The *Lumigraph* approach samples the plenoptic function along a cubic surface around an object of interest [35]. It provides all information that is needed to simulate the light transfer from one region of space on the surface of the cubic setup to all other regions [63]. This cubic representation of the plenoptic function is equivalent to a representation defined by six light slabs as proposed by Levoy and Hanrahan [61]. It allows to synthesize virtual views with full 6 DOF without any constraints concerning the positions and orientation of the virtual viewpoint, as long as it is chosen to lie outside the region bounded by the six light slabs.

Parametrization
As the lumigraph representation is built from six two plane setups, it provides a uniform parametrization of both, position and direction for each of the six two plane setups. However, the uniformity aspect does only hold for a single two plane setup. For the cubic setup of six light slabs, regions of non-uniformity occur at the boundary edges of two adjacent light slabs. In these regions the same boundary sample position is represented in two adjacent light slabs. While being sampled from identical positions for both light slabs the sampling direction abruptly changes by 90°. Thus, the lumigraph cannot be considered to be uniformly sampled in the directional domain.

Geometric Representation
Gortler et al. have shown that, while improving the quality of radiance interpolation, the amount of input samples can significantly be reduced if geometric information about the scene is taken into account to identify ray correspondences. The geometric information can take the form of a coarse triangle mesh, a binary volume [15], or per-pixel depth information [95, 122]. However, Gortler et al. suggest storing an explicit polygonal approximation of the observed object.

Data Representation
As the geometric data is essential for the lumigraph rendering approach, geometry processing is indispensable for image synthesis. While the light field samples are stored according to the two plane light field approach, the explicit geometric 3D model is processed and stored independently. In practice, the geometric model is processed prior to the image synthesis process in a separated task.

Synthesis
To avoid ghosting artifacts which result from incoherent light field samples being interpolated, a geometric scene representation is exploited to ensure rays consistency. Without the object's geometry being considered, a virtual viewing ray is reconstructed from sample rays which intersect the image plane at

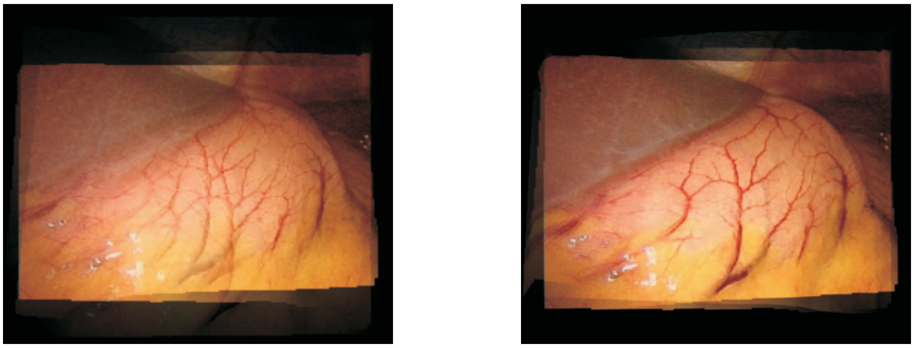


Figure 2.7: Lumigraph rendering results of a stomach data set. Left: View synthesis without depth correction of rays. Right: Rendering results with depth correction of rays being applied for image synthesis based on depth maps. Image courtesy of Vogelgsang and Greiner [122]

the same position. These sample rays, however, are likely to intersect the object at different positions and thus, represent incoherent light field data (see Figure 2.6, left).

With additional geometric information about the observed object being available, the ray-object intersection point P_{Obj} can be determined for a ray $L(s,t,u,v)$. Then, for the ray L and a given $C_i(s_i,t_i)$ one can compute corresponding $I'(u',v')$ for a ray $L'(s_i,t_i,u',v')$ that intersects the object at the same surface point P_{Obj} (see Figure 2.6, right).

Improved rendering results showing less ghosting artifacts for the same (s,t) sample resolution are observed with the depth correction being applied (See Figure 2.7). The effectiveness of the depth correction of rays is dependent on the geometry's level of detail. On the one hand, more precise geometry representations result in improved depth correction. On the other hand, complex raytracing techniques are to be applied to establish ray-object intersections for detailed geometry representations and storage cost are effected by the geometry's precision to a high degree. Rendering efficiency can be improved if fetch based rendering approaches are applied to the lumigraph representation, comparable to the fetch based technique described in Section 2.5.1. In contrast to the straight forward lumigraph rendering approach the fetch based approach's performance is not effected by the target resolution but the amount of fetches being drawn [19,105]. Notice that disparity artifacts [15] occur for situations in which the virtual view spans over a boundary edge of the cubical setup. Then the non uniform sampling of direction lead to visible discontinuities along the edge.

Acquisition

Gortler et al. suggest storing a rough polygonal approximation of the observed object in order to allow for depth corrections. To recover a geometric model of the scene, however, additional effort has to be spent. 3D scanning technology [94] as well as sophisticated stereo vision [92] and image based feature extraction methods [7, 24] are applied to extract the geometric representation. Light field samples can be acquired similar to the two plane representation described in Section 2.5.1.

For flexibility reasons Gortler et al. have presented an acquisition approach which accept input images from arbitrary placed cameras. If a geometric scene representation is available prior to light field sampling and intrinsic as well as extrinsic camera parameters can be evaluated for each input image, an input image can be projected onto the geometry and re-projected into the pre-defined sample position. This technique, known as rebinning [56], then allows to generate a lumigraph representation using commodity imaging devices.

Categorization

The lumigraph representation implements a uniform sampling of the plenoptic function which is invariant in translations, but exhibits non uniform sampling of direction at the boundary regions of two adjacent light slabs. Explicit geometric representations are utilized for depth correction of rays to optimize rendering quality and reducing sampling complexity. The geometry representation, however, is extracted and processed in a separate task. Thus, image synthesis cannot be performed without the geometry being processed in advance.

2.5.3 Spherical Light Field Rendering

Spherical light field rendering techniques overcome the problem of discontinuities observed with multi light slab setups by parameterizing rays using a spherical representation. The use of spherical parametrization schemes provides a symmetric representation of the complete flow of light, which allows for handling arbitrary viewpoints and directions [44]. Several flavors of spherical parameterizations have been published in the past under various names. All of these approaches define the bounding volume around an object of interest to be a spherical volume. Commonly they implement a parametrization which define sample positions to be equally distributed on the surface of the bounding sphere. The parametrization of direction, the representation of individual light field samples, as well as the rendering process, however, differ significantly between approaches. This section discusses different spherical representations.

Spherical Light Fields *Spherical light fields* have been introduced to the computer graphics community by Ihm et al. in 1997 [44]. Spherical light fields define a representation scheme that is based on two spheres. Sample positions are defined by uniformly distributed discrete surface points on the first, the *camera sphere*. For each of the sample positions, a second, so called *directional sphere*, is utilized to parameterize the directional domain for a certain sample position. Using these two spheres, a ray is defined by its intersection with both of these spheres (see Figure 2.8, left).

Parametrization

Sample positions are defined on the surface of the camera sphere using two positional variables (θ_p, ϕ_p). Discrete values of (θ_p, ϕ_p) are applied to formalize sample positions on the camera sphere's surface. Discrete directions are defined using the directional variables (θ_d, ϕ_d) to define surface points on the directional sphere. The directional sphere is positioned tangential to the sample position. Thus, a ray passing through a sample position is explicitly defined by the sample positions and its intersection with the directional sphere $(L(\theta_p, \phi_p, \theta_d, \phi_d))$.

As L can be expressed as a combination of two functions $L(\theta_p, \phi_p, \theta_d, \phi_d) = f_d(\theta_d, \phi_d) = (f_p(\theta_p, \phi_p))(\theta_d, \phi_d)$, the task of sampling the 4D spherical light fields is reduced to the finite approximations of two spheres. Discretization of both, the positional and the directional sphere, is initialized based on an octahedron, where each triangle face corresponds to the eight regular patches on the sphere. Each triangular face is then successively subdivided into four finer triangles. Following this approach arbitrary fine discretizations of the positional- and the directional sphere are achieved. For the positional sphere, each of the vertices of the polyhedron represents a discrete sample position. On the directional sphere, the values of a plenoptic sample is associated with the barycentric center of a triangular face. The subdivision process guarantees the parametrization to be invariant in position and direction, thus providing a uniform parametrization of both domains.

Geometric Representation

The two sphere representation does not integrate any geometric details about the scene.

Data Representation

In practice, up to 65K vertices are generated for the positional sphere and a level 5 subdivision is applied to discretize the directional sphere. With 24 bit color coding this results in about 1.5 GB data storage. The uniform representation, however, allows wavelet compression schemes to be applied to the image data [25, 99, 134]. With wavelet compression being applied a compression ratio of up to 22.4 : 1 can be achieved by Ihm et al. [44].

Synthesis
Light field synthesis is performed using a raycasting approach based on the spherical representation by rendering the smooth shaded triangles of the positional polyhedron. A color is assigned to each vertex such that it corresponds to the color of the directional sphere's fetch which is intersected by the virtual viewing ray. Per-pixel color values within the triangles are interpolated based on barycentric weights.

The quality of this rendering approach is limited by both, the chosen resolution of the directional sphere, as it defines the sample image resolution, and the subdivision level chosen to parameterize the positional sphere. Reducing the amount of sample positions will result in visual details not being sampled and additional loss in image synthesis quality due to interpolation techniques being applied to relatively large triangles. Without the actual scene geometry being taken into account, ghosting artifacts appear for sparsely sampled light fields [44].
Acquisition
With up to 65K positions being used in practice, this spherical approach is suited for artificial light fields to be generated from synthetic data. The acquisition of physical objects, however, is a challenging task. Spherical light fields as proposed above have not been acquired in the past. Nevertheless, spherical gantries as described in Section 2.3 could be utilized to acquire such type of light field.
Categorization
The spherical light field parametrization represents a uniform sampling of position and direction. Image synthesis is performed directly on the input samples without any pre- or post-processing being applied to the input data. With a dense sampling of position and direction, good rendering results are achievable at high storage costs but without geometric assistance.

Two Sphere Parametrization The *two sphere* parametrization implements a spherical parametrization of sample position and direction using two identical uniform spherical representations. Both of these are defined as the spherical convex hull of a scene which is to be captured. The two sphere representation was first introduced to the computer graphics community in 1998 by Camahort et al. [15].
Parametrization
The spherical representation chosen for position and direction is akin to the one chosen by Ihm et al. [44] to parameterize camera space as described in the previous paragraph. Discrete sample positions are achieved by subdividing a spherical approximation based on a polyhedra which provides the most popu-

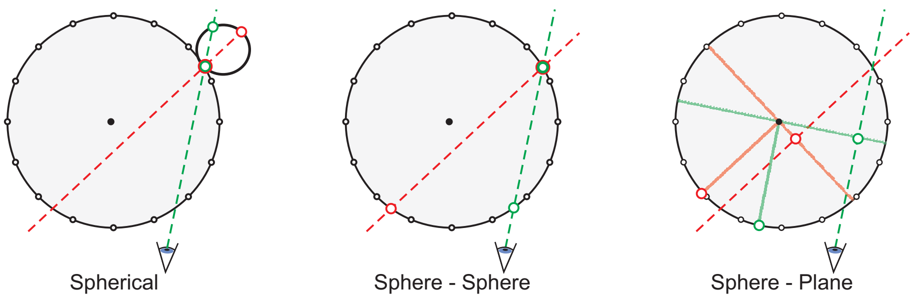


Figure 2.8: Left: Spherical Light Fields use intersection with a positional sphere (large circle) and directional sphere (small circle). Middle: Two-Sphere parameters are determined by intersecting the same sphere twice. Right: Sphere-Plane coordinates consist of the intersection with a plane, and the normal direction of the plane.

lar subdivision of the unit sphere [30]. However, Camahort et al. construct a special polyhedral generator by initially subdividing the 20 faces of an icosahedron. The generator being used for the subdivision process then provides 60 identical faces. By successively applying the subdivision process L times, $4^L \times 60$ faces are generated. In practice, $L = 5$ or $L = 6$ yielding 61K and 245K faces are chosen to create a spherical parametrization for position and direction. Usually, both parameterizations are chosen to be of the same granularity (see Figure 2.8, middle).

In contrast to the camera space parametrization presented by Ihm et al., Camahort et al. define discrete sample positions as well as sample directions to be represented by the barycentric center of the triangular fetches.

Geometric Representation

No geometric information is being represented in the two sphere light field representation.

Data Representation

The huge amount of sample positions and -directions effects storage efficiency to a high degree. Assuming a 24bit RGB color scheme, a total of $N \times N \times 3Bytes$ is consumed (with N being the patch count). Thus, a total of approximately 10.4 GByte of memory is consumed to store a 61K parametrization. With spherical wavelets [98] being applied, storage costs can be significantly reduced by a compression ration of up to 60:1.

Synthesis

Light field synthesis is implemented using a ground truth ray tracing approach [33]. For each synthesized ray, the intersection points of the ray and the unit sphere are computed. In a second step the two intersected patches of the positional and the directional sphere are identified to compute the final color for the pixel being associated with the current ray. Rendering performance is thus proportional to the desired image size. In comparison to light field rendering introduced by Levoy and Hanrahan, this rendering approach takes up to three times longer than rendering a single light slab [15]. The two sphere rendering approach, however, achieves improved rendering quality compared to Levoy's and Hanrahan's approach. Discontinuity artifacts which are observed at light slab boundaries for surround light fields as implemented for the lumigraph by Gortler et al. do not appear. However, improved rendering quality comes at the price of densely sampled light fields and thus increased data volume. Best rendering quality is achieved for spherical parameterizations yielding 20K and above sample positions (see Figure 2.9).

Acquisition

For synthetic scenes, two sphere light field representations are built using a ray tracer which can be instructed to shoot individual rays, joining pairs of points on the sphere to determine the light transport between two fetches. The acquisition of physical objects has not been in focus of Camhort et al.'s work. It could, however, theoretically be implemented using spherical gantries and digital imaging devices.

Categorization

The two sphere parametrization of light fields yields a light field approach that is capable of producing high-quality virtual views from densely sampled light fields without the need for geometric data for depth correction of rays. The uniform sampling of direction and position abet constant rendering quality for the overall surrounding of the scene.

Sphere - Plane Parametrization The *sphere plane* parametrization was introduced by Camahort et al. [15] as an alternative approach to the two sphere light field representation. Thus, the sphere plane parametrization adopt some of the parametrization issues being introduced with the two sphere approach (see Figure 2.8, right).

Parametrization

Discrete sample positions on the spherical surface of the convex hull around an object are determined by subdividing an icosahedron. Each barycentric center of a triangular fetch then represents a sample position. A plpanar image is associated with every sample position. The planar image represents a light field sample for a discrete position. The associated image is generated by parallel projection along the inward looking normal direction of the triangular fetch which defines the concrete sample position. The image plane is defined

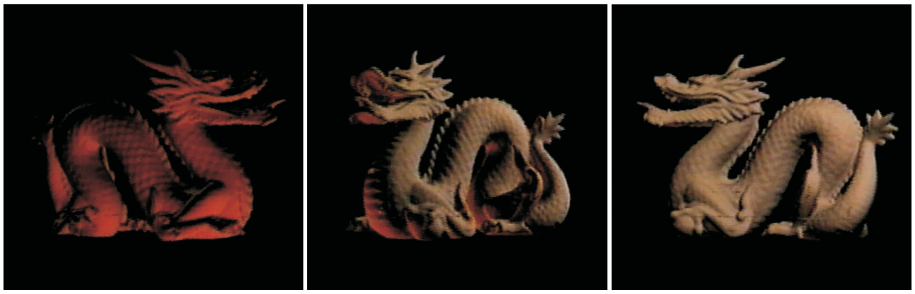


Figure 2.9: Rendering results of the two sphere light field rendering approach. Light field rendered from a two sphere light field sampled at a sample resolution of 65K sample positions. Image courtesy of Camahort et al. [15].

to be oriented orthogonal to the fetch normal and positioned at the center of the unit sphere. For each defined sample position a parallel projection of the synthetic scene is stored. Additionally, depth maps storing per-pixel distances are stored with each light field sample. The depth map captures orthogonal signed distances of the visible object surface to the image plane on a per-pixel basis.

Geometric Representation

Light field samples and depth images are stored as separate textures. While the RGB representation of the captured scene is stored in the light field images, the depth images store an implicit geometric scene representation as orthogonal distances of the object's surface to the image plane.

Data Representation

With two textures being stored for each sample position, the data volume is effected by the amount of sample positions and the sampled image resolution to a high degree. In practice, thousands of light field samples are utilized to represent a complete light field. However, a compression ration of up to 60:1 is achieved with JPEG [83] compression techniques being applied to the image data and Lempel-Ziv [136] compression schemes being applied to the depth map. Then, a light field sampled from 20K positions at a resolution of 256×256 is roughly about 170 MBytes, including the depth maps.

Synthesis

Light field rendering is implemented based on texture mapping techniques. Each sample image is assigned as texture to the vertices of the triangular fetch from which's barycentric center it has been captured. Within the light field synthesis the vertices of each triangle fetch are projected onto the fetch's image

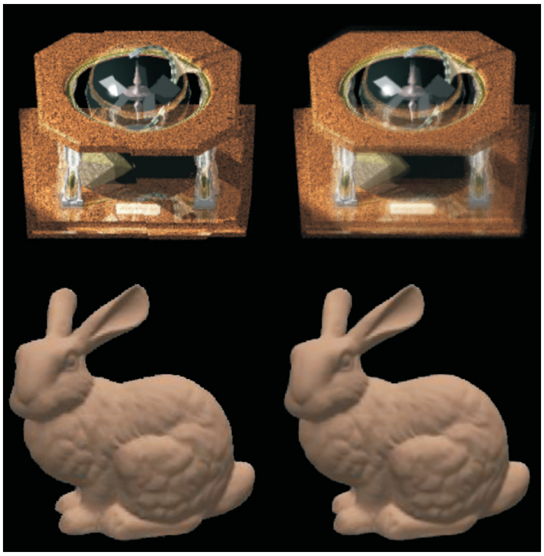


Figure 2.10: Rendering results of the sphere plane light field rendering approach. Top row: Light field synthesized from a dataset containing 1280 sample images at 256×256 pixel resolution. The left image was rendered without blending being applied to fetches. Notice the seams at the fetch boundaries. The right image shows rendering results with blending being applied. Notice the ghosting artifacts a the boundaries of the fetches. Bottom row: Light field of the Stanford Bunny, rendered from a light field containing 20480 samples each at a resolution of 256×256 pixel with depth correction of rays (Left: Without blending, Right: With blending) Image courtesy of Camahort et al. [15].

plane to determine each vertex's texture coordinates, using the virtual viewpoint as center of projection. The texture samples of vertices being shared by adjacent fetches will in the general case provide incoherent visual information as adjacent samples represent orthogonal projections with varying central viewing direction. Thus, this incoherence is exposed as visual seams at the triangular edges of the spherical approximation (see Figure 2.10, left). Applying texture blending on overlapping fetches, the visual discontinuity of sharp edges is omitted at the price of slightly blurring artifacts in these regions (see Figure 2.10, right).

With the available depth map being exploited for depth correction of rays, visible seams and ghosting artifacts are reduced to a reasonable amount (see Figure 2.10, bottom row). For depth correction of rays the depth image is traced to establish a ray-object intersection for three vertices of a single fetch [125, 126]. If the disparity of the determined depth values, however, exceeds a pre-defined threshold the fetch is subdivided at rendering time to account for geometric details of the captured object. This subdivision process is iteratively repeated, until a predefined minimal fetch size is achieved or the depth disparity does not further exceed the given threshold. In the worst case, the subdivision is repeated until the fetch size narrows down to the size of a single pixel.

The subdivision process is driven by the current viewing parameters. Thus, the final topology of the triangle mesh being used for image synthesis varies with changing virtual viewpoints. As a consequence, moving the viewpoint around will lead to noticeable artifacts resulting from topology changes. The overall rendering quality of the sphere plane light field approach is dependent on both, the resolution of the depth map and the resolution of the images. In contrast to the two sphere light field rendering approach the performance is not limited by the target resolution but by the amount of sample positions.

Acquisition

As stated by Camahort et al. this representation is especially suited to generate light field representations from complex synthetic scenes. Although being theoretically possible, physical objects have not been acquired in the past. For synthetic scenes the images are generated with any standard rendering engine by adjusting the view settings and rendering the scene with parallel projection from the desired sample position. Depth maps are easily generated by extracting the z-Buffer depth information and evaluating the distance to the image plane.

Categorization

The uniform representation of this approach allows to continuously synthesize arbitrary views from any position around the object. With the depth correction of rays being applied based on the implicit geometry representation the structure of the spherical approximation is adjusted according to the input data's depth complexity at rendering time. The depth information stored in the depth maps of adjacent fetches steer the subdivision process at run-time. Thus, the final appearance of the subdivided spherical proxy can first be established with all of the depth maps being available.

2.5.4 Unstructured Light Fields

In the past, two major contributions in the field of unstructured light field rendering have been published. Both approaches implement a light field rendering technique which does not require a predefined setup of sample positions and associated input images to synthesize new virtual views. However, both approaches implement different sample representation and image synthesis techniques.

Unstructured Lumigraph Rendering The basic idea of *Unstructured Lumigraph Rendering* is to implement a geometry assisted light field rendering technique that accepts input images from cameras in general positions which are not restricted to a plane or to any specific manifold. It should, however, be general enough to also implement special setups such as the two plane parametrization, a cubic arrangement of light slabs or any sort of spherical parametrization [12]. The work on unstructured lumigraphs has been inspired by *View Dependent Texture Mapping* techniques [22, 23, 86] which apply projective texture mapping for efficient real-time image synthesis [82]. It picks up recent enhancements and extensions to the basic light flied rendering techniques, for rendering digitized three-dimensional models in combination with acquired images [88, 102, 129].

Parametrization

Unstructured lumigraphs define the sample positions to be free of any restrictions in position or orientation as long as they are chosen to lie outside the convex hull of the object of interest. As the sample positions are to be chosen freely without any given constraints on the sample arrangement, uniformity cannot be guaranteed in the general case. If, however, an acquisition device like a gantry is used to acquire light field samples from predefined positions, according to the spherical parametrization, uniformity can be achieved. Then, of course, the benefit of flexible acquisition process is lost.

Geometric Representation

In addition to the light field samples acquired from arbitrary sample positions an explicit polygonal approximation of the scene is to be generated and stored with the set of input images. Typically the polygonal representation is generated in separate task independent of the light field acquisition process. An explicit polygonal approximation of the scene has been declared to be best suited for unstructured lumigraph rendering.

Data Representation

Light field synthesis is dependent on the knowledge of sample positions and imaging parameters. As these are not defined for the unstructured parametrization a prior, internal projection parameters as well as external transformation parameters are to be stored with every sample being acquired. With the amount of sample positions not being defined a prior, assumptions on the memory consumption cannot be made. As a rule of thumb, less samples have to be acquired if a precise geometric representation is available. The granularity of the geometry proxy, however, effects the storage costs to a high degree.

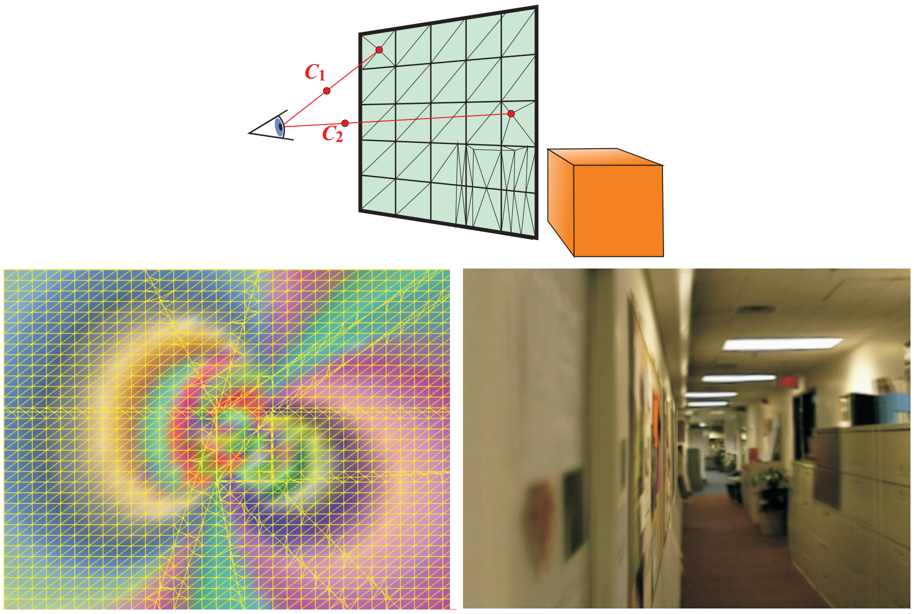


Figure 2.11: Top: Triangulation of the image plane. Bottom Left: Camera blending field. Bottom Right: Rendering results of "Hallway" data set based on the camera blending field. Image courtesy of Buehler et al. [12].

Synthesis

For a virtual view to be synthesized from the input images, the positions of the source cameras' centers are projected into the desired virtual image plane. The projected vertices are then being triangulated and used to reconstruct the interior pixels, according to Heigl et al. [41]. The unstructured lumigraph rendering approach calculates a camera blending field for the desired image plane in a first step and applies projective texture mapping techniques in a second step to synthesize a virtual view. A pixel's blending weight represents a sample camera's contribution to the pixel's final color. The blending weight is calculated with respect to a virtual viewing ray through an image plane pixel r_p. The weight is determined from the angular distance of r_p to the sample camera's central view direction, the sample camera's distance to the observed object, and the sample camera's field of view (FOV). A sample camera's weight is reduced with rising angular distances, increasing distance to the object and with the specific pixel being observed in boundary regions of the FOV.

To efficiently compute the blending field for a certain image plane the blending weight is determined at a set of discrete points on the image plane, only. The discrete points are then triangulated over the image plane and the blending weights are being interpolated. The triangulation of the image plane is performed by projecting the edges of the polygonal proxy to the image plane. All edge-edge crossings are inserted as vertices in the image plane. Additionally, all sample camera positions are projected to the image plane and inserted as vertices. Finally, a dense regular grid of vertices is included on the desired image plane [12]. A constrained Delaunay triangulation is then applied to the vertices of the image plane [100]. For each vertex a set of cameras and their associated blending weights are stored (see Figure 2.11, top and bottom left).

The final image is rendered as a set of projectively mapped triangles. Each triangle is rendered multiple times according to the different sample cameras associated with each of the triangle's vertices and their different textures being mapped to the triangle fetch. Multiple renderings of the same fetch are finally composed by alpha blending according to per-pixel blending weights from the blending field. Since each each polygon is treated independently, smooth transition across polygon edges is not guaranteed. This rendering approach provides sophisticated rendering results for detailed polygonal approximations only. With only a sparse geometric approximation being available, ghosting artifacts become visible in the synthesized image (see Figure 2.11, bottom right).

Acquisition

The acquisition of detailed geometric representations, is a major burden on the acquisition process. For the acquisition of real objects, complex 3D geometry extraction methods are to be employed to generate detailed models of the captured scene. While commodity camera equipment can be used to capture light field samples from arbitrary positions, extra effort has to be spent to capture the scene's geometry. For this reason, only very simplified versions of the geometry are used for image synthesis. In practice geometry representations like boxes or planes are used to represent a captured scene [12]. This simplification comes at the price of visible ghosting artifacts in the final image. If not a very simple representation is chosen, but a detailed model is used instead, complex geometric processing techniques are to be applied prior to the image synthesis process in a separated task to make the geometric representation available.

Categorization

Unstructured lumigraph rendering provides a flexible light field rendering technique that may be used to provide uniformly sampled light fields. But the aim of this approach is targeted at the free-hand acquisition of light fields. For this general case uniformity cannot be guaranteed. The explicit geometric representation of the scene effects all aspects of this light field synthesis approach.

Rendering quality is improved by applying detailed geometric representations, whilst it effects pre-processing and storage costs to a high degree.

Free Form Light Field Rendering *Free Form Light Field Rendering* was published by Schirmacher et al. [96] as an alternative light field rendering approach to unstructured lumigraph rendering. With this approach, however, the optical centers of the sample cameras are considered to lie on a common arbitrary free form surface, while the camera position and orientation can be chosen freely [96].
Parametrization
The free form surface is defined by a convex triangulated polygonal mesh, also called the camera mesh. The camera mesh is built from the sample positions, each defining a vertex of the polygonal mesh. For each camera position an individual image plane is defined. Thus, camera space is defined globally by the camera mesh, while image space is defined separately on a per camera basis by their image planes. Each ray passing through the bounding volume of the observed object can then be defined by the intersection with the camera mesh and the image plane of the associated camera. To ensure each viewing ray to be reconstructible, each camera is to be placed such that the complete silhouette of the observed object is captured. Under this restriction, arbitrary viewing rays passing through the convex hull do always intersect the camera mesh and at least one camera image plane (see Figure 2.12, left). However, uniformity in position and direction cannot be guaranteed, as the sample positions and their orientation is not pre-defined in a uniform manner.
Geometric Representation
The free form light field approach exploits geometric scene details for depth correction of rays. The technique can operate on a variety of geometric representations, such a explicit polygonal representation [35], binary volumes [15], or depth maps [123].
Data Representation
Light field samples are stored separately from the geometric representation. For each light field sample the camera parameters, namely camera transformation and intrinsic camera projection parameters have to be stored in addition. With the camera arrangement to be chosen freely to lie anywhere outside the convex hull of the observed scene, uniformity in position and direction cannot be guaranteed, as well as storage costs are not determinable in advance.
Synthesis
Image synthesis is implemented by rendering the front faces of the textured polygonal camera mesh. Each triangle fetch of the camera mesh is textured according to the input images of the three corresponding cameras which define

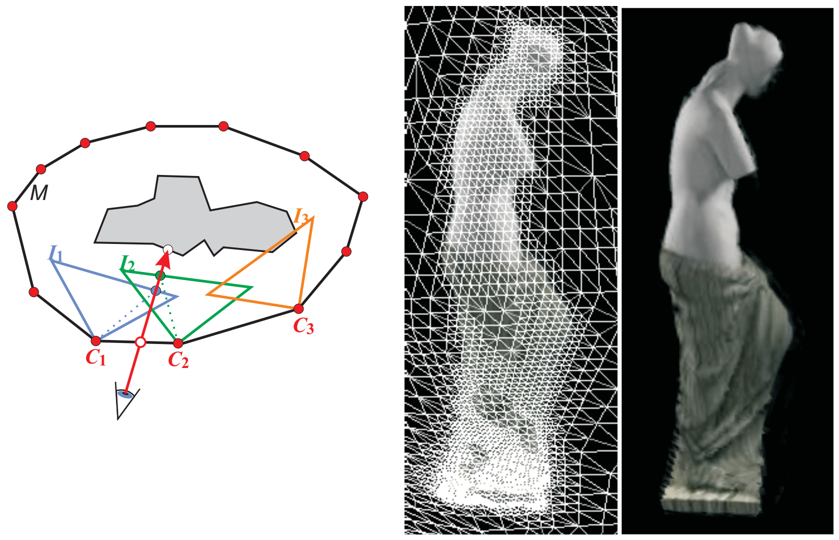


Figure 2.12: Left: Camera Mesh M built from input camera positions C_n, displaying image planes I_n associated with each camera. A viewing ray is defined by the intersection with the camera mesh and the image plane. Middle: Refined camera mesh utilized to determine camera blending weights with depth correction being applied. Right: Rendering results of light field containing 107 input samples with depth correction being applied. Image courtesy of Schirmacher et al. [96].

the vertices of the current triangle. To obtain a triangle's final appearance, its projection onto each of the three camera image planes is rendered separately and alpha blended. If we assume the triangle to be defined by the vertices V_0, V_1, V_2, the triangle is projected to the image plane corresponding to V_0 first, with an alpha value of 1 assigned to V_0, alpha value 0 assigned to V_1, V_2, and the alpha values being linearly interpolated over the triangle. With this procedure being applied accordingly to V_1 and V_2 the resulting three samples are interpolated using the assigned alpha values for the final appearance of the triangle.

If the sampling density is not high enough, this approach leads to serious blurring and ghosting artifacts. With depth correction being applied, these artifacts can be compensated to some degree. With an approximation of the geometry to be known, a ray-object intersection can be determined for each ray emerging from a virtual viewpoint and passing through a vertex. With depth information being available on a per-vertex basis, the vertices of a triangle are then projected onto the approximate geometry surface. An additional depth is estimated for the triangle's center. If the overall depth disparity of the four depth estimations within a triangle exceeds a certain threshold, the triangle is subdivided. This depth estimation and subdivision procedure is repeated until the depth threshold or a minimum triangle size is reached (see Figure 2.12, middle). The minimum triangle size is specified by the target image resolution. Thus, subdivision is repeated recursively until a minimum triangle size of one pixel is reached in the worst case.

With the depth correction being applied to four discrete positions of a triangle, only, high-frequency changes between these depth samples are not accounted for by the depth correction, leading to ghosting artifacts in the final image (see Figure 2.12, right).

The depth correction being applied to the camera mesh results in improved rendering quality. Rendering quality, however, comes at the cost of a performance drop caused by the recursive subdivision process. As the subdivision is steered by per-vertex depth values, which are in turn determined along the ray from the current viewpoint, the topology of the subdivided mesh varies with changing viewpoints. This inconsistent reconstruction of viewing rays is also observed for unstructured lumigraph rendering [12]. While moving the virtual viewpoint around the object, the inconsistence may lead to noticeable artifacts. With the subdivision process being applied to the complete camera mesh at run-time, rendering is performed solely after the acquisition process has been completed.

Acquisition

To allow for depth correction of rays, a geometric representation of the scene is to be generated during the acquisition task. The limitations of the acquisition process being known for unstructured lumigraphs also hold for free form light fields. The acquisition of a scene's geometry is one of the major burdens, while light field samples can be acquired with commodity digital cameras for which the extrinsic and intrinsic camera parameters are known.

Categorization

The free form light field rendering approach implements an efficient way for rendering new virtual views based on a flexible parametrization. Uniformity of this parametrization, however, cannot be guaranteed. With additional geometric information of the scene being acquired, depth correction of rays is applicable. With depth correction of rays being applied, high-quality images can be synthesized. Synthesis then includes a view dependent recursive subdivision process performed at run-time which effects rendering performance. Performance drop and quality gain are strictly coupled with the geometry's level of detail, the minimal triangle fetch size, and the sampling density of the representation.

2.6 Conclusion

Light field rendering techniques have been first introduced to the computer graphics community in 1996 by Levoy and Hanrahan. Since then several enhancements and extensions to the original two plane light field approach have been presented in the past. Each of which is contributing an enhancement or extension to the two plane parametrization. Originally, the two plane light field rendering approach (see Section 2.5.1) implements a parametrization based on two planes. Good rendering results are achieved for very densely sampled light fields, while ghosting artifacts are visible for sparsely sampled ones. The two plane representation implements a uniform parametrization in the positional and directional domain. Virtual view positions, however are limited to a small viewing cone defined by the size and arrangement of the two planes. Two plane light fields are acquirable using a variety of equipment. Camera arrays, gantries and upcoming multilense camera systems can be used to acquire this kind of light fields. Light field samples can be used directly for image synthesis purposes without the need for further processing.

The lumigraph representation extends the two plane approach by using six light slabs to allow the virtual viewing position to be chosen to lie anywhere outside the convex hull of the observed object with 6 DOF. Depth correction of rays is implemented based on a sparse geometric approximation of the scene to reduce ghosting artifacts. However, lumigraph rendering shows noticeable disparity artifacts at the boundaries of two adjacent light slabs. This effect results from non uniform sampling of directions in these regions. While the light field samples can be acquired according to the two plane approach, effort has to be spent on the acquisition of the geometric scene representation. As the geometry is fundamental to the lumigraph rendering technique, it is to processed prior to the rendering task in a separate task. Thus, samples being acquired during the acquisition process can be visualized directly, only, if the geometry has already been processed before.

Spherical light field techniques overcome the problem of visible discontinuities at the boundary regions of two adjacent light slabs by implementing a uniform sampling of positions and directions in a spherical parametrization. All of the spherical approaches presented in Section 2.5.3, however, are in need of a dense sample distribution on the spherical alignment in order to synthesize high-quality virtual views. With depth correction of rays being applied by the sphere plane parametrization, the amount of samples is drastically reduced.

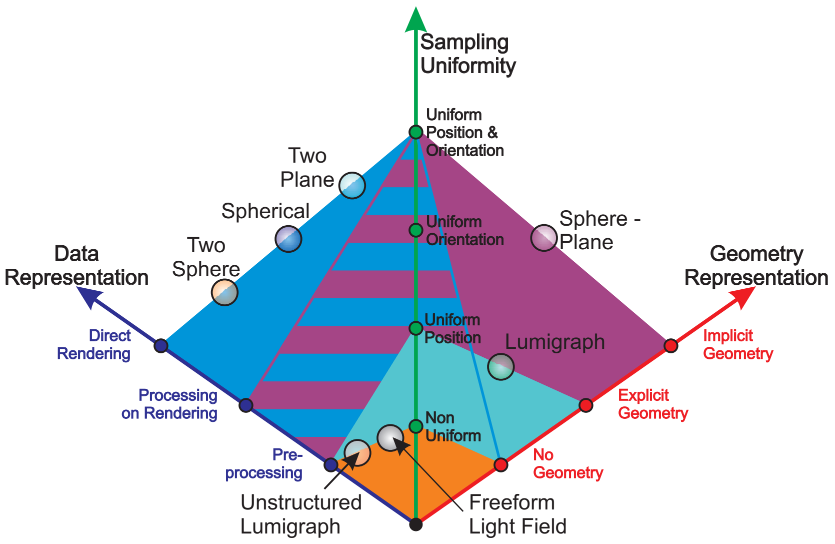


Figure 2.13: Light field rendering approaches categorized by the amount of geometry used for depth correction of rays, data representation and sampling uniformity.

But still, thousands of sample positions are required. Thus, the spherical techniques are especially suited to extract light fields from synthetic objects.

Unstructured light field approaches are focused at accepting input samples from cameras in arbitrary positions. With the sample positions being freely chosen during acquisition, it opens up for a flexible acquisition of light fields, if the camera's intrinsic and extrinsic parameters are known. With the granted flexibility, however, a uniform parametrization cannot be guaranteed for these approaches in the general case. As both approaches rely on some knowledge about the scene geometry, extra effort is to be spent on the acquisition of geometric details which reduces the benefit of easy to use light field acquisition. Extra equipment has to be utilized and costly geometry processing has to be performed prior to virtual view synthesis. With depth correction of rays being applied, high quality rendering results can be achieved for densely sampled light fields.

All of the light field approaches presented in this chapter are capable of synthesizing new virtual views from light field representation at high quality. The achievable quality, however, comes at the high price of extremely dense sampling patterns, very detailed geometric representations or computational costly processing techniques. These issues effect the usability of the presented approaches to a high degree. Two plane light field rendering provides sophisticated rendering results for a restricted region of virtual viewpoints, only, if the light field is sampled very densely. While reducing the sampling count significantly and resolving the virtual point restriction, the lumigraph approach is in need of detailed explicit polygonal geometry to achieve high quality synthesis results of non constant quality. Overall quality is limited by disparity artifacts. Spherical approaches eliminate these artifacts but demand both, densely sampled light fields and detailed geometry to achieve sophisticated quality. While the unstructured light field techniques are aiming at providing a straight forward approach for light field acquisition and rendering, these approaches do not provide a uniform parametrization and exhibit visible artifacts resulting from topology adjustments within the light field synthesis.

This thesis presents a new spherical light field parametrization which combines the benefit of uniformity being implemented by spherical light field techniques and the advantage of geometry assistance in order to reduce sample density. The proposed approach implements a new technique that is capable of synthesizing high-quality virtual views from as few as 42 light field samples based on an implicit geometric representation of the scene. Thus, the presented technique opens up for the acquisition of spherical light fields from physical objects which is not available for spherical approaches presented in the past. A new acquisition technique is presented which captures both, a light field sample and a depth map of the scene in a single exposure and thus opens up for direct data access without the need for costly geometry processing which has been the limiting factor for geometry assisted techniques presented in the past.

3 Spherical Light Field Parametrization with Per-Pixel Depth

This chapter presents a new spherical light field parametrization which provides a uniform sampling pattern that is invariant under both, translation and rotation. The proposed approach efficiently represents combined light field samples and implicit geometric information in a texture based storage format. Sample positions are arranged in a spherical setup based on a spherical approximation being derived from the icosahedron as initial representation. The spherical arrangement of sample positions allows the virtual viewpoint to be chosen with 6 DOF during the light field synthesis process. Light field synthesis can be performed in absence of discontinuity artifacts which are observed for the lumigraph approach and those, which potentially occur for unstructured light fields for non uniform sampled scenes.

The image space parametrization exploits existing environment mapping techniques to represent the radiance along rays passing through a pre-defined sample position. Light field samples are represented as a parabolic environment map which stores combined RGB and depth values on a per-pixel basis. This representation of both, radiance along rays and geometry in a single texture, provides a compact storage scheme, which can be exploited efficiently within light field rendering to generate new virtual views. With the implicit geometry representation being accessible directly as a fundamental part of the proposed parametrization, virtual views can be synthesized precisely at real time frame rates without the need for further geometry processing.

The spherical light field parametrization with per-pixel depth is discussed in detail in this chapter. The spherical parametrization of camera space is outlined in the first section followed by an in detail discussion of the parabolic image space parametrization in the second section. The per-pixel depth representation is explained in the third section. The chapter is closed with an analysis of the sampling scheme and the storage costs followed by a conclusion.

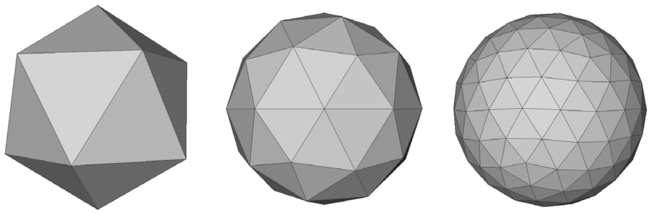


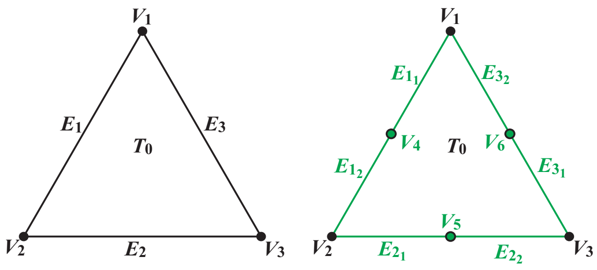
Figure 3.1: Spherical proxies are generated by successively subdividing the 20 faces of an icosahedron (left) into spherical approximations with 42 vertices ($m = 1$, middle) and 162 vertices ($m = 2$).

3.1 Spherical Camera Space Parametrization

Uniform camera space parametrization is achieved by arranging equally spaced sample camera positions on a spherical approximation. Thus, a good spherical 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 the approximation of a sphere [30]. 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 [29, 34]. In contrast to Camahort et al. [15] who construct a special 60 face generator from an icosahedron for the spherical parametrization (see Section 2.5.3) the icosahedron is taken as generator directly for the subdivision process. As will be shown in the next paragraphs, the subdivision scheme being applied to the icosahedron directly provides a flexible and uniform spherical representation.

The subdivision scheme applies a recursive interpolatory subdivision on the solid mesh of the icosahedron (see Figure 3.1). With every iteration each triangle is divided into four (nearly) equilateral spherical triangles. For a top-level triangle T_m defined by vertices V_i, with $i = 1, 2, 3$ which has been subdivided m times a prior, each of it's edges E_j, with $j = 1, 2, 3$, are split into sub-edges E_{j_k} of equal length, with $k = 1, 2$ (see Figure 3.2). New vertices V_l, with $l = 4, 5, 6$ are generated at the split points of the edges. The positions of these vertices V_l are adjusted by projecting the vertices on the unit sphere to maintain a solid spherical approximation. The vertices V_i of the top level triangle T_m and the generated vertices V_l then define the triangulation of the next subdivision level.

Applying the subdivision process m times, a spherical approximation with 20×4^m faces is obtained. In practice, $m = 1$ or $m = 2$ are chosen, yielding 80



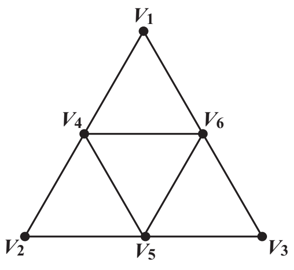


Figure 3.2: The triangle subdivision scheme. A top-level triangle is subdivided into equilateral subtriangles by applying an interpolatory subdivision scheme. The positions of vertices resulting from subdiving an edge into equal sub-edges are adjusted by projecting the vertices on the unit sphere to maintain a solid spherical approximation.

or 320 faces and 42 or 162 vertices, respectively. Note that finer triangulations include all of the vertices of the top level triangulation. Thus, a coarsening of a previously refined spherical approximation is easily achieved afterwards by dropping the inner vertices and adjusting the topology accordingly.

The quality and thus the uniformity of this spherical approximation can be evaluated by means of the ratio of the spherical surface area to the approximated surface area. Ideally, a tessellation into n patches should produce equilateral spherical triangles whose area is $A = \frac{4\pi}{n}$. The icosahedron yields a surface area ratio of 76.08% of A, improving to 92.83% for $m = 1$, narrowing to 96.74% using $m = 2$. Thus, for a spherical light field representation $m \geq 2$ is a good choice. The overall average triangle fetch aspect ratio is about 1.12 (see Table 3.1). Thus, a good uniformity in triangle area is achieved while maintaining the equilateral features of the triangles. As the subdivision process is generated based on an icosahedron yielding 12 positions, this representation provides good spherical approximations while employing a fractional

Sphere Resolution	Minimum Sample Distance	Maximum Sample Distance	Average Sample Distance
12	1.051462	1.051462	1.051462
42	0.618034	0.546533	0.582284
162	0.324920	0.275904	0.299332
642	0.164647	0.138283	0.15073

Table 3.1: Sample distances of spherical parameterizations at different subdivision levels

part of discrete sample positions being used in spherical representations being presented in the past (see Section 2.5.3).

The spherical approximation is stored explicitly as an indexed face set, with each of the vertices defining a sample position. For each vertex, a 4×4 viewing matrix is stored which represents the transformation of world to sample camera coordinates. This matrix is interpreted as extrinsic sample camera parameter for light field acquisition as well as for reconstruction purposes. This spherical approximation of camera positions is denoted as camera sphere in the following.

3.2 Parabolic Image Space Parametrization

To capture the spherical light field according to the spherical camera space parametrization, the radiance along rays has to be captured at each of the discrete sample positions defined by the camera sphere (see Section 3.1). The use of environment maps to capture the incoming light in a texture map has been proven to be an efficient way for representing the radiance along rays passing through a certain point in space [10]. As environment maps record the incident light passing through a single point in space from different directions, each individual environment map of a scene describes a concrete sample of the plenoptic function [71]. In the presented image space parametrization, a raster image of the opposing hemisphere is stored using environment mapping techniques for each camera of the camera sphere.

3.2.1 Environment Mapping Techniques

Various environment mapping techniques have been published in the past. The approaches, however, differ in sampling quality and applicability in the context of light field sampling.

Cubic Environment Maps *Cubic environment maps* [37, 124] utilize six independent perspective images to capture the environment for a predefined position in 3D space. Each of these six images is captured from the center of a cubical setup through each of its faces [36]. Cube maps are either stored using six independent textures or in a single texture using texture atlas techniques [79] to merge these images into a single image based representation. This cubic representation shows fairly good sampling rates. Sampling rates differ by a factor of $3\sqrt{3}$ (≈ 5.2) over all directions. Considering the solid angle covered by a single pixel, i.e. the pixel's area projection onto the unit sphere, it has been shown, that the sampling density slightly increases with the viewing angle in each of the images [39]. Thus, the images show slightly lower sampling rates in the center regions. The memory costs caused by storing six independent images to capture the environment, however, exhibit an severe issue for efficient light field parameterizations.

Spherical Environment Maps A wide spread environment map parametrization used in the computer graphics community is called *spherical environment mapping* [38, 73]. Spherical environment maps are based on the analogy of a small, perfectly mirroring metal ball centered around an object of interest. A single image that an orthographic camera captures when focusing on such a ball from a certain viewing direction can be interpreted as the spherical environment map. Thus, spherical environment maps are more efficient with respect to memory consumption compared to cube maps, as only one single image is needed to represent the environment. However, this spherical parametrization exhibits areas of extremely poor sampling and a singularity [39]. Spherical environment maps show maximum sampling rates for directions opposing the viewing direction and reaches sampling rates close to zero for directions similar to the viewing direction. Additionally a singularity is observed with the spherical parametrization. All points on the sphere with a viewing vector tangential to the sphere show the same point of the environment. As a consequence, this form of environment map is not suitable for a uniform discrete sampling of the plenoptic function.

Parabolic Environment Maps *Parabolic environment mapping* utilizes two environment textures to perform a parabolic environment mapping [40]. The idea behind the parabolic environment mapping technique is similar to that of spherical mapping. Instead of generating the texture by recording the reflection of the environment off a sphere, however, two paraboloids are used, each of which covering an environment hemisphere [2]. Thus, the complete environment is stored in two separate textures, each containing the information of one hemisphere. While being less efficient with respect to storage costs compared to sphere maps, it does not exhibit any singularities. Compared to cube maps it consumes essentially less storage and shows improved sampling rates. The sampling rate varies by a factor of 4 over the complete image. Directions perpendicular to the viewing direction are sampled at a higher rate than directions parallel to the viewing direction [39].

Out of the available environment mapping techniques parabolic environment maps provide the most suitable solution for recording the opposite hemisphere from each of the predefined spherical sample positions. For the parametrization of the hemisphere on the opposite side of a sample position, only one paraboloid is to be extracted.

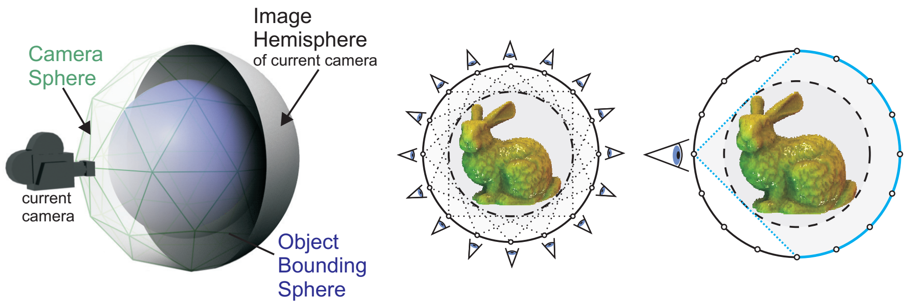


Figure 3.3: Left: Sphere-Hemisphere parametrization of the light field. The object is enclosed in the blue bounding sphere. Virtual cameras are positioned at the vertices of the camera sphere (green). Each camera is recording the opposing hemisphere. Middle and Right: RGB and color values are sampled simultaneous from spherical sample positions on the camera sphere. The camera sphere is to chosen $\sqrt{2}$ times larger than the bounding sphere of the object to ensure that all rays from the current camera through the object boundary sphere will intersect the opposite hemisphere

Note that the camera sphere must be chosen to be larger than 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 (see Figure 3.3).

For each of the spherical camera positions the opposite hemisphere is mapped to a parabolic representation. For each ray emerging from a sample camera position and intersecting the camera sphere at the opposite hemisphere at point S the radiance is mapped to parabolic texture coordinates (u, v) as follows:

$$\begin{pmatrix} u \\ v \end{pmatrix}_k = \frac{1}{2} \begin{pmatrix} \frac{s_x}{1+s_z} + 1 \\ \frac{s_y}{1+s_z} + 1 \end{pmatrix}_k \quad \text{with} \quad S_k = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}_k \tag{3.1}$$

Notice that the opposite hemisphere is captured for each sample position, only. The paraboloid covering the hemisphere in the opposite view direction is not used (see Figure 3.3). Thus, storage costs are minimized by neglecting the second parabolic texture. In practice, a resolution of 256×256 or 512×512 pixels is chosen for the parabolic texture. Examples of source images for the light field are displayed in Figure 3.4.

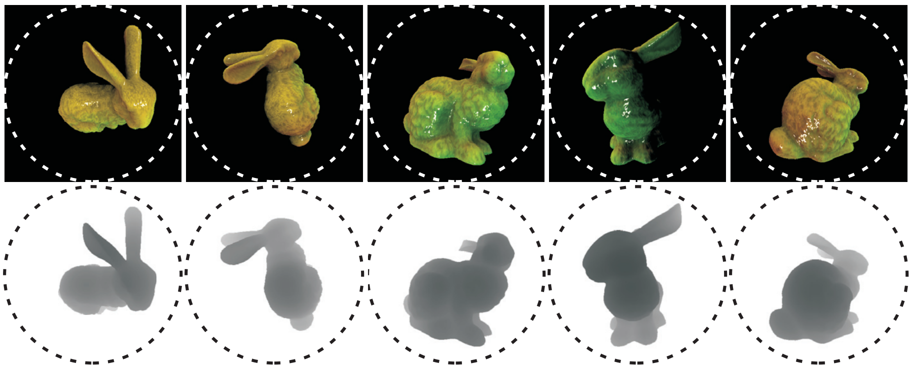


Figure 3.4: Five image samples taken for a spherical light field with 42 cameras. Each image represents a parabolic mapping of the hemisphere for color (top row) and depth (bottom row).

3.3 Geometric Representation

The light field samples acquired from the spherical arrangement of sample positions is supplemented by a representation of the captured scene's geometry. For efficiency reasons a depth map is stored with every light field sample being acquired. Depth maps store the geometry distance for individual rays on a per-pixel basis. The depth map is generated with a resolution according to the light field sample resolution. Using the alpha channel of a captured parabolic light field sample, the depth can efficiently be stored in the fourth channel of the RGBA parabolic map (see Figure 3.4). In the remainder of this work RGB textures with a depth value stored in the alpha channel are reffered to as RGBz textures.

Per-pixel depth values are calculated for each ray as the distance z' from the first intersection point with the object's bounding sphere to the object surface. The ratio of z' to the bounding sphere's ray secant length $z_{\max}$ is then stored as the final depth in the alpha channel of the parabolic texture according to Figure 3.5. This results in a depth value between 0 and 1, which can efficiently be stored as 8 bit value in the alpha channel [85]. Applying an 8 bit quantization on the geometric domain with a depth complexity of 1 yields 256 depth layers, each of which covering a depth interval of $1/256(\approx 0.004)$.

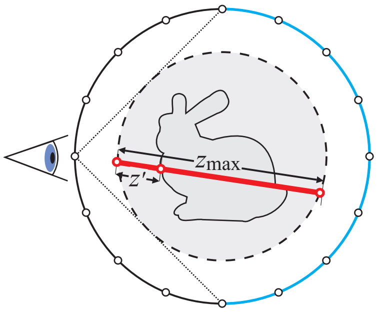


Figure 3.5: The depth value is obtained by dividing the bounding-object distance z' by the bounding sphere's secant length $z_{\max}$.

3.4 Sampling Analysis

According to the Nyquist theorem, in order for a signal to be reconstructed without aliasing the sampling frequency needs to be greater then the Nyquist rate. Chai et al. [16] determine the maximum distance for adjacent sample camera positions ($\Delta s_{c_{max}}$) from the maximum sample rate in the image domain (Δs_i), according to:

$$\Delta s_{c_{max}} = \frac{1}{2\Delta s_i} \tag{3.2}$$

Thus, while limiting the acquisition of light field samples to a uniform image space sampling resolutions in both directions (Δs_i) of $256, 512$ and 1024, a maximum geometric distance of ≈ 0.002, ≈ 0.0009 and ≈ 0.0005, respectively for adjacent sample positions is not to be exceeded in order for the light field synthesis to be performed without aliasing artifacts (see Equation 3.2).

Chai et al. further show, that for a given image space sampling rate, the maximum sampling distance can be increased, if geometric depth information is available with each light field sample being acquired. For N_d discrete levels of depth quantization being available, the maximum sampling camera distance is then determined by:

$$\Delta \frac{s_{c_{max}}}{N_d} = \frac{1}{2\Delta s_i} \tag{3.3}$$

$$\Delta s_{c_{max}} = \frac{N_d}{2\Delta s_i} \tag{3.4}$$

For an 8 bit depth quantization yielding 256 discrete levels of depth, the maximum sampling camera distance ($s_{c_{max}}$) can then be extended to 0.5, 0.25

and 0.125 for image space sampling (Δs_i) of $256, 512$ and 1024, respectively. Thus, with an 8 bit depth representation being available, camera sphere representations resulting from subdividing the initial icosahedron $m \geq 2$ times represent good sampling schemes.

For a subdivision level of $m \geq 2$, the spherical sample setup satisfies the maximum sampling distance constraint (see Table 3.1). Depending on the image space resolution subdivision levels of increased sampling density may be chosed to avoid aliasing artifacts.

3.5 Storage Efficiency and Light Field Compression

The storage efficiency of the light field representation presented above is dependent on the resolution of both, the resolution of the sample image and the sample position density. For each sample position an individual 2D RGBz texture map is stored. Storage costs per sample image (Mem_s) are steered by the image sampling rate s_i as follows:

$$Mem_s = s_i \times s_i \times 4\ byte \tag{3.5}$$

In practice, light field samples are captured at a resolution of 256×256 and 512×512 yielding per sample storage costs of $256 Kbyte$ and $1024 Kbyte$.

The storage costs caused by each light field sample texture are factored by the amount of camera sample positions (N_c) for the complete light field representation. Additional storage costs are caused by storing the spherical approximation being stored explicitly as an indexed face set. Remember, with each vertex, a 4×4 viewing matrix is stored which represents the sample camera coordinate transformation. Storing the viewing matrices using 4 byte floating point values the total memory consumption for a complete light field representation (Mem_{lf}) is then computed according to:

$$Mem_{lf} = N_c \times Mem_s + N_c \times 16 \times 4\ bytes \tag{3.6}$$

Typical sizes of light field representations for a variety of image- and camera space resolutions are listed in Table 3.2.

To reduce the amount of graphics memory consumed by the parabolic maps, commodity hardware-accelerated texture compression schemes can be employed. S3TC texture compression [45] 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 turned out to be sensitive to compression artifacts especially at object boundaries. Fecker et al. [28]

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 3.2: Sizes of typical light fields with and without compression

demonstrated that a compression of a scenery's depth information will cause noticeable loss of quality within light field synthesis. Thus, the DXT3 compression algorithm is used to maintain best results within the rendering task. 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 being generated using the proposed spherical parametrization 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 3.2).

3.6 Conclusion

The spherical light field parametrization with per-pixel depth presented in this chapter represents an innovative parametrization approach which implements a uniform sampling of both position and direction and opens up for virtual view synthesis with 6 DOF for virtual viewpoint selection. It implements an efficient light field representation with respect to storage costs which includes an implicit geometric scene description at minimum additional storage costs based on per-pixel depth values. Light field samples and depth values are stored in a common RGBz parabolic texture map which opens up for standard texture compression techniques to further reduce storage costs.

In contrast to the two plane light field rendering approach presented

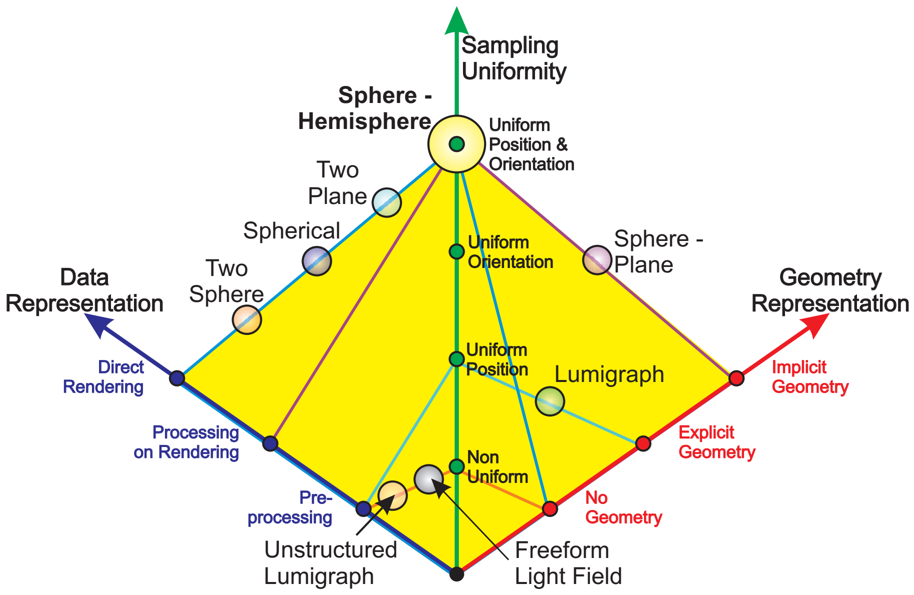


Figure 3.6: Categorization of the sphere-hemisphere fight field parametrization presented in this chapter.

by Levoy and Hanrahan [61] (see Section 2.5.1) this spherical light field parametrization implements a sampling scheme which allows the virtual view-point and virtual view direction to be chosen freely with 6 DOF for virtual view synthesis. The parametrization is chosen sparse enough to open up for efficient storage schemes, but dense enough to allow arbitrary views to be reconstructed without noticeable artifacts. Discontinuities as observed with the lumigraph approach presented by Gortler et al. [35] (see Section 2.5.2) are avoided due to the spherical parametrization being invariant under both translation and rotation. Contrary to free form lightfields [96] and unstructured lumigraphs [12] (see Section 2.5.4) a uniform representation is guaranteed with this spherical parametrization.

The uniform spherical parametrization presented in this chapter is akin to the representations applied for spherical light field rendering presented by Ihm et al. [44] and Camahort et al. [15] (see Section 2.5.3). However, essential less light field sample positions (-90–95%) are needed to sample a complete light field. Storing an 8 bit depth value with every captured ray in a common RGBz parabolic texture map, provides 256 distinct depth layers that significantly reduce the amount of necessary light field samples (see Section 3.4).

With the depth being stored per pixel within the same texture based representation it is accessible directly in conjunction with the RGB data from within a single texture fetch. As will be shown in the following chapter, the proposed representation provides an efficient light field parametrization which facilitates light field synthesis without the need for expensive geometry processing and topology adjustments. The light field representation satisfies the need for a uniform sampling which implements an implicit geometry representation of the scene and provides direct view synthesis capabilities. A comparison to light field parameterizations presented in the past is illustrated in Figure 3.6.

The parametrization presented in this chapter has been presented to the computer graphics community by Todt et al. [115, 116].

4 Spherical Light Field Rendering with Per-Pixel Depth

In this chapter a light field rendering approach is presented which is based on the sphere-hemisphere light field parametrization with per-pixel depth being introduced in chapter 3. The rendering technique exploits the uniform sampling structure and implicit geometric representation to implement an efficient light field synthesis from a sparsely sampled light field representation. Depth correction of rays based on the proposed depth map representation allows for high-quality virtual view generation at real-time frame rates without ghosting or disparity artifacts. Virtual viewpoints can be chosen with 6 DOF for positions outside the convex hull of the represented object.
Image synthesis is implemented on the graphics processing unit (GPU) as a fragment program which efficiently extracts correct image information from adjacent cameras for each fragment by applying per-pixel depth correction of rays based on the parabolic texture representation of image space. Texture samples are exploited within the view synthesis directly without the need for further complex and costly geometry processing or structural topology adjustments.

Two different rendering implementations are introduced in this chapter. While one rendering algorithm implements an iterative refinement approach for rendering light fields, the other approach implements a raycasting technique which provides superior rendering quality at moderate frame rates.

The rendering techniques are parameterizable to be adjustable by means of performance and quality to adapt to changing requirements. Level of detail rendering techniques have been implemented to account for varying object distance and visibility. The optimized rendering techniques allow multiple light fields to be rendered synchronously to generate complex scenes. The efficiency of the uniform light field parametrization in combination with the rendering technique's flexibility provides a solid fundament for the implementation of a web-based remote light field renderer based on a client-server architecture to provide remote access to light field representations of complex scenes.

This chapter describes the GPU based spherical light field rendering with per-pixel depth in detail. The iterative refinement rendering approach and the raycasting technique are explained in the first three sections, followed by a description of level of detail rendering for light fields in the fourth section. The fifth section comprehensively describes progressive light field rendering implemented for remote access to light field data based on a client-server architecture. This chapter is closed with a conclusion in the sixth section.

4.1 Spherical Light Field Rendering

For both of the two light field rendering approaches presented in this chapter, the light field is rendered by rasterizing the front faces of the polygonal camera sphere representation with respect to the virtual viewpoint (P_{eye}). The virtual viewpoint can be chosen with 6 DOF to lie anywhere outside the spherical camera space representation. However, the viewpoint is restricted to be located outside the camera sphere to avoid the vertices of the front faces being culled by frustum culling. Remember, each vertex corresponds to a pre-defined camera sample position (C) (see Section 3.1).

The polygonal mesh is rendered by drawing the set of triangles from a predefined OpenGL display list [76] according to the concrete spherical representation of the light field. For each triangle each of its vertices is assigned one of three distinct color values, namely red (`glColor3f(1.0f,.0f,.0f)`) for camera C_0, green (`glColor3f(.0f,1.0f,.0f)`) for camera C_1, and blue (`glColor3f(.0f,.0f,1.0f)`) for camera C_2. These vertex colors are interpreted as interpolation weights within the light field synthesis. Additionally the parabolic RGBz texture image, the transformation matrix $\mathbf{M}$ of the camera, and the background color are bound as parameters to each of the vertices.

While a common vertex program is utilized which implements standard vertex operations, a customized fragment program is implemented which performs the light field synthesis based on the input parameters (see Figure 4.1). The input parameters such as the sample camera transformation and per-vertex interpolation weights are passed through to the fragment program by the vertex shader. The vertex positions, however, are transformed according to the modelview-projection matrix of the current virtual view.

Within rasterization, per-vertex camera blending weights are interpolated for each triangle being rendered. Within the fragment program interpolated color values and thus interpolated camera weights are available per fragment.

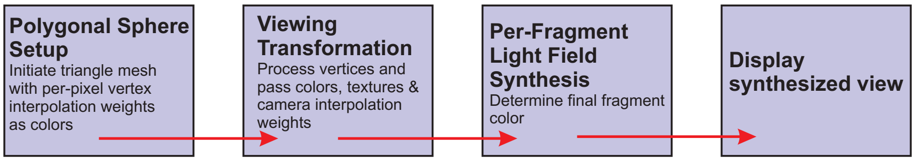

Polygonal Sphere Setup
Initiate triangle mesh with per-pixel vertex interpolation weights as colors
Viewing Transformation
Process vertices and pass colors, textures & camera interpolation weights
Per-Fragment Light Field Synthesis
Determine final fragment color
Display synthesized view

Figure 4.1: Overview of the light field synthesis process. The polygonal mesh of the spherical representation is rendered to synthesize virtual views. A common vertex program implements standard vertex operations and passes per-vertex shader parameters to the fragment program. The customized fragment program performs the light field synthesis to generate a virtual view.

The final color is then determined within the customized fragment program based on the light field synthesis technique. Both of the light field synthesis techniques presented in the following sections implement a specific fragment shader which samples the input textures to extract light field sample data and per-pixel depth values in order to synthesize high quality virtual views.

Note, however, that 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 using OpenGL's standard `GL_LINEAR` interpolation scheme, noticeable render artifacts at the silhouettes appear when interpolating the depth information. Using the nearest neighbour texture lookup scheme (`GL_NEAREST`) ensures appropriate depth information per pixel and avoids depth aliasing artifacts at object boundaries.

4.2 Iterative Refinement

The iterative refinement technique is outlined in Figure 4.2. Four sequential steps are implemented within this light field synthesis approach to establish coherent rays and thus coherent light field samples in order to determine the final fragment color (see Figure 4.3).

First, all variables and parameters, such as the current viewing position and viewing direction are initialized in the initialization step. For the current fragment, a viewing ray is established based on the viewing direction. This ray's intersection with the opposite hemisphere of the spherical representation is evaluated to determine initial texture coordinates and thus initial texture fetches. Based on the initial depth values being extracted from the texture fetche's alpha channel a first object intersection is assessed. The first intersection estimate is then successively refined in the subsequent iterative refinement

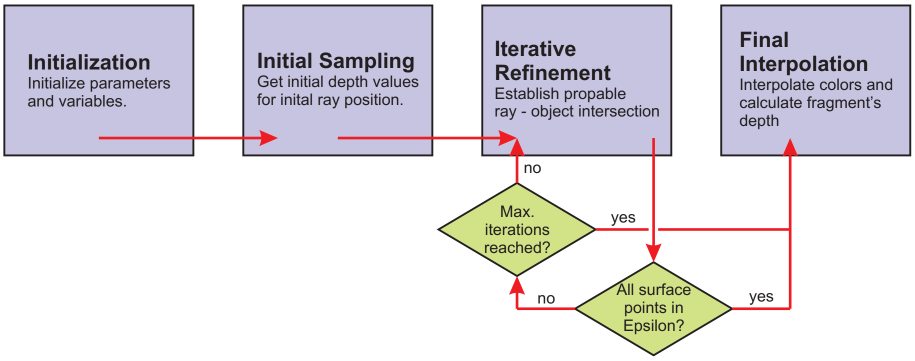

Initialization
Initialize parameters and variables.
Initial Sampling
Get initial depth values for inital ray position.
Iterative Refinement
Establish propable ray - object intersection
Final Interpolation
Interpolate colors and calculate fragment's depth
no
Max. iterations reached?
yes
All surface points in Epsilon?
no
yes

Figure 4.2: The iterative refinement approach employs four sequential steps to establish coherent rays and thus coherent light field samples per fragment.

step. With a ray-object intersection being reliably determined in the iterative refinement process, the final fragment color is evaluated by interpolating appropriate texture fetches.

4.2.1 Iterative Refinement Process

When a triangle is rasterized, each fragment corresponds to a unique position V 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

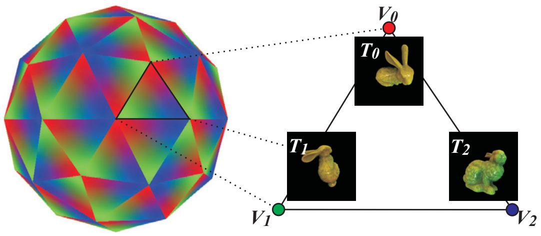


Figure 4.3: Rendering of the polygonal camera sphere representation with distinct colors being assigned to each of a triangle's three vertices. For each vertex a color value representing the sample camera blending weight, the input RGBz texture T_n, and the corresponding camera's transformation matrix are assigned.

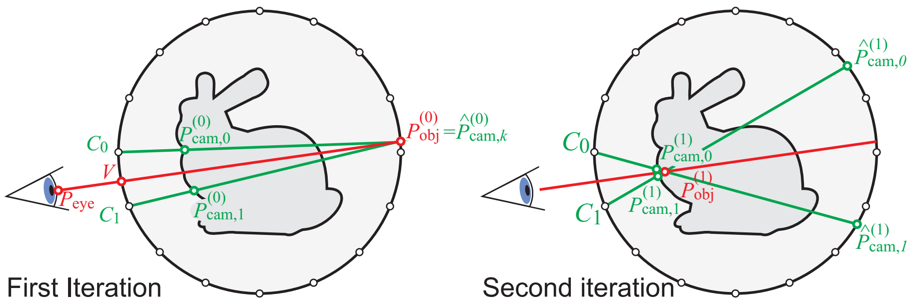


Figure 4.4: First and second step of the iterative depth refinement.

refers to the iteration count (see Figure 4.4).

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

$$S_k^{(i)} \quad = \quad \mathbf{M}_k \, P_{\text{obj}}^{(i)} \qquad \text{with} \quad k \in \{0, 1, 2\}. \tag{4.1}$$

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

$$\begin{pmatrix} u \\ v \end{pmatrix}_k = \frac{1}{2} \begin{pmatrix} \frac{s_x}{1+s_z} + 1 \\ \frac{s_y}{1+s_z} + 1 \end{pmatrix}_k \quad \text{with} \quad S_k = \begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}_k . \tag{4.2}$$

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

$$P^{(i)}_{\text{cam},k} = z \cdot C_k \; + \; (1 - z) \, \hat{P}^{(i)}_{\text{cam},k} \tag{4.3}$$

with $P^{(i)}_{\text{obj}}$ being the object intersection point and $\hat{P}^{(i)}_{\text{cam},k}$ the intersection point being projected onto the sphere using C_k as center of projection. Note that Equation 4.3 is a simple linear interpolation, because z stores the depth value as fractional part of the secant length.

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:

$$P_{\text{obj}}^{(i+1)} = P_{\text{eye}} + \frac{1}{3}\sum_{k=0}^{2}((P_{\text{cam},k}^{(i)} - C_k) \cdot \mathbf{r})\,\mathbf{r} \tag{4.4}$$

with **r** being the normalized direction of the original viewing ray (red line).

As illustrated in Figure 4.4, the iteration can be pursued several times by projecting the updated object point $P_{\text{obj}}^{(i+1)}$ 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 4.3, and eventually calculate an improved intersection point according to Equation 4.4.

The procedure is terminated if the desired accuracy is achieved or a maximum number of iterations is reached. The maximum number of iteration is adjustable at runtime. In practice, however, a maximum number of iterations of 5 turned out to be sufficient.

Accuracy is steered by an adjustable error threshold. The error threshold parameter defines the maximum distance of local estimates ($P_{\text{obj}}^{(i)}$) determined within the iterative refinement process. In case of the maximum divergence of local estimates falling below the error threshold the iteration procedure is terminated. Obviously, ray coherence and as a consequence image synthesis quality is maximized with the error threshold parameter being minimized.

The final color of the fragment is calculated as the weighted sum of the RGB values extracted from the parabolic textures of the different cameras within the final iteration. The weights for each camera correspond to the barycentric coordinates of the fragment with respect to the original triangle. With the primary colors red, green and blue being assigned to the three vertices during geometry setup, the correct weights are automatically calculated through color interpolation during rasterization. With the latest local estimates being averaged and projected to viewing space according to the virtual viewpoint's modelview-projection matrix a depth value is determined for each individual final fragment.

4.2.2 Implementation Details - Iterative Refinement

This section presents in depth details concerning implementation issues of the four sequential steps being performed for light field synthesis within the fragment program (see Figure 4.2).

Initialization Within the initialization the fragment position is determined from barycentric interpolation of the associated triangle's vertex positions (c_n)

taking the input color as interpolation weights. The viewing direction (dir) and the normalized viewing direction ($dirN$) are established based on the virtual viewpoint position (P_{eye}) and the current fragment's position (v) (See Appendix A.1 ll. 31–33 for the corresponding code sequence).

Initial Sampling The initial sampling step establishes the viewing ray's initial intersection with the opposite hemisphere to extract RGB and depth samples from the parabolic textures of the adjacent sample positions as starting point for the iterative sampling process (see Figure 4.5, left).

The viewing ray is noted using a simple ray equation with the virtual viewpoint position P_{eye} being the starting point according to:

$$P_{eye} + \lambda dir = (x, y, z)^T \tag{4.5}$$

With the unit sphere be denoted using the sphere equation

$$x^2 + y^2 + z^2 = 1 \tag{4.6}$$

the ray - sphere intersection is easily determined from:

$$\begin{aligned}
(P_{eye_x} + \lambda dir_x)^2 + (P_{eye_y} + \lambda dir_y)^2 + (P_{eye_z} + \lambda dir_z)^2 &= 1 && (4.7)\\
({P_{eye_x}}^2 + 2\lambda P_{eye_x} dir + \lambda^2 {dir_x}^2)+ & &&\\
({P_{eye_y}}^2 + 2\lambda P_{eye_y} dir + \lambda^2 {dir_x}^2)+ & &&\\
({P_{eye_z}}^2 + 2\lambda P_{eye_z} dir + \lambda^2 {dir_z}^2) &= 1 && (4.8)\\
\lambda^2({dir_x}^2 + {dir_y}^2 + {dir_z}^2)+ & && (4.9)\\
2\lambda(P_{eye_x} dir_x + P_{eye_y} dir_y + P_{eye_z} dir_z)+ & && (4.10)\\
({P_{eye_x}}^2 + {P_{eye_y}}^2 + {P_{eye_z}}^2) - 1 &= 0 && (4.11)
\end{aligned}$$

Substituting term 4.9 with A, term 4.10 with B, and term 4.11 with C the intersection is resolved by determining λ according to Equation 4.12. Note, that the negative solution to λ is ignored as only the intersection with the opposite hemisphere is relevant in this case.

$$\begin{aligned}
A\lambda^2 + B\lambda + C &= 0\\
\lambda^2 + \frac{B}{A} + \frac{C}{A} &= 0\\
\lambda &= -\frac{B}{2A} + \sqrt{\frac{-B}{2A} - \frac{C}{A}}
\end{aligned} \tag{4.12}$$

A, B and C can be expressed as dot products using dir and P_{eye} as follows:

$$A = dir \cdot dir \tag{4.13}$$
$$B = 2(P_{eye} \cdot dir) \tag{4.14}$$
$$C = (P_{eye} \cdot P_{eye}) - 1 \tag{4.15}$$

Thus, with the use of the dot product the intersection of the viewing ray with the opposite hemisphere is efficiently calculated in the fragment program (see Code Sample 4.1).

```
//determine viewing ray - sphere intersection on opposite hemisphere
float A,B,C;
C  = dot(P_eye,P_eye) - 1.0;
B  = 2.0 * dot(P_eye,dir);
A = dot(dir,dir);

float S = max( (B*B - 4.0 * A*C) ,0.0); //components under sqrt
//ignore negative solution - opposite hemisphere, only
float lambda = (-B + sqrt(S)) /A/2.0;

//Sphere intersection
float3 vecS  = (P_eye + lambda * dir);
```

Code Sample 4.1: Initial intersection of the viewing ray with the opposite hemisphere, with respect to the current vertex.

From the initial intersection point, texture coordinates (u, v) are determined according to Equation 4.2 for each adjacent sample camera by transforming the sphere intersection point to local camera coordinates first according to the sample camera's transformation matrix. The parabolic textures are then sampled at the (u, v) texture coordinates to extract RGB and depth values.

With the depth information of the initial sphere intersection to be known for each of the three sample cameras, the iterative sampling is initiated.

Iterative Sampling Within the iterative refinement phase, the initial intersection point and the depth values being evaluated for each of the three adjacent sample cameras are used to determine a valid object intersection point (see Figure 4.5). First, local estimates are determined for each sample position. Then, a new sample position on the viewing ray is determined which is used in the third step to determine error values for the three local estimates. For the new viewing ray sample position the sphere intersection points are updated to establish new depth values per sample camera. Based on the divergence of the updated depth values the iteration is either terminated or restarted in case of the divergence exceeding the given $errorThreshold$ value.

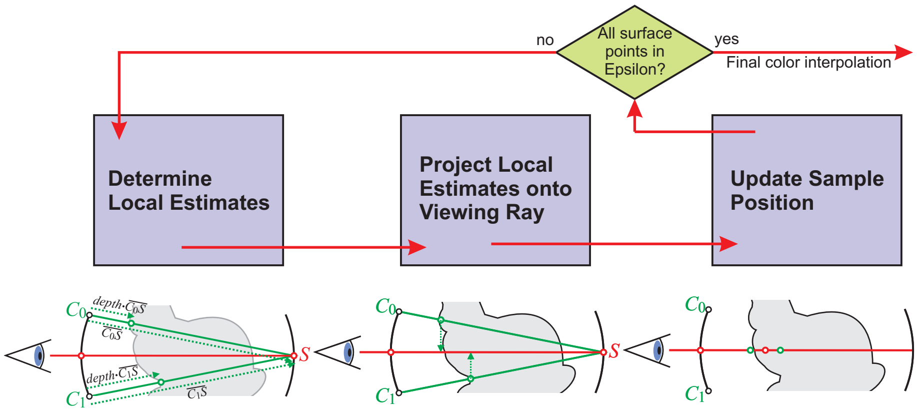

no
All surface points in Epsilon?
yes
Final color interpolation
Determine Local Estimates
Project Local Estimates onto Viewing Ray
Update Sample Position
C_0
$depth \cdot \overline{C_0S}$
$\overline{C_0S}$
$depth \cdot \overline{C_1S}$
$\overline{C_1S}$
C_1
S

Figure 4.5: The iterative refinement phase in detail. Local estimates are determined for each adjacent sample camera by scaling the sampling ray C_0S by the depth value being sampled from the parabolic texture. These estimates are then projected onto the viewing ray in order to determine the new sampling position. If all of the local estimates are positioned within a given error threshold, iteration is stopped.

1. *Local Estimates:* Local estimates are determined for each sample camera by scaling the vector from the sample camera position to the last spherical intersection point by the sampled depth according to Figure 4.5, left.

2. *Update Viewing Ray Sample Position:* The local estimates are projected onto the the viewing ray in order to determine the next sample position on the viewing ray. The updated sample position is determined from the average position of the projected local estimates according to Equation 4.4 (see Figure 4.5, middle and right). In case of more than half of the maximum iterations already processed, however, highly divergent object intersections are assumed. In this case the interpolation operation is replaced by a maximum operation to avoid a deadlock situation (See Section 4.2.3).

3. *Evaluate Error:* For the interpolated new viewing ray sample position the absolute distances to the three local estimates are chosen as error value to evaluate the validity of the reconstructed object intersection points. The absolute distances are weighted by the barycentric weights to ensure more distant camera sample positions to have less influence on the overall error. The overall error is then determined as the squared sum of the absolute distances.

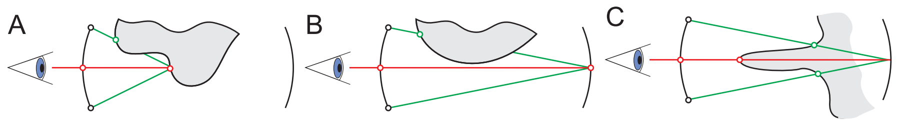


Figure 4.6: Iterative refinement fails at silhouette edges and locations where the correct object point is not visible from all adjacent cameras.

4. *Resample Parabolic Textures:* For the updated viewing ray sample position, new spherical intersection points are established for each adjacent sample camera position according to the ray sphere intersection as performed in the initial step. These new intersection points are then parabolically mapped yielding updated RGB and depth samples to be used in the following iterative run. The next iteration, however, is initialized only, if the evaluated error still exceeds the given *errorThreshold*.

Final Interpolation With the iteration being terminated, the parabolic RGB and depth texture samples being evaluated last, are used for the final interpolation of color values and per-fragment depth evaluation. The final fragment color is interpolated based on the camera interpolation weights being factored by their inverse error values to reduce the impact of sample cameras which observe incoherent object surface points. The fragment's depth is determined by projecting the last known viewing ray sample position to the virtual view's viewing space by applying the virtual view's modelview-projection matrix.

4.2.3 Rendering Quality - Iterative Refinement

Although the proposed iterative refinement 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 4.6. 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. Case C shows a situation where the viewing ray intersects the object at an abrupt and small, but elongated geometric elevation while no intersection can be asserted for any of the adjacent cameras. These cases cannot be handled exactly and will inevitably result in visible ghosting artifacts.

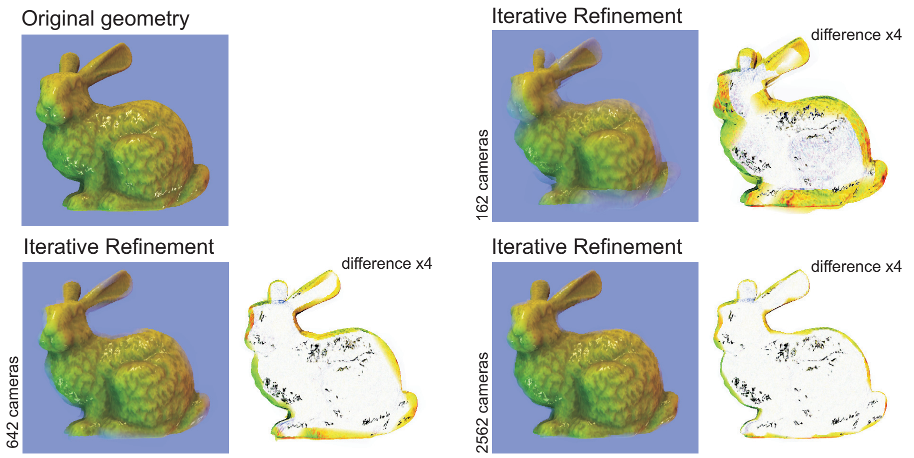


Figure 4.7: Results of the iterative refinement rendering technique at different camera sphere resolutions. Top left shows the original polygonal rendering of the Stanford Bunny model. Top right and bottom row display the iterative rendering results supplemented by a difference image showing the pixel difference from the original geometry multiplied by a factor 4 and inverted.

In order to attenuate such artifacts, the camera estimates $P^{(i)}_{\mathrm{cam},k}$ are examined after a few iterations. If the three estimates are still highly divergent, the point with the closest distance to the camera is discarded. It is then proceeded with the residual two cameras. If no similar local estimates can be achieved within the next few iterations, the averaging in Equation 4.4 is replaced by a maximum operation. This procedure, however, cannot completely eliminate the visual artifacts, especially case C which will inevitably result in ghosting artifacts. If the ghosting is still too strong the only effective measure is increasing the number of cameras.

To evaluate the quality of the iterative refinement approach the well-known *Stanford Bunny* [58] has been chosen as reference object. 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 [35, 61].

The rendering results shown in Figure 4.7 clearly display the effect of increased numbers of sample cameras on ghosting artifacts. It is easily seen that good results are achieved with spherical approximations yielding 162 or above

Images	MAE	MSE	RMSE
162	0,0413227	0,0151801	0,123207
642	0,0253391	0,0824355	0,090794
2562	0,0145189	0,00429155	0,0655099

Table 4.1: Per-pixel color differences of the synthesized image compared to the original rendering. Mean absolute error, mean squared error, and root mean squared error are displayed for light field renderings being synthesized from a varying amount of input samples.

sample positions. ghosting artifacts vanish with the rising amount of sample positions. This effect is best seen at the silhouette edges of the rendered object. Ghosting at the silhouette edges results from inconsistent rays being used for interpolation. The situation at the silhouette edges corresponds to the situation shown as Case B in Figure 4.6. The difference images provided with the rendering results back up this claim. Major errors by means of distance in the RGB color space and the area of false pixel regions are displayed at the silhouettes. With increasing sample camera count, the false pixel regions are minimized.

The visual appearance of concavities and small-to-large depth variations are reconstructed at good quality. The structured surface of the original 3D rendering is clearly resampled in the light field rendering. Concavities as they occur in the transition zone between the bunny's body and his hind legs are reconstructed precisely. The difference image does not show significant pixel differences in these regions. Local differences, however, are observed at the specular highlights which occur in regions of great curvature all over the object's surface. Even with an increased amount of sample cameras this highlight error cannot be reduced significantly.

The observation is approved by a statistical analysis of per-pixel RGB differences. Table 4.1 shows the mean absolute error (MAE), the mean squared error (MSE), and the root mean squared error (RMSE) for the spherical light field representation being displayed in Figure 4.7. From the table it can be seen clearly that all of the error measures significantly drop with rising amount of sample positions. While the RMSE is reduced by about 25% with an increase of sample positions from 162 to 642, the RMSE is halved by using 2562 sample positions instead of 162 samples.

While the sample camera arrangement is chosen to be dense for the iterative rendering approach, far less samples are required compared to spherical rendering approaches presented in the past (see Section 2.5.3) to achieve com-

Images	Input Res.	Target Res. 256 × 256	Target Res. 512 × 512	Target Res. 1024 × 1024
42	256 × 256	155.43 fps	110.4 fps	63.44 fps
42	512 × 512	154.42 fps	107.2 fps	59.65 fps
162	256 × 256	140.21 fps	96.9 fps	62.76 fps
162	512 × 512	130.12 fps	95.6 fps	52.88 fps
642	256 × 256	111.95 fps	78.7 fps	47.74 fps
642	512 × 512	108.33 fps	75.3 fps	46.76 fps
2562	256 × 256	91.59 fps	63.5 fps	39.44 fps
2562	512 × 512	84.65 fps	61.2 fps	37.86 fps

Table 4.2: Rendering performance for the iterative rendering algorithm applied to light fields of varying resolution.

parable rendering quality. Thus, light fields can be reconstructed from more efficient light field representation with respect to storage costs.

4.2.4 Rendering Performance - Iterative Refinement

The rendering performance of the iterative approach allows for virtual view synthesis at real-time frame rates, even for densely sampled light fields including 2562 sample positions (see Table 4.2). With the performance being measured 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, frame rates of up to 155 fps are achieved using the interative refinement approach. For performance measurements the render settings were chosen such that a quality according to the results depicted in Figure 4.7 is achieved (error threshold = 0.1, maximum iterations = 256). For performance evaluation, the spherical proxy being rendered for virtual view synthesis has been rescaled to fit into the target resolution. Thus, 78.53 % of the target view are covered by the spherical proxy. All measurements have been performed using uncompressed light field data.

Table 4.2 shows the rendering performance for a variety of input sample image resolutions captured from spherical sampling setups of varying density. While the resolution of the input textures does not influence the rendering performance significantly, the target rendering resolution does effect the performance to a high degree. With the light field being reconstructed based on pixel wise interpolation, the relation of the target resolution to the rendering performance is obvious. A performance drop of about 35 % in average can be

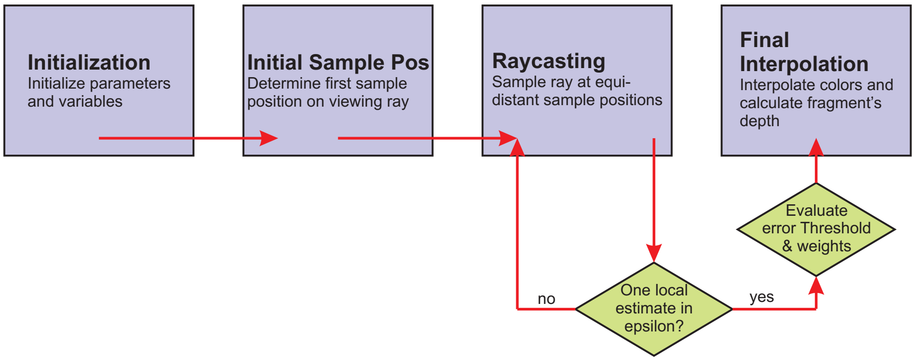

Initialization
Initialize parameters and variables
Initial Sample Pos
Determine first sample position on viewing ray
Raycasting
Sample ray at equi-distant sample positions
Final Interpolation
Interpolate colors and calculate fragment's depth
Evaluate error Threshold & weights
One local estimate in epsilon?
no
yes

Figure 4.8: The raycasting approach is implemented in four phases in order to interpolate color information from coherent rays.

observed with each quadruplication of the target resolution.

4.3 Raycasting Approach

The raycasting approach is implemented in a customized fragment program in four phases according to Figure 4.8. The initialization phase and the evaluation of the viewing ray's first sample provide the starting point for the raycasting process and the initial blending weights for the interpolation. Within the raycasting process the ray-object intersection is established by successively testing for object intersections along the ray being observed from adjacent sample cameras at a ray position. If, for a certain ray position, any of the local estimates being determined based on the adjacent sample's depth information is positioned within a pre-defined epsilon environment, the raycasting is stopped. For those cameras which do not observe an intersection for the last ray sample position, the interpolation weight is reset to 0. Afterwards interpolation weights are normalized and successively used for the final interpolation to determine the final fragment color.

4.3.1 Raycasting Process

When a triangle is rasterized, the fragment program calculates the viewing ray, which corresponds to the fragment. Then the intersection point of the viewing ray with the bounding sphere of the object is calculated according to the ray-sphere intersection performed within the iterative approach (see Section 4.2).

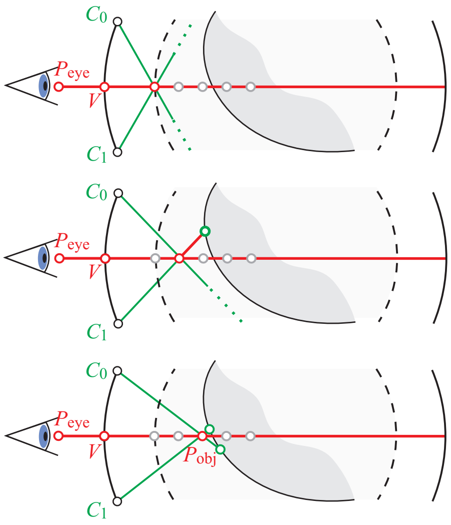


Figure 4.9: The raycasting approach samples the viewing ray stepwise at adjacent positions starting at the intersection point with the object's bounding sphere.

For the raycasting approach, however, the intersection of the viewing ray with the hemisphere facing towards the virtual viewpoint is established as a starting point (see Figure 4.9).

The inner sphere defines the object's bounding sphere and thus limits the region for which valid object surface points can be evaluated. From the ray intersection with the inner object bounding sphere the viewing ray is sampled at a fixed step size, as shown in Figure 4.9. At each step, the assumption is validated that the current ray position is the actual object intersection point (P_{obj}). This point is projected onto the camera sphere using the adjacent camera positions C_k as center of projection according to Equation 4.1. The resulting three intersection points with the sphere are transformed into the viewing coordinate system of the corresponding adjacent camera and are converted to parabolic coordinates (see Equation 4.2). 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 4.3. These points are then compared to the current ray sample position. If one of the local estimates is equal to the ray position within a given epsilon environment, the ray sampling is immediately stopped. This means that at least one camera is found which reliably observes an object intersection at exactly the ray position.

If the remaining two cameras also observe a ray intersection within the

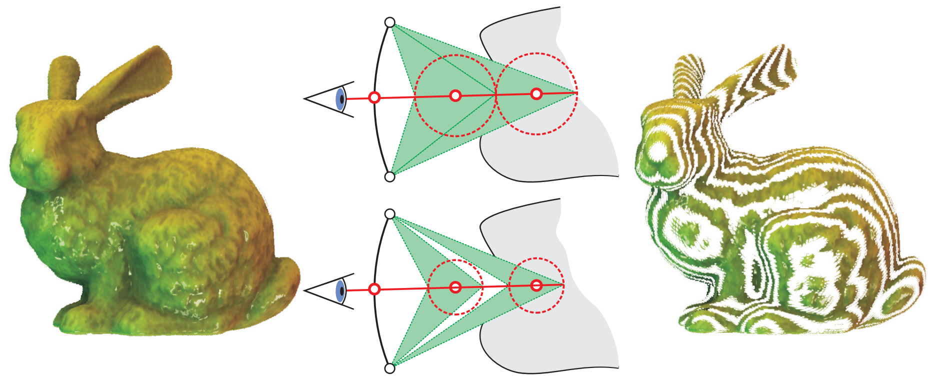


Figure 4.10: Left: Light field rendering of the Stanford Bunny from 162 (512×512) input samples. Middle, top row: Showing recoverable area for epsilon chosen to be equal to half of the step size. Middle bottom row: Epsilon smaller than half of the step size results in unrecoverable areas. Right: Unrecoverable areas appear as gaps in the light field rendering if epsilon is chosen too small with respect to the step size.

epsilon environment, it is assumed that all cameras observe the same point. Then the barycentric weights of the fragment are used for interpolation in order to determine the fragment's final color.

However, if the intersection point for one camera is far away from the ray position, it is assumed that one of the situations outlined in Figure 4.6 occurred. In this case the color information for the respective camera is discarded by setting the corresponding barycentric weight to zero. Afterwards, the interpolation weights are normalized again and the final color is calculated as the weighted sum. In a last step the established ray-object intersection point is being transformed to the virtual view's viewing space to extract per-pixel depth values according to the modelview-projection matrix being set for the virtual view. A light field being rendered from 162 sample positions at an image resolution of 512×512 is shown in Figure 4.10, left.

Note that while the size of the epsilon environment can be chosen freely, it should be chosen according the raycaster's stepsize. Thus, if the stepsize is minimized to exploit more precise object intersections, the epsilon environment should be adjusted accordingly to support the desired precision. In the deviant case, for the stepsize to be chosen larger to increase rendering performance, the epsilon has to be chosen accordingly larger. Choosing the epsilon to be less than half of the step size will result in visual discontinuities as local estimates will not be reconstructible 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 4.10).

4.3.2 Implementation Details - Raycasting

The four phases of the raycasting approach being implemented for light field synthesis within the customized fragment program are discussed in detail in this section (see Figure 4.8).

Initialization The first phase of the raycasting approach implements the initialization of the current fragment position and the determination of the viewing direction according to the initialization performed within the iterative refinement rendering technique presented above (see Section 4.2.2). The raycasting is executed in the scope of a for loop which is initiated depending on the defined stepsize. Due to shader model 3.0 limitations, a maximum loop count may not be exceeded. Thus, up to two nested *for loops* are implemented to overcome this limitations. With the nested loop approach up to 200×200 sampling steps can be implemented (see Appendix A.2 for the complete fragment shader code).

Initial Sample Position The initial sample position can either be chosen freely on the user side by adjusting a *skip* parameter or the first intersection point of the viewing ray with the inner sphere, the object's bounding sphere, is chosen as starting point. While the intersection with the inner sphere defines the first reasonable sample position along the viewing ray, the user may want to adjust this starting point to optimize rendering performance by limiting the sample range along the viewing ray.

A simple ray-sphere intersection is implemented within the fragment shader to determine the sampling starting point. Note that the intersection is calculated for the inner sphere with radius of $r = 1\ \sqrt{2}$. The calculation of the intersection is based on the ray-sphere intersection approach implemented within the iterative refinement fragment program. For the initial sample position, however, the intersection with the hemisphere facing towards the virtual viewpoint is relevant (see Code Sample 4.2).

Raycasting With the initial sampling position as starting point, the raycasting is implemented in three main steps. After stepping forward along the

```
//first position on ray along view dir for first pos.
//skip some samples if skip parameter is set
float3 firstSamplePos = v + skip * dirN;

//determine intersection with inner sphere for initial skip
//if no custom skip is given
float skip;
if(!skip>0.0)
{
  float A,B,C;
  //inner sphere is of size 1.0/sqrt(2)
  C  = dot(P_eye,P_eye) - (1.0/sqrt(2));
  B  = 2.0 * dot(P_eye,dir);
  A = dot(dir,dir);

  float S = max( (B*B - 4.0 * A*C) ,0.0); //components under sqrt
  //first intersection with inner sphere neede, so take min
  skip = min( ((-B + sqrt(S)) /A/2.0), ((-B + sqrt(S)) /A/2.0));
  //take inner sphere intersection as start point
  firstSamplePos = P_eye + skip * dirN;
}
```

Code Sample 4.2: The initial sampling position along the viewing ray is derived from the ray-sphere intersection with the object's bounding sphere if no custom *skip* parameter is given.

viewing ray a given *stepsize* in the first step, texture samples are fetched for the three adjacent sample cameras based on the new sample position to establish local estimates in the second step. In the third step the local estimates are checked against the *epsilon* environment constraint also given as adjustable shader parameter. This *epsilon* test is foundation for the final decision whether to proceed or abort raycasting. In case of raycasting being proceeded the next sample position is initialized and the raycasting procedure is restarted. This is the case if no intersection can be reliably established for any of the adjacent sample positions.

1. *Update Sample Position:* To update the sample position the vector to the last known sample position is simply elongated by adding a *stepsize* part of the normalized viewing direction. The new sample position is constrained by the inner sphere. As the inner sphere limits the defined region of valid object intersection points, all sample positions outside the inner sphere do not contribute to the final image synthesis. Thus, in case of the updated sample position being outside the inner sphere, the current fragment is discarded, as no information is available for this pixel.

2. *Determine Local Estimates:* Local estimates are calculated for each of the adjacent sample cameras from the parabolic depth samples being

extracted based on the ray-sphere intersection of the sample camera rays through the current sample position. Ray-sphere intersection as well as the parabolic texture sampling and the local estimate determination are implemented similar to the iterative procedure (see Section 4.2.2).

3. *Evaluate Epsilon and Error Threshold:* For the three local estimates the absolute distances to the current viewing ray sample position are calculated. In case of either one of the local estimates being closer to the sample position than defined by the *epsilon* parameter, the raycasting is immediately stopped. This means, that at least on camera is observing an object surface point close to the current sample position. The sample cameras observing an intersection within the given threshold will be accounted for in the final interpolation phase. In the deviant case their interpolation weight ($cameraWeight$) is reset to 0. After all, the overall interpolation weights are normalized.

Final Interpolation With the interpolation weights of the sample cameras being adjusted according to the position of their local estimates with respect to the last sample position, the fragment's final color is interpolated similar to the iterative refinement approach. Sample cameras which did not observe an intersection for the last sample position do not contribute to the final result. The fragment's depth is then determined in a last step by applying the modelview-projection matrix to the last viewing ray sample position.

4.3.3 Rendering Quality - Raycasting

The raycasting rendering approach implements an efficient light field rendering technique that applies a precise depth correction of rays to achieve high quality rendering results. While being less efficient by means of rendering performance compared to the iterative refinement method, superior rendering quality is achieved. Silhouette edges and small geometric details are synthesized with a high precision at rendering frame rates of up to 109 fps.

In contrast to raycasting implementations being presented for rendering of displacement maps [125, 126] and relief textures [3, 81, 85], the raycasting approach presented here benefits from multiple input depth maps to handle ambiguous situations as outlined in Figure 4.6. While these related raycasting approaches cannot handle self-occlusion and large variations in depth [6], the raycasting approach presented in this thesis precisely reconstructs the visual appearance from small structures and large geometric variations. Concavities and holes can be reconstructed reliably as depth information can be retrieved

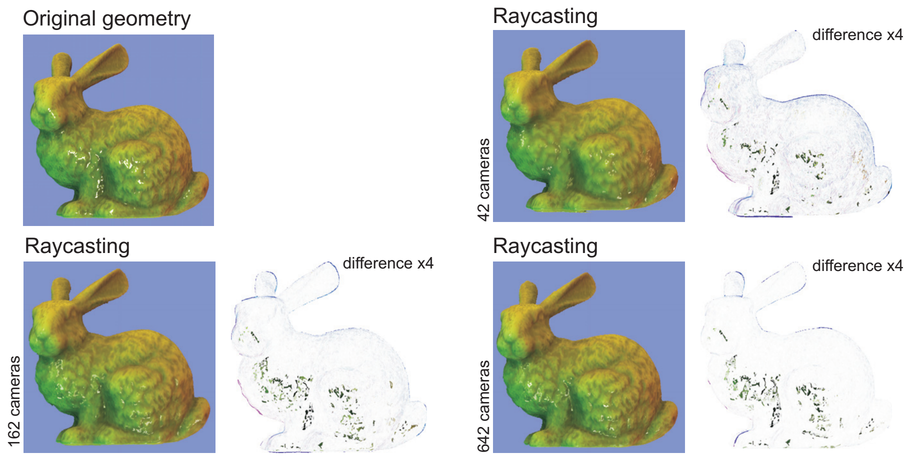


Figure 4.11: Results of the raycasting approach at different camera sphere resolutions. Top left shows the original polygonal rendering of the Stanford Bunny model. Top right and bottom row display the raycasting synthesis results supplemented by a difference image showing the pixel difference from the original geometry multiplied by a factor 4 and inverted.

from three adjacent depth maps to determine the object intersection. Silhouettes and sharp edges are precisely reconstructed (See Figure 4.11).

For comparison reasons the same reference object of the *Stanford Bunny* also used to evaluate the quality of the iterative techniques was chosen to demonstrate the quality of the raycasting approach. Figure 4.11 clearly shows the benefit of raycasting in terms of image quality compared to iterative refinement shown in Figure 4.7. The micro and meso structures of the polygonal *Stanford Bunny* model are reconstructed from only 42 input samples as depicted in the difference image in Figure 4.11. The difference images demonstrate the quality at the silhouette edges, small structures, and concavities. Good synthesis quality is achieved from a fractional amount of input samples being used for iterative refinement light field rendering and related spherical light field techniques presented in the past.

Note that the region of false pixels at the silhouette edges is diminished dramatically compared to the silhouette errors which are appearing with the iterative technique being applied. Silhouettes are reconstructed precisely. Starting from 162 sample positions and above micro and meso structures are precisely reconstructed. Highlights, however, cannot be reconstructed at the same high quality. For specular highlight features the raycasting approach does not show

Images	MAE	MSE	RMSE
42	0,0181901	0,00530386	0,0728207
162	0,0165208	0,00470699	0,0686075
642	0,014703512	0,004048011	0,063623984

Table 4.3: Per-pixel color differences of the light field rendering based on varying sample densities compared to the original rendering, showing the mean absolute error, the mean squared error, and the root mean squared error.

significantly improved rendering quality compared to the iterative approach. Thus, specular highlights exhibit the most visible differences between the original polygonal based rendering and the light field rendering.

Compared to the iterative approach per-pixel differences are significantly smaller. The root mean square error of the raycasting approach is reduced by 40% in comparison to the iterative approach for a light field being sampled from 162 sample positions. However, while for the iterative approach the RMSE could be lowered by a factor of 25% by increasing the sample amount from 162 to 642, the raycasting's RMSE is minimized by a factor of about 7% only. In total the error analysis results approve the superior quality of the raycasting approach (see Table 4.3).

4.3.4 Rendering Performance - Raycasting

The improved rendering quality of the raycasting approach comes at the cost of reduced rendering performance. With the raycasting being applied, the overall rendering performance is reduced by about 30% compared to the iterative approach. But sill, real-time frame rates are achieved for target resolutions of up to 1024×1024 as shown in Table 4.4. For the performance measurements the hardware and rendering setup was chosen to be identical to the iterative refinement performance evaluation. To achieve a rendering quality according to the results depicted in 4.11 the setting have been setup such that the step size was set to 0.01 and the epsilon environment was limited to 0.01. From the table a performance drop of about 38% can be observed with each quadruplication of the target image resolution. Thus, for the raycasting approach the rendering performance is slightly more effected by the target resolution than the iterative approach. The additional drop in performance compared to the iterative approach is about 3%. Again, the influence of the input sample image resolution is negligible in the global view. The amount of sample positions, however, does influence the performance to a high degree. For a target resolution of 512×512

Images	Input Res.	Target Res. 256 × 256	Target Res. 512 × 512	Target Res. 1024 × 1024
42	256 × 256	109.79 fps	79.1 fps	49.71 fps
42	512 × 512	106.23 fps	78.3 fps	44.38 fps
162	256 × 256	81.68 fps	60.4 fps	37.43 fps
162	512 × 512	80.38 fps	57.8 fps	36.41 fps
642	256 × 256	75.41 fps	52.1 fps	29.79 fps
642	512 × 512	73.63 fps	52.2 fps	28.75 fps
2562	256 × 256	63.84 fps	46.1 fps	27.53 fps
2562	512 × 512	62.29 fps	45.8 fps	26.20 fps

Table 4.4: Rendering performance for our raycasting rendering algorithm and the iterative refinement approach applied to light fields of varying resolution with and without texture compression.

as a representative set of measurements, performance is decreased by a factor of 0.58 from 79.1 fps to 45.8 fps with the amount of sample positions being increased by a factor 61 of from 42 to 2562 sample positions.

While the measurements have been performed using a high-end hardware setup, the raycasting algorithms also allows to perform efficiently on other hardware platforms. With the stepsize and epsilon parameters the user can adjust the rendering performance according to his hardware capabilities. However, rendering speed then comes at the price of reduced rendering quality if the parameters are chosen to be too large. In this case discontinuity artifacts are likely to appear. Small geometric details are not recoverable with the step sizes chosen too large.

4.4 Level of Detail for Light Field Rendering

The flexibility of the presented light field rendering techniques opens up for optimization strategies which allow the adjustment of rendering quality to increase rendering performance. By adjusting the rendering quality depending on the object's relevance to the rendered scene, optimized performance can be achieved. With the level of detail (LOD) technique presented in this section, objects in focus are rendered at highest quality whereas distant objects are rendered at lower resolutions. The hierarchical arrangement of sample positions within the spherical approximation resulting from the subdivision of the initial icosahedron (see Section 3.3) makes the implementation of a discrete

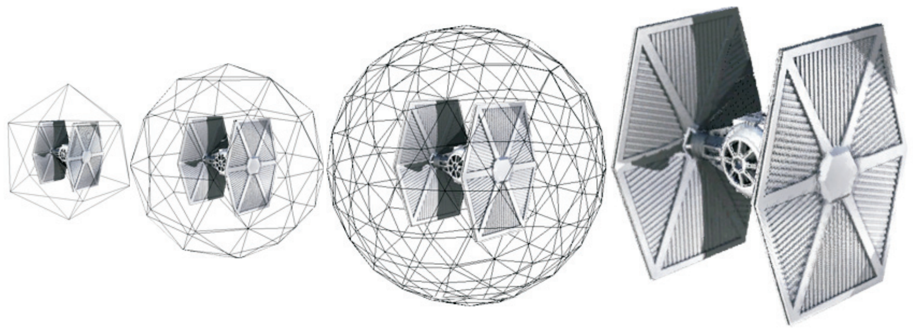


Figure 4.12: From left to right: Light field renditions of a Tie-Fighter light field (162 samples at 512×512) rendered at varying distances and LODs with the spherical proxy geometry being superimposed. The rightmost image displays the light field rendering at maximum rendering quality.

LOD strategy available for light field rendering. 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 4.1). Light fields being rendered with a coarser LOD are reconstructed from fewer light field probes which is in turn reducing the amount of texture switches in the rendering process and thus reducing the GPU workload. The spherical sample positions are arranged such that all of the vertices of a certain LOD are contained in the next finer LOD (see Figure 4.12). 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 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. The LOD to be rendered is determined in classical sense as a tradeoff of resolution vs. geometric quality. A maximum of 5 LODs is available assuming a highest resolution of the spherical approximation with 2562 sample positions for the most detailed representation.

Rendering performance is further adapted to the object distance by adjusting the render settings for the raycasting algorithm to steer the reconstruction quality. As presented in Section 4.3.2 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. Note, however, that the epsilon parameter is to be adjusted according to the chosen stepsize.

The combination of both LOD rendering approaches results in a gain of rendering performance and rendering flexibility which allows light fields to be efficiently displayed. Thus, it provides an ideal solution for rendering multiple instances of light fields simultaneously in a dynamic scene or integrating dynamic high-quality light field renderings in complex polygonal based scenes. This technique provides a powerful tool for the flexible integration of light field renderings in performance critical real-time applications such as games or interactive storytelling and educational presentations.

4.5 Progressive Light Field Rendering

Light field rendering techniques present an ideal solution to provide access to complex data sets. Their usefulness for online-access through web-galleries, however, is limited. The spherical light field parametrization presented in Chapter 3 provides a flexible representation that allows for efficient remote access and client-server based rendering methods of complex light field data.

In this Section an adaption of the spherical light field raycasting approach is presented which is optimized for web based exploration. To provide light field data as a web gallery, the exisiting approach would require a complete light field data set to be downloaded before the first image may be generated. Thus, immediate visual feedback is not available. The user is forced to wait for the complete data set to be downloaded and stored in client memory, even if he might be interested in a certain view upon the data only. High quality light fields also come with significant memory requirements for storing the necessary texture images on the client's local memory.

To these ends the client-server based light field rendering technique presented in this section is well scalable to both the available network bandwidth and the client hardware. This approach allows light field data to be transferred progressively on demand with respect to the interest of the client. The rendering client may automatically adapt the visual quality of the representation to the amount of host memory, and the desired run-time performance. Data that is not required to generate an individual view will not be transferred. Thus, while drastically reducing data transmission, a high-quality light field rendering technique is provided that makes interactive exploration of complex data available at real-time frame rates with 6 DOF. With the proposed rendering technique, visual representations of arbitrary complex data sets may be presented in web galleries.

The idea behind the progressive light field rendering technique is to replace the static tessellation of the camera sphere (see Section 3.1) by a progressive refinement at the client side. This will allow the client to request the images required to synthesize a particular view on demand from the server. The hierarchical nature of subdivision also allows to prioritize the requests with respect to the image quality and available client memory.

Progressive Refinement The base geometry for the local refinement is the icosahedron consisting of 12 camera positions. The cartesian coordinates of the camera positions $\mathbf{c}_i$ are

$$\mathbf{c}_i \quad \in \quad \left\{ \begin{pmatrix} 0 \\ \pm 1 \\ \pm\varphi \end{pmatrix}, \begin{pmatrix} \pm 1 \\ \pm\varphi \\ 0 \end{pmatrix}, \begin{pmatrix} \pm\varphi \\ 0 \\ \pm 1 \end{pmatrix} \right\} \quad \text{with} \quad \varphi = \frac{1+\sqrt{5}}{2} \qquad (4.16)$$

With images from these 12 camera positions which are uniformly distributed on a sphere around the object, the light field rendering approach allows for synthesizing virtual views from an arbitrary viewpoint at a relatively low resolution. In order to synthesize an image from one particular view point $\hat{\mathbf{c}}$, however, only a few of these 12 images are required. If the virtual viewpoint $\hat{\mathbf{c}}$ is known in the sphere's coordinate system, the cameras $\mathbf{c}_i$ needed to be resident in memory for this particular viewpoint can be easily determined by calculating the dot products:

$$\text{if} \quad (\hat{\mathbf{c}} \cdot \mathbf{c}_i) \quad \begin{cases} > 0 & \text{then} \quad \mathbf{c}_i \text{ is required} \\ \leq 0 & \text{then} \quad \mathbf{c}_i \text{ is } \textbf{not} \text{ required} \end{cases} \qquad (4.17)$$

For a web-based application, these are the camera images which must be requested from the server. Whenever the user changes the virtual view point $\hat{\mathbf{c}}$, new images must be requested (or obtained from a network cache), and previously transferred images may become obsolete. Figure 4.13 shows examples taken from the first few steps of progressive subdivision for one particular viewing position.

Due to holes and concavities in the object, however, there may be parts of the object that cannot be seen from any of the 12 camera images which will result in visual artifacts. In order to improve the image quality, more than 12 cameras are required. Additional camera positions are inserted by calculating

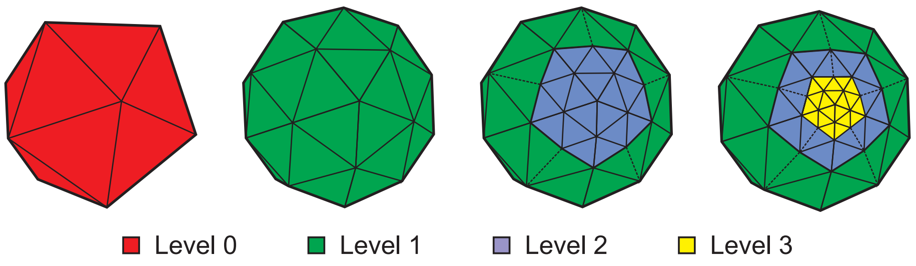


Figure 4.13: The first few steps of the progressive refinement of the camera sphere. Starting with the icosahedron at level 0, hierarchical subdivision is performed view-dependent.

the midpoint of each edge of the sphere tessellation and lift the resulting vertex back to the sphere.

The sphere tessellation must then be adapted to accommodate the newly inserted camera vertex. This way a hierarchical tessellation of the sphere is built. In order to prevent holes in the camera sphere which may result from neighboring triangles of different subdivision level, two simple rules are followed:

1. A triangle may only be further subdivided if all of its neighbors have at least the same subdivision level. This means that the refinement of two neighboring triangles may not differ by more than one level.

2. If two neighboring triangles have different subdivision level, the one with the lower level will be subdivided temporarily to prevent T-vertices from causing holes in the tessellation. Temporary triangles will not be further subdivided.

Figure 4.14 illustrates these rules: The first rule says that the green triangles adjacent to the yellow one in Figure B may not be further refined before the yellow one is subdivided. The temporary tessellation by the second rule is shown by the dashed lines in Figure B and C. If all neighboring triangles are subdivided as in Figure D, the temporary tessellation is replaced by a regular one. Following these rules, vertices can be easily added and removed from the camera sphere while still maintaining a valid view-dependent tessellation without any holes.

The priority of a camera request is calculated from its position $\mathbf{c}_i$ and the refinement level s_i it belongs to, using a heuristic:

$$p(\mathbf{c}_i, s_i) = \frac{1}{s_i^k} \left(\hat{\mathbf{c}} \cdot \mathbf{c}_i\right) \tag{4.18}$$

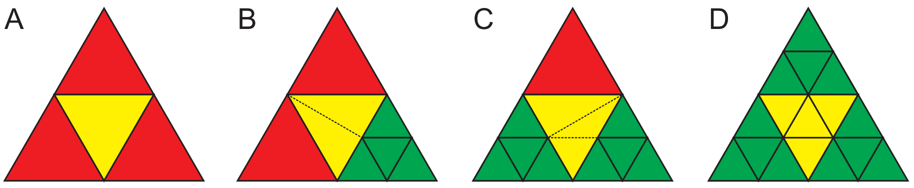


Figure 4.14: Tessellation of the yellow triangle in dependance on the subdivision level of its neighbors. If not all neighboring base triangles are subdivided, a temporary triangulation is used as indicated by the dashed lines. This will prevent holes caused by T-vertices.

The level exponent $k > 0$ is used to balance the progressive subdivision process. A larger value will favor breadth-first, a smaller value depth-first traversal. In practice a level exponent of $k = 0.7$ has been proven to be well suited for most web based light field renderings.

Client - Server Communication The view-dependent tessellation of the camera sphere is performed on the client side. For every newly inserted camera position, the client sends a request for image data to the server. The server is responsible for sending the required image and depth information to the client. A customized network protocol based on TCP has been implemented for this communication. Image requests are identified by a unique hash key which is calculated from the light field id and the requested camera position. The light field server maintains a hash table which refers to the hard disc location of the image data on the server. The communication protocol is supplemented by initialization and security-related messages. The server may simultaneously maintain connections to different clients and different requests from a single client. While the rendering performance on the client side is not effected by the server's response, the request and update rate is effected by the server performance and the network bandwidth.

The client server communication was tested in both LAN and WAN networks. As a rule of thumb the complete network transfer of a light field will be equivalent to transferring the same number of still images in *PNG* format. This holds true if the client caches all images on the local hard disc. The image-based remote rendering framework, however, will also work without a disc cache. In this case, images which become obsolete due to a viewpoint change will be discarded from memory and newly requested from the server if necessary. This allows for efficiently balancing the local memory consumption with the speed of interactivity and visual feedback. Figure 4.15 shows an ex-

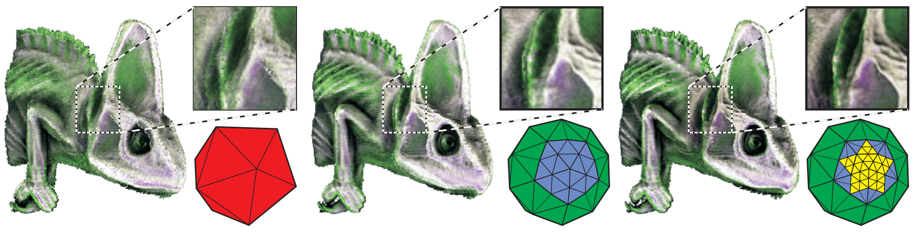


Figure 4.15: Progressive transmission of a light field (UTCT Veiled Chameleon). Images generated from 12, 36 and 63 images resident in client memory (left to right).

ample of the image quality during progressive transmission of the light fight field.

4.6 Conclusion

The tradeoff of rendering speed and synthesis quality has been identified to be the key factor to the overall performance of a light field rendering approach (see Section 1.2). The spherical light field rendering approaches with per-pixel depth correction of rays presented in this chapter provide a comprehensive rendering tool set which provides a flexible and powerful solution to satisfy this challenge. With the geometry assisted per-pixel ray interpolation best rendering quality is achieved without noticeable ghosting artifacts at real-time frame rates. Object silhouettes are reconstructed precisely and visual features from meso and micro structures as well as concavities are reliably reconstructed. The flexible implementation strategy provides an adjustable rendering approach by means of performance and quality to adapt to changing requirements.

With the iterative rendering (see Section4.2) and the raycasting approach (see Section 4.3) two different rendering techniques have been presented. Both of these approaches apply depth correction of rays based on the input data directly. Per-pixel depth information is extracted from the alpha component of the input samples directly to establish ray-object intersections precisely without the need for complex mesh processing to be applied at runtime.

While the raycasting approach provides high-quality rendering results at medium real-time frame rates based on sparsely sampled light fields, the iterative refinement approach presented in this chapter allows for medium rendering quality at high real-time frame rates from densely sampled light fields. Thus, if memory consumption is critical and high-quality rendering results are essential, the raycasting approach provides the optimal solution. If, however, memory consumption is not an issue and slightly ghosting artifacts at the silhouette edges are tolerable but rendering performance is a critical issues, the iterative refinement approach provides an alternative light field rendering solution. Note, however, that only a fractional amount of input samples is mandatory, compared to spherical light field techniques presented in the past (see Section2.5.3), to provide sophisticated rendering results using either one of these two approaches.

Compared to other image based rendering methods based on texture based geometric representations, such as displacement maps [125, 126] and relief textures [3, 81, 85], the raycasting approach provides improved ray-object intersections and thus advanced rendering quality. These approaches are not capable of reconstructing complex view-dependent reflectance and geometric details resulting from self-occlusion, concavities and micro structures [6, 81]. In comparison, the raycasting approach 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.

With the LOD rendering techniques a powerful tool has been presented which allows to adjust the rendering performance and quality according to the user's preferences and dynamic scene changes. The proposed LOD technique is especially suitable for complex scenes being rendered based on multiple light fields or light fields being rendered within the context of complex dynamic scenes on a local machine.

With the progressive refinement rendering approach an alternative rendering technique has been presented in this chapter that allows for rendering of light fields that are not present on a user's local machine. The progressive technique implements a web based rendering approach that provides remote access to light field renditions stored on a remote server. Thus, this approach implements an ideal solution for making light field data available within web based galleries to provide access to sophisticated visualization results from a variety of sources. Such sources include the visualization of simulation data from earth sciences or medicine.

The rendering techniques presented in this chapter have been published by the author to the computer graphics community in [112, 115–117].

5 Acquisition of Spherical Light Fields with Per-Pixel Depth

The spherical light field parametrization presented in Chapter 3 implements a uniform sampling structure which allows high-quality image synthesis at real-time frame rates based on combined RGB and depth input samples which are encoded in RGBz parabolic texture maps. The spherical parametrization allows for light field reconstruction without the need for further geometry processing, neither as a pre-processing step nor within the rendering process. To fully exploit the advantages of this light field representation within the acquisition task an acquisition technique should synchronously provide RGB and depth sampling capabilities in order to integrate combined RGBz samples into a light field representation. Optimally, the acquisition task makes use of state-of-the-art acquisition devices which provide combined RGBz images directly and at high-update rates without costly processing. To further satisfy the demands on acquisition techniques for precision, usability and availability (see Section 1.2) the light field sampling approach should provide precise positioning abilities and make use of commonly available and affordable hardware devices to make a time and cost effective light field acquisition available. Acquisition approaches should ideally assist the user in the sampling task to maximize usability. Usability is optimized by providing immediate visual feedback to provide a tool for quality evaluation within the acquisition task and by guiding the user within the sampling device positioning process.

Unfortunately, light field acquisition techniques presented in the past do not satisfy these requisites. While with the *Light Field Camera* [31] and the *Plenoptic Camera* [78] two flexible solutions for a hand-held acquisition of two-plane light fields have been presented lately, these devices are limited to the acquisition of two-plane light field parameterizations and do not provide per-pixel depth information directly. Thus, these device do not allow for the acquisition of spherical light fields. The limitation to two-plane light field representations does not exist for acquisition setups such as the Stanford Spherical Gantry [60] which are capable of moving an imaging device to any pre-defined sample position around an object of interest. However, the usability of such a device is strictly limited by the financial expense, the limited portability and the missing capability of providing geometric information directly. Acquisition techniques which relief the user of using complex positioning setups sample a light field from arbitrary sample positions using off-the-shelf imaging devices. Though, these approaches either rely on an unstructured light field parameterizations which exhibit known challenges from their non uniform nature (see Section 2.5.4) or do postulate a prior knowledge of the scene geometry in order to re-map the captured sample to predefined sample positions. As a matter of fact, these approaches are not applicable efficiently for the acquisition of spherical light fields with per-pixel depth.

Whether per-pixel depth information is extractable from complete 4D light fields in a post-processing step [21], a costly subsequent process is needed to extract such information. Thus, per-pixel depth information is not available synchronously within the acquisition phase. As a consequence these approaches are not applicable to feed combined RGBz samples to the light field representation directly. Immediate visual feedback is not available for incomplete 4D light fields with per-pixel depth being extracted in a post-processing step.

In this chapter a flexible light field acquisition system is presented which provides synchronized and registered per-pixel depth and color information for each sample at real-time. The system implements an acquisition prototype which makes use of state-of-the-art imaging and upcoming 3D sensor technology to acquire combined RGBz samples which are processed directly to allow for immediate visual feedback within the acquisition process. The prototype demonstrates an efficient and flexible solution towards real-time acquistion of light fields which abet the benefits of spherical light fields with per-pixel depth using future 2D/3D imaging sensors. Precise positioning techniques are utilized to facilitate the acquisition of combined RGBz samples from pre-defined spherical positions according to the proposed spherical light field parametrization. For user guidance immediate visual feedback is granted for quality evaluation within the acquisition process and a positioning guidance system is implemented within the acquisition pipeline to ease the acquisition process.

For artificial scenes being generated from digital content using computer graphics techniques, a light field generation process is presented in this chapter which allows to easily generate spherical light fields from arbitrary complex synthetic scenes. The presented light generation easily adapts to various rendering engines and provides a sophisticated technique to extract light field representations from scenes involving complex rendering techniques which are hard to render in real-time using traditional rendering approaches.

5.1 Light Field Acquisition from Physical Objects

To take advantage of the benefits provided by the spherical parametrization with per-pixel depth, presented in Chapter 3, a light field sampling device is to be utilized which is capable of providing registered depth and color information on a per-pixel basis in real-time. For each light field sample to be acquired the device must be placed such that its position and direction corresponds to the spherical parametrization given at a certain granularity. To achieve a precise and flexible positioning of the device, an accurate and easy to use location system is to be used.

The acquisition device and the location system of the acquisition process presented in this Chapter are explained in detail in this section followed by an in depth description of the processing pipeline being implemented for a flexible light field acquisition in order to acquire complete spherical light fields from physical objects.

5.1.1 Light Field Acquisition Device

For an acquisition device to be suitable for interactive acquisition of spherical light fields it must provide reliable depth information at real-time frame rates on a a per-pixel basis. The depth is to determined with per-pixel time consistence such that the complete image represents a static snapshot of a dynamic scene. Per-pixel depth must be synchronously available and registered with captured color information.

3D Acquisition Technologies There are some technologies and algorithms available for 3D scanning which may be adapted for real-time acquisition of per-pixel depth information. These methods may be divided into active and passive methods. While real-time passive methods based on stereo [27] or silhouettes [69, 70] have been proposed in the past, these approaches typically do not perform well in the absence of scene texture and exhibit computational costly processing. Thus geometry reconstructions based on such passive methods tend to provide instable results at low update rates.

Active methods provide more promising approaches for the interactive acquisition of light fields. Available active range scanning methods may be based on time-of-flight [43, 80, 131, 132], depth from defocus [75], photometric stereo [93] or projected-light triangulation [94]. Of these, the systems based on time-of-flight may provide the most flexible techniques for light field acquisition. They provide full image per-pixel depth information at real-time frame rates, expose the lowest hardware costs and especially do not demand computationally costly processing for geometry extraction. Recently, time-of-flight sensors have been applied in a variety of industrial contexts and have proven their effectiveness within these environments [51]. No research has been done, however, on applying time-of-flight sensors for the acquisition of geometry assisted light fields. Within this thesis a prototype acquisition pipeline is implemented which makes use of a *Photonic Mixing Device (PMD)* time-of-flight sensor in order to interactively acquire spherical light fields with per-pixel depth.

Photonic Mixing Device The *Photonic Mixing Device (PMD)* implements an active 3D measurement device which illuminates an observed scene using modulated, incoherent near infrared light (NIR) (see Figure 5.1). Light being reflected by objects within the scene is captured by pixel sensors commonly arranged in a grid of up to 160×120 resolution. For each of the pixels the incoming optical signal is correlated with the internal system reference signal of the active illumination unit. For this process a sinusoidal signal is assumed as reference signal which is then correlated with the incoming signal. This sampling is performed four times for each pixel but with an internal phase delay being applied. Each sampling then results in an per-pixel evaluation of the correlation function and an additional internal phase shift. From the subsequent sampling the distance related phase shift and thus the distance to the respective object region is calculated. Thus, per-pixel depth information is composed to a depth image of full sensor resolution. The sensor signal evaluation and depth extraction is performed on the pixel sensor and the camera main board in real-time. Signals, sampling results, and the corresponding depth values are accessible directly from a *FireWire* connection at up to 20 fps [49]. For detailed technological background of the PMD sensor the reader is referred to Lange [52].

The PMD sensor results, however, show some artifacts resulting from the implementation principle of this kind of technology. The distinctive features of the resulting data are topic of active research today. Ongoing research results contribute to the improvement of the PMD sensor technology and provide solutions for treatment of artifacts such as the low resolution, depth distortion, or motion blur [50].

Low Resolution Compared to up-to-date RGB imaging sensors currently available PMD sensors provide a relative small resolution of up to 160×120. While most applications are capable of adapting to the low resolution, the resulting large solid angle yields invalid depth values in areas of inhomogeneous depth occurring in situations such as silhouette edges and object boundaries. Lindner et al. have shown that these invalid depth values may be enhanced with an edge-enhanced distance refinement

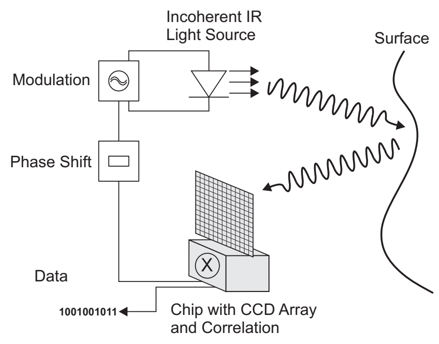


Figure 5.1: The time-of-flight principle of the PMD technology. Image courtesy of Keller and Kolb [49]

approach which discards these so called *Flying Pixels* concerning the sampling of sub-pixel locations while upscaling a PMD depth image [66].

Depth Distortion As the system's theoretically sinusoidal reference signal assumed for the correlation function is not achievable in practice, the extracted depth cannot reflect the correct distance, but comprises systematic errors. This systemic error has been investigated along with statistical uncertainties of the PMD sensor by Rapp et al. [89]. Lindner et al. approved the investigations of Rapp and presented an approach of the lateral and depth calibration of such sensors [64, 65].

Motion Blur With the sampling of the sinusoidal correlation function depending on phase delay and phase shift and the sampling being performed four times for each pixel element, the sensor is sensitive for motion. Motion blur artifacts may arise in dynamic scenes since the correlation results of the subsequent samplings lead to varying distances regarding a pixel region close to object boundaries.

Reflectivity The PMD sensor shows poor measurement results for two extreme reflectivity situations. With the infrared illumination being absorbed by an object in the scene, no incoming light is receivable by the sensor and thus the correlation function cannot be evaluated. On the other extreme, for infrared light being reflected perfectly, the sensor reaches saturation level. As a consequence, the correlation function does not provide any meaningful results. While this occurs to be a special burden on the PMD sensor, other measurement devices using optical

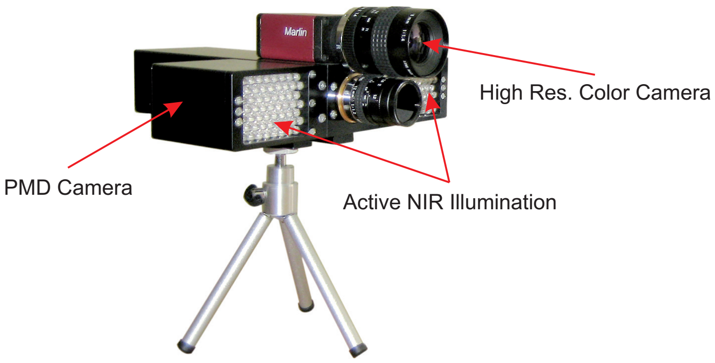


Figure 5.2: PMD camera with an additional high resolution color camera mounted.

principles, e.g. laser scanning systems, suffer from the same problems and do not provide more elaborate measurements in these situations.

2D/3D Camera Setup The combination of PMD depth information with high-resolution RGB data is in focus of current research. While Prasad et al. [87] have presented a hardware solution which combines a commodity CCD and a PMD sensor in a compact mono-ocular camera device using an optical splitter, this device is not available to the public yet. Todt et al. [114], however, have shown that registered RGB and depth information can be captured effectively by combing a high-resolution RGB camera and a PMD camera in a binocular setup (see Figure 5.2). While the proposed combination does provide combined RGBz data on a per-pixel basis, the approach does not take the PMD sensor characteristics into account.

Lindner et al. [68] have presented a data fusion approach which does take the sensor characteristics into account and provides effective counter measures to minimize the effect of PMD specific artifacts as described above. Their data fusion is performed in two main tasks, the distance refinement and the actual data fusion, which are arranged in a data processing pipeline according to Figure 5.3. Note that a simple background and invalid pixel removable is prepended to both of these tasks which discards unreliable distance values based on the distinct evaluation of the four correlation function results.

Distance Refinement The distance refinement implements two optimization steps in order to reduce the amount of invalid pixel and thus false color mapping. First, an extrapolation of valid pixels is applied to restore missing data for the subsequent biquadratic interpolation scheme. The

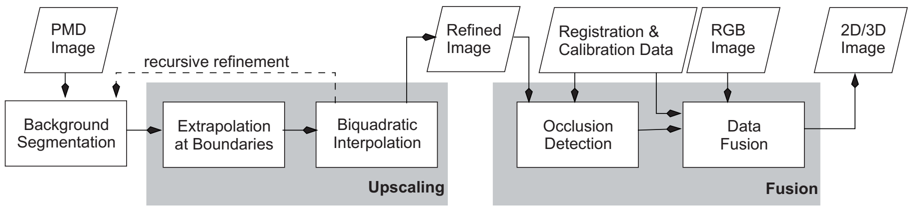


Figure 5.3: The data fusion of PMD depth- and RGB data is performed in a processing pipeline which is conceptually split in two blocks, the distance refinement and the actual data fusion. Image courtesy of Lindner et al. [68]

interpolation scheme implements an upscaling of the valid region while incorporating edge preservation. Upscaling is based on *pyramidal upscaling* [107] which recursively scales an image by a factor of two. With the refinement being applied, the effect of flying pixels is reduced by a reasonable degree.

Data Fusion The data fusion combines both, the refined distance information and the additional high-resolution color image by back-projecting the PMD pixels into world coordinates and a subsequent perspective projection onto the RGB image plane. To avoid an RGB image undersampling and thus a loss of information the projection is implemented based on projective texture mapping [97]. Note that the accuracy of the fusion is strictly dependent on the PMD depth precision. Thus, the PMD device has to be calibrated first using sophisticated calibration techniques to avoid depth distortion [64,65]. Special care must be taken to avoid incorrect mapping caused by occlusion effects which result from the different viewing directions of both cameras. Here Lindner et al. [68] provide a rendering approach comparable to shadow maps [90, 106] which reliably detects and erases occlusion effects.

Due to incoherent camera viewing frustums and the physical offset of the image sensors, the PMD camera does not cover the same viewing volume. Thus, the PMD camera's depth images can be mapped to a subregion of the RGB image, only. In practice the PMD image is mapped to a region covering 77% of the RGB image resolution. As a consequence, combined RGBz data is available at a maximum resolution of approximately 900×675 pixels. While the PMD camera operates at a maximum frame rate of up to 20 fps, the optimization pipeline and the 2D/3D data fusion described above effect the update frame rate. With the optimization being applied the update rate drops by 5 fps. Thus a maximum frame rate of 15 fps is achievable for combined RGBz images.

5.1.2 Pose Tracking

For the acquisition of light field samples an accurate position and orientation tracking system is mandatory for the sampling device. To acquire light field samples from pre-defined spherical sample positions such a system must provide stable and reliable tracking results with low latency to facilitate effective camera alignment. In the past several techniques have been presented to reliably determine camera poses.

A selection of approaches have been presented in the past which perform pose estimation based on captured input images directly and thus are not in need of additional tracking hardware to be installed [130]. However, these approaches do not provide stable tracking results for untextured scene contents and are likely to fail in these situations. Even though advanced image based camera tracking approaches have been presented recently which exploit the benefits of PMD camera devices to stabilize tracking results [9], these techniques exhibit computational complex and costly processing which limit the update rate and thus the interactive, intuitive and effective placement of a sampling device.

Active tracking systems which are in need of additional hardware components commonly attach a tracking target to the tracked device and make use of reference signals to precisely establish the tracking target's pose. Such systems are commonly used in the field of Virtual Reality systems and have proven to be well suited for interactive and accurate placement of devices. Available systems exploit electro-magnetic effects, ultrasonic techniques, or optical systems for pose estimation.

Electro-magnetic systems [20] exploit the effects of electro-magnetism to determine the pose of a tracking target within the magnetic field. These approaches are subject to ferromagnetic interferences and are thus unsuitable for the positioning of a camera covered by metal and attached to a power cord.

Acoustic tracking systems [48], either exploit the time-of-flight or the phase coherence of an ultrasonic signal, to determine the distance and orientation of a receiver relative to a transmitter. Due to the nature of sound acoustic, tracking is relatively slow and effected by temperature, air pressure or changes of the humidity as these aspects effect the acoustic velocity. These systems show poor positing results for environments which exhibit reverberation and echo.

Optical tracking systems [84] make use of infrared based illumination and camera systems to establish reliable and stable pose estimation. With the measurement volume being lit by infrared light sources the reflection of a tracked marker is captured by multiple cameras. For each camera the line of sight to the marker is established and from the intersection points of these lines the 3D

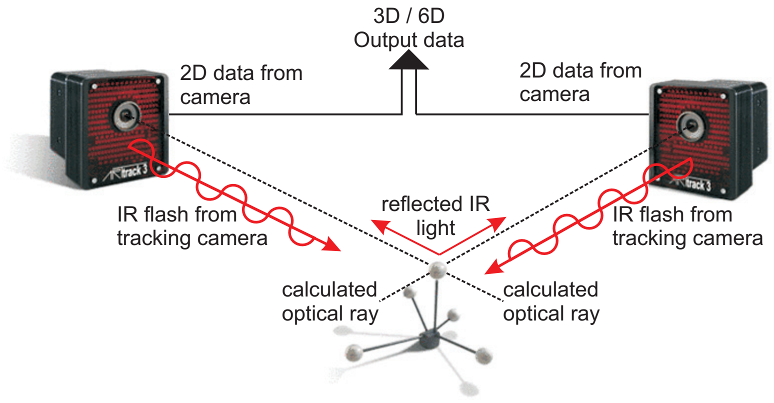


Figure 5.4: Optical tracking principle. The 2D image position of the IR reflections captured by the cameras are utilized to determine 3D positions of single markers and the pose of tracking targets with 6 DOF. Image courtesy of A.R.T. GmbH [4]

position is established (see Figure 5.4). Using multiple markers in a fixed setup, the setup's pose is determinable with 6 DOF. Though, special care must be taken in the setup of such a system to avoid occlusion and ensure visibility of a target in the complete measurement volume from at least two camera postions. These systems provide a precise, flexible and stable solution for the positioning task within the light field acquisition process.

Optical Camera Tracking The DTrack optical camera tracking system implements a high-end tracking system for 3D and 6D tracking of custom tracking targets [4]. The system consists of a PC workstation, two or more tracking cameras, and up to 20 custom tracking targets. The PC workstation runs the tracking software which calculates the 6 DOF target positions and orientations within the tracking volume from the line of sight to the marker's infrared reflection which is captured in 2D by the cameras (see Figure 5.4). The intelligent tracking cameras are equipped with an FPGA for accelerated marker data analysis. The cameras also include an infrared flash to illuminate the tracked objects. Up to 16 cameras can be combined in a tracking system. 3D Markers and 6D Targets can be calibrated for the tracking system and mounted to any custom device. Tracking is performed at an update rate of 60 Hz with a maximum error of 0.5 mm in position and 0.1° in orientation.

Tracking Setup For tracking the position and orientation of the bifocal PMD camera setup a 6D tracking target consisting of five reflective 3D markers is mounted on top of the RGB camera. To ensure visibility for an appropriate tracking volume four ARTrack2 cameras have been installed to the ceiling of a laboratory environment such that a tracking volume of 5 $m \times 3$ $m \times 3$ m is reliably covered. Note that extensive tests have been performed to evaluate the influence of the tracking system's infrared flashing system on the PMD sensor system. Fortunately an effect of the tracking system's flash on the PMD sensor precision could not be diagnosed. A stationary system has been utilized in this case. However, portable systems are also available for more flexible application areas.

Hand-Eye Transformation With the 6D tracking target mounted to the camera system the target's pose is reliably determinable in tracking coordinates. However, the pose of the RGB camera sensor not the tracking target's pose is relevant as reference value to steer the light field acquisition process. Thus, the fixed transformation from the tracked target coordinate system to the camera coordinate system is essential for the acquisition process. The determination of this transformation has been dubbed the *hand-eye calibration problem* and got its name from the robotics community, where a camera (eye) is mounted to the end effector (hand) of a robot. In this context several approaches have been presented in the past to solve this problem using a calibration pattern [54,119,120]. Recently Strobl and Hirzinger have presented a new calibration pattern based approach [108] which is available as a free software toolbox [109]. The toolbox takes a series of calibration pattern images and 6D tracking target poses to extract the desired hand-eye calibration. Using the provided toolbox the *hand-eye transformation* is determined efficiently. Applying the hand-eye transformation to the tracking results yields the imaging sensor pose in the tracking system's coordinate system directly.

5.1.3 Light Field Acquisition Pipeline

With the 2D/3D RGB-PMD camera setup and the accurate pose tracking system, the critical hardware components are available to establish an efficient light field acquisition process. The acquisition process presented in this section is implemented as a processing pipeline taking the RGBz images of the 2D/3D camera setup and the tracked camera pose as input data. Based on these input parameters the light field is sampled from pre-defined positions. Captured light field samples are fed into the spherical light field representation and the (incomplete) light field is being rendered directly and continuously for immediate visual feedback and quality evaluation. See Figure 5.5 for an overview

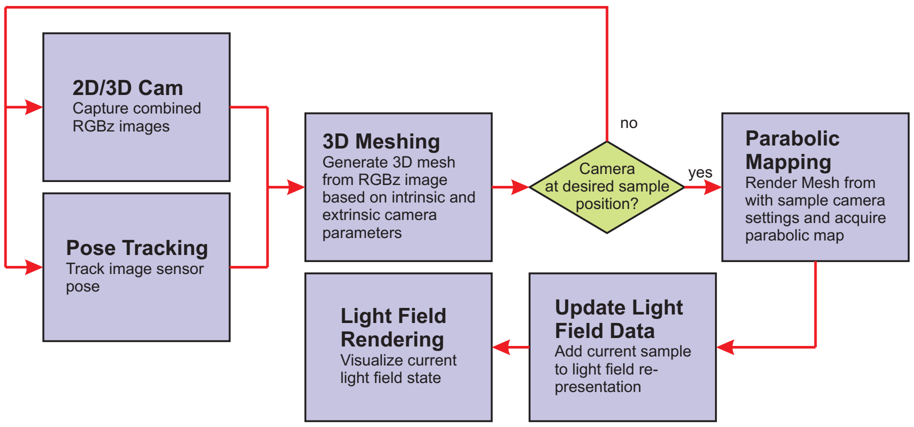

2D/3D Cam
Capture combined RGBz images
Pose Tracking
Track image sensor pose
3D Meshing
Generate 3D mesh from RGBz image based on intrinsic and extrinsic camera parameters
Camera at desired sample position?
no
yes
Parabolic Mapping
Render Mesh from with sample camera settings and acquire parabolic map
Update Light Field Data
Add current sample to light field re-presentation
Light Field Rendering
Visualize current light field state

Figure 5.5: The light field acquisition pipeline: From the captured RGBz image a 3D mesh is generated according to the camera's intrinsic and extrinsic camera parameters. If the camera is positioned according to a pre-defined spherical sample position then the 3D mesh is rebinned to the sample position and a parabolic RGBz map is generated and stored with the light field representation.

of the processing pipeline. The pipeline can be structured in five subsequent steps. After updating the input data, a 3D polygonal mesh is reconstructed from the RGBz data. The camera's pose is evaluated against the pre-defined spherical light field positions and a light field sample is acquired if the camera pose does correspond to a specific sample position. With a light field sample being acquired, the light field representation is updated and the preliminary light field is rendered.

Input Data Update The input data is updated by synchronously polling new data from both, the 2D/3D camera setup and the DTrack tracking system. When polling new data from the camera setup, the sensor data of the RGB and the PMD camera are read out and fed into the PMD processing and 2D/3D fusion pipeline as described in Section 5.1.1. The PMD pipeline implements state-of-the-art optimization and correction functions based on specific PMD characteristics and thus provides optimized RGBz data at a frame rate of 15 fps. The image senor's pose is updated based on the tracking results by applying the hand-eye transformation. Thus, the update step provides the 2D/3D camera's fusionated RGBz image and the current camera setup's pose in tracking coordinates.

3D Mesh Generation From the actual camera setup's pose and the RGBz image data a 3D polygonal mesh is reconstructed based on the intrinsic camera parameters which are available from the PMD processing pipeline described in Section 5.1.1. The mesh is generated first in the camera coordinate system with the image sensor being the origin and the negative z-axis pointing in camera viewing direction. Then, with the tracked pose and the hand-eye transformation matrix to be known, the mesh is transformed to the tracking coordinate system by applying the the tracking transformation matrix and the hand-eye transformation matrix. For clarity and simplicity reasons in this chapter the tracking coordinate system is defined to be identical to the world coordinate system.

Initial Triangle Mesh To generate a 3D mesh from the RGBz images a triangle mesh is rendered using a vertex buffer object. The vertex buffer object is initialized once as a triangle mesh with a resolution equal to the input camera resolution, such that the centers of four adjacent camera pixels are connected by the edges of a quad and the quad being split into two triangles. The mesh is generated in normalized device coordinates (NDC) and a z value of 0 such that it equally spans along the x and y direction in the range of [-1,1]. The updated input RGBz image is mapped to the triangle mesh as texture with the alpha component containing the depth values. The depth values are interpreted as offset values. The mesh is rendered using a customized geometry- and fragment program. While the fragment program performs a simple texture mapping operation only, the geometry shader implements the mesh reconstruction and optimization process.

Within the geometry shader the depth values are fetched from the alpha channel of the input textures. The depth values are then applied to the vertices as offset values in the z component. Thus, from the input mesh being defined with vertex coordinates $V = (x, y, 0)^T$ and $x, y = [-1, 1]$ an offset mesh is generated. The offset mesh expands orthogonal to the X/Y plane in the negative z direction. Note that the depth values are given as radial depth values along per-pixel rays in the 2D/3D camera setup. While the x and y components of the mesh are defined in the camera's normalized device coordinates, the depth values now need to be converted to NDC in order to achieve a consistent representation. This can be easily done by projecting the vertices to the camera image space using the camera projection matric ($M_{C_{Proj}}$). The projection matrix is extracted from the intrinsic camera calibration parameters which are known from the PMD processing pipeline. Note, however, that only the z component is to be adjusted. Thus, for a given mesh vertex $V_{Offset}(x, y, z)$ only the z component is substituted by the projected z component of vertex V_{Proj}.

$$V_{Proj}(x_{Proj}, y_{Proj}, z_{Proj})^T = M_{C_{Proj}} \cdot V_{Offset} \tag{5.1}$$

With the z value being substituted the adjusted vertices $V_{NDC}(x, y, z_{Proj})^T$ now define the mesh in NDC.

From the structure of a regular projection matrix it can be easily seen that the z component of the projected vertex is computed independently of the x and y components. Thus z values (depth values) can be handled and transformed to NDC independently (see Möller and Haines, pp 61–66 [2]).

To finally render the mesh in cartesian camera coordinates the mesh is unprojected again using the inverse camera projection matrix ($M^{-1}_{C_{Proj}}$). Then the V_{NDC} are transformed to camera coordinates V_{Cam} according to:

$$V_{Cam}(x_{Cam}, y_{Cam}, z_{Cam})^T = M^{-1}_{C_{Proj}} \cdot V_{NDC} \tag{5.2}$$

Mesh Optimization In this stage the triangle mesh represents a solid surface in camera coordinates. For boundary regions occurring at sharp edges within the observed object or at the silhouette edges, reconstructed triangles span over a large variation in depth. These triangles occur in regions which are occluded from the camera's point of view and thus do not provide exploitable depth values. If observed from a position being exactly identical to the input camera's pose, these triangles are not visible as they span in line of sight. For slightly varying viewing directions, however, these triangles occur as extremely elongated textured triangles (See Figure 5.6, top middle). As these regions are not visible from the original view, the texture does not carry any meaningful information and the shape, resulting from the interpolation of adjacent depth values, can be interpreted as reconstruction artifact. Fortunately, these regions are identified easily by evaluating the gradient for each vertex based on discrete depth values using finite difference methods within a given triangle. A triangle is culled within the geometry shader, if the gradient exceeds a given threshold value which is defined by the user (See Figure 5.6, top right).

For scenes also including regions of non-interest such as background details and additional information to the sides of the object of interest, clip planes are implemented to provide a flexible tool for object isolation. Up to six clip planes are available which are aligned along the camera

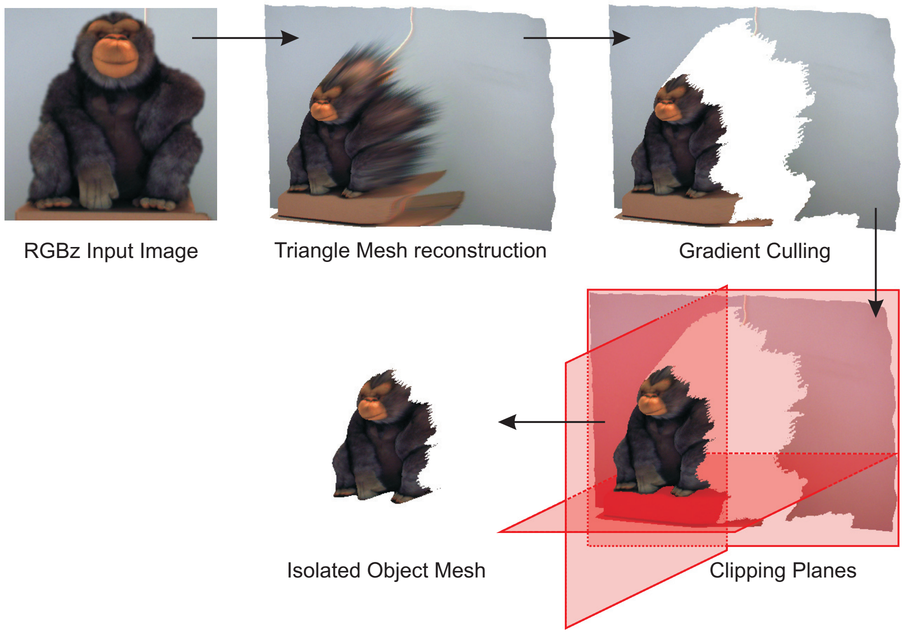


Figure 5.6: Based on RGBz input images (top,left), a triangle mesh is reconstructed in camera coordinates. Top, middle: Elongated triangles resulting from depth interpolation at boundary edges are visible for viewing directions which diverge from the current 2D/3D camera setup pose. Top, right: Reconstruction artifacts are erased by applying a gradient based triangle culling. Bottom, right: Redundant geometric details are culled by adjusting up to 6 clip planes to isolate the object of interest.

coordinate axis and alow to cull 3D content by limiting the maximum and minimum x,y and z values of the reconstructed mesh vertices (See Figure 5.6, bottom right).

With the mesh optimization being performed, an isolated solid surface mesh of the captured object is rendered in camera coordinates. By applying the tracking and hand-eye transformation this mesh is easily transformed to the tracking coordinate system.

Camera Pose Evaluation With each update circle of the light field acquisition pipeline, the camera setup's pose is evaluated based on the tracking results and compared against the pre-defined set of spherical sample positions. Remember that the sample positions are defined by the spherical approximation given by the light field parametrization (see Section 3.1). Thus, given a specific sample resolution of 12, 42, 162 or 642 sample positions, discrete sample positions are pre-defined. Tracking coordinates are evaluated in meters. Thus, the unit sphere is defined with a radius of 1 m centered at the origin of the tracking coordinate system. With the sample positions to be known, the current 2D/3D camera position and central viewing direction can be evaluated with respect to these positions. Note, however, that the spherical representation and the camera position must be defined with respect to the same coordinate system. For clarity reason, the spherical representation and the 2D/3D camera pose are assumed to be given with respect to the tracking coordinate system. Note that different base coordinate systems can be easily defined by applying a common coordinate system transformation to both, the spherical sample coordinates and the tracked camera coordinates.

In the ideal case, the camera is positioned such that the position is identical to a spherical sample position and the camera's viewing direction does correspond to the central sampling direction. The central sampling direction is given by the vector emerging from the sample position and passing through the spherical representation's origin, in this case, the tracking coordinate system's origin. However, positioning the camera according to a sample position and -direction is a challenging task. Thus, for user convenience the pose is evaluated with a certain tolerance.

While defining an epsilon environment around a desired sample position would provide a straight forward solution for a ”soft” pose evaluation, this approach does not take the viewing direction into account. Thus, two distinct epsilon environments have been implemented for each, the camera's direction and its position. To evaluate the camera's pose, the closest sample position is determined first. The closest sample position is determined efficiently by intersecting the ray, emerging from the origin and passing through the camera position, with the spherical approximation. Based on the hierarchical structure of the spherical approximation, a single triangle which is intersected by this ray is identified quickly by establishing the intersection with the sparsest spherical approximation and then subsequently refining the intersection based on the triangles of the next finer discretization level. The closest sample position is determined from the corresponding three vertices (see Figure 5.7, left). For the closest sample position the direction is evaluated first. If the difference in the camera's central viewing direction and the sample position's central sample direction ($\epsilon\alpha$) does not exceed a given epsilon angle the camera's orientation is assumed to be valid. In a second step the camera's absolute distance to the sample position is evaluated. The camera's pose is assumed to be valid if the distance (ϵd) does not exceed a given epsilon distance (see Figure 5.7, middle).

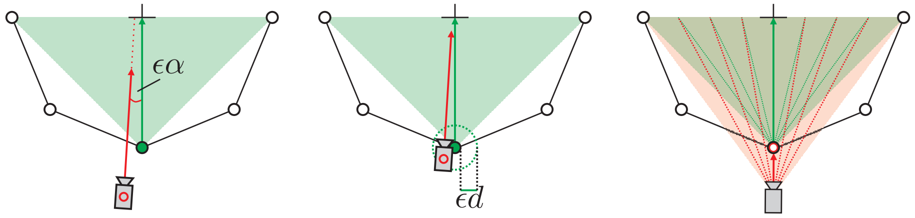


Figure 5.7: The camera's central viewing direction and its position is evaluated with respect to the closest sample direction. Left: Central camera viewing direction within given tolerance ($\epsilon\alpha$), but the camera fails the positional constraint. Middle: Camera within epsilon environment of both, direction and position. Right: The directional constraint is fulfilled and the camera's viewing ray intersects the spherical approximation within the positional epsilon environment but the camera itself is not positioned within the epsilon environment. Note that the camera's central ray corresponds to the central sampling direction, while rising angular errors occur for divergent directions.

However, under certain circumstances the user might not be able to move the camera along a path on the spherical approximation's surface. Thus, for some reasons he might be limited to setup the camera such that the camera pose does fulfill the directional constraint but fails the distance constraint. In these cases the camera pose is still considered to be valid as long as the camera's central viewing ray does intersect the spherical approximation within the given epsilon distance environment. As shown in Figure 5.7, right, the central viewing ray does still correspond the sample position's central viewing direction in these cases and thus provides valid information. Sampling rays to the sides of the central direction, however, exhibit an increasing angular error. The angular effect is factored by the camera's distance to the sample position. While providing a more flexible camera positioning approach, this flexibility comes at the price of directional error to the sides of the central sampling direction. Thus, it is to used under exceptional circumstances, only.

For user guidance a graphical representation showing the spherical approximation and a proxy geometry of the camera setup with the camera's field of view is available during the acquisition task. The closest sample position and the epsilon environment are being highlighted.

Parabolic Mapping and Sample Acquisition With the 2D/3D camera setup being placed according to a pre-defined sample position and the observed object mesh being reconstructed, a light field sample can be acquired by parabolically mapping the mesh from the pre-defined sampling position on the unit sphere.

Depending on the object from which a light field is to be acquired, the object does not necessarily fit into this representation. Remember, the object's bounding sphere has to be defined such that it is centered around the origin with a radius of $\frac{1}{\sqrt{2}}$ (see Section 3.2). Hence, for objects exceeding the dimensions of the unit sphere the reconstructed mesh has to be scaled to fit into the bounding sphere. Obviously, if a scaling is applied to the mesh for one sample position, it has to be scaled accordingly for all of the sample positions to maintain constant proportions.

Then, to acquire a light field sample according to the parabolic image space parametrization, the standard vertex processing step being implemented for the mesh reconstruction is replaced by a customized vertex program, which projects all vertices onto the unit sphere and converts the result to parabolic coordinates. For the generation of parabolic coordinates it is essential to transform the scene such that the sample capture position is placed at $C = (0, 0, 1)^T$, looking along the negative z-axis. Thus, the scene is transformed according to the sample position's viewing matrix which is obtained from the spherical parametrization for the current sample position.

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

1. Projection to the hemisphere
   Each mesh vertex V is projected from the sample point onto the opposite unit hemisphere. The projection is computed by casting a ray from the sample position 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 then converted to parabolic coordinates, according to Equation 4.2

3. Depth Adjustment
   The depth value is calculated as the fractional part of the secant length according to Figure 3.5.

Based on this parabolic mapping, light field samples are acquired from the reconstructed mesh being generated based on the 2D/3D input images of the camera setup being positioned at the sampling position. However, accurate light field samples are achievable only, if the camera is placed exactly corresponding to the position and direction pre-defined by the spherical representation. As mentioned above, an epsilon offset has been implemented for user convenience in the positioning task.

In case of the camera being set up such that its central viewing direction does differ by some degrees, the reconstructed mesh does exhibit visible reconstruction artifacts when being rendered from the pre-defined sample position (see above: 3D Mesh Generation - Mesh Optimization). Without mesh optimization being applied this results in geometric details being visible which are not visible from the 2D/3D camera's point of view and thus do not contain valid information. With the mesh optimization being applied, theses artifacts are removed. In this case, no information is visible in these undefined regions. Thus, with a light field sample being acquired from slightly varying sample positions this will result in no light field information being available for these regions. However, in the reverse this will avoid misleading information to be integrated into the light field representation. As up to three input light field samples are evaluated within the light field rendering process (see Section 4.1) missing information in one of the light field samples will likely to be covered by one of the two other light field samples.

Note, however, that in addition to mesh reconstruction artifacts, the 3D mesh does not represent the scene's correct reflectance characteristics if the variation in direction and position is chosen too large. Thus, the epsilon environment is to be chosen small to avoid dubious information being sampled. In practice an angular epsilon environment of $0.5° - 1°$ for the maximum directional deviation and a corresponding maximum positional error of (8 mm - 15 mm) have proven to result in both, a convenient acquisition process and acquisition results which do not exhibit artifacts resulting from reconstruction artifacts as described above.

Update Light Field and Render The light field sample being acquired in the previous step is integrated into the spherical light field representation directly without further processing. With the first sample being acquired, the light field renderer is capable of providing synthesized images based on the current set of acquired samples at real-time to provide a visual representation for quality evaluation within the acquisition process.

Thus, while acquiring additional samples, the user is capable of evaluating the current light field quality. Impropriate light field samples can be removed from the representation and replaced by new samples to maximize the quality of an acquired light field.

In practice ,this accompanying visualization has proven to be a powerful tool for quality evaluation and visual guidance to optimize acquisition results.

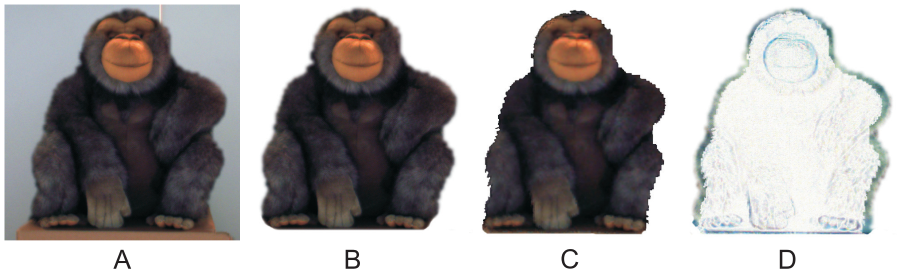


Figure 5.8: Light field acquisition results of a stuffed animal which exhibits diffuse materials only. A: The input RGB image showing the animal with background. B: RGB image with the background being keyed out. C: Rendering of the light field being acquired from 162 sample positions at a resolution of 512×512. D: Difference image showing per-pixel differences of the light field rendering (C) and the input RGB image (B) multiplied by a factor of 4 and inverted.

5.1.4 Acquisition Results

With the 2D/3D sampling hardware, the tracking system and the proposed light field acquisition pipeline, light fields can be efficiently sampled from physical objects. The proposed acquisition reveals the potential of the bifocal PMD camera setup for the acquisition of light fields with per-pixel depth. However, the results expose limitations which are reasoned in the systematic and the current development sate of the young PMD technology.

Figure 5.8 shows light field acquisition results for a light field being captured from a stuffed animal. The physical object features a compact structure and diffuse materials. Thus, the object does not exhibit large variations in depth and practically no sharp boundary edges in the inner region. The silhouette edges, however, provide boundary regions with abrupt and large depth variation with respect to the background. The object's fur does exhibit diffuse reflection properties and thus provides uniform reflection. Regions which are covered by light absorbing materials or reflective materials do not occur.

In Figure 5.8 an RGB image of the stuffed animal (Figure 5.8, A) is shown accompanied by the rendering of the light field (Figure 5.8, C) being acquired using the 2D/3D PMD camera setup. A difference image showing the inverse per-pixel differences multiplied by a factor of 4 is shown in Figure 5.8, D. For comparison reasons the original input RGB input image is displayed with the background being keyed out (Figure 5.8, B).

From the difference image it can be seen clearly that the inner regions of the object are reliably captured and reconstructed. Some minor artifacts occur at

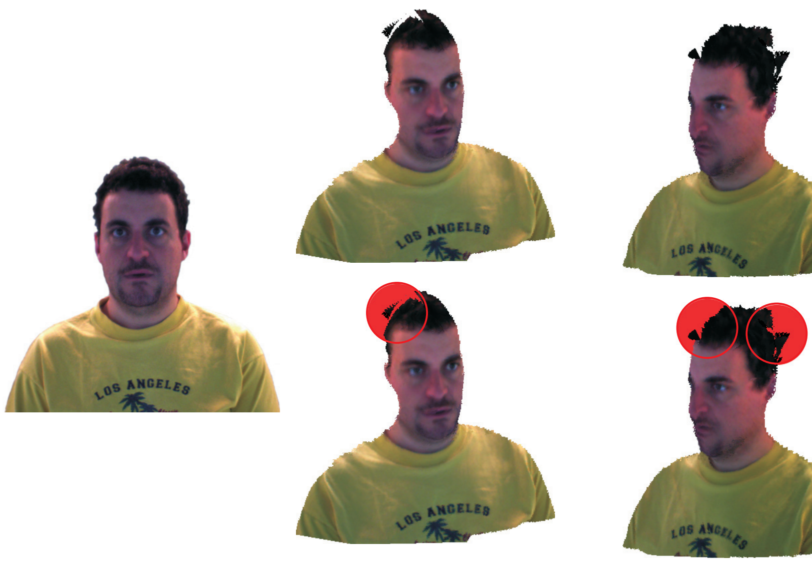


Figure 5.9: Light field being captured from Christof Rezk Salama. Discontinuity artifacts appear in regions of infrared light being absorbed during acquisition. Left: RGB input image with the background being culled. Middle and Right: Light field rendering with the discontinuity artifacts being highlighted in the bottom row.

the image contours such as the nose, the hands or the feet. Solely the silhouette exhibits some more visible artifacts which occur as aliasing like rendering artifacts. These visible stepping artifacts result from the PMD camera's relative low resolution of 160×120 and the resulting large solid angle. The large solid angle yields invalid depth values in areas of inhomogeneous depth occurring in situations such as silhouette edges and object boundaries (see Section 5.1.1). With these invalid regions being culled within the acquisition process, these regions are not present in the light field representation and thus cannot be reconstructed in the image synthesis. As a consequence, the mean absolute error of per-pixel color values resulting from the comparison of the input image and the rendered result is reasonable high (0.0678861). The root mean squared error for the same pair of images is determined accordingly high (0.209604).

While the stuffed animal did provide diffuse materials, only, additional artifacts arise from light absorbing materials. Figure 5.9 demonstrates the effect of infrared light being absorbed within the acquisition. Per-pixel depth cannot be evaluated reliably with the infrared light being absorbed in some image regions. Thus, the polygonal mesh being reconstructed for the sampling process is built from incorrect and incoherent depth information in these regions. These geometric artifacts are sampled within the acquisition process and appear as discontinuities within the light field synthesis process. Figure 5.9 displays the results of a light field being acquired from Christof Rezk Salama. The light field renderings clearly demonstrate the effect of light being absorbed by the black hair. For this region heavy discontinuity artifacts appear.

However, the rendering results displayed in Figure 5.9 also show the strength of the presented approach. Note that ghosting artifacts are minimized to a reasonable amount. While some ghosting artifacts occur in the face region, the Shirt's printing is reconstructed without noticeable artifacts.

For complex scenes covering large variations in depth, which exhibit all variations of materials and expose sharp boundary edges, the proposed acquisition approach is effected by the PMD setup's limitations to a high degree. Figure 5.10 shows the acquisition results of such a scene. Infrared light absorbing materials (labeled A in Figure 5.10) occur as holes or exhibit discontinuity artifacts in the light field rendering. High reflective material causes the PMD sensor to reach saturation level in the acquisition process if infrared light is perfectly reflected. Thus, no valid depth information and as a consequence no geometric detail is available for these regions. These regions are not acquired and appear as holes (labeled B) in the rendering.

For this kind of complex scenes being composed from multiple objects, ghosting and geometric distortion effects appear at boundary edges (labeled C). For the boundary edges depth information cannot be reliably reconstructed. Due to the PMD camera's large per-pixel solid angle, the depth information being acquired for these regions results from a mixture of values being measured for different object surface distances. For small variations in depth the gradient threshold being applied for mesh optimization within the acquisition process does not suffice as a counter measure. This results in geometric distortions being acquired and reconstructed within the light field synthesis. If, however, the gradient threshold is chosen too small, discontinuities and loss of geometric information may appear in region of boundary edges (labeled D). As this complex scene is composed from a selection of relatively small objects, the silhouette artifacts effect the overall quality of the light field rendering to a high degree.

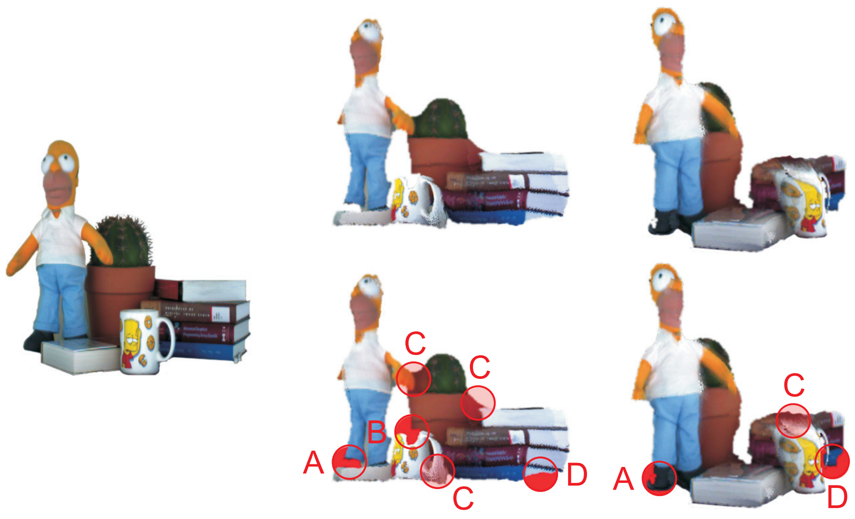


Figure 5.10: Light field rendering of a complex scene containing large variations in depth and light absorbing as well as high reflective materials. Left: RGB input image with the background being keyed out. Middle and right: Light field rendering with artifacts being highlighted and labeled in the bottom row. Label A highlights discontinuity artifacts resulting from light absorbing materials. B: Holes appear in regions of reflective materials. C: Geometric distortion at boundary edges with small variations in depth. D: Loss of geometric information caused by the gradient threshold being chosen too small within the acquisition process.

5.2 Light Field Generation from Synthetic Scenes

The generation of light fields from synthetic scenes based on 3D geometry is implemented according to the acquisition pipeline. However, instead of a light field acquisition device a customized renderer is utilized to extract light field samples. Instead of a mesh being reconstructed from an RGBz image, here a given 3D representation is rendered once for each sample camera being defined by the spherical parametrization to generate a synthetic light field of the synthetic object. For each sample camera the scene is transformed according to the sample camera transformation matrix which is available from the parametrization setup, such that the camera is placed at the position $C = (0,0,1)^T$, looking along the negative z-axis. To ensure the scene to fit into the bounding sphere with radius of $\frac{1}{\sqrt{2}}$ centered around the origin, the entire scene has to be scaled and translated.

Note, that this procedure is conceptually identical to the acquisition approach being utilized to sample a light field from the reconstructed 3D mesh of a physical object (see Section 5.1.3). Thus, the same customized vertex program is utilized to project all vertices onto the unit sphere and to convert the result to parabolic coordinates. This allows for the efficient generation of 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.

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 the sample position's 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.

All of the light fields being used to demonstrate the quality of the light field rendering approach have been generated from synthetic objects using this proposed technique (see Section 4.1). Light fields of the Tie-Fighter model (see Figure 4.12) an Michelangelo's David Statue (see Figure 6.1) have been generated using the *Autodesk Maya* plugin.

5.3 Conclusion

Within this chapter two approaches have been presented to generate spherical light fields with per-pixel depth. While the first approach exploits upcoming depth imaging sensor technology in combination with sophisticated tracking technology to acquire light fields from physical objects, the second approach is aiming at the generation of light fields from synthetic objects.

For synthetic objects spherical light field representations can be generated automatically from any given synthetic 3D representation using a specialized rendering engine. The integration of light field generation capabilities into commercially available 3D modeling packages provides a powerful tool for the flexible generation of sophisticated light fields from any kind of synthetic objects. Light fields being generated from synthetic objects provide high quality representations which are reconstructed precisely using the raycasting light field rendering approach presented in Section 4.3.

For physical objects an acquisition approach has been presented which implements a prototype setup for the interactive acquisition of spherical light fields. The approach makes use of a bifocal 2D/3D camera setup built from a PMD time-of-flight depth imaging sensor and a high resolution RGB camera to acquire combined RGBz information directly without the need for additional processing to extract per-pixel depth information. From the combined RGBz images a polygonal 3D mesh can be reconstructed in world coordinates efficiently using a high precision optical tracking system. With the 2D/3D camera being adjusted according to a pre-defined sample position, a light field sample is easily acquired by rendering the 3D mesh from the sample position.

While state-of-the-art optimization techniques have been implemented to optimize the PMD sensor performance, light field acquisition is still effected by the sensor characteristic to a high degree. The physical object acquisition results demonstrate the main shortcomings of the 2D/3D PMD camera system. Due to the low resolution of the PMD camera boundary edges cannot be acquired precisely, leading to discontinuity and distortion artifacts in the mesh reconstruction and thus in the light field representation. The acquisition is limited to scenes which do not exhibit materials that absorb near infrared light and do not provide high reflective materials. In appearance of one of those materials, the PMD sensor will provide unreliable depth information which effects the mesh representation to a high degree. Thus, the acquisition of light fields is limited to a small selection of single compact objects which provide small variations in depth and uniform reflection characteristics, only.

In the future, however, research in the field of time-of-flight sensor technology will address the problem of low resolution and reflectivity. Current research on PMD sensor calibration [67] and the combination of stereo imaging techniques with PMD sensor information [8, 77] will provide a stable and reliable technique for the acquisition of combined RGBz images. With the combined mono ocular PMD camera setup [87] which provides perfectly aligned and synchronized RGBz images, an improved PMD sensor will be available in the near future. Thus, upcoming PMD techniques will perfectly support the light field acquisition process presented in this chapter and improve the acquisition results to a high degree.

6 Conclusion and Future Work

This work has presented a new light field technique based on a uniform spherical parametrization which implements per-pixel depth correction of rays to synthesize new high-quality virtual views at real-time frame rates.
The spherical light field parametrization presented in this thesis implements a uniform sampling of observation space and allows view synthesizes to be performed with 6 DOF. The light field representation stores an RGBz parabolic texture map which combines both RGB and depth values per pixel. Thus, it implements an efficient light field representation with respect to data volume which includes an implicit geometric scene description at minimum additional storage costs.

In contrast to unstructured light field rendering approaches presented by Schirmacher and Buehler [12, 96] the representation is guaranteed to be invariant under both, translation and rotation. Due to the uniformity of the proposed representation, discontinuity artifacts as they are observed with the lumigraph approach presented by Gortler et al. [35] do not appear. While being akin to spherical parametrization presented in the past [15, 44], the amount of light field samples being mandatory for a high-quality image synthesis is drastically reduced by a factor of over 90%–95%.

The light field rendering techniques presented in this dissertation exploit the efficient spherical representation in order to generate high-quality virtual views. Both implementations, the iterative refinement approach and the ray-casting technique, implement a per-pixel depth correction of rays. With the depth correction being applied, virtual views are generated without noticeable ghosting artifacts at real-time frame rates. Visual features from meso and micro structures as well as concavities are precisely reconstructed and silhouette edges are reliably resampled.

With the level of detail rendering approach and the progressive refinement light field rendering, two efficient and flexible rendering techniques have been presented in this thesis which allow light field rendering techniques to be utilized in a variety of application areas. The LOD technique provides a powerful tool for rendering performance management which allows light field techniques to be efficiently integrated into performance critical real-time applications such as computer games. With the progressive client server based rendering technique, light field representations can be flexibly shared in web-based applications in order to provide access to sophisticated visualization results from a variety of sources.

For the acquisition of such a spherical light field representation two different techniques have been presented. A straight forward rendering based approach was presented which provides an efficient tool set for the generation of spherical light fields with per-pixel depth from synthetic scenes. This synthetic generation method allows light fields to be efficiently generated from arbitrary complex scenes using a customized rendering engine. This work has demonstrated that this light field generation technique can be easily integrated in any kind of 3D rendering application. For the acquisition of light fields from physical objects a new sampling approach has been introduced in this work which makes use of upcoming 3D sensor technology to synchronously acquire combined RGBz images. The acquisition approach provides immediate visual feedback for quality evaluation of (incomplete) light fields within the acquisition process. This work demonstrated that emerging 3D time-of-flight sensors can be used to efficiently acquire light fields with per-pixel depth without the need for complex geometry evaluation and geometry processing steps. However, this work also showed that acquisition artifacts arise under certain circumstances resulting in poor light field representations. The author is confident that current research focusing on this young technology and future 3D sensor systems will solve these limitations. Then, these devices provide the optimal solution for the acquisition of light field representations with per-pixel depth.

With the spherical parametrization, the high-quality rendering approaches and the light field generation technique presented in this dissertation, sophisticated techniques are available today which are capable of promoting the application of light field rendering techniques in many new application areas.

Light field rendering techniques are ideally suitable for the presentation of medical data which is traditionally in need of sophisticated rendering algorithms to provide high quality results. The author has shown that the presentation of such data sets can be efficiently implemented using light field techniques [91]. The progressive light field rendering approach presents an efficient solution for the presentation of such kind of data.

Engineering disciplines which are in need of publishing very large 3D representations but do not want to grant access to the underlying structural information will profit from light field representations to a high degree. The efficient parametrization presented in this thesis provides the ideal solution for a compact exchange of 3D representations while retaining the sensible structural information.

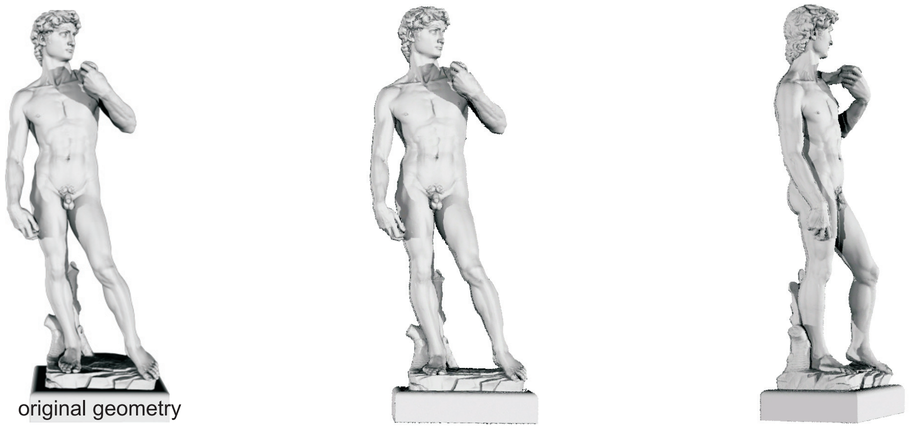


Figure 6.1: Light field rendering in the context of cultural heritage. Left: The original polygonal based rendering of the simplified version (500k polygons) of the original geometry of David obtained from *The Digital Michelangelo Project* [62] rendered using Mental Ray for Maya in 39 seconds. Middle and Right: Light field rendering from three composed light fields, each sampled from 162 sample positions at an image resolution of 512×512. Rendering is performed at a frame rate of 26.5 fps.

With the advent of computer graphics technology in the field of cultural heritage presentation systems [42], museums are starting to integrate visualizations of synthetic content in their exhibitions in order to grant interactive access to virtual 3D representations [113, 118]. Here, the sophisticated light field rendering approaches presented in this work provide an efficient solution to the high-quality presentation of ancient artifacts (see Figure 6.1).

The benefits of the light field techniques presented in this dissertation will open up for new ways of data visualization and analysis. The author has demonstrated the applicability of light field synthesis for object recognition as a proof of concept in his research activities [111]. This conceptual work will inspire future research in the field of object recognition. With the spherical light field representation proven to be flexible and adjustable at run-time, it will allow for efficient data manipulation in future research projects. Especially the potential of adjusting the lighting conditions of the light field and thus relight the virtual scene being represented by the light field is in focus of future research activities.

A
Appendix

A.1 Iterative Refinement Fragment Shader

```
void main(
      float3 c0            : TEXCOORD0, //sample position coordinates
      float3 c1            : TEXCOORD1,
      float3 c2            : TEXCOORD2,
      float3 P_eye         : TEXCOORD3, //virtual viewpoint position
      float3 colorIn       : COLOR,     //interpolation weights
      uniform float4x4 ModelViewProj,   //virtual view modelview-proj

      uniform sampler2D DecalMap0 : TEXUNIT0,//parabolic light field
      uniform sampler2D DecalMap1 : TEXUNIT1,//sample textures
      uniform sampler2D DecalMap2 : TEXUNIT2,

      uniform sampler2D DepthMap0 : TEXUNIT3,//parabolic depth maps
      uniform sampler2D DepthMap1 : TEXUNIT4,
      uniform sampler2D DepthMap2 : TEXUNIT5,
      uniform float4 backgroundColor,
      uniform float  errorThreshold, //steering parameter for quality
      uniform float  maxIterations,  //max iteration count
      uniform float  skip,           //size of empty region
                                     //inside sphere to be skipped

      uniform float3x3 CamToWorld0, //transformation matrices of
      uniform float3x3 CamToWorld1, //sample cameras
      uniform float3x3 CamToWorld2,

      out float4 color : COLOR,
      out float oDepth : DEPTH)
{
  //fragment's position based on barycentric weight from sample
  vertices
  float3 v   = c0 * colorIn.x  + c1 * colorIn.y + c2 * colorIn.z;
  float3 dir = v-P_eye;          //the view direction
  float3 dirN = normalize(dir); //normalized view direction

  //extract interpolation weights from interpoalted color
  //x,y,z correspond to c0,c1,c2, accordingly
  float3 cameraWeight;
  cameraWeight=colorIn;
  //initialize the 3 camera color values with background color
  float4 color0 = backgroundColor;
  float4 color1 = backgroundColor;
  float4 color2 = backgroundColor;

```

```
//determine viewing ray - sphere intersection on opposite hemisphere
float A,B,C;
C  = dot(P_eye,P_eye) - 1.0;
B  = 2.0 * dot(P_eye,dir);
A = dot(dir,dir);

float S = max( (B*B - 4.0 * A*C) ,0.0); //components under sqrt
//ignore negative solution - opposite hemisphere, only
float lambda = (-B + sqrt(S)) /A/2.0;

//Sphere intersection
float3 vecS  = (P_eye + lambda * dir);

//direction to sphere interesection point
float3 dir0 = mul(CamToWorld0, vecS);
float3 dir1 = mul(CamToWorld1, vecS);
float3 dir2 = mul(CamToWorld2, vecS);

//texture coordinates for sphere intersection
//based on parabolic mapping
float3 UV0;
float3 UV1;
float3 UV2;
UV0.x = dir0.x / 2.0 / (1.0 - dir0.z) + 0.5;
UV0.y = dir0.y / 2.0 / (1.0 - dir0.z) + 0.5;
UV0.z = dir0.z;

UV1.x = dir1.x / 2.0 / (1.0 - dir1.z) + 0.5;
UV1.y = dir1.y / 2.0 / (1.0 - dir1.z) + 0.5;
UV1.z = dir1.z;

UV2.x = dir2.x / 2.0 / (1.0 - dir2.z) + 0.5;
UV2.y = dir2.y / 2.0 / (1.0 - dir2.z) + 0.5;
UV2.z = dir2.z;

// xyz components refer to camera 0, 1, and 2, respectively.
float3 depthV;

// obtain the depth samples
depthV.x = (UV0.z < 0.0)? tex2D(DepthMap0,UV0.xy).r : 1.0;
depthV.y = (UV1.z < 0.0)? tex2D(DepthMap1,UV1.xy).r : 1.0;
depthV.z = (UV2.z < 0.0)? tex2D(DepthMap2,UV2.xy).r : 1.0;

//initialize parameters evaluated within iteration
//ensure absolute error >1 and >errorThreshold
float fError = errorThreshold + 1.0;
float fIter = -1.0;  //iterations performed
float3 vecS0 = vecS; //starting intersection points for
float3 vecS1 = vecS; //each sample cam corresponds to
float3 vecS2 = vecS; //initial sphere intersection
float3 dist;         //distance of the local estimes for 3 cams
float3 vecG;         //current position on ray
float3 direction0;   //direction of rays emerging from each of
float3 direction1;   //the three sample cameras
float3 direction2;

//refine local estimates until maximum iterations
//or errorThreshold reached
for(int a = 0; a < maxIterations; a++) {
  cameraWeight=colorIn; //reset camera weightsto barycentric
```

```
    // vectors from a vertex (camera) to its corresponding
    // intersection point with the geometry
    float3 dirG0 = depthV.x * (vecS0-c0);
    float3 dirG1 = depthV.y * (vecS1-c1);
    float3 dirG2 = depthV.z * (vecS2-c2);

    // intersection points for each camera
    float3 vecG0 = c0 + dirG0;
    float3 vecG1 = c1 + dirG1;
    float3 vecG2 = c2 + dirG2;

    // the distances are projected onto ray from actual camera to the
    //3D fragment position v)
    dist.x = dot(dirG0,dirN);
    dist.y = dot(dirG1,dirN);
    dist.z = dot(dirG2,dirN);

    //interpolated in barycentric coordinates
    float dd;
    //project local estimates on viewing ray if for the first few
    //iterations (current iteration<maxIterations/2) for pos on ray
    if (a < (maxIterations/2)) {
      dd = (dist.x + dist.y + dist.z)/3.0;
    } else { //take the maximum distance otherwise
      dd = max(dist.x,max(dist.y,dist.z));
    }
    // determine interpolated intersection point for the fragment
    vecG = v + dirN*dd;

    //calculate the error as distance from position on ray
    float3 vecError;
    vecError.x = length(vecG-vecG0);
    vecError.y = length(vecG-vecG1);
    vecError.z = length(vecG-vecG2);
    vecError *= cameraWeight;
    // the scalar error is calculated as the sum of the squared
    // distances between the interpolated intersection point
    // and the intersection point of the individual camera
    fError = dot(vecError,vecError);

    //re-weight the cameras according to their errors
    vecError/=dot(vecError,1..xxx); //normalize errors
    cameraWeights*=vecError; //adjust weights

    // prepare for the next iteration/color interpolation we
    // intersect the camera rays with the unit sphere = solving
    // 3 quadratic equations simultaneously
    // (similar to first intersection of viewing ray)
    float3 d0G = vecG-c0;
    float3 d1G = vecG-c1;
    float3 d2G = vecG-c2;

    float3 AG;
    float3 BG;
    C = dot(v,v) - 1.0;

    BG.x  = 2.0 * dot(v,d0G);
    BG.y  = 2.0 * dot(v,d1G);
    BG.z  = 2.0 * dot(v,d2G);

    AG.x = dot(d0G,d0G);
    AG.y = dot(d1G,d1G);
    AG.z = dot(d2G,d2G);

    float3 SG = max( (BG*BG - 4.0 * AG*C) ,0..xxx);
    float3 tG = (-BG.xyz + sqrt(SG.xyz)) /AG.xyz/2.0;

    //if the depth is within the inner region of the sphere
    //limited by parameter skip,
    if(depthV.x<(1.0-skip) && depthV.x>(skip))
    {
      vecS0 = c0 + tG.x * d0G;
      //transform the sphere points into the vertex-camera's
      //local coordinate system
      direction0 =  mul(CamToWorld0, vecS0 );
      // transform sphere point to parabolic map
      UV0.x = direction0.x / 2.0 / (1.0 - direction0.z) + 0.5;
      UV0.y = direction0.y / 2.0 / (1.0 - direction0.z) + 0.5;
      UV0.z = direction0.z;
      // sample the parabolic map for each vertex-camera
      color0=(UV0.z < 0.0)?tex2D(DecalMap0,UV0.xy):backgroundColor;
      depthV.x = (UV0.z < 0.0)? tex2D(DepthMap0,UV0.xy).r : 1.0;
    }else{ //prevent local estimates in back to influence next ray pos
      cameraWeight.x = 0.0;
      color0 = backgroundColor;
    }

    if(depthV.y<(1.0-skip) && depthV.y>(skip))
    {
      vecS1 = c1 + tG.y * d1G;
      direction1 =  mul(CamToWorld1, vecS1 );
      UV1.x = direction1.x / 2.0 / (1.0 - direction1.z) + 0.5;
      UV1.y = direction1.y / 2.0 / (1.0 - direction1.z) + 0.5;
      UV1.z = direction1.z;
      color1 = (UV1.z < 0.0)? tex2D(DecalMap1,UV1.xy) :
      backgroundColor;
      depthV.y = (UV1.z < 0.0)? tex2D(DepthMap1,UV1.xy).r : 1.0;
    }else{
      cameraWeight.y = 0.0;
      color1 = backgroundColor;
    }

    if(depthV.z<(1.0-skip) && depthV.z>(skip))
    {
      vecS2 = c2 + tG.z * d2G;
      direction2 =  mul(CamToWorld2, vecS2 );
      UV2.x = direction2.x / 2.0 / (1.0 - direction2.z) + 0.5;
      UV2.y = direction2.y / 2.0 / (1.0 - direction2.z) + 0.5;
      UV2.z = direction2.z;
      color2 = (UV2.z < 0.0)? tex2D(DecalMap2,UV2.xy) :
      backgroundColor;
      depthV.z = (UV2.z < 0.0)? tex2D(DepthMap2,UV2.xy).r : 1.0;
    }else{
      cameraWeight.z = 0.0;
      color2 = backgroundColor;
    }
    cameraWeight /= dot(cameraWeight,1..xxx); //normalize weights

    // if the error is below the threshold, exit the for loop
    if (fError < errorThreshold) {
      break;
    }
    fIter += 1.0;
  }

  fIter /= maxIterations; //ratio with respect to maximum iterations

  //determine depth for fragment by projecting to virtual view
  float4 depthPoint = mul(ModelViewProj, float4(vecG,1.0));
  depthPoint/=depthPoint.w;
  oDepth=(depthPoint.z+1)/2.0;

  //set sample cameras color to background if weight is 0
  if(!(cameraWeight.x > 0.0))
    color0 = backgroundColor;
  if(!(cameraWeight.y > 0.0))
    color1 = backgroundColor;
  if(!(cameraWeight.z > 0.0))
    color2 = backgroundColor;

  //interpolate color weights
  color = color0*cameraWeight.x+color1*cameraWeight.y+
        color2*cameraWeight.z;
}
```

Code Sample A.1: Iterative refinement light field rendering fragment program.

A.2 Raycasting Fragment Shader

```
void main(
      float3 c0            : TEXCOORD0, //sample position coordinates
      float3 c1            : TEXCOORD1,
      float3 c2            : TEXCOORD2,
      float3 P_eye         : TEXCOORD3, //virtual viewpoint position
      float3 colorIn       : COLOR,     //interpolation weights
      uniform float4x4 ModelViewProj,   //virtual view modelview-proj

      uniform sampler2D DecalMap0 : TEXUNIT0,//parabolic light field
      uniform sampler2D DecalMap1 : TEXUNIT1,//sample textures
      uniform sampler2D DecalMap2 : TEXUNIT2,

      uniform sampler2D DepthMap0 : TEXUNIT3,//parabolic depth maps
      uniform sampler2D DepthMap1 : TEXUNIT4,
      uniform sampler2D DepthMap2 : TEXUNIT5,

      uniform float  errorThreshold,//steering parameter for quality
      uniform float  epsilon,       //size of epsil. environment
      uniform float  stepsize,      //raycasting stepsize
      uniform float  skip,    //size of empty region
      //inside sphere to be skipped
      uniform float4 backgroundColor,
      uniform float earlyBackgroundTest,

      uniform float3x3 CamToWorld0,//transformation matrices of
      uniform float3x3 CamToWorld1,//sample cameras
      uniform float3x3 CamToWorld2,

      out float4 color : COLOR,
      out float oDepth : DEPTH)
{
  float breakTest     = 0.0;//control flag for early exit in for loop
  //determine size of for loop, use 2 vars because max loops is
  limited
  float maxIterationsI  = 1;
  float maxIterationsJ  = (2.0-skip)/stepsize;
  if(maxIterationsJ>200)
  {
    maxIterationsJ    = 200;
    maxIterationsI    = (((2.0-skip)/stepsize)/200)+1;
  }
  //fragment's position based on barycentric weight from sample
  vertices
  float3 v = c0 * colorIn.x  + c1 * colorIn.y + c2 * colorIn.z;
  float3 dir = v-P_eye;          //the view direction
  float3 dirN = normalize(dir); //normalized view direction

  //first position on ray along view dir for first pos.
  //skip some samples if skip parameter is set
  float3 firstSamplePos = v + skip * dirN;

  //determine intersection with inner sphere for initial skip
  //if no custom skip is given
  float skip;
  if(!skip>0.0)
  {
    float A,B,C;
    //inner sphere is of size 1.0/sqrt(2)
    C  = dot(P_eye,P_eye) - (1.0/sqrt(2));
    B  = 2.0 * dot(P_eye,dir);
    A = dot(dir,dir);

    float S = max( (B*B - 4.0 * A*C) ,0.0); //components under sqrt
    //first intersection with inner sphere neede, so take min
    skip = min( ((-B + sqrt(S)) /A/2.0), ((-B + sqrt(S)) /A/2.0));
    //take inner sphere intersection as start point
    firstSamplePos = P_eye + skip * dirN;
  }

  //current position on ray along view dir for first pos. skip some
  samples
  float3 vecG = firstSamplePos;

  //extract interpolation weights from interpoalted color
  //x,y,z correspond to c0,c1,c2, accordingly
  float3 cameraWeight;
  cameraWeight=colorIn;
  //initialize the 3 camera color values with background color
  float4 color0 = backgroundColor;
  float4 color1 = backgroundColor;
  float4 color2 = backgroundColor;

  //spherical intersection points of sample cameras
  float3 vecS0, vecS1, vecS2;
  // xyz components refer to camera 0, 1, and 2, respectively.
  float3 depthV;//depth sampled from texture

  //start raycasting in two for loops
  for(int i = 0; i < maxIterationsI; i++)
  {
    for(int j = 0; j < maxIterationsJ; ++j) {

      vecG += stepsize * dirN; //determine next sample position on ray
      //check if we are still in the inner sphere
      //discard pixel if no intersection within inner sphere
      if(skip>0.0 &&
        length(firstSamplePos-vecG)>(1-skip+1.0/sqrt(2)))
      {
        discard;
      }else if(length(firstSamplePos-vecG)>(1.414213562)){
        discard;
      }
      //determine dirs from sample cams to ray sample pos
      float3 d0G = vecG-c0;//vector from sample camera position
      float3 d1G = vecG-c1;//to current ray sample position
      float3 d2G = vecG-c2;
      d0G=normalize(d0G);//normalized dir to sample pos
      d1G=normalize(d1G);
      d2G=normalize(d2G);

      //determine ray - sphere intersection on opposite hemisphere
      //for each sample camera ray through current view ray sample pos
      float3 A,B,C;
      C.x  = dot(c0,c0) - 1.0;
      C.y  = dot(c1,c1) - 1.0;
      C.z  = dot(c2,c2) - 1.0;

      B.x  = 2.0 * dot(c0,d0G);
```

```
        B.y  = 2.0 * dot(c1,d1G);
        B.z  = 2.0 * dot(c2,d2G);

        A.x = dot(d0G,d0G);
        A.y = dot(d1G,d1G);
        A.z = dot(d2G,d2G);

        float3 S = max( (B*B - 4.0 * A*C) ,0..xxx);//component under
        sqrt
        //ignore negative solution - opposite hemisphere, only
        float3 lambda = (-B.xyz + sqrt(S.xyz)) /A.xyz/2.0;

        //Sphere intersections
        vecS0 = c0 + lambda.x * d0G;
        vecS1 = c1 + lambda.y * d1G;
        vecS2 = c2 + lambda.z * d2G;

        //direction to sphere interesection point
        float3 direction0 =  mul(CamToWorld0, vecS0 );
        float3 direction1 =  mul(CamToWorld1, vecS1 );
        float3 direction2 =  mul(CamToWorld2, vecS2 );

        //texture coordinates for sphere intersection
        //based on parabolic mapping
        float3 UV0;
        float3 UV1;
        float3 UV2;
        UV0.x = direction0.x / 2.0 / (1.0 - direction0.z) + 0.5;
        UV0.y = direction0.y / 2.0 / (1.0 - direction0.z) + 0.5;
        UV0.z = direction0.z;

        UV1.x = direction1.x / 2.0 / (1.0 - direction1.z) + 0.5;
        UV1.y = direction1.y / 2.0 / (1.0 - direction1.z) + 0.5;
        UV1.z = direction1.z;

        UV2.x = direction2.x / 2.0 / (1.0 - direction2.z) + 0.5;
        UV2.y = direction2.y / 2.0 / (1.0 - direction2.z) + 0.5;
        UV2.z = direction2.z;

        // obtain the color samples
        color0 = (UV0.z<0.0)?tex2D(DecalMap0,UV0.xy) : backgroundColor;
        color1 = (UV1.z<0.0)?tex2D(DecalMap1,UV1.xy) : backgroundColor;
        color2 = (UV2.z<0.0)?tex2D(DecalMap2,UV2.xy) : backgroundColor;

        // obtain the depth samples
        depthV.x = (UV0.z<0.0)?tex2D(DepthMap0,UV0.xy).r : 1.0;
        depthV.y = (UV1.z<0.0)?tex2D(DepthMap1,UV1.xy).r : 1.0;
        depthV.z = (UV2.z<0.0)?tex2D(DepthMap2,UV2.xy).r : 1.0;

        // vectors from a vertex (camera) to its corresponding
        // intersection point with the geometry
        float3 dirG0 = depthV.x * (vecS0-c0);
        float3 dirG1 = depthV.y * (vecS1-c1);
        float3 dirG2 = depthV.z * (vecS2-c2);

        // object intersection points for each camera (local estimates)
        float3 vecG0 = c0 + dirG0;
        float3 vecG1 = c1 + dirG1;
        float3 vecG2 = c2 + dirG2;

```

```
      //absolute distances of local estimates from view ray sample pos
      float3  dist;
      dist.x = length(vecG-vecG0);
      dist.y = length(vecG-vecG1);
      dist.z = length(vecG-vecG2);

      //determine minimal distance
      float minDist = min(dist.x,min(dist.y,dist.z));
      //if one local estimate wihin epsilon environment
      if ((minDist < epsilon)){
        //Check all cams if their local estimates are within error
        //threshold, if not set weight to 0
        if (dist.x > errorThreshold) {
          cameraWeights.x = 0.0;
          color0=backgroundColor;
        }
        if (dist.y > errorThreshold) {
          cameraWeights.y = 0.0;
          color1=backgroundColor;
        }
        if (dist.z > errorThreshold) {
          cameraWeights.z = 0.0;
          color2=backgroundColor;
        }
        //normaliez camera weights
        cameraWeights /= dot(cameraWeights,1..xxx);

        //stop raycasting
        breakTest = 1.0;
        break; //inner loop

      }
    }
    if (breakTest > 0.0)
      break; //outer loop
  }
  //discard if no intersection established
  if (breakTest<1.0)
  {
    discard;
  }
  //determine depth for fragment by projecting to virtual view
  float4 depthPoint = mul(ModelViewProj, float4(vecG,1.0));
  depthPoint/=depthPoint.w;
  oDepth=(depthPoint.z+1)/2.0;

  //Interpolate color from cameraWeights
  color = color0 * cameraWeights.x  +
        color1 * cameraWeights.y +
      color2 * cameraWeights.z;
}
```

Code Sample A.2: Raycasting light field rendering fragment program.

Bibliography

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In M. S. Landy and J. A. Movshon, editors, *Computational Models of Visual Processing*, pages 3–20. MIT Press, Cambridge, MA, 1991.

[2] T. Akenine-Möller and E. Haines. *Real-Time Rendering.* A K Peters, Ltd., Wellesley, MA, USA, 1999.

[3] Andujar, C., Boo, J., Brunet, P., Fairen, M., Navazo, I., Vazquez, P., Vinacua, and A. Omni-directional relief impostors. In *Computer Graphics Forum (Proc. Eurographics EG'07)*, pages 553–560, 2007.

[4] A.R.T. GmbH. The DTrack System. Novmeber 2008. http://www.ar-tracking.de/, last visited: 01.11.2008.

[5] I. Ashdown. Near-field photometry: A new approach. *Journal of the Illuminating Engineering Society*, 22(1):163–180, 1993.

[6] L. Baboud and X. Décoret. Rendering geometry with relief textures. In *GI '06: Proc. Graphics Interface 2006*, pages 195–201. Canadian Information Processing Society, 2006.

[7] P. A. Beardsley, P. H. S. Torr, and A. Zisserman. 3d model acquisition from extended image sequences. In *ECCV '96: Proceedings of the 4th European Conference on Computer Vision-Volume II*, pages 683–695, London, UK, 1996. Springer-Verlag.

[8] C. Beder, B. Bartczak, and R. Koch. A combined approach for estimating patchlets from pmd depth images and stereo intensity images. In C. B. Hamprecht, F. A.;Schnrr, editor, *29th Annual Symposium of the German Association for Pattern Recognition, DAGM 2007*, volume 4713, pages 11–20. Springer Berlin / Heidelberg, 2007.

[9] C. Beder, I. Schiller, and R. Koch. Real-time estimation of the camera path from a sequence of intrinsically calibrated pmd depth images. In *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume Vol. XXXVII. Part B3a, pages 45–50, Beijing, China, 2008. XXI. ISPRS Congress.

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

[11] J. F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics (Proceedings of SIGGRAPH 78)*, pages 286–292, Aug. 1978.

[12] C. Buehler, M. Bosse, L. McMillan, S.Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proc. ACM SIGGRAPH*, pages 425–432, 2001.

[13] E. Camahort. 4d light-field modeling and rendering. *Ph.D. Thesis, University of Texas at Austin*, 2001.

[14] E. Camahort and D. Fussel. A geometric study of light field representations. *Technical Report TR-99-35, Department of Computer Science, University of Texas*, 1999.

[15] E. Camahort, A. Lerios, and D. Fussell. Uniformly sampled light fields. Technical report, Austin, TX, USA, 1998.

[16] J.-X. Chai, X. Tong, S.-C. Chan, and H.-Y. Shum. Plenoptic sampling. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[17] B. Chen. Novel methods for manipulating and combining light fields. *Ph.D. Thesis, Stanford University*, 2006.

[18] S. Chen. QuickTime VR: An image-based approach to virtual environment navigation. In *Proc. ACM SIGGRAPH*, pages 29–38, 1995.

[19] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields. *ACM Trans. Graph.*, 21(3):447–456, 2002.

[20] A. T. Corporation. Flock of birds tracking system. *Ascension Technology Corporation Product Presentation*, November 2008. http://www.ascension-tech.com/realtime/RTflockofBIRDS.php, last visited: 15.11.2008.

[21] D. F. Dansereau and L. T. Bruton. Gradient-based depth estimation from 4d light fields. In *ISCAS (3)*, pages 549–552, 2004.

[22] P. Debevec, Y. Yu, and G. Boshokov. Efficient view-dependent image-based rendering with projective texture-mapping. Technical report, Berkeley, CA, USA, 1998.

[23] P. E. Debevec. *Modeling and Rendering Architecture from Photographs.* PhD thesis, University of California at Berkeley, Computer Science Division, Berkeley CA.

[24] P. E. Debevec. Image-based modeling and lighting. *SIGGRAPH Comput. Graph.*, 33(4):46–50, 2000.

[25] B. L. B. J. DeVore, Ronald A.; Jawerth. Image compression through wavelet transform coding. *IEEE Trans. Inform. Theory*, 38(2):719–746, 1992.

[26] M. Faraday. Thoughts on ray vibrations. *Philosophical Magazine*, XXVIII:3, May 1846.

[27] O. Faugeras, B. Hotz, H. Mathieu, T. Viéville, Z. Zhang, P. Fua, E. Théron, L. Moll, G. Berry, J. Vuillemin, P. Bertin, and C. Proy. Real time correlation based stereo: algorithm implementations and applications. Technical Report RR-2013, INRIA, 1993.

[28] U. Fecker, A. Guenegues, I. Scholz, and A. Kaup. Depth Map Compression for Unstructured Lumigraph Rendering. In J. Apostolopoulos and A. Said, editors, *Visual Communications and Image Processing 2006*, Proceedings of SPIE-IS&T Electronic Imaging, Bellingham, WA, 2006.

[29] G. Fekete. Rendering and managing spherical data with sphere quadtrees. In *VIS '90: Proceedings of the 1st conference on Visualization '90*, pages 176–186, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[30] P. C. Gasson. Geometry of spatial forms: Analysis, synthesis, concept formulation and space vision for cad. 1983.

[31] T. Georgiev, C. Zheng, S. K. Nayar, D. Salesin, B. Curless, and C. Intwala. Spatio-angular resolution trade-offs in integral photography. In *Proc. Eurographics Symposium on Rendering*, pages 263–272, 2006.

[32] A. Gershun. The light field. *J. Math. and Physics*, 18:51–151, 1939. trans. by P. Moon and G. Timoshenko.

[33] A. S. Glassner, editor. *An introduction to ray tracing.* Academic Press Ltd., London, UK, UK, 1989.

[34] J. S. Gondek, G. W. Meyer, and J. G. Newman. Wavelength dependent reflectance functions. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 213–220, New York, NY, USA, 1994. ACM.

[35] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proc. ACM SIGGRAPH*, pages 43–54, 1996.

[36] N. Greene. Applications of world projections. In *Proceedings on Graphics Interface '86/Vision Interface '86*, pages 108–114, Toronto, Ont., Canada, Canada, 1986. Canadian Information Processing Society.

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

[38] P. Haeberli and M. Segal. Texture mapping as a fundamental drawing primitive. In *Fourth Eurographics Workshop on Rendering*, pages 259–266. Eurographics, June 1993.

[39] W. Heidrich. *Modeling and Rendering Architecture from Photographs.* PhD thesis, University of Erlangen, Computer Graphics Group, Erlangen.

[40] W. Heidrich and H.-P. Seidel. View-independent environment maps. In *HWWS '98: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 39–ff., New York, NY, USA, 1998. ACM.

[41] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. J. V. Gool. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *DAGM-Symposium*, pages 94–101, 1999.

[42] T. Horz, A. Pritzkau, C. Rezk-Salama, S. Todt, and A. Kolb. Gaming Technology in Cultural Heritage Systems. In *Proc. 8th Int. Conf. on Intelligent Games and Simulation*, 2007.

[43] G. Iddan and G. Yahav. 3D imaging in the studio. In *Proc. SPIE*, volume 4298, page 48, 2000.

[44] I. Ihm, S. Park, and R. K. Lee. Rendering of spherical light fields. In *PG '97: Proceedings of the 5th Pacific Conference on Computer Graphics*

and Applications, page 59, Washington, DC, USA, 1997. IEEE Computer Society.

[45] K. Iourcha, K. Nayak, and Z. Hong. 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).

[46] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE MultiMedia*, 04(1):34–47, 1997.

[47] S. Kang. A Survey of Image-Based Rendering Techniques. In *VideoMetrices, SPIE*, volume 3641, pages 2–16. ACM Press, 1999.

[48] I. Karaseitanidis and A. Amditis. A nouvel acoustic tracking system for virtual reality systems. In D. Talaba and A. Amditis, editors, *Product Engineering: Tools and Methods Based on Virtual Reality*, pages 99–122. Springer Science and Business Media B.V., 2008.

[49] M. Keller and A. Kolb. Real-time simulation of time-of-flight sensors. *Simulation Modelling Practice and Theory*, page submitted, 2008.

[50] M. Keller, J. Orthmann, A. Kolb, and V. Peters. A Simulation Framework for Time-Of-Flight Sensors. In *Proc. of the Int. IEEE Symp. on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 125 – 128, 2007.

[51] A. Kolb, E. Barth, and R. Koch. ToF-Sensors: New Dimensions for Realism and Interactivity. In *Conf. on Computer Vision and Pattern Recognition (CVPR), Workshop on ToF Camera based Computer Vision (TOF-CV)*, 2008.

[52] R. Lange. *3D time-of-flight distance measurement with custom solid-state image sensors in CMOS/CCD-technology*. PhD thesis, University of Siegen, 2000.

[53] J. Lengyel. The convergence of graphics and vision. *Computer*, 31(7):46–53, 1998.

[54] R. K. Lenz and R. Y. Tsai. Calibrating a cartesian robot with eye-on-hand configuration independent of eye-to-hand relationship. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(9):916–928, 1989.

[55] R. Levin. Photometric characteristics of light controlling apparatus. *Illuminating Engineering*, 66(4):205–215, 1971.

[56] M. Levoy. Computational imaging in the sciences. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 7, New York, NY, USA, 2006. ACM.

[57] M. Levoy. Light fields and computational imaging. *Computer*, 39(8):46–55, 2006.

[58] M. Levoy. The stanford 3d scanning repository. *Stanford University*, October 2008. http://graphics.stanford.edu/data/3Dscanrep/, last visited: 09.10.2008.

[59] M. Levoy. The stanford multi-camera array. *Stanford University*, October 2008. http://graphics.stanford.edu/projects/array/, last visited: 07.10.2008.

[60] M. Levoy. The stanford spherical gantry. *Stanford University*, October 2008. http://graphics.stanford.edu/projects/gantry/, last visited: 07.10.2008.

[61] M. Levoy and P. Hanrahan. Light Field Rendering. In *Proc. ACM SIGGRAPH*, pages 31–42, 1996.

[62] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 131–144, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[63] R. R. Lewis. *Light-driven global illumination with a wavelet representation of light transport*. PhD thesis, Vancouver, BC, Canada, Canada, 1998.

[64] M. Lindner and A. Kolb. Lateral and Depth Calibration of PMD-Distance Sensors. In *Advances in Visual Computing*, volume 2, pages 524–533. Springer, 2006.

[65] M. Lindner and A. Kolb. Calibration of the intensity-related distance error of the PMD TOF-Camera. In *SPIE: Intelligent Robots and Computer Vision XXV*, volume 6764, pages 6764–35, 2007.

[66] M. Lindner and A. Kolb. Data-Fusion of PMD-Based Distance-Information and High-Resolution RGB-Images. In *Proc. of the Int. IEEE Symp. on Signals, Circuits & Systems (ISSCS)*, volume 1, pages 121 – 124, 2007.

[67] M. Lindner, A. Kolb, and T. Ringbeck. New Insights into the Calibration of TOF Sensors. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Workshop on ToF Camera based Computer Vision (TOF-CV)*, 2008.

[68] M. Lindner, M. Lambers, and A. Kolb. Data fusion and edge-enhanced distance refinement for 2D rgb and 3D range images. 2007.

[69] Y. Matsumoto, H. Terasaki, K. Sugimoto, and T. Arakawa. A portable three-dimensional digitizer. In *NRC '97: Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling*, page 197, Washington, DC, USA, 1997.

[70] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 369–374, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[71] L. McMillan and G. Bishop. Head-tracked stereoscopic display using image warping. In *Proc. ACM SIGGRAPH*, pages 39–46, 1995.

[72] L. McMillan and S. Gortler. Image-based rendering: A new interface between computer vision and computer graphics. *SIGGRAPH Comput. Graph.*, 33(4):61–64, 2000.

[73] G. Miller and C. Hoffman. Illumination and reflection maps: Simulated objects in simulated and real environments. In *ACM SIGGRAPH Course Notes for Advanced Computer Graphics Animation*, 1984.

[74] P. Moon and D. Spencer. The photic field. *MIT Press*, 1981.

[75] S. K. Nayar, M. Watanabe, and M. Noguchi. Real-time focus range sensor. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 995, Washington, DC, USA, 1995. IEEE Computer Society.

[76] J. Neider and T. Davis. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Release 1.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1993.

[77] C. Netramai, O. Melnychuk, J. Chanin, and H. Roth. Combining pmd and stereo camera for motion estimation of a mobile robot. In *The 17th IFAC World Congress July*, 2008. accepted.

[78] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. Light Field Photography with a Hand-Held Plenoptic Camera. Technical Report 02, Computer Science Tech Report CSTR, April 2005.

[79] Nvidia. Improve batching using texture atlases. White Paper, 2004. Nvidia Corporation, Santa Clara, CA, USA.

[80] T. Oggier, B. Bttgen, F. Lustenberger, G. Becker, B. Regg, and A. Hodac. Swissranger sr3000 and first experiences based on miniaturized 3D-ToF cameras. In *Proc. of the First Range Imaging Research Day at ETH Zurich*, 2005.

[81] M. M. Oliveira, G. Bishop, and D. McAllister. Relief texture mapping. In *SIGGRAPH '00: Proc. Conf. on Computer graphics and interactive techniques*, pages 359–368. ACM Press/Addison-Wesley Publishing Co., 2000.

[82] H. P. M. Paul S. Heckbert. Interpolation for polygon texture mapping and shading. *State of the Art in Computer Graphics: Visualization and Modeling*, pages 101–111, 1991.

[83] W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Kluwer Academic Publishers, Norwell, MA, USA, 1992.

[84] T. Pintaric and H. Kaufmann. A rigid-body target design methodology for optical pose-tracking systems. In *VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 73–76, New York, NY, USA, 2008. ACM.

[85] F. Policarpo, M. M. Oliveira, and a. L. D. C. Jȯ Real-time relief mapping on arbitrary polygonal surfaces. In *I3D '05: Proc. Symposium on Interactive 3D graphics and games*, pages 155–162. ACM, 2005.

[86] D. Porquet, J.-M. Dischler, and D. Ghazanfarpour. Real-time high-quality view-dependent texture mapping using per-pixel visibility. In *GRAPHITE '05: Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 213–220, New York, NY, USA, 2005. ACM.

[87] T. Prasad, K. Hartmann, W. Weihs, S. Ghobadi, and A. Sluiter. First steps in enhancing 3D vision technique using 2D/3D sensors. In V. Chum, O.Franc, editor, *11th Computer Vision Winter Workshop 2006*, pages 82–86, 2006.

[88] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. G. Shapiro, and W. Stuetzle. View-base rendering: Visualizing real objects from scanned range and color data. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 23–34, London, UK, 1997. Springer-Verlag.

[89] H. Rapp, M. Frank, F. Hamprecht, and B. Jhne. A theoretical and experimental investigation of the systematic errors and statistical uncertainties of time-of-flight cameras. 2007.

[90] W. Reeves, D. Salesin, and R. Cook. Rendering antialiased shadows with depth maps. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 283–291, New York, NY, USA, 1987. ACM Press.

[91] C. Rezk-Salama, S. Todt, and A. Kolb. Raycasting of Light Field Galleries from Volumetric Data. *Computer Graphics Forum*, 27(2):839–846, 2008.

[92] I. D. Rosenberg, P. L. Davidson, C. M. R. Muller, and J. Y. Han. Real-time stereo vision using semi-global matching on programmable graphics hardware. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 89, New York, NY, USA, 2006. ACM.

[93] H. Rushmeier, G. Taubin, and A. Guéziec. Applying shape from lighting variation to bump map capture. In *Eurographics Rendering Workshop 1997*, pages 35–44, June 1997.

[94] S. Rusinkiewicz, O. Hall-Holt, and M. Levoy. Real-time 3d model acquisition. *ACM Trans. Graph.*, 21(3):438–446, 2002.

[95] H. Schirmacher, W. Heidrich, and H.-P. Seidel. High-quality interactive lumigraph rendering through warping. In *Graphics Interface*, pages 87–94, 2000.

[96] H. Schirmacher, C. Vogelgsang, H.-P. Seidel, and G. Greiner. Efficient Free Form Light Field Rendering. In *Proc. Vision Modeling and Visualization*, pages 249–256, 2001.

[97] M. Segal, C. Korobkin, R. van Widenfelt, J. Foran, and P. Haeberli. Fast shadows and lighting effects using texture mapping. In *Proc. SIGGRAPH*, pages 249–252, 1992.

[98] S. Seitz and C. Dyer. View morphing. In *SIGGRAPH 96*, pages 21–30, 1996.

[99] J. M. Shapiro. Embedded image coding using zerotrees of wavelet coefficients. pages 124–141, 2001.

[100] J. R. Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In M. C. Lin and D. Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[101] H.-Y. Shum, S.-C. Chan, and S. B. Kang. *Image-Based Rendering*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[102] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 299–306, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[103] H.-Y. Shum and S. B. Kang. A review of image-based rendering techniques. In *IEEE/SPIE Visual Communications and Image Processing (VCIP)*, pages 2–13, Perth, Australia, June 2000. Institute of Electrical and Electronics Engineers, Inc.

[104] P.-P. Sloan, M. F. Cohen, and S. J. Gortler. Time critical lumigraph rendering. In *SI3D '97: Proc. of the 1997 symposium on Interactive 3D graphics*, pages 17–ff., New York, NY, USA, 1997. ACM.

[105] P.-P. Sloan and C. Hansen. Parallel lumigraph reconstruction. In *PVGS '99: Proceedings of the 1999 IEEE symposium on Parallel visualization and graphics*, pages 7–14, Washington, DC, USA, 1999. IEEE Computer Society.

[106] M. Stamminger and G. Drettakis. Perspective shadow maps. In *Proc. SIGGRAPH*, pages 557–562, 2002.

[107] M. Strengert, M. Kraus, and T. Ertl. Pyramid methods in GPU-based image processing. In *Proc. Vision, Modeling and Visualization (VMV)*, pages 169–176, 2006.

[108] K. H. Strobl and G. Hirzinger. Optimal Hand-Eye Calibration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4647–4653, Beijing, China, October 2006.

[109] K. H. Strobl, W. Sepp, S. Fuchs, C. Paredes, and K. Arbter. DLR CalDe and DLR CalLab.

[110] I. E. Sutherland. Sketchpad: A man-machine graphical communication system. In *AFIPS Spring Joint Computer Conference*, pages 329–346, Detroit, Michigan, USA, 1963.

[111] S. Todt, M. Langer, C. Rezk-Salama, A. Kolb, and K. Kuhnert. Spherical Light Field Rendering in Application for Analysis by Synthesis. *Int. J. on Intell. Systems and Techn. and App. (IJISTA), Issue on Dynamic 3D Imaging*, 2008.

[112] S. Todt, C. Rezk-Salama, L. Brückbauer, and A. Kolb. Progressive Light Field Rendering for Web Based Data Presentation. In *Proc. Workshop on Hyper-media 3D Internet*, page to appear, 2008.

[113] S. Todt, C. Rezk-Salama, T. Horz, A. Pritzkau, and A. Kolb. An Interactive Exploration of the Virtual Stronghold Dillenburg. In *Eurographics 2007 (Cultural Heritage)*, pages 17–24, 2007.

[114] S. Todt, C. Rezk-Salama, and A. Kolb. Real-Time Fusion of Depth and Light Field Images. ACM SIGGRAPH Posters, 2005.

[115] S. Todt, C. Rezk-Salama, and A. Kolb. Fast (Spherical) Light Field Rendering with Per-Pixel Depth. Technical report, 2007.

[116] S. Todt, C. Rezk-Salama, and A. Kolb. GPU-Based Spherical Light Field Rendering with Per-Fragment Depth Correction. *Computer Graphics Forum*, page to appear, 2008.

[117] S. Todt, C. Rezk-Salama, and A. Kolb. Light Field Rendering for Games. In *Proc. Theory and Practice of Computer Graphics*, pages 27–33, Best Student Paper Presentation Award, 2008.

[118] S. Todt, C. Rezk-Salama, and A. Kolb. Virtuelle Rekonstruktion und Exploration der Schlossanlage Dillenburg. In *Proc. Symposium Virtuelle Welten*, page to appear, 2008.

[119] R. Tsai and R. Lenz. Real time versatile robotics hand/eye calibration using 3d machine vision. pages xx–yy, 1988.

[120] R. Tsai and R. Lenz. A new technique for fully autonomous and efficient 3d robotics hand/eye calibration. 5:345–358, 1989.

[121] C. Vogelgsang and G. Greiner. Hardware accelerated light field rendering. Technical Report 11, IMMD 9, University Erlangen-Nürnberg, 1999.

[122] C. Vogelgsang and G. Greiner. Adaptive lumigraph rendering with depth maps. Technical Report 3, IMMD 9, University Erlangen-Nürnberg, 2000.

[123] C. Vogelgsang and G. Greiner. Ray-tracing in depth maps for image-based rendering. Technical report, IMMD 9, University Erlangen-Nürnberg, 2001.

[124] D. Voorhies and J. Foran. Reflection vector shading hardware. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 163–166, New York, NY, USA, 1994. ACM.

[125] J. Wang, X. Tong, J. Snyder, Y. Chen, B. Guo, and H.-Y. Shum. Capturing and rendering geometry details for btf-mapped surfaces. *The Visual Computer*, 21(8-10):559–568, 2005.

[126] L. Wang, X. Wang, X. Tong, S. Lin, S. Hu, B. Guo, and H.-Y. Shum. View-dependent displacement mapping. *ACM Trans. Graph.*, 22(3):334–339, 2003.

[127] B. Wilburn, N. Joshi, V. Vaish, E.-V. Talvala, E. Antunez, A. Barth, A. Adams, M. Horowitz, and M. Levoy. High performance imaging using large camera arrays. *ACM Trans. Graph.*, 24(3):765–776, 2005.

[128] B. Wilburn, M. Smulski, H.-H. K. Lee, and M. Horowitz. The light field video camera. In *Proc. of Media Processors, SPIE Electronic Imaging*, 2002.

[129] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. H. Salesin, and W. Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 287–296, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[130] K. Xu, K. W. Chia, and A. D. Cheok. Real-time camera tracking for marker-less and unprepared augmented reality environments. *Image Vision Comput.*, 26(5):673–689, 2008.

[131] Z. Xu, R. Schwarte, H. Heinol, B. Buxbaum, and T. Ringbeck. Smart pixel – photonic mixer device (PMD). In *Proc. Int. Conf. on Mechatron. & Machine Vision*, pages 259–264, 1998.

[132] G. Yahav, G. J. Iddan, and D. Mandelbaum. 3D imaging camera for gaming application. In *Digest of Technical Papers of International Conference on Consumer Electronics*, pages 1–2, 2007.

[133] J. C. Yang, M. Everett, C. Buehler, and L. McMillan. A real-time distributed light field camera. In *Rendering Techniques*, pages 77–86, 2002.

[134] A. Zandi, M. Boliek, E. L. Schwartz, and A. Keith. Compression with reversible embedded wavelets with an enhanced binary mode. In *ICASSP '96: Proc. of the Acoustics, Speech, and Signal Processing, 1996. on Conference Proceedings., 1996 IEEE International Conference*, pages 1962–1965, Washington, DC, USA, 1996. IEEE Computer Society.

[135] C. Zhang and T. Chen. A self-reconfigurable camera array. In *Proc. Rendering Techniques*, pages 243–254, 2004.

[136] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23:337–343, 1977.