Methods for transform, analysis and rendering of complete light representations

Lukas Ahrenberg

PhD Thesis

Dissertation zur Erlangung des Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
der Naturwissenschaftlich-Technischen Fakultät I
der Universität des Saarlandes

Eingereicht am 22. Mai 2007.

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken
Germany

Bibliografische Informationen der Deutschen Bibliothek Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.ddb.de abrufbar.

Typset in LaTeX.

Eingereicht am 22. Mai 2007 in Saarbrücken durch
Lukas Ahrenberg
MPI Informatik
Stuhlsatzenhausweg 85
66 123 Saarbrücken
ahrenberg@mpi-inf.mpg.de

Betreuender Hochschullehrer – Supervisor
Prof. Dr. Marcus A. Magnor, Technische Universität Braunschweig, Germany

Gutachter – Reviewers
Prof. Dr. Hans-Peter Seidel, Max-Planck-Institut für Informatik, Germany
Prof. Dr. Marcus A. Magnor, Technische Universität Braunschweig, Germany

Dekan – Dean
Prof. Dr. Thorsten Herfet, Universität des Saarlandes, Saarbrücken, Germany

Promovierter akademischer Mitarbeiter – Academic Member of the Faculty having a Doctorate
Dr. Bodo Rosenhahn, Max-Planck-Institut für Informatik, Germany

Datum des Kolloquiums – Date of Defense
16. Juli 2007 – July 16, 2007

Abstract

Recent advances in digital holography, optical engineering and computer graphics have opened up the possibility of full parallax, three dimensional displays. The premises of these rendering systems are however somewhat different from traditional imaging and video systems. Instead of rendering an image of the scene, the complete light distribution must be computed. In this thesis we discuss some different methods regarding processing and rendering of two well known full light representations: the light field and the hologram.

A light field transform approach, based on matrix optics operators, is introduced. Thereafter we discuss the relationship between the light field and the hologram representations. The final part of the thesis is concerned with hologram and wave field synthesis. We present two different methods. First, a GPU accelerated approach to rendering point-based models is introduced. Thereafter, we develop a Fourier rendering approach capable of generating angular spectra of triangular mesh models.

Kurzfassung

Aktuelle Fortschritte in den Bereichen der digitalen Holographie, optischen Technik und Computergrafik ermöglichen die Entwicklung von vollwertigen 3D-Displays. Diese Systeme sind allerdings auf Eingangsdaten angewiesen, die sich von denen traditioneller Videosysteme unterscheiden. Anstatt für die Visualisierung ein zweidimensionales Abbild einer Szene zu erstellen, muss die vollständige Verteilung des Lichts berechnet werden. In dieser Dissertation betrachten wir verschiedene Methoden, um dies für zwei verschiedene gebräuchliche Darstellungen der Lichtverteilung zu erreichen: Lichtfeld und Hologramm.

Wir stellen dafür zunächst eine Methode vor, die Operatoren der Strahlenoptik auf Lichtfelder anzuwenden, und diskutieren daraufhin, wie die Darstellung als Lichtfeld mit der Darstellung als Hologramm zusammenhängt. Abschliessend wird die praktische Berechnung von Hologrammen und Wellenfeldern behandelt, wobei wir zwei verschiedene Ansätze untersuchen. Im ersten Ansatz werden Wellenfelder aus punktbasierten Modellen von Objekten erzeugt, unter Einsatz moderner Grafikhardware zur Optimierung der Rechenzeit. Der zweite Ansatz, Fourier-Rendering, ermöglicht die Generierung von Hologrammen aus Oberflächen, die durch Dreiecksnetze beschrieben sind.

Summary

This dissertation presents a number of different contributions related to light field and computer generated holography (CGH) research. In short, the projects regard light field transformation, wave field analysis and two different methods for wave field and hologram rendering.

The overall motivation behind this thesis is an interest in complete light representations such as the light field and the hologram. These representations are capable of encoding the full near field of a scene. They could be used as rendering targets in true three dimensional display technologies, taking over the role that the image or video frame has today.

Below we summarize the different projects in this thesis.

Light field transforms The project focuses on adopting linear operators from ray optics in a light field framework. It is shown how propagation between planes, rotation, interfaces and thin lenses can be expressed using a matrix notation. This representation allows a chain of optical elements to be expressed as a multiplication of each operator matrix. By adopting a plane-slope representation, the light field can be propagated through the chain simply by a coordinate transform. It is shown how the matrix describing the transform can be seen as a change of the coordinate frame in ray space. Thus, this notation allows for very efficient light field transforms. We present a framework for wavelet compressing the light field and show how a hierarchical hexadeca-tree representation can be used to allow for fast rendering.

Wave field analysis In this project the motivation is to investigate the relationship between the light field and wave field representations. We discuss the principle of each representation, as well as the physical model of light. In the resulting analysis we argue that a conversion preserving the near field may not be possible without reconstructing scene depth information. We discuss briefly how this may be achieved using different methods. We also present a time-frequency approach which is exemplified by the short-term Fourier transform. This method approximates the wave field locally as a sum of planar waves, using the angular spectrum. An example is given using a wave field reconstructed from a phase-shift hologram.

GPU-based computer generated holography This project shows how holographic interference patterns can be generated from 3D point objects using programmable graphics hardware. We present an approach that renders the bipolar distribution of a wave field using a fragment shader that is

customly generated to deliver optimal performance. The motivation behind the project is to find an efficient way of implementing hologram generation on the GPU. We analyze the problem as well as earlier work in the field. The resulting program uses a tradeoff between multipass rendering and fragment shader load. Our program tailors a fragment shader in runtime to optimize the efficiency and take the limitations of current hardware into account. The resulting shader contains the code necessary to render the superposition of as many points as supported by the GPU in an unrolled loop, and is used in a multiple rendering algorithm. We have used our program to generate output directly on an experimental SLM-based display setup.

Fourier rendering for wave field synthesis The motivation behind this work is an idea of a new strategy for computer generated holograms from polygonal models. An efficient way of transporting wave fields between parallel planes is based on the angular spectrum. This method, however, requires transforming the wave field of each planar surface into the frequency domain. While previous approaches sampled the polygons and Fourier transformed the resulting 2D image, we compute the Fourier transform of a general triangle analytically. This has several advantages as the wave field is not sampled until it is propagated all the way to the hologram plane. Therefore, our technique does not suffer from the need to rotate and filter the Fourier coefficients like previous methods. We present the theory behind the approach and derive an analytic expression for the wave field of a general triangle assuming simplified material properties. We also present a proof of concept implementation, and resulting wave fields that can be used for holographic display.

Zusammenfassung

Thema der vorliegenden Dissertation sind Verfahren zur schnellen und realistischen Darstellung anhand von digitalen Hologrammen und Lichtfeldern. Die Schwerpunkte sind Transformation von Lichtfeldern, Analyse von Wellenfeldern und Methoden, um Wellenfelder sowie Hologramme in Echtzeit zu visualisieren.

Motiviert wurde die Arbeit von der Überlegung, daß Lichtfeld und Hologramm eine vollständige Darstellung des Lichts in einer Szene ermöglichen. Als solche könnten sie in zukünftigen vollwertig dreidimensionalen Darstellungstechnologien die Rolle übernehmen, die momentan dem zweidimensionalen Videobild zukommt.

Im folgenden fassen wir die verschiedenen Projekte kurz zusammen.

Transformation von Lichtfeldern Ziel des ersten Abschnitts ist es, die Wirkungsweise der linearen Operatoren der Strahlenoptik in den mathematischen Rahmen der Lichtfelder zu übertragen. Wir zeigen, wie Lichttransport zwischen Ebenen, Rotation, Materialübergänge und dünne Linsen mit Hilfe einer Matrixnotation dargestellt werden können. In dieser Darstellung kann eine Kette von Operationen durch einfache Matrixmultiplikation abgebildet werden. Ein Lichtfeld, welches auf einer Ebene in verschiedene Richtungen definiert ist, durchläuft eine Koordinatentransformation, wenn es einer solchen Transformationen ausgesetzt wird. Die Matrix, die die Transformation beschreibt, kann dann als Basiswechsel im Strahlenraum betrachtet werden. Auf diese Weise erlaubt unsere Notation eine sehr effiziente Behandlung von Lichtfeldern. Abschliessend stellen wir ein praktisches Konzept vor, wie durch eine Wavelet-Kompression des Lichtfeldes und hierarchische Darstellung in einem Baum ein schnelles Rendering ermöglicht wird.

Analyse von Wellenfeldern Der zweite Abschnitt widmet sich der Untersuchung von Zusammenhängen zwischen den Darstellungen als Lichtfeld oder Wellenfeld. Wir diskutieren dabei die zugrundeliegenden Prinzipien und Theorie des Lichts beider Darstellungen. Die anschliessende Untersuchung zeigt, daß im allgemeinen kein Wechsel zwischen beiden möglich ist, wenn das Nahfeld erhalten bleiben soll. Erst die Gewinnung zusätzlicher Information in Form von Tiefeninformation in der Szene macht einen solchen Übergang möglich, und wir untersuchen verschiedene Methoden, die dazu geeignet sind. Eine davon ist der Zugang über eine Fouriertransformation in kurzen Zeitfenstern, wobei das Wellenfeld lokal als Summe ebener Wellen dargestellt wird. Als Beispiel rekonstruieren wir das Wellenfeld eines phasenverschobenen Ho-

logramms.

GPU-basierte Hologrammsynthese Im dritten Abschnitt zeigen wir, wie holographische Interferenzmuster aus 3D-Punkten mit Hilfe von programmierbarer Grafikhardware erzeugt werden können. Ziel ist die effiziente Implementation der Hologrammsynthese auf einer GPU. Die Analyse des Problems sowie früherer Arbeiten auf dem Gebiet führt uns zu einem neuen Zugang, in dem wir eine optimierte Balance zwischen Multipass-Rendering und Berechung der Wellenüberlagerung ein einem einzelnen Shader herstellen.

Unser Algorithmus nutzt einen speziell angepassten Fragment-Shader, um die bipolare Verteilung eines Wellenfeldes zu erzeugen. Dieser Shader wird zur Laufzeit generiert und optimiert, um die Möglichkeiten der eingesetzten Hardware zu berücksichtigen. Dabei überlagert er die Wellen von so vielen Punkten, wie die GPU ohne die Verwendung von Schleifen ermöglicht. Weitere Überlagerungen werden durch zusätzliche Rendering-Durchgänge berechnet. Die praktische Verwendbarkeit des Systems wird auf einem experimentellen SLM-basierten Display gezeigt.

Fourier-Rendering zur Synthese von Wellenfeldern Der letzte Abschnitt entwickelt die Idee für eine neue Strategie, um Hologramme aus Polygonmodellen zu berechnen. Eine naheliegende Methode, das Wellenfeld zwischen parallelen Ebenen zu transportieren, führt über das Winkelspektrum, erfordert aber eine Fouriertransformation des Wellenfeldes jeder einzelnen Fläche. Frühere Zugänge führten über ein Sampling der einzelnen Polygone und anschließende Fouriertransformation des entstandenen 2D-Bildes. Im Gegensatz dazu berechnen wir die Fouriertransformation für ein allgemeines Dreieck analytisch und führen ein Sampling erst durch, wenn das Wellenfeld bis zur Hologrammebene propagiert wurde. Diese Methode hat verschiedene Vorteile, insbesondere können wir darauf verzichten, Fourierkoeffizienten zu rotieren und zu filtern.

In der theoretischen Untersuchung unseres Zugangs leiten wir eine analytische Darstellung des Wellenfeldes eines allgemeinen Dreiecks her, wobei wir ein vereinfachtes Materialmodell annehmen. Eine experimentelle Implementierung beweist die praktische Durchführbarkeit des Zugangs, die resultierenden Wellenfelder können direkt in holographischen Displays verwendet werden.

Preface

The contributions presented in this thesis are based on work I did as a researcher at the Max-Planck-Institut für Informatik in Saarbrücken, Germany, the years 2002 – 2007. My interest in numerical optics, holography and wave optics for computer graphics was spurred by my supervisor, Marcus Magnor. After I had been at the institute for some time, he handed me a few books on linear and Fourier optics and suggested that I have a look. In one way or another, several threads leading up to this dissertation sprung from the experiments and project that followed.

Most, but not all, of the contributions have been published in journals and conference proceedings. The publications have been integrated in this thesis in revised and extended form. *Light Field Rendering using Matrix Optics* [7] was presented at *WSCG 2006*. It is now part of Chapter 4. The discussion on a light field to hologram mapping in Chapter 5 is the result of a long process of understanding holograms in a computer graphics setting. It is related to the work in [127], which is the result of a collaboration with Remo Ziegler and his colleagues at ETH Zürich. It will be presented at *Eurographics 2007*. The new rendering challenges that are introduced by full light displays have been one of my main interests, and something I would really like to pursue also in the future. The point-based method presented in Chapter 6 was originally published in *Optics Express* [4]. It was while revising this approach that I had the idea of developing a completely new analytic method. This work lead to the contribution in Chapter 7. At the time of writing, the basics of Fourier rendering is also being prepared for publication.

During my stay at the MPI I also published some results in areas that do not lie directly within the scope of this thesis. *A Mobile System for Multi-Video Recording* [5] (*CVMP 2004*) and *External camera calibration for synchronized multi-video systems* [46] (*WSCG 2004*) are the results of successful collaborations with Ivo Ihrke concerning multi-video recording. Another joint project with Ivo resulted in the paper *Volumetric Reconstruction, Compression and Rendering of Natural Phenomena from Multi-Video*

Data [6] (*Volume Graphics 2005*).

Max-Planck-Institut für Informatik is a wonderful environment for a researcher, and I am very grateful for the chance to study there. My gratitude goes to my supervisor, Professor Marcus Magnor, who supported my research; providing resources and time, as well as patience with my temper. He also introduced me to numerical optics for which I am very grateful. I would also like to thank Professor Hans-Peter Seidel and Christian Theobalt from the AG4 for their support. Special thanks to Ellen Fries, secretary of the independent research groups, who probably have saved my life, or at least sanity, a couple of times during the years. I owe her for that, naturally.

In both the real world as well as in many virtual realms, Bastian Goldlücke has been a good friend ever since my very first day at the institute. He has proof-read my math and made me understand how much I had forgotten and how much I still have to learn. Ivo Ihrke has been an excellent source of numerical wisdom, beer, friendship and discussions. Ivo read early drafts of this thesis and have helped out in numerous ways.

Many thanks to Philip Benzie and Professor John Watson at the University of Aberdeen for showing me the practice of holography and for successful collaborations. They were also excellent hosts during my research visits. I would like to acknowledge Remo Ziegler of ETH Zürich for fruitful discussions as a part of our collaboration. Thomas Naughton and his team at NUIM invited me as a seminar guest speaker, and during the discussions that followed I had several moments of insight. I still have not had time to test all ideas that came to me that day. Ulf Assarsson at Computer Engineering, Chalmers University of Technology arranged so that I could visit the department while writing up parts of the thesis, for which I am very grateful.

Being at a place like the MPI allows one to make a lot of exceptional acquaintances, and all of life is not work. I have made a lot of good friends, too many to name them all here, but a huge thanks to all the people of NWG3 and AG4 whom I have gotten to know. Anyway, a small special greeting to the following people that have not yet been mentioned: Robert Bargmann has been an excellent friend, sharing many moments of joy and some of pain. He has been my partner for the occasional pint, squash and swimming. (He also has a car.) Timo Stich, old office mate, helped me out a lot during my stay in Braunschweig. He also receives a special acknowledgement, together with Kristian Hildebrand, for trying their best, but still loosing numerous times playing NHL 2005. On the other hand, Grzegorz Krawczyk beat me every time we played squash. I guess it made humble, at least for a while. Thanks! Andrei Lințu, always organized and helpful, has a hell of a fighting spirit and taught me a thing or two about being positive.

Staying away from home, in another country, is hard at times. No matter

the distance, or how many excellent friends and colleagues you have, homesickness still hits you from time to time. I have been lucky to have good friends and family at home, who have always been there for me, encouraging me. A big thanks to Daniel, Henrik, Ingemar, Lars and Jens–Petter of the *hiss* emailing list. They provided much needed breaks, feedback, news from home and gaming whenever I had the time to stop by. I owe so much to Anna for supporting my decision to pursue a Ph.D. at the MPI, and for encouraging and helping me. I have no words describe how much I appreciate it, so I guess "Thank you!" will have to do. The same goes for my family who accepted to see me all too seldom. Though, they did give me a really nice fishing rod for my birthday. Maybe now is time to try it out.

Lukas Ahrenberg,
May 2007

Contents

Chapter 1
Introduction

This thesis presents research related to transforms, analysis, rendering and synthesis of complete light distributions such as the light field and the hologram. Human society has a long history of two dimensional image synthesis, stretching from painting on a cave wall to rendering on a computer monitor. It is, however, only relatively recently that technology has taken the first steps towards true three-dimensional display systems. Such a system allows all viewers within its proximity to experience their own true image of the object. The perceived view is *full* or *complete* in such a sense that it carries all the visual attributes, such as parallax and depth, of an object. While the technical solutions for such displays are still in their early stages, we know that they will require a full representation of the light leaving an object, as well as novel rendering techniques.

Representations for full light distributions have been discussed in computer graphics for quite some time now, and in the field of optics even longer. In computer graphics light fields or lumigraphs are considered, while the optics community talks about holograms or wave fields. In this work we consider holographic rendering methods, but also analyze the hologram from a computer graphics perspective and show how to relate the hologram to a light field. We will however start by discussing light and the motivation behind this thesis.

1.1 Light and true three dimensional viewing

Light is what we usually call the part of the electro-magnetic spectrum that can be perceived using our eyes. For an object to be visible, light must travel from its surface to our eyes.

A digital camera can also be used to take a picture of an object. The light emitted from the scene is focused through the optics of the camera lens onto a light sensitive chip that registers color and intensity. Taking it one step further, inspired by the human visual system, one can try to retrieve information from the recorded picture. This is the science of *Computer Vision*. Likewise, in *Computer Graphics* methods are developed to generate synthetic pictures for display on a monitor. The monitor is made up of millions of small light emitting elements that can be set to a specific color and intensity. Combining computer vision and computer graphics so that images are digitally recorded, processed and redisplayed on a computer screen we have *Image-based Rendering*.

Cameras do not record the complete light of an object however, nor does a monitor display the complete light. A camera records a *picture* showing how the object looks from the position of the camera. Standard monitors do the same. If two persons standing well apart are looking at a monitor displaying a CG rendering of a coffee cup, both will see the same picture. While, if the cup would have been placed where the monitor stood it would look significantly different to both of them. Actually, it would look different to the left and right eyes of one person, that is how depth is perceived.

This is why it is very uncommon to try to reach into a photograph of a fruit basket and grab an apple for a bite. What we see is the light from colors on a plane, and if no special concern is taken to fool the human visual system we know it for a picture.

But, what if the complete light could be recorded and played back? In that case, all light leaving said fruit basket, traveling in all directions would be saved. Later light of identical structure could be re-emitted, and the eye would perceive it as a solid, three-dimensional object. It would be much more convincing to the visual system, and one would have to rely on the other senses to perceive the illusion, e.g. by reaching out and trying to touch the scene.

There are of course stereo displays and other technical solutions geared towards giving a personal three-dimensional experience. However, a true three-dimensional display is independent of the kind of instrument used for viewing, or the amount of users. It emits the full light from a scene and acts as a window to a virtual world.

The window analogy is a useful one when describing the difference to traditional display and recording techniques. When viewing something on a computer display it is like watching a painting or a photograph. The three dimensional world is projected onto a two dimensional plane before reaching our eyes and the depth cues are lost. Watching a true three-dimensional display will be like looking out through a window. All light from the out-

side world passing through the window is reaching the eyes, thus depth is perceived.

1.2 Motivation

Over the last few years promising digital technologies, showing some of the features needed to construct complete three dimensional displays, have arisen. Modern digital holography, boosted by the development of high resolution Charge Coupled Devices (CCDs) and Spatial Light Modulators (SLMs), has been developed by optical engineers during the last decade. Meanwhile, light field recording and rendering has been developed in the computer graphics community, and are now supported by autostereoscopic display systems.

In this thesis we will present research in the area of both technologies. We will also show how the representations are related and how the information contained in a hologram can be interpreted as a light field under certain assumptions.

While the majority of technology today is of rather low resolution and performance it obeys the basic principles of a complete light system. The display is still a two dimensional surface, but the representations implicitly encode depth information which is reconstructed during playback. Thus it acts as if the glass in the window analogy above had a "memory" and could store and playback the light passing through it. The scene is not projected to this surface, instead the visual information is encoded.

Such display technology clearly requires new methods for analysis and rendering. These challenges are our main motivation behind the contributions presented in the different Chapters of this thesis. It is also worth noting that although true 3D display technologies will be of great use in research, medicine and the industry, wave front synthesis may also be used for other applications than rendering. Wave fields are for instance used to model optical tweezers. This is a holographic construction that can be used to affect the momentum of very small particles and thus move them around. This is very useful for microscopy and micro structure engineering.

1.3 Contributions

While this thesis only offers some contributions in the form of wave field analysis and synthesis, we hope that the reader will find them valuable. It is clear that several hurdles remain before systems for true 3D display are

commonly available, but we are convinced that they can be overcome and that one strategy is the study of wave field rendering.

The main contributions in this thesis can be summarized as

- An adoption of ray optics operators to a light field framework
- A discussion on the relationship between the light field and the hologram
- An method for approximate conversion between wave field and light field.
- A strategy for acceleration of computer generation using programmable graphics hardware.
- A novel method for computer generated holography from triangular meshes.

1.4 Outline

The work in this thesis regards several aspects of holography and light field processing that together bridges the fields of computer graphics and optics.

The next chapter introduces the reader to the basics of geometric and wave optics, the theory of holography and light field rendering, as well as the principles of an experimental holographic system. General scientific contributions and literature in the fields of computer graphics and digital holography are also reviewed. Chapter 3 on the other hand discusses work more directly related to the contributions of this thesis.

In Chapter 4 we present a method for approximating a set of optical elements for light field rendering as 4×4 matrix operators. This allows for fast transformations of light fields as viewed through a chain of optical elements.

Following that, Chapter 5 investigates the relationship between wave field and light field. We discuss the basics of both representations and show their differences. We also introduce the angular spectrum based on Fourier optics and show how to approximate a hologram in the form of a light field.

Chapter 6 regards accelerating the computation of digital holograms through the use of graphics processors. We introduce a point-based method that generates a custom shader in run time to suite the current rendering architecture. This also allows it to render larger models than some of the previous work.

A novel method for wave field synthesis is presented in Chapter 7. We derive an analytic solution for the angular spectrum of a general triangle. We

also show how this result can be used to render a wave field for triangular mesh models. This method does not require a per triangle Fourier transform as previous methods did, neither is interpolation of the angular spectrum needed.

Finally, Chapter 8 concludes the work. We summarize the contributions in the thesis and discuss possible venues for future research.

Chapter 2
Background

This chapter will explain the basic concepts and terminology used in this dissertation.

We arranged this chapter so that the first sections are fairly general, giving a brief introduction to basic concepts, while the later ones discuss terminology and work directly related to the scientific contribution of this thesis. It is of course impossible to give a full introduction to optics and holography on these few pages. We have tried to provide references to textbooks and other sources on information whenever available.

Section 2.1 introduces the reader to the geometric and wave models of light. Section 2.2 thereafter, discusses how to perceive or record an image. Section 2.3 builds on this to introduce the light field and the hologram as two strategies to overcome the limitations of normal photography. Section 2.4 finally, turns the problem around by discussing how to display a three dimensional image. That section also presents the experimental setup we have used in our work on computer generated holography. The chapter is concluded with a brief summary.

2.1 Models of light

The nature of light is not something that is easily explained. There are however several different models to describe its behavior. Techniques described in this text will either be using the ray or the wave model of light.

In computer graphics the classic *geometric model* is most commonly used, as features found in standard CG models tend to be much larger then the wavelength of light and do not give rise to any noticeable degree of diffraction. When it comes to hologram rendering however, diffraction plays an important

role, thus for some of the work presented in this thesis we will also be using the *wave model of light*.

We recommend readers looking for a complete introduction to the nature of light to consult the excellent book *Principles of Optics* by Born and Wolf [11]. However, as the following chapters will work with both the geometric and wave models of light a short introduction is in place.

2.1.1 The geometric model of light

The geometric model describes light as rays traveling from a light source through space. The model is therefore also commonly referred to as the *ray model* of light. It has roots at least as far back as the ancient Greek culture. In the centuries B.C. Euclid studied the properties of light in his *Optica*, and proposed that light travels in straight lines.

When we refer to a *light ray* in this thesis we will usually mean a construction that has an origin (the *light source*) and a direction of propagation. The power of the ray, the *radiance*, is assumed to be constant along the ray. Thus a ray can be described by the following attributes

$$\{\mathbf{p}, \mathbf{v}, L\}. \tag{2.1}$$

Where $\mathbf{p}$ and $\mathbf{v}$ are vectors in $\mathbb{R}^3$ and denotes ray origin and direction respectively. L is the light radiance, which is a scalar constant along the ray. In most general discussions, monochromatic light will be considered for simplicity. If color images are desired, the methods can be generalized in accordance with standard computer graphics principles, where the color intensity is considered as a coordinate in RGB-space.

In Chapters 4 and 5, rays without known origin are considered. These are parameterized as intersecting a reference plane $\Pi \subset \mathbb{R}^3$ defined as

$$\Pi = (\mathbf{o}_\Pi, \{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}), \tag{2.2}$$

where $\mathbf{o}_\Pi$ is a point in Π and $\{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}$ spans a base in Π. We will denote the normal to the plane $\mathbf{n}_\Pi$. Assuming that not two rays intersect the plane in the same point from the same direction a ray can now be described by

$$\{\mathbf{x}, \mathbf{d}, L\}. \tag{2.3}$$

In the above equation, $\mathbf{x} \in \mathbb{R}^2$ and $\mathbf{d} \in \mathbb{R}^2$ denotes the ray intersection with Π, respective the ray direction components along the basis vectors of Π.

To view light propagation as happening along rays has many advantages. To start with it is intuitive. Light travels in a straight line between two points:

"Nature always acts by the shortest course" (Pierre de Fermat, 1601–1665). In addition, interaction with the light through lenses and other optical instruments can be solved by geometric constructions, making it computationally effective in many situations. Due to the vector nature and the possibility of linear operations, geometrical optics is the model of choice for many numerical applications. It is by far the most commonly used light model in computer graphics and is valid in most situations where the size of the object interacting with the light is much bigger than a wavelength.

There is however one important characteristic of light that can not be easily described by the geometric model: diffraction. This phenomena occurs when the light passes an obstacle the size of a wavelength, and was one of the main reasons for alternative models of light to be developed.

2.1.2 The wave model of light

The wave theory of light was originally developed as an alternative to the geometric model during the 17th century. Some of its earliest advocates and founders were Robert Hooke and Christian Huygens. The reason for looking for alternatives to the already accepted model was that geometric optics could not explain all observed properties of light at that time. One of the most noteworthy characteristics, and the one interesting for holography and the work in this thesis is diffraction.

However, before we start to address this phenomenon, we will introduce the basics of the wave model.

Basic principles

Light is generally an electromagnetic wave, and can thus be described by Maxwell's equations. In the cases where the wave properties of light are applied in this dissertation however, a simpler scalar wave model is sufficient. We will not perform the full derivation of this model, but will only briefly discuss some steps in order to identify some important properties. A full derivation as well as valuable discussions can be found in *Principles of Optics* by Born and Wolf [11]. In the discussion below we will follow the argumentation of Kreis [53].

Propagation of light in vacuum is described by the wave equation derived from the Maxwell equations.

$$\nabla^2 \mathbf{E} - \frac{1}{c_0^2} \frac{\partial^2 \mathbf{E}}{\partial t^2} = 0. \tag{2.4}$$

In the above equation $\mathbf{E}$ is the electric field strength, c_0 is the speed of light in vacuum and t is the time variable. ∇^2 denotes the Laplacian operator.

The general light wave is a transverse wave, meaning that it oscillates in all directions orthogonal to the direction of propagation. The different directions of vibration are called polarizations and are the reason that the electric field strength is a vector quantity. For the purposes of the methods in this thesis it is however safe to assume that the light is plane polarized. That is, it only oscillates in one plane, and thus Eq. 2.4 reduces to the scalar wave equation

$$\frac{\partial^2 E}{\partial z^2} - \frac{1}{c_0^2}\frac{\partial^2 E}{\partial t^2}, \tag{2.5}$$

assuming that the wave propagates along the z direction. Due to the linearity of Eq. 2.5 the sum of two solutions is also a solution. We will call this the *superposition principle* and it is an important property for many of the methods presented in this thesis.

There are of course several solutions to the scalar wave equation. For the applications in this dissertation we will however consider only plane waves and spherical waves. We will start with the planar wave front which has constant phase in planes perpendicular to the light propagation direction. To describe these waves we start with the harmonic solution to Eq. 2.5 under monochromatic light of wavelength λ

$$E(z,t) = E_0 \cos(\frac{2\pi}{\lambda}z - \omega t). \tag{2.6}$$

Assuming E has maximum phase at $(z = 0, t = 0)$ in the above equation, E_0 is the real amplitude of the wave, while $\omega = \frac{2\pi c_0}{\lambda}$ is called the angular frequency. It is worth noting that the term $\frac{2\pi}{\lambda}$ is called the wave number, and is often shorthanded as k in some physics literature.

A useful simplification here is to use Euler's formula and introduce a complex wave. Thus, Eq. 2.6 is written as

$$E(z,t) = \frac{1}{2}E_0 \exp(i(\frac{2\pi}{\lambda}z - \omega t)) + \frac{1}{2}E_0 \exp(-i(\frac{2\pi}{\lambda}z - \omega t)). \tag{2.7}$$

In reality it is only the real part of a complex $E(z,t)$ that has any physical interpretation. If we keep this in mind we can remove the second part of Eq. 2.7 and have

$$E(z,t) = \frac{1}{2}E_0 \exp(i(\frac{2\pi}{\lambda}z - \omega t)). \tag{2.8}$$

Equation 2.8 describes a planar wave front propagating in the direction of the z axis. A planar wave front propagating in a general direction can be

described by introducing the wave vector

$$\mathbf{k} = \frac{2\pi}{\lambda} [d_x, d_y, d_z]^T . \tag{2.9}$$

The vector $[d_x, d_y, d_z]$ denotes the direction vector of the light in the world coordinate frame, and $\| [d_x, d_y, d_z] \| = 1$ so that the elements make up the components projected onto the **xyz** basis vectors. Finally

$$\mathbf{r} = [x, y, z]^T \tag{2.10}$$

denotes the spatial position. We can thus write a planar wave front traveling in the direction $\mathbf{k}$ as

$$E(\mathbf{r}) = A_0 \exp(i(\mathbf{k} \cdot \mathbf{r})) \tag{2.11}$$

where A_0 is called the complex amplitude.

The spherical wave front has constant phase on equiradial distances from the light source. Such a wave must then satisfy the following scalar wave equation:

$$\frac{1}{r}\frac{\partial^2}{\partial r^2}(rE) - \frac{1}{c^2}\frac{\partial^2 E}{\partial t^2} = 0. \tag{2.12}$$

Using the harmonic spherical solution to this equation, and employing the simplifications analog to those discussed above, the equation for a spherical wave front is:

$$E(r) = \frac{A_0}{r} \exp(\frac{2\pi i}{\lambda} r). \tag{2.13}$$

Concluding, we have two simplified expressions that can be used to model light sources in the wave model. The planar wave front as presented in Eq. 2.11 and the spherical wave front, Eq. 2.13. Intuitively, the spherical wave front is a point light source, irradiating equally light in all directions. The planar wave front on the other hand can be thought of as originating from a light source at an infinite distance. As the phase is constant on planes, this means that the light does not spread out, which corresponds to parallel rays in the geometric model.

Finally, due to the assumptions noted in the above derivations, the type of light dealt with has the following properties:

- monochromatic and coherent
- plane polarized
- propagation in vacuum
- maximum amplitude at source

- physical interpretation only for real part

From the first two points of the assumptions it is clear that we are dealing with a laser type of light. The other assumptions are also worth keeping in mind, as they are valid in an ideal situation and may or may not be met in real experiments. However, for most cases the numerical methods based on these assumptions will function well also in real world display situations.

Theory of diffraction

One of the main reasons for using waves to describe light propagation is that it allows for diffraction. This is a phenomenon that occurs when waves get disrupted by objects in their path. The effect occurs for all types of waves, although in this dissertation we will only use the fact that light can get visibly diffracted. The observed effect of diffraction is that the waves get spread out at sharp edges of the obscuring object. Some everyday examples include water waves that fan out around rocks and other obstacles, sound waves that diffract at the edges of a doorway allowing us to hear a conversation although we are not in the room, and the rainbow patterns reflected from a compact disc as light diffracts in the small gratings. Although all object edges diffract light, visible effects usually only occur when the size of the diffracting object is at the order of a wavelength. This is due to the fact that the spread of the diffracted light is inversely proportional to the size of the object.

The first scientific mentioning of diffraction is by F.M. Grimaldi in 1665. It was also an effect of diffraction, the presence of light in a geometric shadow that led Hooke to propose that light may propagate as a wave. Diffraction is the phenomenon that occurs when a wave front passes an obstacle, or through an opening of the size near the order of a wavelength. This can be seen in Figure 2.1, where a wave travels from left to right through two small apertures. Modeling the light propagation as waves, the light gets distorted when passing the obstacle. In the case described in Figure 2.1 the small slits will effect the wave front in such a way that the apertures themselves can be thought of as two new light sources. This means that instead of the original source on the left, emitting one wave front, we have two wave fronts to the right of the barrier. When these waves hit a diffuser screen or a CCD to the right of the obstacle they interfere with each other, and their amplitudes are added together. Thus regions of different amplitude can attenuate or cancel each other out, creating the interference pattern that can be observed in real world experiments of this kind.

The geometric model of light will not yield this result. For the setup described in Figure 2.1, a simple construction will yield two pencils of rays

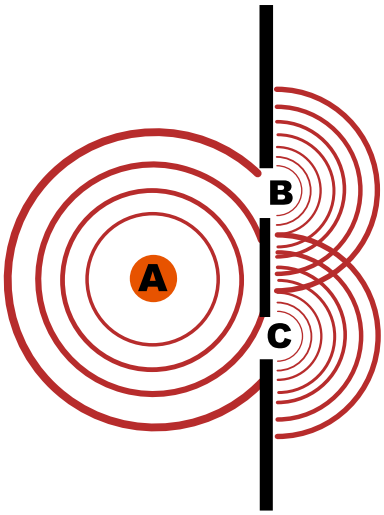


Figure 2.1: Young's experiment; A classic diffraction example. A point light source in A emits light in all directions. The light passes through two small slits, but is otherwise obscured by an opaque obstacle. Treating the light as waves will then have the effect that each slits can be viewed as the source of two new light waves, B and C. Measuring the light intensity in some plane at the right of the obstacle yields an image based on the interference between these two light waves. This model fits well with results from practical experiments, where such patterns are observed.

originating from A, one through each aperture. According to this model, an observer on the right side of the obstacle should just notice two bright regions; the rest would be in shadow. This contradicts the interference pattern observed in reality and was also what led Hooke to advocate the wave model which could explain the phenomenon.

Huygens' principle A good way of explaining diffraction is using *Huygens' principle*. Named after the Dutch 17th century scientist Christian Huygens this principle of wave propagation states that: *Each point on an advancing wave front can be considered a source of a secondary spherical wave. The value of the wave front at some later stage can be written as the superposition of the secondary waves.* Figure 2.2 illustrates this principle. Diffraction can thus be thought of as canceling out some of these secondary waves in the region of the obstacle, and affecting the resulting wave front. This is exactly what happens in Figure 2.1, where the incoming wave front is canceled out in all but two narrow slits. Huygens' principle states that the light passing

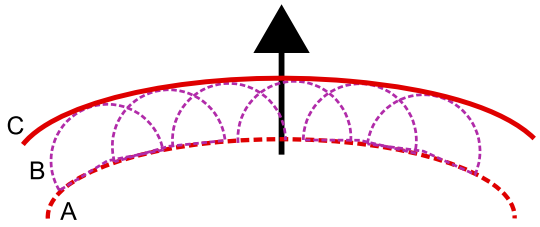


Figure 2.2: Illustrating Huygens' principle: A wave front at A can be described as the source of a set of new spherical waves, B. The wave front at some later position, C, can be described as the envelope of all the spherical waves.

through these slits will act as two new light sources emitting spherical waves. Thus the interference pattern observed to the right of the screen.

Rayleigh-Sommerfeld diffraction The diffracted wave front formed by some distribution in a specific plane can be more formally described using the Rayleigh-Sommerfeld integral

$$U(\mathbf{p}) = -\frac{1}{\lambda}\iint_{\Pi} A(\mathbf{s})\frac{\exp\left(\frac{2\pi i}{\lambda}\|\mathbf{p}-\mathbf{s}\|\right)}{\|\mathbf{p}-\mathbf{s}\|}\frac{(\mathbf{p}-\mathbf{s})\cdot\mathbf{n}}{\|\mathbf{p}-\mathbf{s}\|}\,d\mathbf{s}. \tag{2.14}$$

Where A is the known complex amplitude in some plane $\Pi \in \mathbb{R}^3$, $\mathbf{p} \in \mathbb{R}^3$ is assumed to lie in the positive half-space of Π, $\mathbf{n} \in \mathbb{R}^3$ is the positive normal to Π. The integral expresses the value of the wave field U in the point $\mathbf{p}$ as resulting from diffraction at the plane Π. The scenario is shown in Figure 2.3.

Intuitively, the Rayleigh-Sommerfeld integral reminds of the Huygens-Fresnel principle. The term $A(\mathbf{s})\frac{\exp\left(\frac{2\pi i}{\lambda}\|\mathbf{p}-\mathbf{s}\|\right)}{\|\mathbf{p}-\mathbf{s}\|}$ can be considered a spherical wave, and we are integrating over a set of complex amplitudes. The integral does not directly translate to the principle however, even though it explains the effects. Instead the Rayleigh-Sommerfeld integral should be considered a formal description of diffraction theory. Its derivation is outside the scope of this dissertation, but is worth studying together with the other major theory called Kirchhoff diffraction. Both are thoroughly described in [11] and [36].

We will use Rayleigh-Sommerfeld diffraction as the main formal description of diffraction, and our main methods will originate from this description.

The Fresnel approximation The full Rayleigh-Sommerfeld integral can be costly to compute, and approximate methods have been suggested. One of the most used is the so called Fresnel approximation, which has been extensively used in previous computer generated holography and numerical

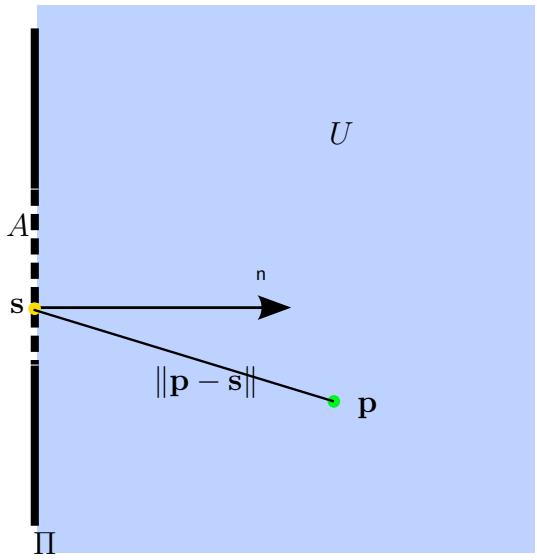


Figure 2.3: Light gets diffracted in the plane Π by some occluder. The complex wave amplitude in Π is described by A. $\mathbf{n}$ is normal to the plane. The resulting wave front in a point $\mathbf{p}$ can be calculated using the Rayleigh-Sommerfeld integral given in Eq. 2.14.

optics research. In this dissertation we base our CGH methods directly on the Rayleigh-Sommerfeld integral, but use the Fresnel approximation when discussing holograms and light fields in Chapter 5.

Starting from Eq. 2.14, we assume the plane Π to be the plane at $z = 0$ with the normal $\mathbf{n} = [0, 0, 1]^T$. Writing $\mathbf{p} = [x_p, y_p, z_p]^T$ and $\mathbf{s} = [x_s, y_s, 0]^T$ we have

$$r = \|\mathbf{p} - \mathbf{s}\| = \sqrt{(x_p - x_s) + (y_p - y_s) + z_p}. \tag{2.15}$$

For a finite aperture, when $z_p \gg \sqrt{(x_p - x_s)^2 + (y_p - y_s)^2}$, we may approximate r in the denominator by $r \approx z_p$. Inserting this approximation and Eq. 2.15 into Eq. 2.14 gives us

$$\begin{aligned} U(x_p, y_p, z_p) &= -\frac{1}{\lambda} \iint_{\Pi} A(x_s, y_s) \frac{\exp\left(\frac{2\pi i}{\lambda} r\right)}{z_p^2} (\mathbf{p} - \mathbf{s}) \cdot \mathbf{n} \, dx_s dy_s \\ &= -\frac{1}{\lambda z_p} \iint_{\Pi} A(x_s, y_s) \exp\left(\frac{2\pi i}{\lambda} r\right) dx_s dy_s. \end{aligned} \tag{2.16}$$

Above we use the fact that $(\mathbf{p}-\mathbf{s})\cdot\mathbf{n} = \left[(x_p - x_s)^2, (y_p - y_s)^2, z_p^2\right] \cdot [0, 0, 1]^T = z_p$ to simplify the equation.

Note that we can not use $r \approx z_p$ in the exponent however. This is where the Fresnel approximation comes into play. It is based on the binomial

expansion of Eq. 2.15, assuming that it can be adequately approximated by the two first terms

$$r = \sqrt{(x_p - x_s)^2 + (y_p - y_s)^2 + z_p^2} = z_p \sqrt{(\frac{x_p - x_s}{z_p})^2 + (\frac{y_p - y_s}{z_p})^2 + 1}$$
$$\approx z_p (1 + \frac{1}{2}(\frac{x_p - x_s}{z_p})^2 + \frac{1}{2}(\frac{y_p - y_s}{z_p})^2). \tag{2.17}$$

The formal condition for the approximation to be valid is according to [36]

$$z_p^3 \gg \max(\frac{\pi}{4\lambda}((x_s - x_p)^2 + (y_s - y_p)^2)^2). \tag{2.18}$$

Inserting Eq. 2.17 in (2.16) results in the Fresnel approximation

$$\begin{aligned} U(x_p, y_p, z_p) = &-\frac{1}{\lambda z_p} \iint_{\Pi} A(x_s, y_s) \\ &\exp(\frac{2\pi i}{\lambda} z_p (1 + \frac{1}{2}(\frac{x_p - x_s}{z_p})^2 + \frac{1}{2}(\frac{y_p - y_s}{z_p})^2))\, dx_s dy_s \\ = &-\frac{\exp(\frac{2\pi i}{\lambda} z_p)}{\lambda z_p} \iint_{\Pi} A(x_s, y_s) \\ &\exp(\frac{\pi i}{\lambda z_p}((x_p - x_s)^2 + (y_p - y_s)^2))\, dx_s dy_s. \end{aligned} \tag{2.19}$$

Interference

The interaction of two wave fronts through superposition is called interference and, as discussed above, the superposition can be regarded as a new wave front. Interference is one of the corner stones of holography, and will be used when this technique is discussed below. We will therefore briefly present the basic theory of interference in order to see what happens to the amplitude and phase of two interfering wave fronts. In doing so we will follow the example given in [99].

Consider two monochromatic waves of the same wavelength and polarization

$$U_1 = A_1 \exp(i\phi_1) \tag{2.20}$$

$$U_2 = A_2 \exp(i\phi_2). \tag{2.21}$$

The superposition is simply the sum of the wave fronts

$$W = U_1 + U_2. \tag{2.22}$$

However, calculating the magnitude of W we see that

$$\begin{aligned} \|W\|^2 &= \|U_1 + U_2\|^2 = (U_1 + U_2)(U_1 + U_2)^* \\ &= A_1^2 + A_2^2 + 2A_1A_2\cos(\phi_1 - \phi_2) \end{aligned} \tag{2.23}$$

where * denotes the complex conjugate. Thus, the magnitude of the superpositioned waves is the sum of the magnitudes of the individual waves, plus the so called interference term $2A_1A_2\cos(\phi_1 - \phi_2)$. This term has a maximum when the phase difference is

$$\phi_1 - \phi_2 = 2n\pi \text{ for } n = 0, 1, 2... \tag{2.24}$$

and a minimum at

$$\phi_1 - \phi_2 = (2n + 1)\pi \text{ for } n = 0, 1, 2... \tag{2.25}$$

These cases are called constructive and destructive interference respectively, and as can be seen from Eq. 2.23 they either amplify or cancel out the magnitude.

The important observation here is that the magnitude of two interfering wave fronts is dependent on the phase difference. A single wave, as in Eqs. 2.20 or 2.21 is not. It is this phase difference in the interference term that leads to the fringe patterns sometimes observed when two laser sources overlap, or when two ripples in a pond meet.

A valid question to ask in this situation is why we do not observe interference patterns in our daily lives. After all, light is all around us and must surely interfere all the time. In fact it does, but the effects are smoothed out because most light sources emit light in a fairly broad range of the spectrum. This kind of light contains a band of different wavelengths and is called incoherent. Coherence is a measure of how well two light waves interfere, and for instance laser light, containing only a single wavelength exhibits what is called temporal coherence. In this dissertation coherent light is assumed for the holographic experiments unless something else is specified. For further reading on coherence we refer to [11] or the sections on interferometers in [53] and [99].

A word on speckle

Image speckle is a general problem when using coherent light sources, and are often heard as one of the main reasons for not using holographic techniques. In short, speckle is an interference pattern that occurs due to random variations in the object surface. It is visible when using coherent light to illuminate diffuse surfaces.

Each surface element on an object with diffuse material properties reflects light in all directions. We can see this as if each element is the source of a diffracted wave. Each wave will have a random phase due to the diffuse material properties and thus create an interference pattern which is observed as a speckle noise over the image.

The speckle pattern produced is almost independent of the surface structure, but the intensity over the image follows the negative exponential probability distribution [42, 99].

As described in the above section, interference is generally only visible when using coherent light, and then so is speckle noise. However, coherent light is a prerequisite for interference which in turn is the foundation for holographic recording as we will see in Section 2.3.1. Thus, speckle is an inherit problem of holography and other coherent lighting. Figure 2.4 shows a numerical reconstruction of a phase shift hologram. Note the speckle pattern over the image.

Figure 2.4: Numerical reconstruction of a digital phase shift hologram depicting a chess knight. Distance between model and hologram plane was $0.46\ m$ **and the hologram has a size of** 512×512 **pixels. Each pixel has a size of** $9\ \mu$**m. Note the speckle pattern as the irregular structure over the hologram.**

The issue of speckle reduction is outside the scope of this dissertation, however the theme is an active research area and several methods on speckle

reduction has been proposed. According to the section on the topic in [42] the most common method is to average out the speckle pattern by multiple exposures. As speckle is such an integrated part of the wave front recorded in a hologram, it does not affect the efficiency of the methods proposed in this thesis. However, the existence must of course be taken into consideration when examining visual results and renderings.

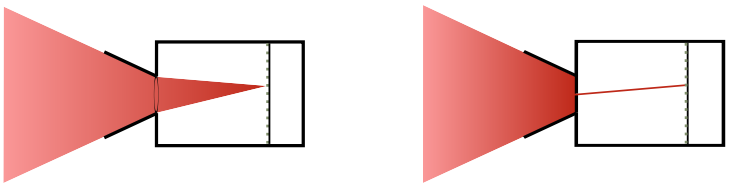
2.2 Image formation

In the previous section we presented a couple of models for representing and propagating light through space. Now we will address the question of image formation as well as the general principles behind light recording and display. We will try to give a general overview of the problematic and thus lay the foundation for the future sections dealing with holography and light field rendering.

Light receptors of today, weather they are biological or man-made, measure light power. Simply told, this is the energy of all light reaching the sensor per time unit. In order to form an image, intensity is measured in several points, leading to a structure of receptors ordered on a two dimensional surface. As an example, in the human eye light is focused through the cornea and the lens onto a photo sensitive neural area in the eye called the retina. This is basically a process where the light is projected onto the receptor area at the back of the eye. The same principle holds for the camera, but using glass optics and a CCD instead.

The amount of light reaching each receptor is dependant on the aperture, or opening of the optical system. Figure 2.5 shows a conceptual sketch of an aperture and a pinhole camera. Using an aperture, light from all directions passing through the opening reaches each one of the receptors on the image plane. Clearly, measuring the total power, means that the radiance of the individual directions is lost in this case. This destroys the three-dimensional information carried by the propagating light. Vision systems, such as our brain, reconstruct some of the information and provide a 3D sensation using advanced processing techniques.

Using an idealized pinhole camera instead, the aperture is now so small that only light from one direction is measured per receptor. In theory, using an infinitely high resolution receptor area the incoming radiance from each light direction could be captured separately. Due to several practical and physical limitations this is not possible however.



(a) Aperture Camera (b) Pinhole Camera

Figure 2.5: Camera models. Light passes through an aperture and is focused using optics onto some light sensitive material. The images show the light coming in to one picture element for (a) a camera with an aperture opening and (b) a pinhole camera.

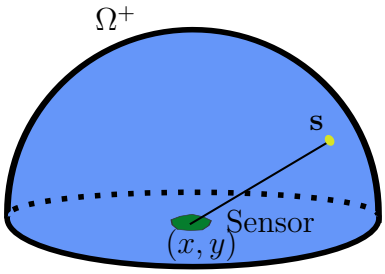


Figure 2.6: The hemisphere of incoming light directions at a sensor.

2.2.1 Image formation in geometrical optics

As described in Sect. 2.1.1 we define radiance as a property associated with a ray. Thus the total incoming power at a receptor positioned at coordinates (x, y) in some image plane Π is

$$I_\Pi(x, y) = \int_{\Omega^+} L_\Pi(x, y, \mathbf{s})\Gamma(\mathbf{s})\, d\mathbf{s}. \tag{2.26}$$

In the above equation $L_\Pi : \mathbb{R}^4 \to \mathbb{R}$ is a function on Π that for a specific position and direction yields the incoming radiance. Γ is a direction dependent attenuation function, and Ω^+ is hemisphere of incoming light directions. The principle is shown in Figure 2.6. I_Π is called the irradiance on Π. When we refer to an image in this text, it usually means the normalized irradiance field on some plane. Likewise the intensity will mean the element of such a field.

It is clear from Eq. 2.26 that the radiance along the incoming individual light rays is lost. Thus, it is not possible to determine the light directions without some kind of deconvolution process. As this is very hard in most cases, techniques to record both radiance and direction of the incoming light rays would be desirable.

2.2.2 Image formation in wave optics

The power of the wave fields described in Sect. 2.1.2 is measured as their magnitude. Thus for a general complex valued wave field W_Π in a plane Π we can assume the measured power to be proportional to

$$I_\Pi(x, y) = \|W_\Pi(x, y)\|^2. \tag{2.27}$$

Just as in the case of geometrical optics, the normalized absolute field will be referred to as an image in this text.

Measuring the power of the light only, removes the phase from the complex valued wave field. Again, just as in the case of the geometrical model this destroys any knowledge of light directions. The simplest example of this is to consider the equation for a planar wave, 2.11, were the light direction is explicitly encoded in form of the wave vector in the phase.

$$\begin{aligned}\|E(\mathbf{r})\|^2 = \|A_0 \exp(i(\mathbf{k}\cdot\mathbf{r}))\|^2 &= (A_0 \exp(i(\mathbf{k}\cdot\mathbf{r})))(A_0 \exp(i(\mathbf{k}\cdot\mathbf{r})))^* \\ &= A_0 A_0^* \exp(i(\mathbf{k}\cdot\mathbf{r}))(\exp(i(\mathbf{k}\cdot\mathbf{r})))^* = \|A_0\|^2. \end{aligned} \tag{2.28}$$

Thus, it is clear that the directional information is lost also in this case.

2.3 Full light recording

While the contributions of this dissertation mainly regard analysis, manipulation and synthesis of holograms and light fields, these concepts did historically origin from the desire to capture and reconstruct the full light of a scene. Thus, a section on recording is in place to introduce the holographic method and the light field.

As we have seen in the previous section recording of light works by measuring the power of the total incoming radiance on some image surface. This projection results in a loss of directional information which is the basis of many of the cues used to experience a 3D sensation. A simple example is by looking at the lack of depth experienced from a photograph compared to the real scene.

Thus, it is clear that in order to capture the full light, that is the whole incoming light, as it passes through the aperture of the camera we need methods and structures to encode the directional information of the captured scene. Below we briefly present the background of the two main approaches used in this thesis. The hologram and the light field.

2.3.1 Principles of holography

Holography is a technique for wave front reconstruction presented by Dennis Gabor in 1948 [29, 30, 31]. He was awarded the Nobel price for his discovery in 1971. Gabor wanted to tackle aberration problems in electron microscopy, but the general technique was applicable in a broader field of optics. The holographic method requires coherent light, and as such it did not take off as a practical method until the laser was introduced in the 1960's. From that time onwards however several improvements were made to the technique, and today there are many different types of holographic techniques.

The basic principle however, is to record the interference of the object wave field with a reference wave field. The resulting interference pattern will then encode the phase difference between the object and reference waves, thus reconstruction is possible.

The holographic setup is fairly simple. A collimated light source illuminates the object which is to be recorded, at the same time as a reference light (often from the same light source) directly exposes the recording medium. The reflected light from the object will then interfere with the reference light as described in Sect. 2.1.2 which can be recorded as an intensity fringe pattern. This setup is illustrated in Figure 2.7.

Given an object wave front W_o and a reference wave front W_r we can express the process as

$$\begin{aligned} I &= \|W_r + W_o\|^2 = (W_r + W_o)(W_r + W_o)^* \\ &= W_r W_r^* + W_r W_o^* + W_o W_r^* + W_o W_o^*. \end{aligned} \tag{2.29}$$

The intensity recording of this interference is commonly referred to as a hologram.

The actual recorded hologram intensities are a linear function of I, as the hologram values are dependent on the recording media and exposure time. However, according to Schnars and Jüptner [100] the constant factor of the transform can be dropped in digital holography. This leaves just a scale factor, dependent on exposure, but constant over the hologram surface. As we are mainly concerned with hologram rendering in this thesis, where the output has to be scaled to the dynamic range of the display device, we will drop also this factor and assume that the hologram is perfectly recorded and reproduced.

To reconstruct the wave field from the hologram the recorded intensity pattern is exposed to the same coherent light source that was used to record the object. The pattern will diffract the incoming light, and part of the resulting wave front contains the original object wave. To see why this is true, consider illuminating the hologram in Eq. 2.29 by the reference wave

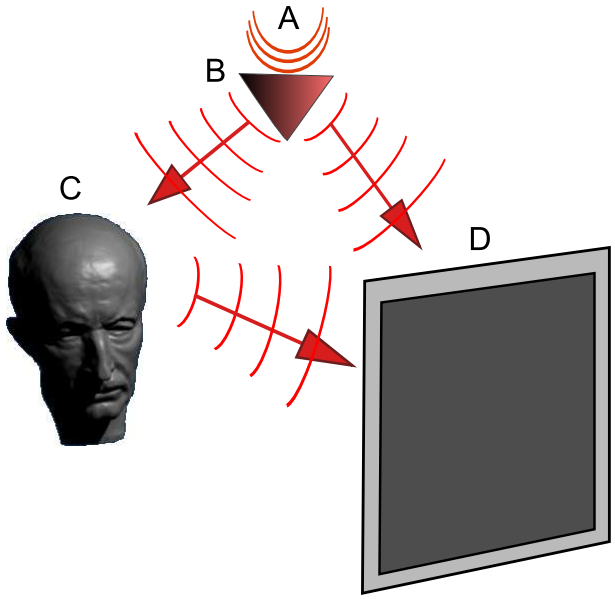


Figure 2.7: Hologram recording setup. A coherent light source in A illuminates a beamsplitter, B. Part of the light illuminates the object at C and is reflected in direction of the recording medium at D. The hologram is the recorded interference pattern between the reference light coming directly from B and the reflected light carrying the object wave.

W_r as described above. This yields the following expression

$$W = IW_r = (\|W_r\|^2 + \|W_o\|^2)W_r + W_r^2 W_o^* + \|W_r\|^2 W_o. \tag{2.30}$$

A more explicit argument can be given by inserting the expression of general complex valued reference and object waves, $W_r = A_r \exp(i\phi_r)$ and $W_r = A_o \exp(i\phi_o)$, in the above equations. This leaves us with the following expression for Eq. 2.29

$$\begin{aligned} I &= W_r W_r^* + W_r W_o^* + W_o W_r^* + W_o W_o^* \\ &= A_r A_r^* + A_r A_o^* \exp(i(\phi_r - \phi_o)) + A_r^* A_o \exp(i(\phi_o - \phi_r)) + A_o A_o^* \\ &= \|A_r\|^2 + \|A_o\|^2 + A_r A_o^* \exp(i(\phi_r - \phi_o)) + A_r^* A_o \exp(i(\phi_o - \phi_r)). \end{aligned} \tag{2.31}$$

And the reconstruction becomes

$$\begin{aligned} IA_r \exp(i\phi_r) = (\|A_r\|^2 + \|A_o\|^2)A_r \exp(i\phi_r) + \\ A_r^2 A_o^* \exp(i(2\phi_r - \phi_o)) + \|A_r\|^2 A_o \exp(i\phi_o). \end{aligned} \tag{2.32}$$

In both Eq. 2.30 and Eq. 2.32 the third term is identical to the original object wave, only multiplied by the magnitude of the reference wave. This magnitude only influences the brightness of the image [100], and we thus have a reconstruction of the original object wave. The image created by this wave front is called the *virtual image*.

The original wave front is only a part of the reconstruction however. The two other terms also corresponds to wave fronts. The first is the so called *zero order*, it corresponds to the amount of reference light passing through the hologram without being diffracted. The second term forms the *real image*. This is a distorted view of the original object located at the opposite side of the hologram plane from the virtual image. Thus, the full light observed is more than the original wave front. It also contains the zero order from illuminating with the reference wave, as well as the distorted real image. This can make it hard to view a clear image of the object as reconstructed from a so called inline setup where laser, object and recording medium all are centered on the optical axis of the system. This is the kind of setup that was originally described by Gabor.

One commonly used solution to this problem is to tilt the reference wave, creating a so called off-axis hologram. Such a setup has the effect of spatially separating the real and virtual images so that they lie on each side of the zero order light. This approach was suggested by Leith and Upatnieks who made several important contributions to the early development of modern holography [56, 57, 58]. Off-axis holography will not be further discussed in this Section, as the basic holographic principle is the same as in the inline case. For a more in-depth description on this matter textbooks such as [99, 53, 42] should be consulted.

2.3.2 Light field recording

The term light field was introduced in optics by Arun Gershun in 1936 in a paper on the radiometric properties of 3D space [33]. The light field concept as used in computer graphics was introduced at the annual SIGGRAPH conference in 1996 by Levoy and Hanrahan [61] and Gortler et al.(using the name *lumigraph*) [37]. Both papers describe the same general idea, which however differs slightly from Gershun's original definition. We shall be using the computer graphics definition of the concept in this thesis and refer to it as a light field.

The light field is closely related to the plenoptic function [2]. However, the concept is simplified in such ways that it becomes suitable for computer graphics rendering. The plenoptic function, in its most general form, is defined as a

$$\mathcal{P} : \mathbb{R}^7 \rightarrow \mathbb{R}. \tag{2.33}$$

That is, $\mathcal{P}(x, y, z, t, \theta, \phi, \lambda)$ is a function describing the radiance of a ray passing through the spatial point (x, y, z) with the directional components (θ, ϕ) at time t for wavelength λ. The light field is a specialized version of this function, assuming a static scene, a discrete number of wavelengths and empty (occlusion-free) space between the observer and the object. The two first assumptions simply let us remove the time and wavelength dimensions as they are not used. The last assumption might require a bit more explanation. What it in fact states is that the light field can be measured at a plane instead of a volume. As there is nothing obscuring the light path between the object and the observer the radiance along the ray will always be kept constant. This means, the same light will be measured in all planes in front of the object, but at slightly different configurations. The third dimension of the spacial coordinate can safely be ignored.

This leads to a light field defined as

$$\mathcal{L} : \mathbb{R}^4 \rightarrow \mathbb{R} \tag{2.34}$$

that is a mapping from a ray-space to radiance. Intuitively $\mathcal{L}$ can be thought of as a function that gives the radiance for a ray going though a plane, where two coordinates describe the position in the plane and two describe the direction of the ray relative to the plane.

In practice there are a couple of ways to implement the spatial and directional parameterizations. In [61, 37] as well as in the majority of subsequent computer graphics work the so called plane-plane parameterization has been used. In this setup the light rays are parameterized by the coordinate pairs arising from the intersection with two planes. Figure 2.8 shows this setup.

There are also other parameterizations such as the plane-sphere and the sphere-sphere variants [17, 16] that have advantages in certain situations. In this thesis however I will use a special case of the plane-plane configuration. Depicted in Figure 2.9, it stores the first plane coordinate together with the ray direction as components along the plane basis vectors. Chapter 4 discuss this further and also defines Eq. 2.34 more rigorously.

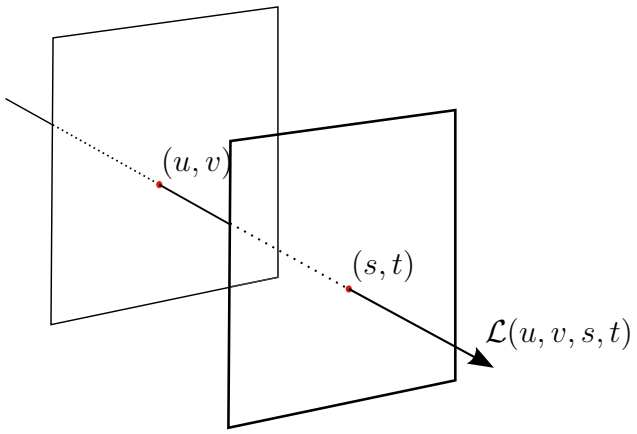


Figure 2.8: The plane-plane parameterization. The light field ray is parameterized as the intersection coordinates with the camera (uv) and focal (st) planes.

Light field recording is usually performed using a single, multiple, or plenoptic cameras. Recently there has also been some interest in synthetic light field rendering. Below is a short comparison of the different techniques.

Single camera light field recording Used in the first light field related papers, this technique requires a static scene. A digital camera is moved from position to position around the scene, taking pictures. This can be done either manually as in [37] or by an automated setup as in [61]. Several issues have to be taken into consideration in this setup. The lighting has to be constant, i.e., moving the camera must not create unwanted shadows or change the light in any other way. In the works cited above it is also important to acquire pictures in a regularly sampled grid around the object. This requirement has later been relaxed by Buehler et al. [15].

Multi-camera light field recording The multiple camera setup is by far the most common recording environment for video-based rendering [74]. Such a setup delivers high resolution dynamic video from multiple viewpoints. Early versions of multi-camera recording setups include the works of

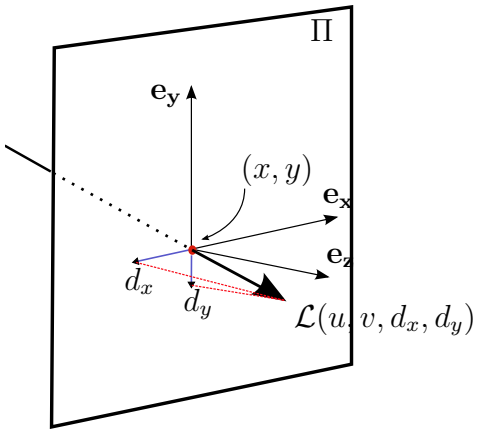


Figure 2.9: The plane-direction parameterization that we use in Chapter 4. This representation describes the ray by its intersection coordinate with a reference plane, Π, and its directional components along the basis vectors that spans Π. This can be thought of as a special case of plane-plane parameterization, using an individual focal plane at distance 1 from every camera plane point.

Narayanan, Randers and Kanade who present a domelike structure of synchronized video cameras [87]. Several similar systems with the same goal of capturing multi-video data, have arisen since then [122, 113, 5, 84]. However, most of the papers cited above consider systems targeted at other video-based rendering techniques than light fields. This includes applications such as 3D reconstruction [34], visual hull rendering [83, 63] and motion tracking [18]. Due to economical and technical reasons these systems usually employ a sparse camera setup arranged to cover the scene from different viewpoints. For light field acquisition, however, the number of cameras in such a setup is often not sufficient. In recent years camera array systems such as the Stanford Multi-Camera Array developed by Wilburn et al. [119] and the MIT Distributed Light Field Camera described by Yang et al. [124] have been introduced. These systems arrange a huge number of cameras into a planar grid, and the recorded images are interpolated to form a light field.

A multi-camera recording system can deliver high resolution dynamic light fields, but it also exhibits a greater degree of complexity. Some issues that need to be resolved when considering a multi-camera system is inter-camera synchronization, calibration and storage. Generally this means that the systems require some infrastructure and are more suitable for studio usage.

Plenoptic camera light field recording A plenoptic camera is a single camera that captures a whole light field in one shot. A device for this purpose was first proposed by Adelson and Wang [3], and has recently been redesigned and improved by Ng et al. [88, 89]. The basic principle is to break up the incoming light so that every unique ray passing through the camera aperture is imaged on a unique receptor on the sensor. This concept thus presents a solution to the aperture problem discussed in Sect. 2.1.1. Thus, the plenoptic camera works like a tightly packed array of pinhole cameras. The refocusing of the light is performed using a micro lens array. Thus, a standard digital camera CCD can be used to record the images, acting like a super sensor divided into several sub sensors registering the ray bundles. Each of the small images represents a pinhole picture. This is a very elegant technique for light field recording and has recently also been adapted to other areas such as microscopy [62]. There is a tradeoff to this method however. The plenoptic camera records a full light field on just one standard CCD. Thus, the sensor has to be tiled as described above. In a multi-camera system each camera records a single image from its viewpoint using the full CCD, while in the plenoptic case each tile acts as a camera. Thus, the plenoptic camera captures lower resolution images leading to a lower resolution light field. Despite this, the plenoptic camera holds a lot of promise as imaging sensor technology improves.

Synthesized light fields The above acquisition methods all focus on image- or video-based rendering, which also is the origin of the light field technology. Synthetization of light fields has mostly been used to generate test cases and thus not been the main focus for improvement. The light field representation however is very closely related to integral imaging and autostereoscopic display systems. Isaksen et al. [47] showed how light fields could be used to form lenticular images and Halle [40] discussed methods for multi-view rendering already in 1998. Different types of autostereoscopic and integral display systems are readily becoming available and so the interest in efficient multi-view rendering is on the rise [125, 43, 45]. A good introduction to autostereoscopic display technologies and its role for multi-view computer graphics can be found in [41].

2.4 Holographic displays and computer generated holography

The main contributions in this thesis are two methods to synthetically generate holograms. The foundations for computer generated holography (CGH) lies in a class of devices known as Spatial Light Modulators (SLMs). As the name gives away these are devices that allow for wave front modulation. More specifically they can, in the right scenario, be used to manipulate the amplitude or phase of light. Such a device could be used to recreate a generic wave front, containing the total light of a scene and thus allow true three dimensional graphics.

2.4.1 Spatial Light Modulators

There are many types of SLMs on the market today, employing different technological solutions. Two common types are Liquid Chrystal Displays (LCDs) and Digital Micromirror Devices (DMDs). LCD based SLMs use the same general principle as computer displays, employing a matrix of liquid crystal cells embedded in an electrode wiring. By employing an electrical field the crystal material can be arranged to affect the polarization of the light and thus the transparency. As the wiring needs to be run within the LCD structure pixel fill factor can be a problem with LCD displays. A Digital Micromirror Device is a chip covered with a huge number of very small aluminum mirrors. The mirrors are mounted on yokes and can be rotated to an off-state using an electrostatic field. Thus, by setting the mirrors on or off a binary image can be formed by choosing which light is reflected towards the observer. Grayscales can be created by flipping the mirror on and off with a high frequency, reflecting just a portion of the light and thus decreasing the intensity by a form of time multiplexing. DMD devices are usually praised for the almost complete fill factor of the chip, which is due to the fact that the mirrors can be tightly packed.

A recent technology that can be seen as a hybrid between LCD and DMD is the Liquid Crystal on Silicon (LCoS) devices. These are constructed as a matrix of reflective elements, just as the DMDs. However, instead of the mechanical mirrors liquid crystal elements are used as mirroring elements. This allows for a much higher fill-rate than with the traditional LCD construction.

We will not delve deeper into the specific technical solutions of SLMs in this thesis. A comparative table of LCDs and DMDs for holographic displays can be found in [112]. The methods within this thesis are independent of the specific type of SLM used. The only thing assumed is that a spatial light

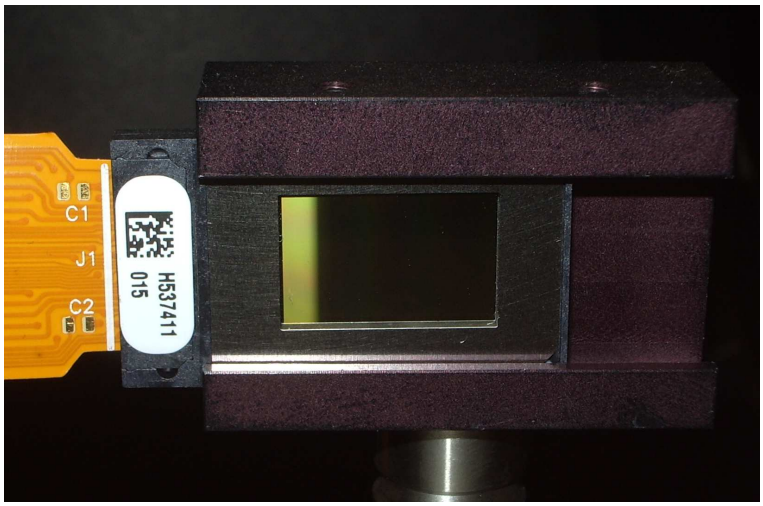


Figure 2.10: A close-up of the type of SLM used in our experiments. A Brillian 1080 with a resolution of 1920×1200 **pixels, and individual pixel size** 8.1μ**m. Photograph by Philip Benzie.**

modulator is an addressable device with $M \times N$ intensity valued elements arranged into a matrix, much like a computer monitor.

2.4.2 Principle of a SLM-based holographic display

Spatial Light Modulators can be used to construct a holographic display system. As described in Sect. 2.1.2 diffraction occurs as light interacts with fine, detailed patterns. This is also the basics of traditional holographic reconstruction as described in Sect. 2.3.1. Today, SLMs with pixel sizes of a few micrometers can be bought off the shelves. While this is at least a factor 10 larger than the grain size offered by photographic film, it still offers the possibility to prototype holographic displays with fairly good resolution. The general principle is the same as that of traditional reconstruction. Coherent light interacts with the interference pattern either by transmission through the SLM or by reflection. The diffracted light will then create the wave front. Figure 2.11 shows the layout of such a system.

In contrast to film-based holography however, using a SLM provides us with the same advantage as using a computer monitor over a photograph: the intensity pattern shown on the device can be rewritten and changed. Thus, with the possibility to create arbitrary interference patterns, we can indirectly control the wave field that is emitted. By developing methods to compute holographic interference patterns, 3D images of virtual objects can be displayed. This is the basic principle of the holographic display setup.

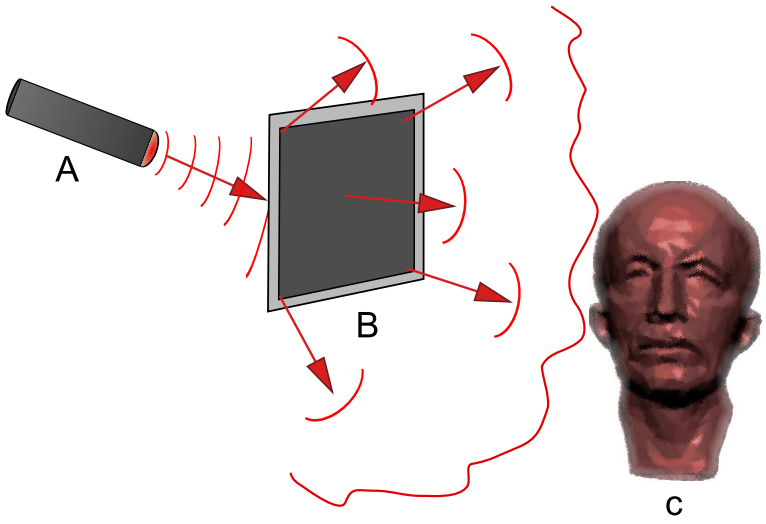


Figure 2.11: Basic principle of hologram viewing. A laser in (A) illuminates a SLM located at (B) with coherent light. The light gets diffracted and an object wave field is emitted. An observer will perceive a 3D model of the object

The main contributions of Chapters 6 and 7 considers effective methods for hologram creation which is at the heart of the computational holographic process.

As mentioned above, a SLM does not have the same high resolution as the type of photographic film used in traditional holograms. The pixel size of the SLM affects the maximum diffraction angle of the light, and therefore the possible maximum object size and required focal distance. The diffraction angle of a SLM is usually expressed using the grating equation [104, 99]

$$\sin\theta = \frac{\lambda}{2\Delta p}. \tag{2.35}$$

Where θ is the maximum angle of diffraction possible, Δp is the pixel size of the SLM and λ is as usual the wavelength of the light source. We would like to point out that this hints at that a general holographic system requires very high resolution spatial light modulators and high pixel counts. Slinger et al. [104] estimates that displaying an object of size 0.5 meters while allowing for a field of view of 60° require about one terrapixels.

2.4.3 An experimental holographic display setup

Although there are several brands of SLMs readily available on the market today, holographic display systems are still in their early stages. We have

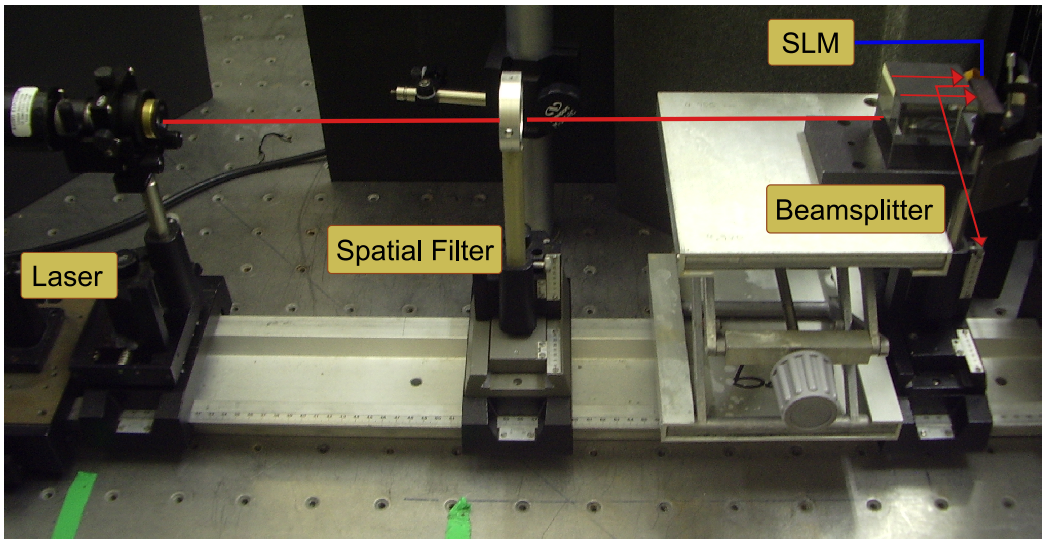


Figure 2.12: Picture of the SLM based holographic viewing setup used in some of the experiments in this thesis. A HeNe laser illuminates a beamsplitter through a spatial filter. A Brillian 1080 reflective SLM is mounted on the far side of the beamsplitter. Light directed at the SLM is diffracted and reflected back trough the splitter and towards a viewer located perpendicular to the laser beam. Original photograph by Philip Benzie.

tested the techniques developed in this dissertation using an experimental setup build by researchers at the University of Aberdeen. The principle of the system is very similar to the one described in Sect. 2.4.2. Figure 2.12 depicts the main body of the system showing laser, polarizer, beamsplitter and SLM. The light source is a Helium Neon laser ($\lambda = 633$ nm) that shoots light through a spatial filter. A beamsplitter redirects the light so that it reflects on the SLM and passes back out towards the viewer, located at a right angle to the original beam.

The spatial light modulator is a Brillian 1080. This model has a maximum resolution of 1920×1200, and a pixel size of 8.1 μm. Inserting these values, and the laser wavelength of 633 nm in Eq. 2.35 we find the maximum diffracting angle of the setup to be

$$\theta = \sin^{-1} \frac{633 \times 10^{-9}}{2 \times 8.1 \times 10^{-6}} \approx 2.2^{\circ}. \tag{2.36}$$

2.5 Summary

In this chapter we have introduced some of the basic concepts used in this dissertation. The geometric model of light and the light field concept will be used in Chapters 4 and 5 where we discuss transforms of light fields and holograms. Chapters 6 and 7 focus on methods for computer generated holography. Wave optics and the concept of diffraction are the foundation of the theories presented there. These two chapters also present results displayed on the experimental SLM setup discussed in Section 2.4.3.

Chapter 3
Related work

While we gave references to a broad range of background work in the previous Chapter, we will now focus on work related more directly to the contributions in this thesis. This includes traditional important work in computer graphics and optics, as well as recent publications in light field analysis and computer generated holography.

We will begin our study in Section 3.1 by reviewing work in computer graphics applying the wave theory of light. Although this dissertation does not primarily contribute to diffraction for traditional computer graphics, our strategy is related to some of these rendering techniques and could be used in the area in the future.

Thereafter, in Section 3.2, we will continue by presenting work related to light field transform, rendering and compression. This is work directly related to our contributions in Chapter 4.

There has been some work, in both optics and computer graphics to transform between light field and hologram like representations. In Section 3.3 we will review these papers from the point of view we take in Chapter 5.

Finally, Section 3.4 discusses work related to our contributions in computer generated holography. This relates to both Chapters 6 and 7 and tries to give a broad overview of work in the field.

3.1 Wave optics in computer graphics

Traditionally, computer graphics (CG) research has mostly been concerned with the ray theory of light. This is due to computational efficiency and that many wave-effects such as diffraction can be ignored while scattering may be modeled or approximated using simplified versions of the Bidirectional Reflectance Distribution Function (BRDF). There are however exceptions.

For instance some materials have surface properties producing clearly visible diffraction effects. One such example is the everyday compact disc, which produces a distinctive rainbow reflection pattern due to diffraction in the disc's gratings.

Using the so called *geometrical theory of diffraction* (GTD) [52], diffraction effects could be integrated in a ray-tracing environment. However, to our knowledge this theory has not been used much in the computer graphics community. One exception is the work by Tsingos [117] where it is used to model diffraction at wedges. GTD has also been incorporated in Monte Carlo ray tracing [27], however this implementation is targeted towards optic simulations and not computer graphics. In this dissertation we do not further consider the geometric theory of diffraction, however it may be an interesting venue for further research.

The first major attempt at using the wave model of light for CG rendering seems to have been Moravec's SIGGRAPH paper *3D Graphics and the Wave Theory* in 1981 [86]. Moravec suggests using the wave model in computer graphics to model reflection and illumination. His method employs propagation of wave fronts between planes in the scene. This approach is however computationally very expensive as each pass requires the current wave front to be convolved by a transmission function.

Lucente introduced the concept of computer generated holograms in computer graphics [70, 69]. In contrast to other work discussed in this Section, Lucente's goal was computer graphics for holographic display systems, and not the use of wave optics methods in traditional graphics. Many of the later CGH rendering methods targeting real time display are related to the work of Lucente. There is a difference in focus to the work presented in this dissertation however. We are mainly interested in synthesizing the wave field and base our algorithms on implementations of interference directly. The work by Lucente is mostly based on rendering using so called holographic elements (hogels). We will return to how these are related to light fields below.

In his work on *Diffraction Shaders* Stam used the wave theory of light to show how diffraction due to surface reflection could be implemented in a shader [107]. This result is important, as it shows how to compute the BRDF analytically, including diffraction effects, if the height field of the surface is given as a Gaussian random process. There has also been work targeted at rendering CDs and CD-like surface diffraction reflection [111, 23]. These approaches however are specifically targeted at computing the reflectance given the microstructure of the material.

Recently, Ziegler et al. have presented a framework for integrating holograms in computer graphics [128]. The report summarizes digital holography and holographic rendering from a computer graphics perspective. It also dis-

cusses the hologram as a representation for CG rendering. In [127] Ziegler et al. also show how holograms can be computed from a light field representation of the scene using depth reconstruction. This work is related to the discussion on wave fields and light fields in Chapter 5, which also contributed somewhat to the cited paper.

Although this dissertation is not directly concerned with holography in the contents of traditional computer graphics rendering, the above works are still of relevance. Many CG methods are designed with efficiency and visual quality in mind. Meaning that even if the methods may not be directly applicable in holographic rendering, they may still provide insights in numerical optics that classical simulation methods do not.

3.2 Light fields

In Chapter 4 we consider methods regarding optical operators for light fields. As presented in Chapter 2, the computer graphics concept of a light field, as used in this thesis was introduced in 1996 by Levoy and Hanrahan [61] and Gortler et al. [37].

A lot of work related to light field transforms has been connected to morphing. Zang et al. [126] presented a method to perform warping between two light fields using corresponding rays and blending. Chen et al. [19] took a different approach where the light fieldis subdivided into volumetric parts that are deformed individually before being stitched together again. Our work differs from morphing methods in that we only consider optical operators. This means that we do not define an explicit target state of the light field before transforming. Rather we focus on describing the parts of the transform using a series of operators.

Heidrich et al. have presented an advanced lens model that well can be used for light field rendering [44]. The model describes a mapping of all rays passing through the lens, and can thus be seen as a light field transform. While our approach is based around the thin lens model and thus not as powerful, it allows us to combine different operators in a chain to also take rotation and interface effects into account.

Isaksen et al. have presented important work on reparametrizing light fields, and shown how to express the ray space in different frames [47]. This is very useful in many applications as it allows for variable reconstruction and focal planes. It also allows to model aperture and depth of focus effects. Another important work on focal effects in light field rendering is *Fourier Slice Photography* by Ng [89]. In this work the image formed by a light field is expressed as a slice in its four dimensional Fourier space. While we

perform our rendering using ray propagation in the spatial space, and are currently not addressing depth of focus effects, it is interesting to note that Ng base his slice lookup method on a plane to plane function very similar to our propagation operator. It may thus be possible to integrate our operators in a similar framework.

Several different methods have been proposed for light field rendering in the past few years. Sloan and Hansen have developed a number of methods using ray - light slab intersection tests targeted at parallel architectures [105]. In [15] Buehler et al. present a general method for rendering by blending views from the original light field using a blending field derived from geometrical and view information. Goldluecke et al. have developed a GPU-based method for dynamic light field rendering using a warping algorithm [35]. The matrix optics method developed in Chapter 4 expresses the ray transport from the light field to the image plane as a linear transform. This allows us to easily model the light field under influence of optical elements such as lenses and interfaces.

Densely sampled light fields require high bandwidth, and consequently data compression is essential. In [61] vector quantization followed by entropy encoding is used. Wong et al. use spherical harmonics to represent the directional part of a light field [120]. In [85] block encoding and the DCT is suggested as compression method. Magnor and Girod have presented a pair of codecs based on disparity compensation [73]. Wood et al. have developed vector quantization and principal component analysis methods for irregularly sampled surface light fields [121]. Another PCS approach is presented by Lelescu and Bossen [59], while Chen et al. factorize an approximation of the light field into a set of two dimensional functions and apply vector quantization and block coding [20].

We use wavelet compression of the light field data. This allows for high compression ratios and efficient storage and rendering. The efficiency for light field encoding has already been reported in several publications; Lalonde and Fournier use wavelets to store light fields in a hierarchical data structure [54], while Peter and Straßer introduce a wavelet representation that allows for efficient storage and progressive transmission of light fields [91]. In [71] a light field acquisition, compression and representation system based on a hierarchical wavelet structure is presented. Our approach to data representation is similar to the work of Peter and Straßer, while the basic wavelet tree accessing philosophy has ideas in common with the method of Lalonde and Fournier. However, in contrast to both approaches, we perform interactive image reconstruction using matrix optics.

3.3 Wave field analysis

To our knowledge, not much work has been performed on relating the light field and the wave field. One notable exception is [51] where Kartch shows how to map from Lucente's hogel representation [68] to a light field. The method assumes far field viewing and is based on Fourier transforming the hogel data. It is thus very similar to the example we give in Chapter 5. The hogel has the same role as the Fourier window in our framework. Thus the both methods can be expected to produce similar results. However, Kartch's method is a transform between hogel and light field representations which produces exact results for holographic stereograms only. Our transform should be seen as an approximate method for wave field to light field conversion as discussed in Chapter 5.

Abookasis and Rosen present a method to synthesize holograms from multiple images [1]. The algorithm can be seen as computing a hologram from a light field. Each image is weighted by direction dependent basis functions and the results are summed together to form a hologram. However, as no depth information is known, these holograms are only valid in the far field.

Ziegler et al. also show how holograms can be computed from a light field representation of the scene using depth reconstruction [127]. The paper is targeted towards a light field - hologram mapping, and thus attempts to address some of the difficulties pointed out in Chapter 5.

3.4 Computer generated holography

According to Tricoles [116], the first work regarding computer generated holography (CGH) was published in 1966 by Brown and Lohmann [13]. The paper considers optical spatial filtering, and the holograms were computed using a direct Fourier transform and output on a plotter. Nevertheless, the work showed that it was possible to inversely compute wave fronts given an object and sparked interest for further research.

The areas of application for CGHs ranges from optical computing, via dynamic filtering to 3D displays. This thesis is primarily concerned with the latter, and so we will mainly address works related to holographic display technology from recent years. Readers interested in the early development of CGH may want to refer to [55]. A literature survey of the twenty first years of CGH research can be found in [116].

While most techniques reported here are based on propagating the light from the object to the hologram plane, the opposite is of course possible. Stein et al. have presented an approach using ray-tracing [108]. The method

computes holographic patterns from a set of 3D points by computing the distances between hologram plane and object points and then treating each point as a light source. The basic premises of this technique are thus not different from the ones we use in Chapter 6. However, the paper may suggest an interesting possibility to couple holographic rendering with a computer graphics grade ray-casting for accelerated rendering methods.

3.4.1 Hardware accelerated hologram synthesis

Due to significant computational load associated with hologram synthesis, most methods targeted towards interactive holographic display make use of some kind of special purpose hardware.

Holographic rendering hardware as well as techniques for real-time display were presented in 1994 by researchers in the MIT Media group [106, 66, 68, 118, 70]. This implementation was reported to be approximately 50 times faster than a workstation of its day. Lately work by Bove et al. has been presented that takes advantage of the programmable graphics pipeline of modern GPUs to drive a similar display [12]. Another hardware architecture, HORN, has been developed at the Department of Medical System Engineering at Chiba University [48, 49, 101, 102, 50]. The later versions use a clever scanline-based iterative recurrence formula [103]. The latest reported HORN-model has an architecture allowing several boards to be coupled in parallel, the system is reported to be about 1000 times faster than a CPU implementation [50].

Special purpose hardware must be custom built however. This an expensive and time-consuming process. In this thesis we focus our attention towards using the commonly available Graphics Processing Units (GPUs) in order to synthesize holographic interference patterns.

Computer generated holography using the so called fixed-function (non-programmable) pipeline on a graphics workstation was proposed in 1999 by Ritter et al. [95]. They made use of an early generation of graphics hardware to render holograms by using them for quick table look-up and summation of the point source distributions. Another method was presented by Petz and Magnor [92]. They showed that the Fresnel zone plate of a point source could be approximated by scaling and translating the zone plate of a precomputed, known source. They then used texture combiners to perform fast transformation and summation of the Fresnel zone plates from precomputed images.

Both techniques applied state-of-the-art graphics hardware which at that time did not allow for much programming freedom, hence the use of precomputed zone plates as approximations. Using modern GPUs, programming

restrictions are not so limited, allowing for more accurate algorithms to be implemented.

Matsushima and Takai have presented some very interesting reoccurrence formulas aimed at hologram creation [82]. In their paper they derive three different iterative formulas for computing holographic scanlines from point models. They also perform error analysis and show how the formulas can be optimized for efficiency. Their method is interesting, however, it must be implemented on a device allowing scanline access to the frame buffer memory. The methods are thus hard to implement efficiently on GPUs for instance.

Haist et al. use a multi-pass GPU-method to compute so called *holographic tweezers* [38]. This is a light trap construction used in microscopy and often implemented using a CGH pattern. Although, not targeted towards 3D models or display, the methods used to compute the tweezers are similar to the ones used for CGH. Haist et al. have used GPU shaders similar to ours to accelerate the computation. The implementation differs from the one we present in Chapter 6 however, as it is multi-pass only. Our approach creates a customized shader and unrolls much of the loops thus creating a method with potentially less overhead.

Very recently Masuda et al. presented a GPU implementation for CGH computation [75]. By implementing CGH calculations in a GPU shader they managed to achieve video frame rates for a hologram image size of 800×600 pixels. However, their method is restricted by the number of GPU registers to a maximum of 100 object points. Although the method presented in Chapter 6 is implemented on similar hardware (their GPU was an NVIDIA GeForce 6600, our's an NVIDIA GeForce 7800) the algorithms differ significantly. Among other things, our method does not suffer from a fixed maximum number of points as we have tailored the algorithm to adhere to the programming model imposed by the GPU. In order to speed up the computations, we also use NVIDIA's Scalable Link Interface (SLI) to parallelize multiple GPUs for CGH rendering.

3.4.2 Hologram synthesis from surface models

All methods discussed above perform CGH rendering at interactive rates and are targeted at display systems. However, the trade-off for this performance is the use of light-weight, point-based models to arrive at simpler numerical methods. As the complexity of these methods rise with the number of points, it is generally hard to construct convincing holograms of detailed or solid objects as the number of points required is too high. We therefore propose an approach that uses triangular surfaces as its base primitive. It can handle much more realistic objects at almost the same algorithmic complexity.

In accordance with many other CGH methods, we perform light transport using the angular spectrum of plane waves. This concept is related to the Fourier spectrum of a wave field and will be further introduced in Chapters 5 and 7.

The Fourier transform has a long history as a tool in computer generated holography, due to the potential computational speed increase and the possibility to use the angular spectrum of plane waves. Already in 1967 Lohmann and Paris published work where the Fourier transform was used to compute Fraunhofer holograms [65].

In 1988 Leseberg and Frère introduced a method to create CGH from tilted planes [60]. This approach is shown to work well, but is based around the Fresnel approximation. In our work, we perform rotation and transport of tilted planes based on work by Tommasi and Bianco [114] and Kreis [53].

The bulk of angular spectrum techniques are targeted at transporting wave field distributions between planes. Consequently, the early Fourier-based CGH algorithms considers point distributions in a plane. A good introduction to development of Fourier holography, as well basic CGH techniques in general, can be found in [14] by Bryngdahl and Wyrowski.

Lately, however there has also been some work on CGH from polygonal objects. In these techniques each planar surface in the model is transported to the hologram plane and the distributions are superpositioned. These methods are primarily targeted towards realistic rendering, and can not be used for interactive holographic display today. Matsushima has presented work on CGH rendering from surface objects incorporating both shading and texturing [79, 76].

Ziegler et al. [128] have reported on a framework for incorporating and rendering both synthetic and real holograms. They treat polygonal models in a similar way to Matsushima and also use graphics hardware for the computations.

The methods presented above use the angular spectrum of the light distribution in the angular patch when performing light transport, just as our approach in Chapter 7. This can be efficiently performed by just a multiplication of a transfer function. However, as the polygon is defined spatially, a direct implementation requires a Fourier transform for each polygonal surface and thus puts an effective bound on the usefulness for interactive displays.

In contrast, our methods performs the rendering directly in Fourier space, and thus the per-surface FFT can be avoided together with interpolation of the sampled Fourier coefficients. Matsushima has estimated interpolation to take up 44 % of the CPU time when using the previous method outlined above [78]. With our approach we have an analytic description of the wave field during the whole transport, and can thus avoid interpolation completely.

To our knowledge this is the first time holographic rendering has been performed directly in the angular spectrum.

Chapter 4

Light field transforms by matrix optics

This chapter presents a model for light manipulation inspired by the field of matrix optics [32, 25]. The basic idea is to treat optical elements for ray manipulation as linear functions described by matrices. Thus, the series of operators for a complex optical system can be concatenated to a single operator. This yields an elegant and very effective method for manipulation of light fields.

4.1 Introduction

Since the introduction of image based light fields [37, 61] a number of different rendering techniques have been suggested. The standard methods today include both image-space and object-space algorithms. The former are based on ray-tracing, ray-casting or view morphing, while the latter typically involve texture mapping. In this chapter we investigate the use of matrix optics for light field rendering. In comparison to other light field rendering methods, no ray - plane intersection test has to be performed for every image pixel. Ray tracing based methods can be cumbersome if the imaging system involves a series of optical elements such as lenses or material interfaces causing refraction. In this case the imaging method requires tracing rays between every element, each transforming them in some way. Using matrix optics, a set of operators such as light propagation, thin lenses and dielectric interfaces can be represented using matrices, allowing us to model the whole process as a single matrix. This enables us to model a light field under the influence of an arbitrary series of optical operators by performing a linear transform of its elements. Rendering is a special case of this where a camera

model is constructed from lens and propagation operators. This can be used to create something alike to a "virtual optical bench", a test environment for optical manipulation of light fields.

Our contribution in this chapter is twofold: First we adopt matrix optics to light field modelling. Secondly, we have implemented a system using wavelet compression to render light fields directly from the hierarchical wavelet representation. We achieve interactive to real-time frame rates for densely sampled and compressed light fields.

4.2 Background

Inherited from physics, the ray model of light is by far dominant in computer graphics applications. In image-space systems rays are traced from the camera and out into the model space. Rays are either traced from camera plane to object, or the other way around. Generally, the camera has some kind of optical setup that focuses the rays onto the image plane. Some scenes do also contain optical elements such as interfaces that cause refraction. When rays pass through such elements they change directions according to the specifications of the system. In the simplest of cases the camera is a so called pinhole camera without any optics. This is the classical and most dominant camera model of computer graphics, often just represented by a projection matrix. The name comes from the fact that the iris of the camera is a "pinhole", i.e. the aperture is so small that only one ray will hit every pixel in the image plane. There are however many situations both in computer graphics and optics design and visualization when a more general model is needed. One such case is when propagating light fields through space.

The model we will adopt for this task is called matrix optics. It is a very simple linear model, but as we will see it has a number of strengths for fast and simple matrix operations. We will only briefly touch the foundations of matrix optics here, a more general introduction can be found in [25] and [32]. Matrix optics defines linear operators for a number of optical elements as well as propagation of light between planes in space. This gives an elegant way of computing propagation and optical manipulations of light fields. Standard ray-casting or ray-tracing based methods must compute the ray-path between the individual optical elements, while in matrix optics all operators can be combined using matrix multiplication. The model introduced here is an extension to the matrix operators in optics, suitable for light field transformations.

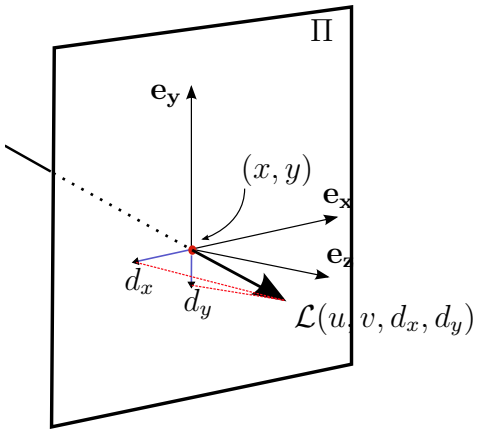


Figure 4.1: The plane-direction parameterization describes the ray by its intersection coordinate with a reference plane, Π, and its directional components along the basis vectors that spans Π.

4.3 Definition of ray space and light field

As discussed in Chapter 2, light fields represent all light that passes through a plane in space. In practical situations, the plane will have finite extent and can thus be regarded as an aperture. The light field itself can be regarded as a function giving the radiance of a specific ray passing through the plane. In this section we will define the light field and the space of rays on which it is defined.

To do this we will use the position-direction light field representation shown in Figure 4.1. From this Figure, let $\Pi \subset \mathbb{R}^3$ be a plane with an associated coordinate system in 3D space

$$\Pi = (\mathbf{o}_\Pi, \mathbf{n}_\Pi, \{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}), \tag{4.1}$$

where $\mathbf{o}_\Pi$ and $\mathbf{n}_\Pi$ is the origin and normal of Π, and $\{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}$ are the vectors spanning the plane.

The ray space on Π consists of all light rays intersecting the plane,

$$\mathcal{R}_\Pi = \mathbb{R}^2 \times \mathbb{R}^2. \tag{4.2}$$

Thus, a ray passing through Π is described as a point in ray space with homogeneous coordinate

$$\mathbf{r} = [\mathbf{x}, \mathbf{d}, 1]^T \in \mathcal{R}_\Pi. \tag{4.3}$$

Above $\mathbf{x} = [x_1, x_2]$ denote the plane coordinates in the frame $\{\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi\}$, and $\mathbf{d} = [d_1, d_2]$ the directional component along Π's basis vectors.

A light field on Π is a mapping

$$L : \mathcal{R}_\Pi \to \mathbb{R}. \tag{4.4}$$

Given a ray $\mathbf{r}$, the light field L yields the radiance along that ray.

To highlight the advantage of the position-direction representation of the rays, consider the following task: A ray is propagating from plane Π_1 to a parallel plane, Π_2, at a distance z. The ray has coordinate $\mathbf{r}_1 = [\mathbf{x}_1, \mathbf{d}_1, 1]^T$ in Π_1. Using the plane direction representation we can simply compute the new ray coordinate as

$$\mathbf{r}_2 = [\mathbf{x}_1 + \mathbf{d}_1 z, \mathbf{d}_1, 1]^T . \tag{4.5}$$

Thus, if we know the light field on Π_1 to be L_1 it is easy to compute the light field on Π_2 by the inverse propagation

$$L_2 = L_1([\mathbf{x}_2 - \mathbf{d}_2 z, \mathbf{d}_2, 1] . \tag{4.6}$$

We are now ready to define a set of matrix operators for transforming ray space.

4.4 Matrix optics operators for light field transformation

Now that we have defined the ray space on a plane, as well as the light field representing the radiance function on this, we can begin to consider what happens to a ray space under transformation. Our goal is to find operators that transforms the ray space coordinates in accordance with standard light operations in geometric optics. To do this we follow the practice from matrix optics literature [18]. Specifically we are interested in two types of operations. The first is propagation of light: given a light-field in one plane, what does the light-field in another plane look like? The second considers different types of refraction, i.e. what happens to a light-field that is viewed through a thin lens or an interface?

4.4.1 Propagation operators

A propagation $\mathbf{P} : \mathcal{R}_\Pi \to \mathcal{R}_{\Pi_p}$ means a transformation of the ray space $\mathcal{R}_\Pi$ of plane Π to the ray space $\mathcal{R}_{\Pi_p}$ of some other plane

$$\Pi_p = (\mathbf{o}_{\Pi_p}, \mathbf{n}_{\Pi_p}, \{\mathbf{e}_1^{\Pi_p}, \mathbf{e}_2^{\Pi_p}\}). \tag{4.7}$$

Any point in $\mathcal{R}_\Pi$ and its image in $\mathcal{R}_{\Pi_p}$ under the propagation $\mathbf{P}$ must correspond to the same ray in world space.

We assume that all propagation takes place in free space, i.e. that there are no occluding objects between Π and Π_p.

Mathematically, let $W_\Pi(\mathbf{r})$ be the world space ray of $\mathbf{r} \in \mathcal{R}_\Pi$, given by the base point $\mathbf{o}_\Pi + [\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi]\mathbf{x}$ and the direction $\mathbf{n}_\Pi + [\mathbf{e}_1^\Pi, \mathbf{e}_2^\Pi]\mathbf{d}$.

Then

$$\mathbf{P} : \mathcal{R}_\Pi \rightarrow \mathcal{R}_{\Pi_p} \tag{4.8}$$

is a propagation operator if and only if

$$\forall \mathbf{r} \in \mathcal{R}_\Pi : W_\Pi(\mathbf{r}) = W_{\Pi_p}(\mathbf{P}\mathbf{r}). \tag{4.9}$$

We will now introduce two different propagation operators: the transport operator, which propagates between parallel planes, and the rotation operator which propagates between rotated planes.

The transport operator

The transport operator, $T_\mathbf{v}$, propagates the light to a plane parallel to Π offset by some vector, $\mathbf{v} = [v_1, v_2, v_3]^T \in \mathbb{R}^3$

$$\Pi_\mathbf{v} = (\mathbf{o}_\Pi + \mathbf{v}, \mathbf{n}_\Pi, \{e_1^\Pi, e_2^\Pi\}). \tag{4.10}$$

It can be written as a 5×5 matrix

$$\mathbf{T}_\mathbf{v} = \begin{bmatrix} 1 & 0 & v_3 & 0 & v_1 \\ 0 & 1 & 0 & v_3 & v_2 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.11}$$

which has the desired properties. Intuitively, this corresponds to transportation along the ray direction and a translation in the plane. Figure 4.2 shows an example of a pure transportation.

The rotation operator

The rotation operator maps the ray space of a plane Π to a ray space of a rotated plane $\Pi_{\mathbf{S}_j^\theta}$, where $\mathbf{S}_j^\theta$ denotes rotation of θ around basis vector $\mathbf{e}_j^\Pi, j \in [1, 2]$:

$$\Pi_{\mathbf{S}_j^\theta} = (\mathbf{o}_\Pi, \mathbf{S}_j^\theta \mathbf{n}_\Pi, \{\mathbf{S}_j^\theta \mathbf{e}_1^\Pi, \mathbf{S}_j^\theta \mathbf{e}_2^\Pi\}). \tag{4.12}$$

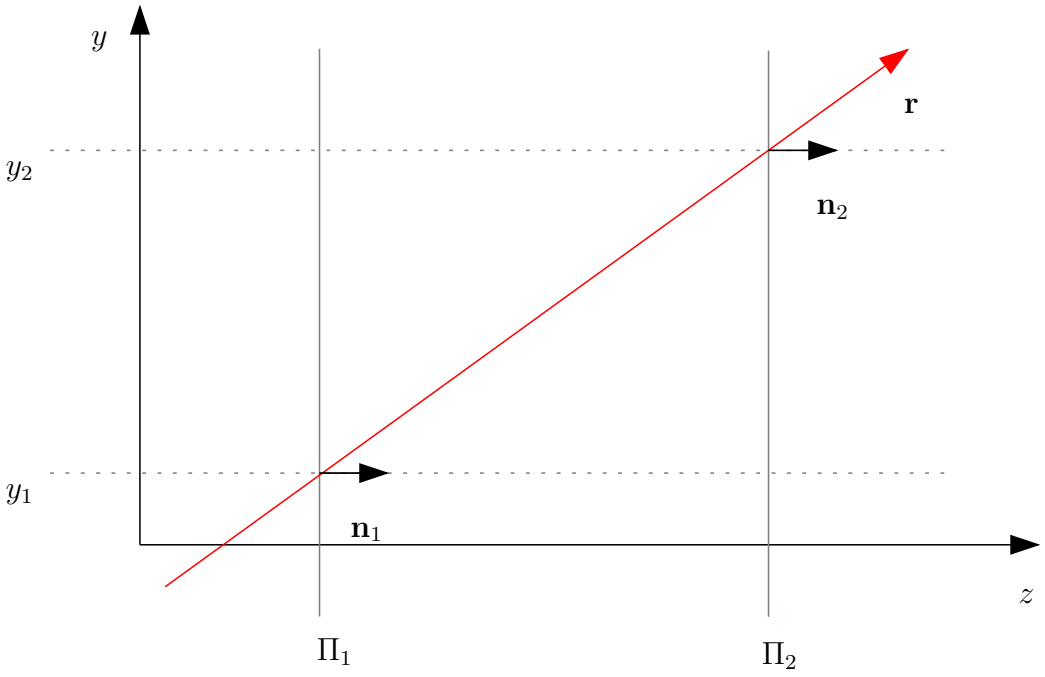


Figure 4.2: An example of ray propagation. At the plane Π_1, the ray r has coordinate $\mathbf{r}_{\Pi_1} = [y_1, d]^T$. d is the directional deviation of r from the normal $\mathbf{n}_1$, and can intuitively be thought of as the slope of r. For an empty space propagation to plane Π_2, the spatial y-component is updated by the transport along the ray direction, while the directional component is unaffected.

The matrix for ray-space transformation corresponding to the plane rotation of Π around $\mathbf{e}_1^\Pi$ is

$$\mathbf{R}_1^\theta = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1/\cos\theta & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & -\tan\theta \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.13}$$

Rotation around $\mathbf{e}_2^\Pi$ follows by symmetry.

A general rotation does not yield a linear operation in ray space. However, it can be adequately approximated if the *paraxial approximation* of ray optics is applied. A linear approximation can be used for rays which lie close to the optical axis. We will use this approximation both for rotations and for elements with curved surfaces in the next section. Figure 4.3 shows an example where the plane has been rotated around the origin of the plane Π_1.

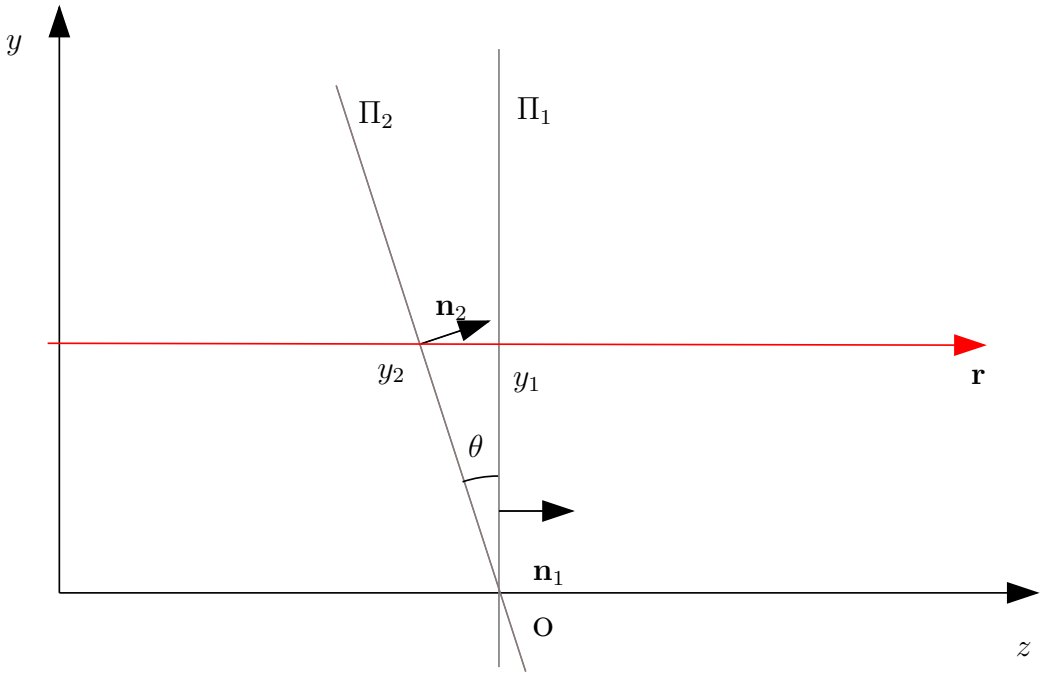


Figure 4.3: Π_2 is a plane rotated by θ around the origin of Π_1, and r is a ray traveling in the normal direction of Π_1, intersecting the planes in y_1 and y_2 respectively. To find the ray coordinate in Π_2, observe that y_2 is scaled proportionally to y_1, given by the triangle $\overline{oy_1y_2}$. As r intersects Π_2 at an angle of $-\theta$, the direction will be offset by $-\tan\theta$.

4.4.2 Lens and interface operators

In this section we present operators which map the ray space on Π onto itself.

$$\mathbf{I} : \mathcal{R}_\Pi \to \mathcal{R}_\Pi. \tag{4.14}$$

This kind of operator changes the ray direction, and can be used to model elements such as interfaces and thin lenses.

Interfaces

Interfaces are used to model light transition from one medium to another. If the materials have different refractive indices, a perturbation of the ray direction will occur when passing through the material boundary.

The matrix for a planar interface is

$$\mathbf{P}_{n_1,n_2} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & \frac{n_1}{n_2} & 0 & 0 \\ 0 & 0 & 0 & \frac{n_1}{n_2} & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.15}$$

where n_1 and n_2 are the refractive indices of the source and destination materials.

For a circularly curved interface the perturbation of the directional component depends on the plane coordinate component of the ray. The operator matrix is written as

$$\mathbf{C}_{n_1,n_2,r} = \begin{bmatrix} 1 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & \\ \frac{1}{r}(\frac{n_1}{n_2} - 1) & 0 & \frac{n_1}{n_2} & 0 & \\ 0 & \frac{1}{r}(\frac{n_1}{n_2} - 1) & 0 & \frac{n_1}{n_2} & \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.16}$$

As with the planar interface n_1, n_2 denotes the refractive indices, while r is the radius of curvature. A positive r yields a convex interface, while negative values result in a concave one.

The thin lens operator

A lens is considered "thin" if the light propagation within the lens material can be neglected. Thus, the thin lens acts by perturbating ray directions, and has the matrix

$$\mathbf{H}_f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ -1/f & 0 & 1 & 0 & 0 \\ 0 & -1/f & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.17}$$

for a focal length of f.

The matrix varies the directional component of a light field coordinate as a linear function of the plane component. Figure 4.4 depicts an intuitive example of a thin lens.

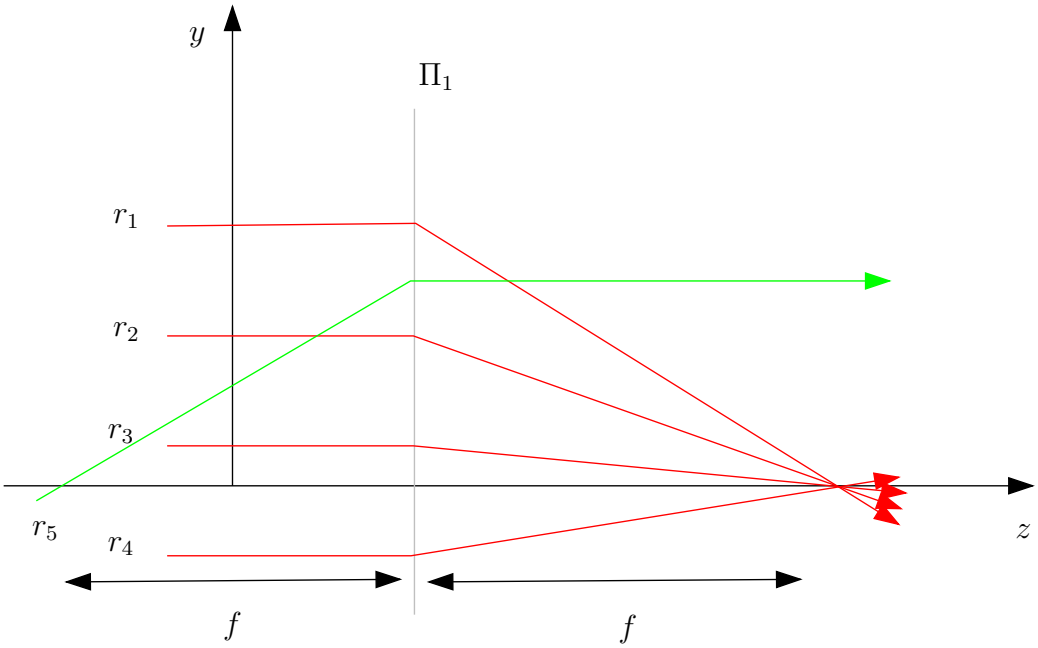


Figure 4.4: A thin lens in plane Π_1. The rays r_1 to r_4 arrive perpendicular to p_1. Their directions are perturbed depending on the distance from the origin of Π_1 so that all intersect at a distance of f in front of Π_1. r_5 on the other hand, intersects the z-axis a distance of f *in front* of Π_1 and will emanate normal to the plane. A general ray will have its directional component offset by $-y_1/f$.

4.5 Image formation

We will now show how a two dimensional image can be formed from a 4D light field .

Let

$$\Gamma = (\mathbf{o}_{\mathbb{R}^3}, \mathbf{e}_3, \{\mathbf{e}_1, \mathbf{e}_2\}) \tag{4.18}$$

be the image plane located at the world space origin, where $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ is the standard basis in $\mathbb{R}^3$. To form an image on Γ, the intensity in the image plane at a point $\mathbf{x}$ is computed using the general camera model

$$I_\Gamma(\mathbf{x}) = \int_{A_\mathbf{x}} \omega(\mathbf{d}) L(\mathbf{M}[\mathbf{x}, \mathbf{d}, 1]^T)\, d\mathbf{d}. \tag{4.19}$$

L is a light field defined on $\mathcal{R}_\pi$. $A_\mathbf{x}$ is the set of all ray directions intersecting $\mathbf{x}$ through the camera aperture, $\mathbf{M}$ is the matrix transforming from $\mathcal{R}_\Gamma$ to $\mathcal{R}_\Pi$ via any optical elements present, and ω is a weighting function used to grade rays dependent on their direction.

However general, this model is computationally expensive. A common practice in real-time computer graphics rendering is to use the pinhole camera model. This is a special case of the general model where the aperture of the camera is considered a single point in space, yielding only a single ray per point in the image plane. If we let the weight $\omega = \delta_0$, so that only rays with $\mathbf{d} = \mathbf{0}$, i.e. perpendicular to the image plane, are considered the camera integral reduces to

$$I_\Gamma(\mathbf{x}) = L(\mathbf{M}[\mathbf{x}, \mathbf{0}, 1]^T). \tag{4.20}$$

For $\mathbf{M} = \mathbf{1}$ or $\mathbf{M} = \mathbf{T_v}$ this will render an orthographic view of the light field . Perspective views can simply be rendered by including a lens matrix in $\mathbf{M}$. Thus, a perspective camera with focal length f, viewing a light field from a distance of t, would have the matrix

$$\mathbf{M} = \mathbf{T_v}\mathbf{H}_f \tag{4.21}$$

where $\mathbf{v} = [0, 0, t]$.

4.6 Wavelet compression of light field data

In this section we propose a 4D wavelet compression scheme to reduce the memory and storage requirements of the light field representation.

4.6.1 Wavelets

Wavelet compression is nowadays a well-known approach to data reduction. We will therefore just give a brief introduction to the part of the theory useful for our application and point to other sources, such as [21] and [110], for a thorough introduction.

We will assume that f is the light field and can be written as a linear combination of basis functions

$$f = \sum_{i=0}^{N-1} c_i B_i \tag{4.22}$$

where N is the number of data elements.

Wavelet theory is based on two sets of basis functions, the scaling functions

$$\phi_{k,n} = 2^{\frac{k}{2}}\phi(2^k - n), \tag{4.23}$$

and the wavelet functions

$$\psi_{k,n} = 2^{\frac{k}{2}}\psi(2^k - n). \tag{4.24}$$

The scaling function down-samples a part of the signal, *scaling* its size and preserving the low frequency, smoothed information. The wavelet function, on the other hand, is chosen so that it can represent the high-frequency information, that is, the details lost by the low-pass filtering of the scaling function. Both the scaling and wavelet functions are scaled and translated versions of a mother function, and form a multi-resolution hierarchy. We will use these properties later when constructing our data-structure for representing the wavelet compressed data. Once a set of wavelet basis functions and corresponding scaling functions are chosen, the recovery of coefficients can be thought of as a recursive process. Start with the original data and use the scaling functions to down-sample the data to half its size, computing the scaling coefficients of this level. At the same time the wavelet functions compute another set of coefficients that represent the high-frequency information.

Now we take the down-sampled data and repeat this process, ending up with half of the original data size, and a new set of coefficients. This can be done until only one scaling function is needed to represent the down-sampled data. Starting from this one scaling value, $s_{0,0}$, the scaled function of the next higher resolution is computed as

$$f^1 = s_{0,0}\phi_{0,0} + w_{0,0}\psi_{0,0}, \tag{4.25}$$

and for any other level after this as

$$f^{m+1} = f^m + \sum_t w_{m,t}\psi_{m,t} \tag{4.26}$$

where the sum over t represents the different translations on this scale. Thus f is a linear combination of the basis functions $\phi_{0,0}$ and $\psi_{k,n}$, $0 \leq k < M$, where M is the number of scaling levels. Wavelet basis functions have compact support, and for our application we consider only orthonormal basis functions. For our four-dimensional data we construct the basis functions as the tensor product of one-dimensional wavelet and scaling functions. This corresponds to the different combinations of the wavelet or scaling function along the coordinate directions.

4.6.2 Compression

For many data sets a low pass filtering will yield a good approximation of the original set, and thus the high frequency wavelet coefficients will be small. The basic principle of compression is to threshold small coefficients to zero.

For basis functions with good interpolating properties, many coefficients can be dropped without degrading data quality. This technique results in high compression ratios, storing only non-zero coefficients.

From the new sequence of coefficients, we keep only the ones that are non-zero in value, as well as their corresponding basis functions. In order to store this information and allow for both small size and fast access, we need a compact data structure.

4.6.3 The Hexadeca-tree data structure

Because a wavelet basis function has local support and is used to refine the value of a coarser scale function, the support of a basis function of side s with index k will span the region of support of the $s/2$ sized functions of index $k+1$. That is, for a basis function, $\psi_{k,t}$, of scale $k > 0$, one can find basis functions of a coarser scale $j < k$, so that their support contains the support of $\psi_{k,t}$. Just as observed in section 4.6.1, the support of the basis functions divide the original support of the data set into sub sets.

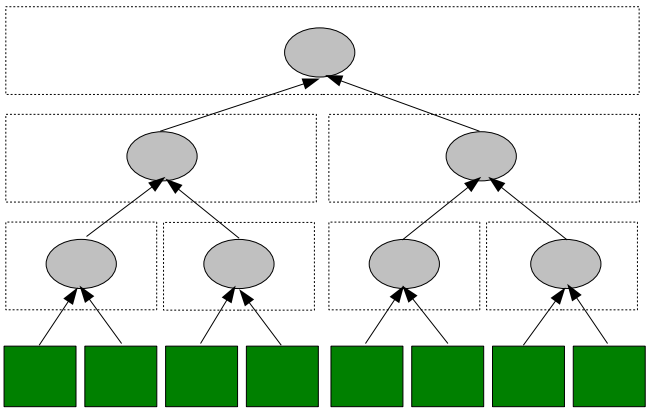


Figure 4.5: A binary tree representation. The squares represent the original data-points, and the ellipse nodes contain a basis function and the corresponding coefficient. The dashed box around the node indicates the width of the support of the function.

The basis functions form a space partitioning tree analogous to quad– and octrees in 2D and 3D. Using this tree we let each node represent all basis functions of a specific scale and translation. The child-nodes are those basis functions refining the value along their parent node's support, subdividing them. Figure 4.5 shows a binary tree for the 1D case of the situation described above. In this case, there are two basis functions: one wavelet and one scaling function per node. However, for reconstruction purposes it is sufficient to

store only the wavelet basis functions and their corresponding coefficients, except for the root node which also contains the scaling coefficient and the scaling basis function. Higher-dimensional trees will have more functions of the same support, as there are more ways to combine the basis functions. Generally, for a n-dimensional space, $2^n - 1$ wavelet basis functions will have the same support. All basis functions with the same support are represented by one tree node. In our 4D case, there are 16 different basis functions defined having common support. Of these, the 15 wavelet basis function coefficients are stored in the nodes. The 16:th, being the scaling function coefficient, is not required for reconstruction, except at the coarsest scale.

At the top level of the tree, we have the coarsest scale, represented by the scaling function coefficient and the wavelet coefficient refining it. The support of the root node being effectively the whole data size. The child-node will then subdivide the support of its parent in a hierarchical manner.

If all basis functions are multiplied with the corresponding coefficients and summed up, the result is the original function. In our 4D approach, the result is not a binary tree, but one where each node has 16 children, representing the 16 equally sized sub-cubes of a 4D hypercube. We refer to this structure as a *hexadeca-tree*.

We can reduce the memory usage of the data-structure drastically by pruning the tree in a bottom-up approach after compressing the coefficients. Because the leaf-nodes with zero-coefficients do not contribute to the reconstructed signal, they can be removed from the tree. If all children of a node are removed, it becomes a leaf and the same test can be applied again until we find a node that can not be removed. Figure 4.6 shows a simple, binary tree example.

4.6.4 Implementing the data-structure

The pruning method described above leaves us with a smaller tree than we started with. But it cannot do anything about those nodes which are not leaves and still contain many zero-valued coefficients. This case can be common when dealing with 4D data, as each node contains several coefficients. If only one of these coefficients of a leaf is non-zero, it is kept. Therefore we choose to have a data structure of dynamic size to represent the nodes of the hexadeca-tree.

A schematic of the node data-structure is depicted in Figure 4.7. For each node we store a static and a dynamic part. The static part contains the node position and two 16 bit masks. The 'Coefficient Mask' specifies which of the possible basis functions in the node have non-zero coefficients. The 'Child Mask' defines which of the children exist in a pruned tree. The dynamic part

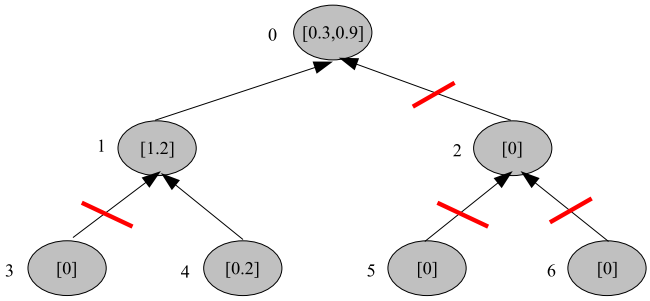


Figure 4.6: Pruning the nodes from the bottom up in a binary space partitioning tree. Starting with node 6, we can remove it because its coefficient is zero. Node 5 can be removed for the same reason, and node 2 now becomes a leaf. This node can be removed as it is 0. Node 4 is saved and node 3 is removed, but then the pruning stops since there are no more leaves to check.

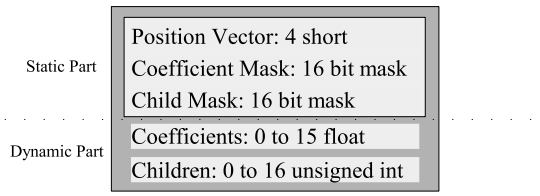


Figure 4.7: The node data-structure. The position vector and the masks make up the static part, present in every node, while the coefficient and the children sections are of dynamic size.

contains just the non-zero coefficients and index offsets to any child nodes, as indicated by the masks. Every node stores basis functions having the same support, which are the ones resulting from the tensor product producing the four-dimensional basis functions from the one-dimensional mother wavelet. The reason we only need to store 15 coefficients per node, although there are 16 different functions resulting from such an operation, is that one of them is the scaling function, which is not needed for reconstructing the data. The only scaling function coefficient needed is the one at the coarsest scale, as indicated in section 4.6.1, and this one is stored first in the array as a special case.

The compressed and pruned tree is an efficient representation for transmission and rendering. There is, of course, some potential overhead in this representation, as in the worst case all 15 coefficients will be set in the dynamic section. However this is rarely the case. The nodes are stored breadth-first in an array. This facilitates progressive decoding so that time-,

transmission- or memory-critical applications need only read and decode a part of the tree to obtain approximated rendering results. The approach is similar to the spatial orientation trees in the SPIHT codec for images [98].

4.7 Proof of concept: a matrix optics rendering system

In the previous section we have shown how matrix operators can be used to transform light fields. In this section we will discuss our implemented framework.

4.7.1 Light field data structures

We have chosen to implement two different ways of representing the light field. The first is a raw light field table, which we will call the *direct* representation, the second is the *wavelet compressed* representation suitable for huge light fields described in Section 4.6.3. The former is faster as it basically is a four-dimensional image representation. For each spatial coordinate x_1, x_2, we can look up the RGB value of any ray of direction d_1, d_2. However, as the data size of light fields often gets large, we have also implemented the alternative to use a wavelet compressed representation as described below.

4.7.2 Rendering

As seen in Section 4.5, Eq. (4.20) can be used to render an image from a light field, L. Our framework implements this equation, and computes the image formation matrix $\mathbf{M}$ by having the user specifying a chain of optical elements.

For the direct light field representation, a value lookup is straightforward, and Eq. (4.20) can be implemented directly. However for the wavelet compressed representation, the light field must be reconstructed before it can be evaluated. Instead of reconstructing the full light field we have developed an access method that takes the hierarchical structure of the wavelet tree into consideration. This method is similar to the work of Lalonde and Fournier [54], but integrates it with our operator framework.

Given some image formation matrix $\mathbf{M}$, and an image plane coordinate $\mathbf{u} \in \Gamma$, let $\mathbf{v} = \mathbf{M}\mathbf{u}$. Observe that if $\mathbf{v}$ is located in the support of a specific node in the hexadeca-tree, it is bound to be located in the support of one of that nodes' children. As the wavelet functions have a value of 0 outside their

supports, the only nodes that affect $L(\mathbf{v})$ are the single parent-child chain of nodes whose support contains $\mathbf{v}$. Thus,

$$L(\mathbf{v}) = \sum_{i \in \Omega} c_i B_i(\mathbf{v}), \tag{4.27}$$

where Ω is the set of all indices of basis functions B with support containing $\mathbf{v}$.

From (4.20) and (4.27) we then have the reconstruction expression

$$I_\Gamma(\mathbf{x}) = \sum_{i \in \Omega} c_i B_i(\mathbf{M}[\mathbf{x}, \mathbf{0}, 1]^T). \tag{4.28}$$

As the children of a node sub-divide the support, it is simple to compute Ω from the root node. Given that a node contains $\mathbf{v}$, it is only necessary to check in which of the node's children the point lies until a leaf node is reached. The reconstruction sum will thus take the form of a traversal through the space partitioning tree. This is depicted in Fig. 4.8.

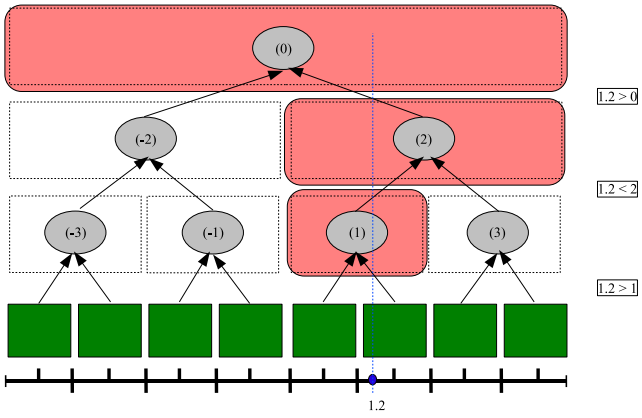


Figure 4.8: Traversal of a simple node tree, to reconstruct the value at position 1.2. The method starts at the root node, and checks in which child the coordinate lies. Nodes visited during traversal are highlighted.

We know that the support of the next basis functions in the sum will be contained within the support of the current. The number of summations needed to reconstruct a value in a direct approach will thus be the depth of the tree, which is logarithmically dependent on the resolution of the data. This straightforward method can be directly implemented as an image-space algorithm.

4.7.3 Results

We have implemented a software framework of our matrix optics representation of light fields. If required, the light field can be stored in a progressive, wavelet compressed data structure. The software renderer computes the current view in a texture and uses OpenGL to map it onto a polygon filling the screen.

For testing we have used both a synthetic light field and the 'buddha4' data set freely available from the Stanford light fields archives. The synthetic data set have a resolution of $128^2 \times 64^2$ samples, each point on the sampling plane covering an angular region of 120×120 degrees. This has proven to be a good balance between spatial and directional resolution while still keeping the original data size manageable. The buddha data set has a resolution of $256^2 \times 32^2$ samples.

The framework lets us implement and test a range of different optical setups. The most interesting application in computer graphics is of course to construct a 'camera' that lets the user interactively view a light field. Figure 4.9 shows a set of views from our synthetic light field . Figure 4.10 shows four images rendered from the buddha data set. The camera was constructed using two thin lens operators offset by propagation operators. The camera transform was modeled by a rotation and another propagation operator. Aside from light field viewing, we believe that the availability of other operators, such as interfaces, will allow for easy testing of a range of optical configurations.

We have performed renderings from both a direct and a wavelet compressed representation of the test data set. Rendering speeds are presented in Table 4.1. The machine used is a Linux-PC with an Intel Xenon 2.8GHz CPU and 2 Gigabytes of RAM.

	No AA
Direct	417 fps
Wavelet	79 fps

Table 4.1: Frame rates for rendering a $128^2 \times 64^2$ synthetic light field.

As can be seen from Table 4.1 the rendering speeds for the uncompressed data representation greatly exceeds those of the wavelet compressed data, making it a preferred choice if the whole data set can be fit into main memory. However, many light field data sets are huge, and may require compression. Nevertheless, our rendering algorithm achieves interactive framerates.

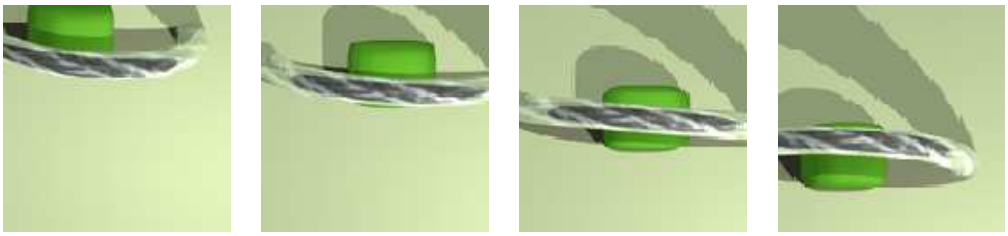


Figure 4.9: Four views of a test light field as taken by a perspective camera. The camera was constructed by combining lens and propagation operators. Placement relative to the light field is controlled by a rotation and a propagation operator.

4.8 Conclusions

We have presented a set of light field transformations inspired by matrix optics. This allows us to model optical elements such as lenses and interfaces into the image formation process. On this basis, we have implemented a real-time light field rendering framework. The framework can handle uncompressed light fields as well as wavelet compressed ones. The wavelet compression scheme builds a hierarchical representation of the light field, enabling a fast way of accessing the data that integrates well with the presented transformations.

We believe matrix optics proves an elegant solution to modeling some optical phenomena for light field rendering. The ability to freely combine the operators of different optical elements into one single matrix results in a lot of flexibility. It should also be noted that the framework is not restricted to pure image-based rendering. Many computer graphics problems can be posed as a sampling or transport of a light field. In such situations this framework can be used to model mappings of light fields through optical elements.

An interesting application for this framework would be to incorporate it into Fourier Slice Photography [89]. This should prove to be an easy but powerful extention to the FSP algorithm.

Figure 4.10: Four views of the buddha light field.

Chapter 5
Wave field analysis

In this chapter we will discuss some features of the hologram from a computer graphics perspective. In Chapter 2 we argued that the hologram is similar to a light field in that they both are methods to represent the full light traveling through an aperture.

This chapter introduces some further details regarding holography and wave front analysis. We also investigate the relationship between the wave field, as used in holography, and the light field, commonly used in computer graphics. Recent advances in computer holography and image-based rendering have opened up both fields to new applications.

5.1 Introduction

In computer graphics the light field is a common representation for image based data. During the last ten years similar techniques have been used, not only for pure image based methods, but also in ray tracing, visual hull reconstruction, shading etc. Similarly, in image processing light field data are widely used in a number of shape and light reconstruction techniques. Digital holography has recently seen a lot of development as better CCD chip technology has become available [100, 26]. While it still can not compare to traditional holography in terms of resolution, several unique features such as phase shift techniques and fast recording procedures have made it widely used.

An analysis of the relationship between light field and wave field representations may prove useful for future applications in computer generated holography, image processing and computer graphics.

In the discussions below we will use the same basic concept of the light field as used in Chapters 2 and 4. For convenience we will use angles to denote

ray directions instead of directional components when it is appropriate.

For a hologram, we will actually use the complex valued object wave instead of the intensity distribution. We will refer to this as the wave field. The reason for using the complex wave, is that we always can reconstruct a complex valued wave field from a hologram (Sect. 2.3.1). There are also digital techniques, such as phase-shift holography [123] to reconstruct the object wave without the zero order wave. Thus, we will assume that we have access to this wave field.

We will start this chapter by investigating three important concepts from optics literature. First we will recapitulate how ray directions are related to wave fronts. Second, hologram sampling theory is discussed, and third we introduce the concept of the angular spectrum from Fourier optics.

5.2 Rays from a wave perspective

As described in Chapter 2, in this thesis, a ray is considered an entity with constant radiance traveling in some direction through space. We will now ask ourselves how this is related to the wave front, which can be considered an equivalent representation. The answer is that rays always travel perpendicular to the geometric wave front [11].

A general wave disturbance, $W : \mathbb{R}^3 \rightarrow \mathbb{C}$ can be written as

$$W(\mathbf{x}) = A_0 \exp\left(\frac{2\pi i}{\lambda} S(\mathbf{x})\right). \tag{5.1}$$

In the above equation, the function $S : \mathbb{R}^3 \rightarrow \mathbb{R}$ is called the *eikonal*. The geometric wave front is defined as the surface in the disturbance of constant magnitude. This is the case when

$$S(\mathbf{x}) = K, \tag{5.2}$$

where $K \in \mathbb{R}$ is some constant.

Comparing to how the spherical and planar wave fronts where defined in Section 2.1.2, we see that this is coherent with the definition of the wave front given there. For instance, the planar wave front would have the eikonal

$$S_{\mathrm{p}}(\mathbf{x}) = [d_x, d_y, d_z] \cdot \mathbf{x}. \tag{5.3}$$

Which is the equation of a plane when S_p is held constant. $[d_x, d_y, d_z]^T$ is normal to the plane and can be considered as ray direction. Thus, a planar wave front can be considered an infinite set of parallel rays traveling in the direction of the plane normal. Using the same argument, the spherical wave

front is a set of rays traveling radial in all directions from some point in space.

Thus given a single ray, we know that there is a wave front having a local tangent plane perpendicular to the direction vector in every point along the ray. However, the ray does not tell us anything about the global curvature of the wave front. This is equivalent to the fact that a ray provides full information of light direction and radiance in a point but nothing on depth. The ray can originate from an object at any distance.

5.3 The wave field

In Chapter 2 we mostly considered wave *fronts* as the complex valued light wave from some light source, but have also talked about wave *fields* in connection with holograms. In this section we will briefly discuss the wave field and how it is related to the wave fronts of object points.

We will consider the wave field to be the complex light distribution measured in some 2D plane in 3D space. Thus, the *wave field* will represent the superposition of all wave fronts in a plane. Huygens' principle (see Section 2.1.2) tells us that this is indeed a full description of all light in the scene. We can use the information of a wave field to represent the total light for propagation and rendering. However, it is intuitively clear that this representation also looses information of the wave front locally. The wave field represents the light distribution in a plane, however, the wave front is constant valued on the eikonal (Section 5.2) which has a general shape. Thus, the wave field measures the wave front at different depths i.e. at different travel times. It does not contain any explicit information on the single wave front curvatures, and without the knowledge of scene depth it does not say anything about individual light directions.

Moreover, the wave field of a general scene is the superpositioned wave fronts from all light emitting sources, i.e. all scene points. This means that in order to reconstruct directions and depth of all points this sum would need to be deconvolved.

5.4 The angular spectrum

The Fourier transform of a wave front is called the *angular spectrum* [11, 36], and the coefficients are seen as amplitudes of plane waves propagating through space in different directions. There is a direct analogy between the direction of propagation and Fourier frequency. Below we give an intuitive

argument for this, using reasoning similar to the one given in [36]. First, let A be the Fourier spectrum of the wave front W;

$$A = \mathcal{F}\{W\}. \tag{5.4}$$

Expressing W using the inverse transform we then have, explicitly

$$W(x,y) = \iint_{-\infty}^{\infty} A(\psi,\omega)\exp\left(i2\pi(\psi x + \omega y)\right) d\psi d\omega. \tag{5.5}$$

The exponential part of the above equation can be interpreted as a plane wave by rewriting it as

$$\begin{aligned}\exp\left(i2\pi(\psi x + \omega y)\right) &= \exp\left(i\frac{2\pi}{\lambda}(\psi x\lambda + \omega y\lambda)\right)\\ &= \exp\left(i\frac{2\pi}{\lambda}(\alpha x + \beta y)\right)\exp\left(i\frac{2\pi}{\lambda}z\gamma\right)|_{z=0}\\ &= \exp\left(i\mathbf{k}^T\cdot\mathbf{r}\right)\end{aligned} \tag{5.6}$$

where $\alpha := \psi\lambda$ and $\beta := \omega\lambda$. The last step in the equation denotes a planar wave, where

$$\mathbf{r} = [x, y, z]^T \tag{5.7}$$

is the position vector, and

$$\mathbf{k} = \frac{2\pi}{\lambda}\left[\alpha, \beta, \gamma\right]^T, \gamma = \sqrt{1-\alpha^2-\beta^2}. \tag{5.8}$$

is the wave vector. Note that the magnitude of the wave vector is $\frac{2\pi}{\lambda}$ by definition. This is how we can calculate the third component of the vector in Eq. 5.8.

Now, we can use Eq. 5.5 to interpret W as a superposition of plane waves

$$W(x,y;z) = \iint_{-\infty}^{\infty} A(\psi,\omega)\exp\left(\frac{2\pi i}{\lambda}\left[\alpha, \beta, \gamma\right]\cdot[x,y,z]^T\right) d\psi d\omega. \tag{5.9}$$

The propagation direction of each component is by the definition of a plane wave determined by the wave vector. Finally, as shown in Eq. 5.6 there is a direct relation between the frequency coordinates and the propagation direction. Thus, with the help of the geometric setup in Figure 5.1, we can relate the directions as angles directly to frequencies as

$$\begin{aligned}\theta &= \cos^{-1}\psi\lambda\\ \phi &= \cos^{-1}\omega\lambda\end{aligned} \tag{5.10}$$

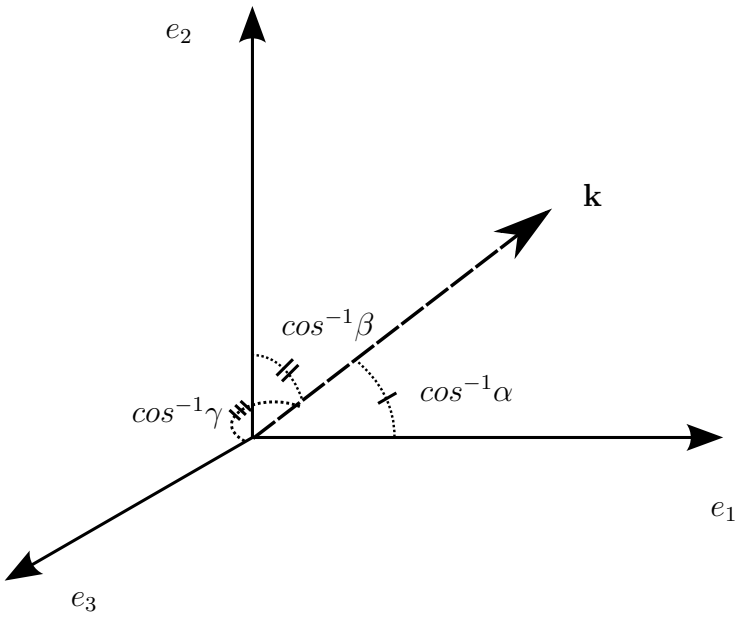


Figure 5.1: The directions of the wave vector, k, as inverse cosines of its vector components α, β, γ.

The angular spectrum of a wave front is a very powerful tool. It allows us to analyze the contents of a wave field as directional components, by expressing the superposition of arbitrary wave fronts as a collection of plane waves. We will use the angular spectrum formulation of a wave field several times in this thesis. First in this chapter, but also in Chapter 7 where we will show how light propagation can be performed in the angular spectrum representation.

5.5 The hologram and the light field

Now we are at the point where we can discuss the relationship between the hologram and the light field representations. As stated earlier we will let the wave field in the hologram plane represent the hologram information.

The light field is, as discussed in Chapters 2 and 4, a mapping from ray space to intensities. In Section 5.2 above, we saw that a ray described the local direction of a wave front. Thus, it samples the light directions at every point in the light field plane.

We assume that each point in the scene is the source of a unique spherical wave front, and that the normal of this wave front is unique for any point in space. That is, occlusion works so that any ray can be traced back through

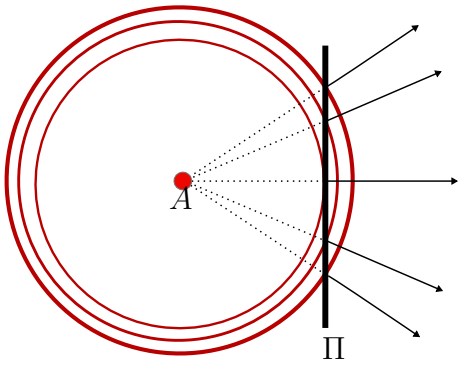


Figure 5.2: Side view of point source and wave fronts. A point source, A, with position $[x_s, y_s, z_s]^T$ emits light with amplitude a. In the plane Π we measure both the wave field and the light field. As can be seen wave fronts of different time, i.e. of different propagation distance is measured in the plane. For the light field the ray direction is always perpendicular to the wave front. The wave field is the complex value of the wave front in Π, dependent on the distance to A.

free space to a single scene point. This means that the light field can also be seen as a mapping from the local wave front directions to the wave magnitude.

The wave field on the other hand is a superposition of all wave fronts at a specific plane, Π, in space

$$W(x, y, z) = \sum_j W_j(x, y, z),\ [x, y, z] \in \Pi. \tag{5.11}$$

The question now is: given one of the representations, is there a point wise mapping to the other? We will argue that no simple local mapping between the two is possible without explicit information on the scene structure. The basic reason for this is that the wave field is a sum of wave fronts, each of which is constant valued along their eikonal, while the light field consists of rays which by definition are constant valued in the direction of propagation. As the ray direction is always normal to the wave front the curvature or phase is needed to convert back and forth. This property in turn can not be exactly reconstructed without knowledge of which distance the light has traveled before reaching the plane of the wave field. Thus, depth, i.e. scene structure information is needed.

To see this we will consider the simplest of all cases, namely a single point light source, located in a point $[x_s, y_s, z_s]^T$. This source emits a spherical wave front, and the amplitude at the source is considered to be a. We will

consider the wave field and the light field to be measured in the same plane Π at $z = 0$. The situation is illustrated in Figure 5.2. Consider the value measured in some spatial point $[x, y; z = 0]^T$. The wave field value can be computed as

$$W(x, y) = a\frac{\exp\left(\frac{2\pi i}{\lambda}r\right)}{r}. \tag{5.12}$$

Where $r = \sqrt{(x - x_s)^2 + (y - y_s)^2 + z_s^2}$. The 2D light field slice in this spatial point will be zero valued everywhere except at $\frac{x}{z_s}, \frac{y}{z_s}$ where it will be

$$L(x, y, \frac{x}{z_s}, \frac{y}{z_s}) = \|a\|^2. \tag{5.13}$$

Now, consider computing the light field from the wave field. This requires us to find the squared magnitude, $\|a\|^2$, of the wave field. Remember that the measured value we have is just the complex valued $W(x, y)$. Computing the magnitude of this number gives us an equation

$$\begin{aligned}\|W(x, y)\|^2 = aa^* \frac{\exp\left(\frac{2\pi i}{\lambda}r\right)}{r}\left(\frac{\exp\left(\frac{2\pi i}{\lambda}r\right)}{r}\right)^* \\ = \frac{aa^*}{r^2} = \frac{aa^*}{(x - x_s)^2 + (y - y_s)^2 + z_s^2}.\end{aligned} \tag{5.14}$$

which is dependent on two unknowns, the light source amplitude and position vector. Thus, we can not solve it for either of them. For the same reason, we can not know the angular coordinate of the light field, $(\frac{x}{z_s}, \frac{y}{z_s})$. Thus, without knowledge of the light source position it is not possible to compute a light field.

The other way around, going from a light field, $L(x, y, \alpha, \beta) = \|a\|^2$, to a wave field, we know the magnitude of the source amplitude and local direction of propagation (α, β). However, attempting to express a spherical wave using just this information leads to the following equation

$$W(x, y) = (c \pm di)\frac{\exp\left(\sqrt{(x - x_s)^2 + (y - y_s)^2 + z_s^2}\right)}{\sqrt{(x - x_s)^2 + (y - y_s)^2 + z_s^2}}. \tag{5.15}$$

Where we let $\|a\|^2 = (c + di)(c - di)$. Even if we assume a to be real valued and substitute $z_s = \frac{x}{\alpha} = \frac{y}{\beta}$, we still need to know the relative light source position (x_s, y_s) in order to compute the correct value. Thus, we can conclude that it is not possible to map directly between wave field and light field based on a single point measurement.

On the other hand, the above examples are generally possible to solve if we consider the measurement in several points. By setting up a system

of equations the unknowns can be solved for. Doing so, the original point source position is recovered.

However, while a system of equations is easy to set up if we know that the source is a single point light, it is not possible in the general case. This is simply due to the fact that every point in the scene must be considered a source of a spherical wave front of unknown location. Thus, to create a mapping in either direction the scene structure is needed. Either as the position and amplitude of all the point sources or as the eikonal function of the superpositioned wave fronts. These representations are equivalent for free space propagation, and both require depth reconstruction for both light field and wave front.

In the light field case, reconstructing depth is closely related to finding rays of constant intensity. Thus, the problem is equivalent to plenoptic scene reconstruction [109], and similar heuristic methods can be applied. It should however be noted that we have now moved from the concept of an exact mapping to discussing optimization methods to recover scene structure. This approach for light field to wave field conversion is further investigated in [127]. We face a similar problem when trying to construct a light field from a wave field. In general, the value of W will be an interference between a huge number of wave fronts, as expressed in Eq. 5.11. The sum must either be deconvolved to find the addends, or the phase function of W, meaning that the depth of the scene must be found. Doing so requires phase unwrapping [53, 99], shape measurement [72] or some other signal processing technique to correlate the wave front values over the wave field.

In the next Section we will introduce a method based on the sliding window Fourier transform that can be used to approximate the directional component from a wave field locally as a plane wave.

5.6 Time-frequency analysis of a wave field

In this section we will show how the local frequency spectrum of a wave front can be approximated by a light field. To do this we will first give an introduction to time-frequency analysis of wave fronts. Then we will show how the local Fourier frequencies of a wave front can be interpreted as ray directions.

5.6.1 The local frequency spectrum

If $f(x')$ denotes a function in the spatial domain we can transform it into the frequency domain by the Fourier transform

$$F(\psi) = \int_{-\infty}^{\infty} f(x') \exp\left(-i2\pi x'\psi\right) dx'. \tag{5.16}$$

In doing so however, we lose all information on the spatial localization of the frequencies. Just as the original function representation, f, does not tell us anything of the frequencies in each spatial point of the function.

In many applications however, it would be beneficial to have localization in both frequency and space. This is the goal of time-frequency analysis. Several different transforms exist for this purpose. In this thesis we will consider the short term Fourier transform, also known as the sliding window transform [24]

$$F(x, \psi; h) = \int_{-\infty}^{\infty} f(x')h^*(x' - x) \exp\left(-i2\pi\psi x'\right) dx'. \tag{5.17}$$

F is called the *local frequency spectrum* of the function $f(x')$. h is a window function localized at the origin. The role of the window function is to suppress the signal outside a region of interest and thus create a localization. Note that F is a function of both location and frequency and that the transform thus effectively creates a two-dimensional function from a one-dimensional one. This is not an operation that is generally possible however, and it is governed by the Heisenberg-Gabor inequality [24]. Intuitively this can be thought of as a conservation of information and corresponds to Heisenberg's uncertainty principle in mechanical physics: we can not get more information out of a signal or function than is there from the beginning. Thus mapping from one to two dimensions will mean a tradeoff between spatial and frequency information.

In Eq. 5.17 the tradeoff is directly influenced by the support of the window function h. If h is a Dirac pulse, and thus has infinitely narrow support, there will be perfect localization in the spatial domain but none in the frequency domain. F is reduced to $f(x)$ for $\psi = 0$. On the other hand, if $h \equiv 1$ with an infinite, constant support Eq. 5.17 corresponds to the standard Fourier transform. Thus, F will have perfect localization in frequency but none in space. A good localization in one domain has the consequence of poor localization in the other.

5.6.2 Local frequency spectrum of a wave field

The previous section considered one-dimensional functions, however the theory works in arbitrary dimensions. We can therefore compute a local frequency spectrum of a wave field. As the wave field is two dimensional, the spectrum will be four dimensional. That is, we transform the mapping $W : \mathbb{R}^2 \to \mathbb{C}$ to a new mapping $S : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{C}$, giving the complex amplitude for a coordinate in spatial-frequency space. In our case, using the short term Fourier transform we use a two-dimensional version of Eq. 5.17

$$F(x, y, \psi, \omega; h) = \iint_{-\infty}^{\infty} W(x', y')h^*(x'-x, y'-y) \exp\left(-i2\pi(\psi x' + \omega y')\right) dx'dy'. \tag{5.18}$$

Now, consider how the local frequency spectrum can be interpreted physically. We start by observing that the short time Fourier transform can be thought of as a set of normal Fourier transforms, albeit performed on sub-sets of the function. This can be seen by noting that as the Fourier integral in Eq. 5.18 does not depend directly on ψ and ω, we can consider the product of W and the window function h to be a new function

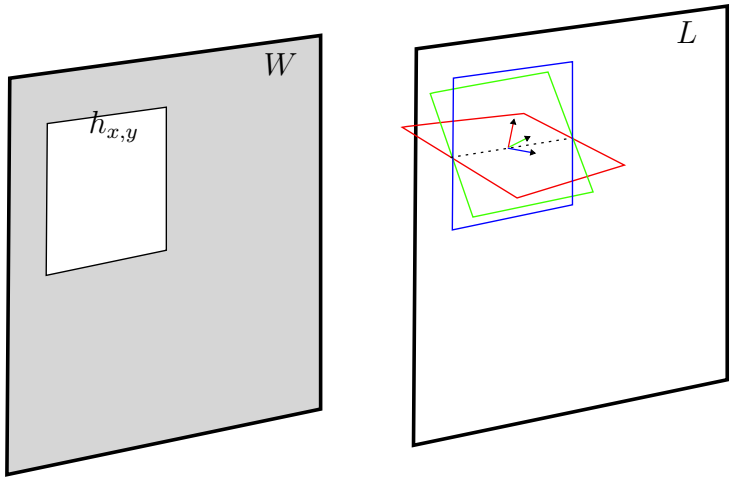
$$g(x', y', \psi, \omega) = W(x', y')h^*(x' - x, y' - y). \tag{5.19}$$

Thus for every combination of ψ and ω a unique Fourier transform of g is computed.

Moreover, a wave field masked by a window function is a subsection of the full distribution and can be treated as a new wave field. Worth noting here is that in a real world situation such a masking operation would be performed using an aperture, generating diffraction at the edges. In this theoretical setup however we will just consider it as masking out a sub-portion of a given wave front and treating it as a new wave front. Thus, by noting that Eq. 5.18 can be seen as a set of independent Fourier transforms and treating a windowed wave field as a new wave field, we can interpret the short time Fourier transform of a wave field as a set of angular spectra.

As we have seen in Section 5.4, Fourier transforming a wave front W yields the angular spectrum A. This can be interpreted as a superposition of plane waves, and the frequencies can be thought of as denoting the wave vectors, giving direction of propagation. Thus, by Fourier transforming a window of the original wave field we are efficiently approximating the incoming wave fronts by planar "patches". Each patch is approximating the wave field by the local far field.

This means that for a discretized wave field, using a $M \times N$ sized window with position (x, y), we would get $M \times N$ plane waves. As we have seen



$$A = \mathcal{F}\{h^*_{x,y} W\} \quad \rightarrow \quad L(x, y, \cos^{-1} \psi\lambda, \cos^{-1} \omega\lambda)$$

Figure 5.3: Transforming a partial wave front W to directions via the angular spectrum A. The rectangle denotes the Fourier window, $h_{x,y}$. After masking the wave front W with the window function, the Fourier transform over the resulting partial wave front can be seen as a set of plane waves traveling in directions determined by the angular coordinates (ψ, ω).

earlier, the coordinates in the angular spectrum are directly correlated to ray directions.

Given a wave field, W, defined on a plane Π, we denote the angular spectrum from a window centered at $(x, y) \in \Pi$ as $A_{(x,y)}$. From Eq. 5.6 we see that $A_{(x,y)}(\psi, \omega)$, for some coordinate pair (ψ, ω) in the Fourier domain of Π, can be interpreted as the complex amplitude of a plane wave. Thus, we can define our mapping $S : \mathbb{R}^4 \to \mathbb{C}$, computing the complex amplitude from propagating any of the plane waves to some coordinate in Π as

$$S(s, t, \alpha, \beta) = A_{(x,y)}(\frac{\cos\alpha}{\lambda}, \frac{\cos\beta}{\lambda}) \exp{(\frac{2\pi i}{\lambda}(\alpha(s-x) + \beta(t-y)))}. \quad (5.20)$$

Where (s, t) is a point in Π and (α, β) are plane wave directional angles.

Within the specific window we can thus approximate a light field as the magnitude of S

$$L(s, t, \alpha, \beta) = \|S(s, t, \alpha, \beta)\|^2 \quad (5.21)$$

where $A_{(x,y)}$ is the local angular spectrum and (ψ, ω) is the frequency coordinates. Figure 5.3 illustrates this.

From this it is intuitively clear how the window position and size affects the result. A smaller window will give us smaller patches, thus better spatial localization. However, this means that there will be higher uncertainty regarding the exact direction of the plane wave vectors. That is, the angular resolution of the light field will be low. On the other hand, a large window will allow for a high angular resolution, but all parallel ray directions within the window would have the same intensity, leading to low spatial resolution.

5.6.3 A physical interpretation

According to Goodman [36], we can compute the effect of a thin lens on a wave front by

$$W_f(x, y) = W_\Pi(x, y)P(x, y) \exp{(-\frac{\pi i}{\lambda f}(x^2 + y^2))}. \quad (5.22)$$

Where, W_Π is the wave field in the plane of the lens, and f is the focal length of the length. P is the aperture function, defined as 1 inside the opening and 0 outside.

If we assume the lens plane to lie in $z = 0$ and the viewing distance to be large enough, we can compute the complex amplitude in some plane at a distance z_p by the Fresnel approximation, introduced in Chapter 2. Inserting

Eq. 5.22 in Eq. 2.19 results in

$$U(x_p, y_p; z_p) = \frac{\exp\left(\frac{2\pi i}{\lambda} z_p\right)}{\lambda z_p} \iint_{\Pi} W_{\Pi}(x_s, y_s) P(x_s, y_s) \exp\left(-\frac{\pi i}{\lambda f}(x_s^2 + y_s^2)\right) \exp\left(\frac{\pi i}{\lambda z_p}((x_p - x_s)^2 + (y_p - y_s)\right) dx_s dy_s. \quad (5.23)$$

The polynomial in the exponent of the Fresnel approximation can be expanded to yield

$$(x_p - x_s)^2 + (y_p - y_s)^2 = x_p^2 + y_p^2 + x_s^2 + y_s^2 - 2(x_p x_s + y_p y_s)^2. \quad (5.24)$$

Inserting this result into Eq. 5.23 we find that

$$\begin{aligned} U(x_p, y_p; z_p) &= \frac{\exp\left(\frac{2\pi i}{\lambda} z_p\right)}{\lambda z_p} \iint_{\Pi} W_{\Pi}(x_s, y_s) P(x_s, y_s) \exp\left(-\frac{\pi i}{\lambda f}(x_s^2 + y_s^2)\right) \\ &\quad \exp\left(\frac{\pi i}{\lambda z_p}(x_p^2 + y_p^2 + x_s^2 + y_s^2 - 2(x_p x_s + y_p y_s)\right) dx_s dy_s \\ &= \frac{\exp\left(\frac{2\pi i}{\lambda} z_p\right) \exp\left(\frac{\pi i}{\lambda z_p}\right)(x_p^2 + y_p^2)}{\lambda z_p} \iint_{\Pi} W_{\Pi}(x_s, y_s) P(x_s, y_s) \\ &\quad \exp\left(-\frac{\pi i}{\lambda f}(x_s^2 + y_s^2)\right) \exp\left(\frac{\pi i z_p}{\lambda z_p}(x_s^2 + y_s^2)\right) \\ &\quad \exp\left(\frac{-2\pi i}{\lambda z_p}(x_p x_s + y_p y_s)\right) dx_s dy_s. \quad (5.25) \end{aligned}$$

Now, by choosing our reconstruction distance to be the same as the focal length, $f = z_p$, the first and second exponential inside the integral will cancel out and we have

$$U(x_p, y_p; z_p) = K(x_p, y_p, z_p) \iint_{\Pi} W_{\Pi}(x_s, y_s) P(x_s, y_s) \exp\left(\frac{-2\pi i}{\lambda z_p}(x_p x_s + y_p y_s)\right) dx_s dy_s. \quad (5.26)$$

In the above equation we let K denote the phase function in front of the integral. If we perform the variable substitution $u = \frac{x_p}{\lambda z_p}$ and $y = \frac{y_p}{\lambda z_p}$ Eq. 5.26 can be interpreted as a Fourier transform of $W_{\Pi}(x_s, y_s)P(x_s, y_s)$ weighted by the function K.

In the following we will assume a simple imaging system where the lens focus the incoming light onto an image, measured on a distance z_p, in the

same plane as U. As noted in Section 2.2, the image intensity is proportional to the magnitude of the wave front, and from Eq. 5.26 we have

$$\begin{aligned} I(x_p, y_p) &= \|U(x_p, y_p; z_p)\|^2 \\ &= k\| \iint_{\Pi} W_{\Pi}(x_s, y_s) P(x_s, y_s) \exp\left(\frac{-2\pi i}{\lambda z_p}(x_p x_s + y_p y_s)\right) dx_s dy_s \|^2 \end{aligned} \tag{5.27}$$

where

$$k := \frac{K(x_p, y_p, z_p) K^*(x_p, y_p, z_p)}{\lambda^2 z_p^2} = \frac{1}{\lambda^2 z_p^2} \tag{5.28}$$

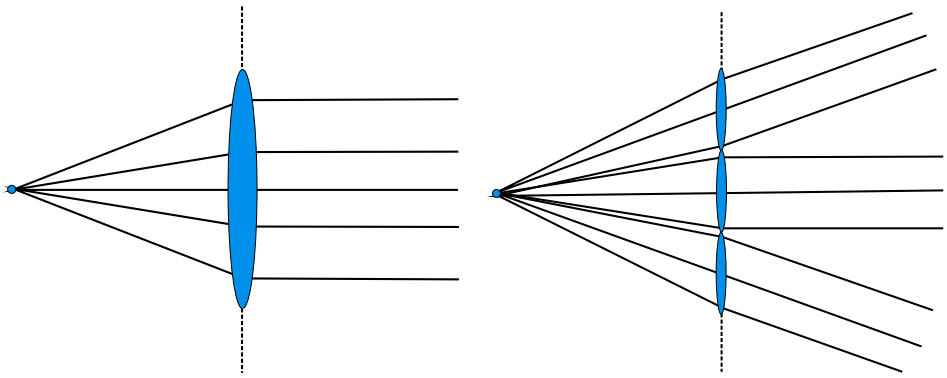
is constant over the whole image.

Thus, this makes it possible to interpret the windowed Fourier transform of a wave field, as a collection of lenses spread over a plane. Much like the lenticular array described by Ng et al. in their work on the plenoptic camera [88].

The Fourier window corresponds directly to the aperture function. The size of the aperture in a real imaging system affects the depth of field. So does the windowing function. Ultimately, one would like to have an idealized, pinhole model to have as large depth of field as possible. This is however not possible in reality as we are working with discretized wave fields. In this case, using a too small aperture masks out most of the samples leaving only a few coefficients. Again, we have the tradeoff between spatial and angular resolution.

The Fourier transform itself will perform an operation similar to that of a lens and focus light into bundles of parallel rays. Considering this is what makes it apparent that a collection of lenses translated in a plane is needed. This corresponds to sliding the Fourier transform. Consider Figure 5.4(a) where light from a point located on the optical axis at focal distance is imaged through a lens. Thus rays from this point will leave the lens parallel, destroying all information on direction. If we are using a collection of lenses however, as in Figure 5.4(b), the translation of each lens' individual optical axis relative to the object point results in an approximation of the light direction.

Visualizing the process in this way, makes it apparent that the numerical method suggested in the previous section is similar to the production of holographic Stereograms [96, 10, 39], and also shares the same traits. However, in contrast to this traditional physical process, our approach is numerical.



(a) Imaging using a single lens. (b) Imaging using multiple lenses.

Figure 5.4: Imaging using single (a) and multiple (b) lenses. A single lens transforms all rays emerging from a single point to parallel rays. Using a number of lenses tiled on a plane, the resulting ray bundles have an unique direction depending on the lens position.

5.6.4 Test results

We have performed a test on a real phase shift digital hologram. It depicts a chess knight at a distance of 0.46 m from the hologram plane. Four holograms were recorded with phase shifts of 0, $\frac{\pi}{2}$, π and $\frac{3\pi}{2}$ using a 532 nm laser. Using phase shifting techniques as described in [123] we reconstructed a 512×512 wave field with pixel spacing 9 μm.

To exemplify the method, Figures 5.5(a) and 5.5(b) show two reconstructions from 128×128 windows from different positions within the wave field. The large windows means that we have a good angular resolution. Each pixel in the reconstruction can be viewed as a light field direction sample. This is the reason for the relatively good reconstruction quality. These Figures can be compared to a reconstruction of the full hologram, as shown in Figure 2.4 in Chapter 2. There are however very little spatial differences between the two images. This is due to two facts: First, we have relatively large windows, meaning that huge areas of the incoming wave fronts are assumed to be linear. This makes sense, as the distance 0.46 m is large compared to the hologram side, $512 \times 9\mu\text{m} \approx 0.5$ cm. At this distance, the curvature of spherical wave fronts can be approximated as planes over a small area.

To exemplify what happens if we use a smaller window, Figures 5.5(c) and 5.5(d) depict the result from using 32×32 samples. As can be seen, the quality is much worse. This is due to the fact that we have less ray direction samples. This leads to a situation where a sample has a larger "foot print" as it will have to represent a pencil of directions corresponding

to the uncertainty in angular position. Thus the degradation of resolution in the reconstructions and the blurring we see in the Figures. Note that the speckle noise has also become larger.

Thus, we conclude this test by noting that it is possible to compute a light field from a wave field by using time-frequency analysis. By using the fractional Fourier transform we can approximate the wave fronts as piecewise linear via the angular spectrum, thus interpreting the coefficients as rays originating from a planar wave front locally. This can be used to build a light field as in Eq. 5.21. However, due to the unavoidable uncertainty principle of time-frequency analysis balance has to be kept between spatial and angular resolution.

5.7 Summary and conclusion

In this chapter we have compared the light field and the wave field. We have found that both representations capture the full visual information of a scene through free space. However, the fundamentally different way in which depth information is implicitly encoded makes it hard to convert between the two without first reconstructing the original scene.

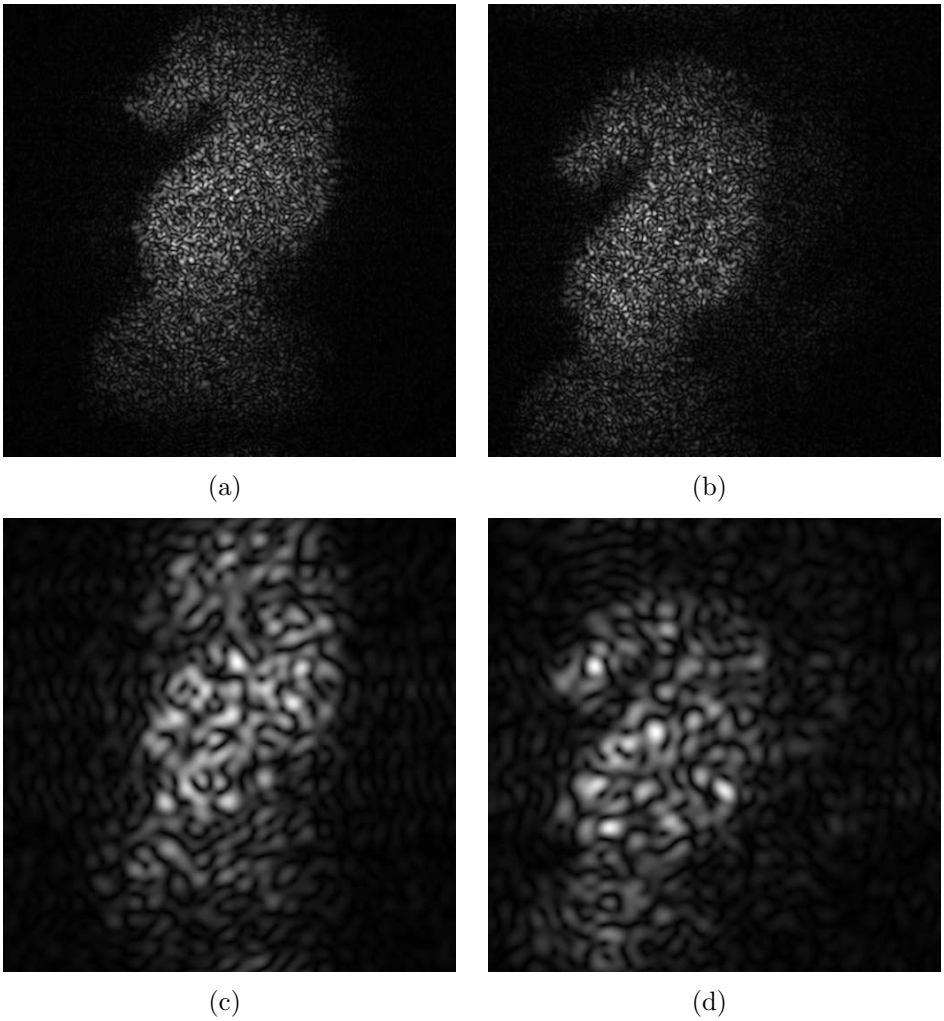
We have also shown how the angular spectrum can be used to interpret a hologram as a collection of plane waves. Extending this, we proposed to use the windowed Fourier transform to approximate light field information from a wave field. As a local approximation, plane waves are used. The depth information for these patches is lost. Thus, this method should produce adequate results for far field approaches, but loose some information in the near field. Our result is similar to the transform presented by Kartch [51]. In this work, Kartch shows how to map between a holographic stereogram, represented using hogels, as introduced by Lucente [68], to a light field. Just as in the windowed Fourier case, the hogel represents the local frequency spectrum around some point in the hologram.

In theory, the mapping could also be inverted and used to approximate a hologram from a light field. We have not tried this approach ourselves, but Abookasis and Rosen have reported success in synthesizing holograms from multiple view images [1]. In their method, the images are weighted using an exponential function dependent on viewing direction. This should correspond to our interpretation of the directions as plane waves, and hints at the methods being similar.

From this we may gather that the full hologram should be viewed as a complement to the light field . The two representations share many common traits, however there seems to be no simple way of mapping between the two

without reconstructing the complete object wave at a specific time, and thus also reconstruct the depth.

A one-way wave field to light field transform should however be possible by simulating a camera viewing the object scene through the hologram aperture. There may also be powerful time-frequency methods available that can do a good job of correlating the values of a wave field to reconstruct the wave fronts and thus the light field. In this thesis we have shown how the windowed Fourier transform may be used, however in the future it would be interesting to apply other methods. One specific transform we would like to have a closer look at is the Wigner distribution [9].



(a) (b)

(c) (d)

Figure 5.5: Reconstructions from the windowed Fourier transform. (a) Window size 128×128 positioned in the upper left corner of the hologram. (b) Window size 128×128 positioned in the lower right corner of the hologram. (c) Window size 32×32 positioned in the upper left corner of the hologram. (d) Window size 32×32 positioned in the lower right corner of the hologram.

Chapter 6

GPU-based computer generated holography

This chapter introduces a method to efficiently use programmable graphics hardware for holographic interference pattern generation. While GPU based methods have been proposed earlier, as discussed in Chapter 3, our method is efficient and does not require the Fresnel approximation. Further more; our dynamic shader generation allows us to tailor the program to handle large models.

6.1 Introduction

Recent advances in three dimensional display technologies have shifted attention to the possibility of true holographic displays. One way of realizing this is through the use of Spatial Light Modulators (SLMs). The performance and resolution of SLMs are increasing rapidly, and so consequently, holographic displays of increasing spatial bandwidth product can be built. To drive these, Computer Generated Holograms (CGHs) rendered from point models are commonly used. However, generating a hologram from a set of point samples is a computationally intensive task. In this chapter we propose a method that takes advantage of parallel graphic processing units (GPUs) to perform the computation. Although development of special purpose hardware for CGH rendering has been reported [118, 49, 101, 102, 50], this type of equipment is expensive and must be custom built. In contrast, graphic boards are readily available today, and are especially constructed to accelerate numerical operations frequently used in computer graphics. Thus, a parallel GPU system is an attractive alternative to both expensive custom-built hardware and to much slower CPU-based approaches.

6.2 The CGH model

In this section we will discuss the steps needed in order to render a holographic intensity pattern from a 3D model. The target system is described in Chapter 2, Section 2.4.3. The target SLM has a maximum spatial resolution of 1920×1200 and an intensity resolution of 8 bits.

6.2.1 Using a point based 3D model

The goal of our method, as described above, is to compute the holographic pattern from a 3D model. We consider the scene to be made up from a set of points written as

$$\mathcal{P} = \{p_1 ... p_N \in \mathbb{R}^3\}. \tag{6.1}$$

An example of such an object is given in Figure 6.1 depicting a set computed from the Stanford bunny. We assume that the material is diffuse so that light reflected off the model can be considered scattered in all directions.

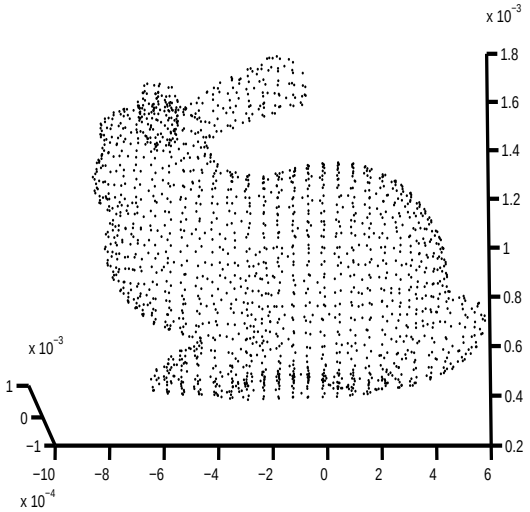


Figure 6.1: A point set generated from the Stanford bunny. This version of the model contains about 1800 points.

Given these preconditions we can treat the object as being made up from a number of self luminous points. Each points acts like a spherical light source and

$$a_j \in \mathbb{C},\ j = 1...N \tag{6.2}$$

is the complex amplitude of point p_j.

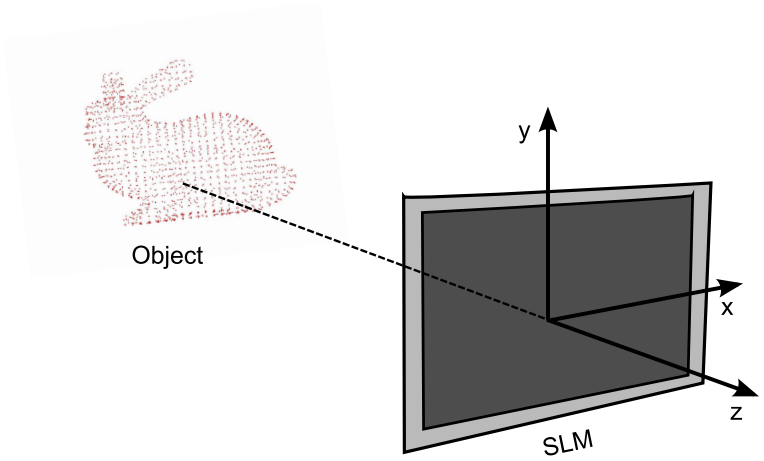


Figure 6.2: We use a coordinate system where the SLM lies in the xy-plane and is centered at the origin. The object is assumed to be placed at some distance along the negative z-axis.

From the definition of a spherical wave front (Eq. 2.13) the complex amplitude of point j at some position $\mathbf{d}$ in space can be written as

$$W_j = \frac{a_j \exp\left(\frac{2\pi i}{\lambda}\|\mathbf{d} - \mathbf{p}_j\|\right)}{\|\mathbf{d} - \mathbf{p}_j\|}. \tag{6.3}$$

6.2.2 Wave field formation

In order to compute the hologram pattern we need to compute the superpositioned wave field in the plane of the SLM. We adopt the coordinate system shown in Figure 6.2. The SLM lies in the xy-plane centered at the origin. The object points are assumed to lie somewhere along the negative z-axis.

From the superposition principle we know that the total object wave at position $\mathbf{d} \in \mathbb{R}^3$ is the sum of all individual point sources. Using Eq. 6.3 we have

$$W_o(\mathbf{d}) = \sum_{j=1}^{N} \frac{a_j \exp\left(\frac{2\pi i}{\lambda}\|\mathbf{d} - \mathbf{p}_j\|\right)}{\|\mathbf{d} - \mathbf{p}_j\|}. \tag{6.4}$$

Thus by computing this sum for each pixel position in the SLM we have the complex disturbance in this plane. The intensity-valued hologram pattern is then the magnitude of the interference between this object wave and a reference wave as described in Section 2.3.1.

6.2.3 The bipolar model

While it is possible to compute the exact holographic pattern by simulating the physical process directly we will employ a simplified method suggested by Lucente [67].

Assuming that we have computed the object wave W_o using Eq. 6.4 the hologram is (Eq. 2.29)

$$I = \|W_r + W_o\|^2 = W_r W_r^* + W_r W_o^* + W_o W_r^* + W_o W_o^*. \tag{6.5}$$

Where W_r is the reference wave.

Following the argumentation of Lucente we identify the terms of Eq. 6.5. $W_o W_o^*$ is the object wave interfering with itself. Thus this term includes the interference between all pairs of object points. Lucente argues that this term is unnecessary for reconstructions and commonly also produces artifacts in the reconstructed wave field. $W_r W_r^*$ denotes the reference wave self interference. This is a major source of the zero term in the reconstructed wave front, and serves no direct purpose for the object wave.

Removing these terms from Eq. 6.5 we have the following approximation

$$I \approx I_b = W_r W_o^* + W_o W_r^* = 2\Re(W_o W_r^*). \tag{6.6}$$

I_b is called the bipolar intensity distribution as the intensities can have both positive and negative values.

A negative intensity neither makes sense in reality, nor can be mapped to a SLM. However, it is clear that the original intensity distribution in Eq. 6.5 is positive everywhere and thus that I_b acts as an offset from two removed terms. As the intensity distribution will need to be scaled to fit the 8 bit dynamic range of the SLM we can actually also offset the bipolar distribution before scaling to fit into a positive 8-bit range.

We will assume the reference wave W_r to originate in a light source with distance far enough for it to be treated as a plane wave over the SLM area. Thus $W_r = a_r \exp(\frac{2\pi i}{\lambda}\phi_r)$. Inserting this together with the expression for W_o (Eq. 6.4) in Eq. 6.6 we have

$$\begin{aligned} I_b(\mathbf{d}) &= 2\Re((a_r \exp(\frac{2\pi i}{\lambda}\phi_r))^* \sum_{j=1}^{N} \frac{a_j \exp(\frac{2\pi i}{\lambda}\|\mathbf{d}-\mathbf{p}_j\|)}{\|\mathbf{d}-\mathbf{p}_j\|}) \\ &= 2\Re(a_r \sum_{j=1}^{N} \frac{a_j \exp(\frac{2\pi i}{\lambda}\|\mathbf{d}-\mathbf{p}_j\| - \phi_r)}{\|\mathbf{d}-\mathbf{p}_j\|}) \\ &= 2a_r \sum_{j=1}^{N} \frac{a_j \cos(\frac{2\pi}{\lambda}\|\mathbf{d}-\mathbf{p}_j\| - \phi_r)}{\|\mathbf{d}-\mathbf{p}_j\|}. \end{aligned} \tag{6.7}$$

Denoting $\mathbf{p}_j$ by the coordinates $[x_j, y_j, z_j]^T$ and the SLM coordinate $\mathbf{d} = [x, y, 0]$ we arrive at our final bipolar equation

$$I_s(x,y) = 2a_r \sum_{j=1}^{N} \frac{a_j \cos(\frac{2\pi}{\lambda}\sqrt{(x-x_j)^2+(y-y_j)^2+z_j} - \phi_r)}{\sqrt{(x-x_j)^2+(y-y_j)^2+z_j}}. \tag{6.8}$$

A further common simplification of (6.8) is to use the Fresnel approximation, i.e, the object size is assumed to be much smaller than the distance between the object and the hologram plane; $(x_{\max}, y_{\max}) \ll z_{\text{object}}$. This approximation is made for numerical reasons as square roots can be computationally expensive. However, as distance computations occur frequently in computer graphics algorithms, it is reasonable to assume that GPUs are optimized for these operations. This is also confirmed by practical experimentation, and thus we work directly with Eq. 6.8 and do not have to restrict ourselves to the case of Fresnel diffraction.

6.3 Programmable graphics hardware

Special-purpose graphics hardware was developed to perform the computationally intensive transformations needed by many of today's CAD and computer games applications. The GPU architecture is based around the *rendering pipeline*, a step-by-step sequential approach that allows for a very fast and, at some stages, parallel hardware implementation. On modern GPUs the pipeline is also programmable. This allows software routines, so called shaders, to be uploaded to the graphics card and inserted to bypass fixed functionality at certain places in the pipeline. The programs can be written in a high-level language and sometimes use such features as loops and conditional branching. It is important to note however, that although the shaders are written in high level programming languages and allow for much freedom, they are still just part of the graphics pipeline, and have to process data in an input-output fashion. Thus, many problems and algorithms have to be reposed to be efficiently implemented on a GPU. Although this might seem restrictive from a programming point of view, it is also what makes the GPU a special-purpose processor and accounts for much of its increased computational efficiency over standard CPUs. Therefore, special care must be taken when using a GPU for general purpose problems. Complex or ill-posed shaders can clog the pipeline down and actually slow down the processing speed.

A huge class of algorithms, for instance integrals and sums, rely on repeated processing of the data, i.e. looping. This also includes the CGH algorithm. Whilst loops are supported within the newer shader standards, they

can only be used in a rather restrictive manner when programming a GPU. The reason for this is that very few GPU architectures support true branching in hardware. Keeping an efficient branching mechanism can be very hard in a pipeline design. In order to still serve some looping functionality, the shader compiler usually employs loop-unrolling. I.e. the instructions that will be repeated within the loop are written out in a long sequence before compile time. However, this sequence will of course require more program memory as well as temporary registers. We have found that on the GeForce 6 and 7 series cards that were used for testing, the number of free memory registers limits the number of consecutive operations. Depending on how many other instructions that are performed, this number typically is in the range of one to two hundred.

Thus, due to compiler and hardware limitations it is often only possible to loop a few times while performing operations that make heavy use of GPU registers.

A different approach than to perform the loop inside a shader is to repose the problem so that it uses multi-pass rendering of the entire graphics pipeline. That is, the result is computed by repeatedly running the entire graphics pipeline and summing up the results. One such pass will create some overhead, as many operations of the pipeline may have to be executed that are not needed for the computation. However it allows for many more sequential operations than a loop in a single shader. In the next section we will present how we use a combination of in-shader looping and multiple passes to create a fast and extendable CGH rendering implementation.

6.4 Implementation

In this Section we will discuss how we implement the bipolar method in Eq. 6.8 given the programming constraints discussed in Sect. 6.3. The type of computation hardware used is an NVIDIA GeForce 7800 GTX graphics card, and a Brillian 1080 reflective spatial light modulator for display. The SLM has a maximum resolution of 1920×1200 elements and a pixel pitch of 8.1 μm. The software was implemented using the OpenGL graphics library and the GPU shaders were written in OpenGL Shading Language (GLSL) [97].

There are two types of shaders in modern day GPUs: the vertex and fragment shaders. As the CGH algorithm operates on all pixels in an image most of the logic will lie in the fragment program. Thus the word shader from here on we will typically refer the fragment shader unless otherwise mentioned.

6.4.1 A double loop approach

Analyzing the bipolar CGH equation, it is clear that a summation of N terms is needed for each pixel of the SLM. A standard CPU implementation would solve this by a simple loop. On a GPU this would lead to an algorithm like the one outlined in Fig. 6.3(a) performed for every fragment in the pipeline, i.e. the whole summation would be performed within a single fragment shader program. Due to the restrictions discussed in Sect.6.3, a somewhat different strategy is necessary. Although loops are available on modern graphics hardware, the size is restricted to a few hundred iterations. This is too few for anything but the simplest objects.

An alternative, used frequently in computer graphics before looping in shaders was available, is to use multiple rendering passes and functionality known as *blending*. Using blending, the intensity, as output from sequential executions of the graphics pipeline to the screen image, is combined using a linear operation. This means that the pipeline is run once for every term in the sum and the results are accumulated, see Fig. 6.3(b). However, this method also has drawbacks as there in general is an overhead associated with running multiple passes of the pipeline. For a small number of points it is thus faster to perform the operations in a shader program.

We combine both methods to create an algorithm that can handle an arbitrary number of points while still taking advantage of the GPU for hardware acceleration. Fig. 6.3(c) depicts our algorithm. In short, for a set of N object points we perform S summations in the shader while performing P passes so that $N = SP$. Thus we have split the single loop into two nested ones. This allows us to find a balance between the number of operations performed in the shader for each pass and the number of passes, thus optimizing performance.

6.4.2 Program structure

As mentioned above, we implemented our shader programs using GLSL. The GLSL code is not compiled until runtime which serves as a great advantage to us as it makes it easy to write a hologram shader generator. In order to adopt our program to different kinds of hardware we generate a suitable fragment shader on the fly. The basic shader code is available as a program template that contains the operations that should be carried out before and after the loop. However instead of the loop we have inserted a special token that will be replaced with the correct code when our program executes. The shader code needed to compute the bipolar contribution from one point is stored in a string. At the beginning of our program a custom shader is generated by

Algorithm 6.1: Generating the fragment shader by manually unrolling the rendering loop.

Data: Shader program template (prog), loop code (term), the number of model points(N)
Result: A tailored CGH shader (prog) compiled and loaded to the GPU, the number of shader calls needed (P)

```
1  Query the GPU for maximum number of registers and store the
   result in Rmax;
2  Let Ruse = number of registers needed by the fixed code in the
   template;
3  Let S = Rmax - Ruse;
4  Let P = N/S+1;
5  for i = 1 to S do
6  |   Find the loop token in prog ;
7  |   Replace the token with the code string in term;
8  |   if i != S then
9  |   |   Insert loop token after the code;
10 |   end
11 end
12 Compile and load the shader program to the GPU;
13 return P
```

Algorithm 6.1. The token in the template is replaced by the correct number of copies of the loop code string.

Pseudocode for the basic steps in our main implementation can be found in Algorithm 6.2. The object point coordinates are loaded as textures to the GPU. In order to trigger rendering a point primitive is rendered.

This version of our code assumes hardware support for float blending. As some older hardware architectures support 16 bit float buffers, but not float blending, we have also implemented support for blending through a so called ping-pong approach. The basic principle is very simple; the result from the previous pass is sent as a texture to the current rendering pass. Thus, the old value can be added to the new one in the fragment shader. The two buffers are interleaved, so that the first time buffer A acts as texture and B is the rendering target. The next pass B contains the most recent version of the sum and is used as texture while A is overwritten, and so on.

Once the rendering is started the simple fragment shader described in Algorithm 6.3 is called for the vertex connected with the rendered point primitive. This shader is really simple, however it performs one important

Algorithm 6.2: The basic steps of our program.

Data: The object points (*points*)
Result: Holographic intensity pattern in screen buffer

```
1  Check for hardware support ;
2  Create a 16 bit FP frame buffer ;
3  Turn on FP blending ;
4  Set up a floating point texture rendering target ;
5  Call Algorithm 6.1 ;
6  Set up orthographic projection ;
7  for j = 1 to P do
8  |    Load points[(j − 1)(N − P) : j(N − P)] as a 1D texture ;
9  |    Render a point primitive in the middle of the screen ;
10 end
11 Normalize the frame buffer to make sure the results lie in the
   positive 8 bit range ;
```

Algorithm 6.3: The general structure of our vertex program. Note that the point primitive size is set to cover the whole screen, ensures that the fragment shader is executed for every pixel.

Data: Vertex coordinate data
Result: Sends on vertex information to the graphics pipeline

```
1  Transform the vertex to SLM coordinate system ;
2  Set point size to cover the whole screen ;
3  Make sure to forward the texture coordinates to the fragment
   shader ;
```

task, which is to set the size of the rendered primitive to the same size as the screen. This causes the GPU to call the fragment shader for every point in the frame buffer.

Now the turn has come to our custom made fragment shader. The pseudo-code of an expanded shader can be found in Algorithm 6.4. For simplicity we have assumed the amplitude of all point sources to be 1 and the reference wave phase to be 0. The unrolled lines of code repeatedly sum the bipolar intensity distribution from each point stored in the texture memory. Thus, as the shader is called once for each fragment in the target buffer, the total bipolar contribution from all S points in all pixels of the image is computed.

As the main program has a loop rendering P points, each with a different texture containing S points, the total N point bipolar distribution is com-

Algorithm 6.4: Fragment shader. The local image plane coordinate is reprojected to the global coordinate system. Then the bipolar intensity from the S object points stored in the texture is computed sequentially and accumulated in sum. Instead of a loop, the code on lines 4 to 8 have been repeated S times.

Data: Point coordinates as a 1D texture (points), Local fragment coordinate (x)
Result: The accumulated bipolar intensity for fragment x

```
1   Let sum = 0;
2   Reproject x to global coordinate system and store result in d ;
3   Let texturePos = 0;
4    // Generated code chunk 1
5   p = texture_lookup(points, texturePos);
6   r = distance(d, p) ;
7   sum = sum + a_j cos((2π/λ) r) / r ;
8    texturePos = texturePos + 1 ;
    ...
508 // Generated code chunk S
509 p = texture_lookup(points, texturePos);
510 r = distance(d, p) ;
511 sum = sum + a_j cos((2π/λ) r) / r ;
512 texturePos = texturePos + 1 ;
513 fragmentColor = sum;
```

puted at the end. The only thing that remains is to offset the distribution to positive values and normalize the intensity values to lie in the 0...255 8-bit range.

The implementation works with 16-bit floating point precision which can be handled completely in hardware by modern graphics processors. Thus, the method is not as prone to numerical errors as earlier methods were due to the fix-point 8-bit representation of the intermediate results.

6.4.3 A note on SLI

Scalable Link Interface (SLI) is a technology that allows several graphics boards to be connected in parallel in order to increase rendering speeds. This technology automatically distributes the rendering tasks over the GPUs and

the load balancing is handled at the driver level. We have confirmed that our implementation works in single GPU mode as well as on a two GPU SLI-equipped system.

6.5 Results

We have tested our software running on a GeForce 7800 GTX PC connected to a Brillian 1080 SLM. The PC was equipped with an extra GeForce 7800 GTX board for dual GPU SLI rendering. For optical reconstruction we used a 10 mW, 633 nm HeNe laser. The maximum resolution of the SLM is 1920×1200 pixels. We have performed tests at both full, and half, 960×600, resolution. The test objects used had from 100 to 10000 points.

Resolution	960×600	960×600 SLI	1920×1200	1920×1200 SLI
100 pts	0.017 s	0.017 s	0.033 s	0.033 s
500 pts	0.094 s	0.041 s	0.327 s	0.172 s
1000 pts	0.161 s	0.083 s	0.619 s	0.318 s
2000 pts	0.307 s	0.161 s	1.204 s	0.630 s
10000 pts	1.486 s	0.785 s	5.848 s	3.115 s

Table 6.1: Rendering times; 100 **to** 10000 **points for two SLM resolutions using single and dual GPUs.**

Table 6.1 and Figure 6.4 presents table and corresponding plot of the times in seconds to render an interference pattern using our software for a few test sets. Note that the rendering time scales almost perfectly linearly with the number of points for both resolutions. With the exception of the object set consisting of just 100 points, where the GPU can do the computations in just one pass, using the SLI setup with two graphic boards effectively doubles the performance. Moreover, as the number of output pixels is quadrupled from the lower 960×600 to the higher 1920×1200 resolution, so are the rendering times. This demonstrates that the algorithm is suitable for graphics hardware. The rendering time is directly dependent on the number of input and output points. There is no extra overhead associated with increasing either the number of points or the resolution of the SLM. As can be seen from Figure 6.4 we achieve interactive frame rates for fairly densely sampled objects. Two thousand points are rendered in just 0.6 seconds at full SLM resolution, and in less than 0.2 seconds at half resolution. Smaller objects, in the order of a few hundred points, can be rendered at real time frame rates of 30 frames per second and above. Figure 6.5 shows photographs of off-axis

reconstructions. The models have 200, 1800 and 8000 points respectively; render times are 15, 2 and 0.4 frames per second on a 1920×1200 SLM.

6.6 Conclusions

In this chapter we have presented a novel method for using a commodity graphics processor to generate holographic patterns for SLM-based holographic displays. The algorithm is designed to fit the pipelined model of the GPU and takes high advantage of the parallel architecture. Interactive rates of 10 frames per second are achieved for one thousand object points at a resolution of 960×600 pixels.

We have shown that the rendering time increases linearly with the number of points and the SLM pixel count. Thus, the algorithm fits the programming architecture imposed by the hardware, and can be expected to perform proportionally better with future generations of GPUs. We have also shown that the performance can be doubled by using two graphic boards in a parallel configuration. Today's SLI standard allows for up to four GPUs to be configured in this way, which potentially would quadruple the speed.

Future research on this project will consider color, occlusion and more complex primitives. Since GPUs are designed to work with multi-channel color data, it should be trivial to adopt our program to render three patterns, each one for a different wavelength, without additional computational load. These patterns could then be multiplexed to create a color hologram.

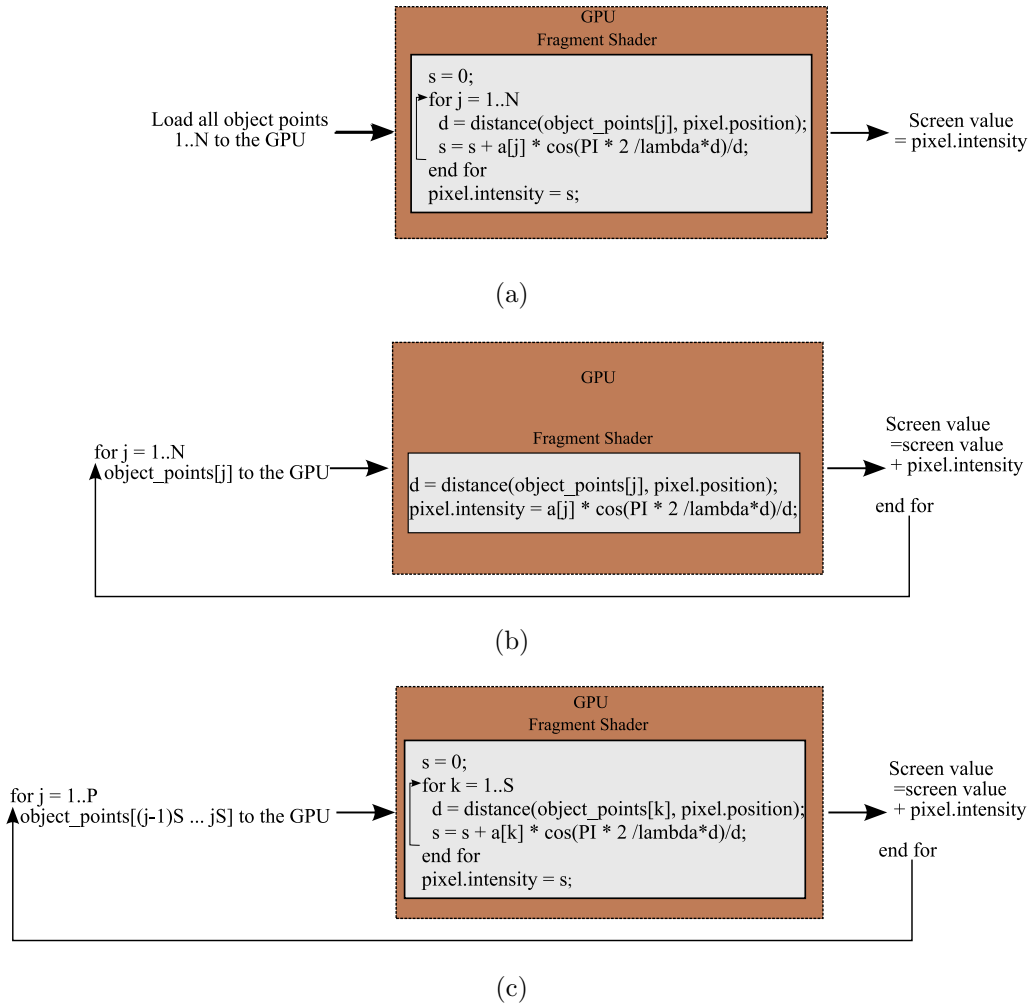


Figure 6.3: Three different algorithmic layouts for computing the distribution from N object points on a GPU. The fragment shader is called once for every pixel. (a) Process all points in one pass. Loop in shader. (b) Process one point per pass. Multi-pass. (c) Process $N = PS$ points. P passes and S summations in shader.

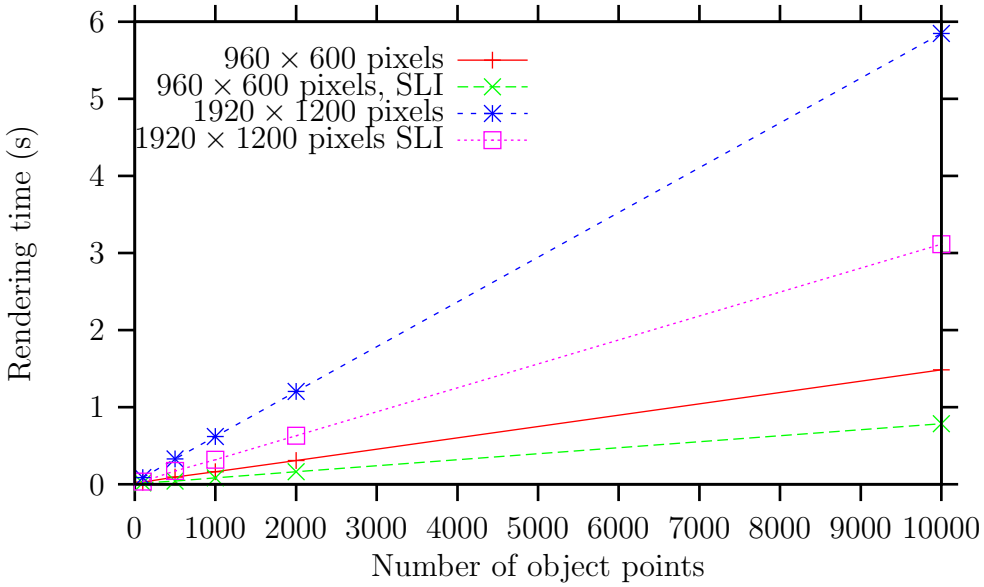
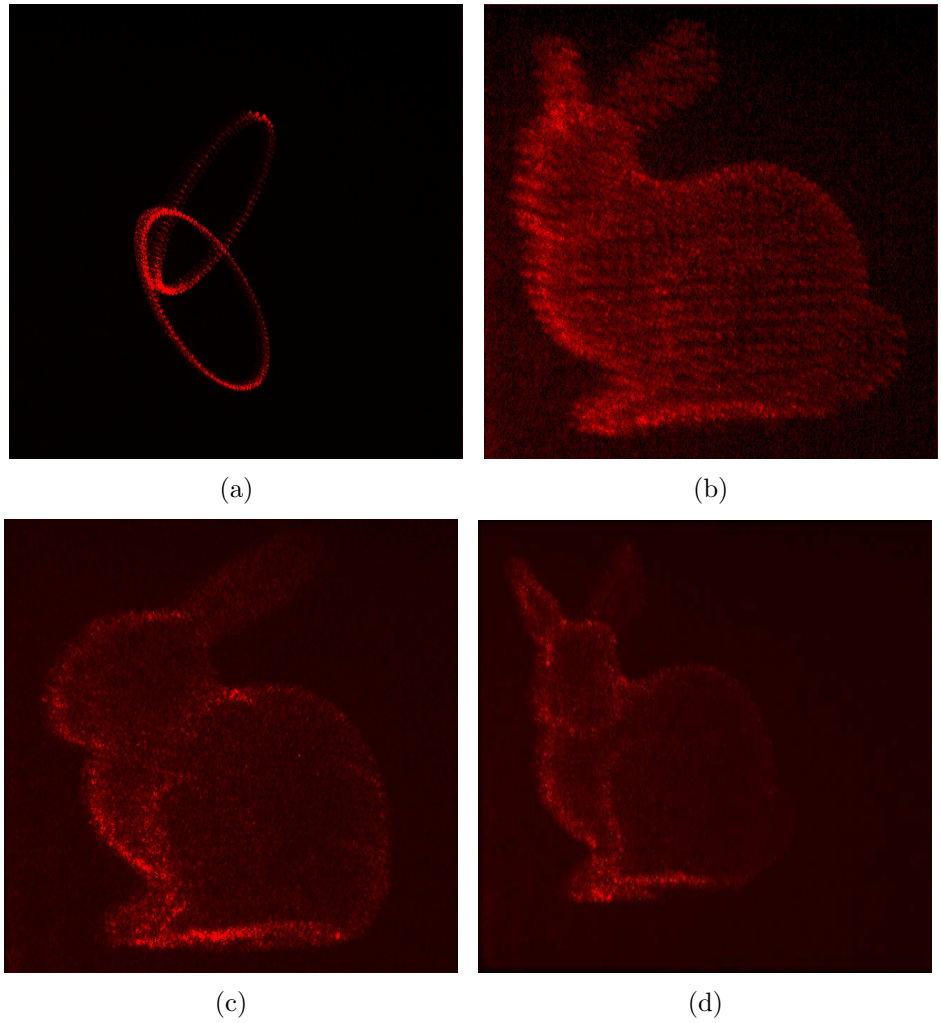


Figure 6.4: Plot of the times presented in Table 6.1. The rendering time scales almost perfectly linearly for 100 **to** 10000 **points. Using the SLI setup doubles the performance, with the exception of the** 100 **point case where the GPU can do the computation in just one pass.**



(a) (b)

(c) (d)

Figure 6.5: Off-axis reconstruction. The models has 200 **(a),** 1800 **(b) and** 8000 **(c,d) points.**

Chapter 7
Fourier rendering

This chapter presents a novel algorithm for fast rendering of holographic display patterns based on polygonal models. Previous CGH methods either compute a Fourier spectrum for each planar surface in a 3D scene or used a per point approach. In contrast, our method computes the light distributions in the angular domain analytically, and only requires a single FFT for the whole wave field. This approach is efficient and beneficial from a sampling perspective. Using previous methods, care has to be taken to properly sample the 3D object. With the method proposed in this chapter, the wave field is expressed analytically through the whole propagation step, and thus sampling is only necessary in the hologram plane.

7.1 Introduction

Holographic display systems were introduced in computer graphics over a decade ago with the work of Lucente and Galyean [67, 68, 70]. Since then, experimental setups for holographic displays have moved to reflect technological advances, such as higher resolution spatial light modulators and full parallax displays. There has been some work in both the graphics and optics communities on improving the CGH rendering algorithms, however much of this work has followed the technological advances, presenting speedups based on GPU rendering, etc.

One of the main reasons for this is that rendering holographic interference patterns from 3D models is a computationally heavy task. Due to the fact that a portion of the incoming light typically diffracts when it gets reflected off the model, every surface point contributes with some light to the whole hologram. Thus, the complexity of a direct method is $\mathcal{O}(N \times M)$ where N is the number of object points and M the number of hologram samples.

This is fundamentally the complexity of methods such as [92, 50, 75] as well as the method we propose in Chapter 6. They are all based on point primitives and then employ different strategies to compute the superpositioned light distribution from these points on a hologram plane. The general drawback of point-based methods is that the 3D object needs to be sampled. The number of point samples required to faithfully represent a solid object is dependent on its surface area, as well as the distance to the hologram plane. Full sampling of the surface using points is, however generally too costly in practice. As an example, consider the resulting rendering of Chapter 6, Figure 6.5(c). The model was constructed of 8000 points, but it is still clearly visible that it is no solid structure.

An alternative to point sampling the whole surface of the 3D object is to try to use more complex primitives than points. The most common way to represent a 3D surface object is through a mesh of polygons. Each polygon is a closed 2D structure made up from a set of vertices connected by edges. The surface is considered solid inside the polygon. The distribution from each polygon can be computed using the Rayleigh-Sommerfeld integral and superpositioned in the hologram plane to form the full object distribution. Computing the Rayleigh-Sommerfeld integral directly is, however, of the same complexity as the per point approach. Instead, one may use the propagation between parallel planes using the angular spectrum, which is based on the Fourier transform of the wave field. This opens up for use of the Fast Fourier Transform (FFT) and is the strategy used by Matsushima [76]. Basically, the wave field in the polygon plane is Fourier transformed, multiplied with a transfer function representing the propagation to the hologram plane, and transformed back to the spatial domain.

For T triangles this requires $T + 1$ number of 2D FFTs and $T \times M$ additions. There are fast FFT libraries available nowadays, e.g. [28], allowing this method to be implemented relatively efficiently. There are however some drawbacks. In order for the FFT to work, the polygons still need to be sampled in the surface plane. Moreover, as most polygon planes are not parallel to the hologram plane, the Fourier image needs to be rotated and resampled before propagation. Thus, the polygons need to be discretized at an early stage and care has to be taken not to introduce sampling artifacts.

We present a novel method that is based on rendering the light distribution of triangles directly in frequency space. This allows for fast evaluation, shading and propagation of light from 3D mesh objects. The method is attractive because its complexity is only dependant on the hologram resolution and the polygon count of the 3D model, just as in classic computer graphics. In contrast to the FFT-based method outlined above there is no need to sample the triangle before propagation. We have derived an analytic formula for

the frequency distribution of a diffusely shaded triangle. When transforming the angular spectra we can then compute the frequency distribution of a general triangle in the hologram plane. This eliminates the need of a per polygon FFT. In addition the wave front is only sampled when recorded in the hologram plane, mimicking the physical process, where the hologram quality is only dependent on the resolution of the recording medium.

7.2 Computer generated holography of surface objects

In this section we will discuss the basic concepts of our method and show how they can be put together to an efficient approach for CGH rendering. We will only cover the techniques briefly here, the theory will be thoroughly presented in Section 7.3.

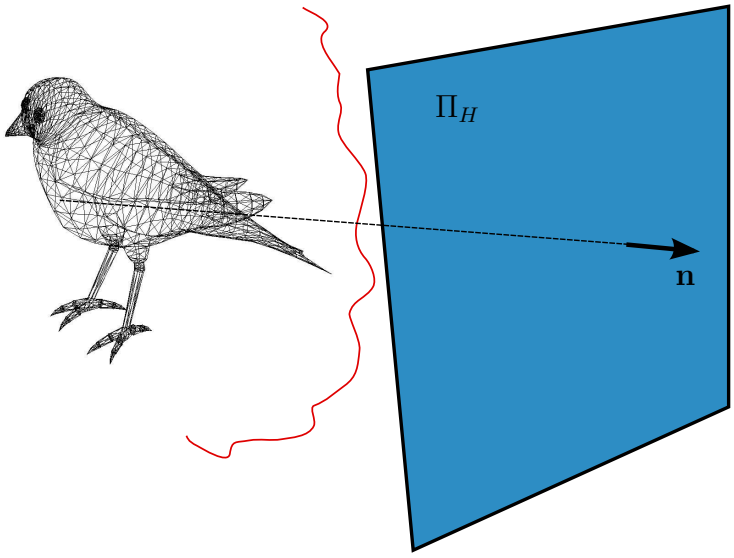


Figure 7.1: The goal of our method is to compute the wave field in the hologram plane, Π_H. This distribution is the sum of the distributions of all triangles in the object mesh.

The input scene is assumed to be a 3D model made up from T surfaces. Each face of the model is a 2D polygon. We want to compute the total complex valued light distribution from the object, measured in some defined plane in space. The light distribution will be referred to as the wave field and its location as the *hologram plane*. This is illustrated in Figure 7.1.

If we somehow can compute the portion of light in the hologram plane steaming from the individual polygons, the superposition principle tells us that we can add these together to form the distribution of the whole object

$$U = \sum_{i=1}^{T} U_j. \tag{7.1}$$

Where U_j is the light distribution from polygon j in the hologram plane.

The general problem can thus be seen as twofold:

1. Shading: Given material properties and incoming light compute the outgoing wave field of a polygon

2. Given the wave field of a polygon: propagate it to the hologram plane.

Regarding the first problem on the list; if the material properties of the polygon, as well as the illumination, are known we can compute the amplitudes of the reflected light using standard computer graphics shading methods. We can thus assume that at least the intensity values of the polygons are known, and will call the wave field defined in the plane containing polygon j as W_j. Thus, we start by covering the second point in depth.

The question is, how can the distribution in some other plane, specifically the hologram plane be efficiently computed? One answer lies in the methods working with the angular spectrum [36]. The angular spectrum is introduced in Chapter 5, and the foundation and equations for propagation are given in Section 7.3.3 below. The general idea, however is that given a wave field W_Π the propagated field W_Γ at some plane Γ parallel to Π at some distance, d, can be expressed as

$$W_\Gamma = \mathcal{F}^{-1}\{\mathcal{F}\{W_\Pi\}K(d)\}. \tag{7.2}$$

In the above equation $\mathcal{F}\{\}$ denotes the Fourier transform, and K is a specific transfer function dependent on the distance, d.

By definition this propagation is only possible between parallel planes. However, this is clearly not the general case as the polygons are rotated with respect to the hologram plane. Fortunately, there are methods to rotate the angular spectra, so that they may be transferred to planes tilted with respect to the original are [114, 115, 80, 81]. We will denote this operator $\mathcal{R}$ for this overview and discuss it further in depth in Section 7.3.3.

Using these two methods, Eq. 7.1 can be more explicitly written as

$$W_H = \sum_{i=1}^{T} \mathcal{F}^{-1}\{\mathcal{R}_j(\mathcal{F}\{W_j\})K_j\} = \mathcal{F}^{-1}\{\sum_{i=1}^{T} R_j(\mathcal{F}\{W_j\})K_j\}, \tag{7.3}$$

where $\mathcal{R}_j$ is the rotation operator, asserting that the angular spectrum in the plane of polygon j lies parallel to the hologram plane. K_j is the transfer function transporting the j:th rotated spectrum along the optical axis to the hologram plane.

Equation 7.3 is an efficient method for light propagation if the term $\mathcal{F}\{W_j\}$ can be computed. This is related to the first problem on the list above, and can be formulated as: *How can the angular spectrum of the reflected light be calculated?* In an implementation this could be done in one of three ways:

1. Compute W_j by shading the polygon then perform the Fourier transform on the fly

2. The whole $\mathcal{F}\{W_j\}$ can be precomputed and stored as a sampled angular spectrum

3. Express $\mathcal{F}\{W_j\}$ using an analytic function.

The first case is possible to implement using the FFT algorithm, but will require a transform per polygon each rendering step. It also requires W_j to first be rendered and then be sampled, something that could introduce artifacts. Likewise, the second method requires sampling of the angular spectrum. This method is also unpractical to implement due to very large memory requirements. For each polygon a precomputed wave field, with size of the same order as the hologram, has to be stored. It also does not allow for dynamic lighting due to the precomputation.

The third method, to derive an analytic expression of the angular spectrum for a general shape is an intriguing option. It has several benefits:

- There is no need to sample the object to points

- It is possible to render solid objects

- Ideally the complexity of the algorithm scales with the number of polygons and hologram samples only

- Dynamic lighting is possible

- The resulting wave field is dependent on hologram sampling only.

In this chapter we will show how to compute the angular spectrum of a general diffusely reflecting triangle. We will also construct an algorithm based on this theory and implement a proof of concept renderer.

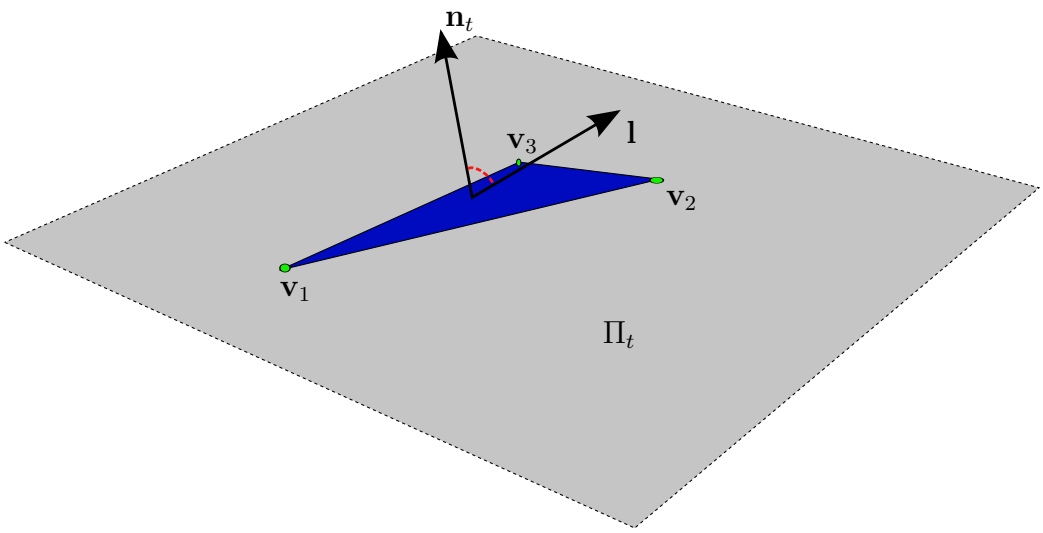


Figure 7.2: Triangle model. Each triangle is defined by three vertices: $\mathbf{v}_1$, $\mathbf{v}_2$, $\mathbf{v}_3$, which also spans the triangle plane, Π_t. $\mathbf{n}_t$ is normal to the triangle plane, and the vector $\mathbf{l}$ is the direction towards a distant light source.

In the following sections, we will assume that the 3D object is made up from triangles, each one defined by three coplanar vertices spanning a plane, Π_t. This plane will be called the triangle plane. The material of the object is assumed to be a perfectly diffuse and the light source is assumed to lie far away. This allows us to set a constant amplitude, i.e. flat shading, over a single triangle, defined as

$$a_t = \mathbf{n}_t \cdot \mathbf{l}. \tag{7.4}$$

In the above shading equation, $\mathbf{n}_t$ denotes the triangle plane normal and $\mathbf{l}$ is the direction vector of lighting. Figure 7.2 shows the above terms in relation to each other.

Based on these assumptions, the next section will show how to derive an expression to calculate the angular spectra of a triangle in a general reference plane.

7.3 Theory

In this section, we will derive the mathematical foundations of the method outlined in the previous section. We will do this in three steps. First, to formulate an analytic expression of the Fourier spectrum of a general triangle, we derive the analytic formula for calculating the two dimensional Fourier

transform of a right triangle. Second, we show how to relate the Fourier spectrum of the general triangle to this function. Third, we will revise some strategies for rotating and transporting the angular spectra of a wave field from literature.

7.3.1 The Fourier transform of a right triangle

Figure 7.3(a) depicts a right triangle with the side 1 and one vertex in the origin. We will call this triangle, with vertices in the points $(0,0)$, $(1,0)$, $(1,1)$, Δ and define the function $f_\Delta : \mathbb{R}^2 \to \mathbb{R}$ as

$$f_\Delta(x,y) = \begin{cases} 1 & \text{if } (x,y) \text{ lies inside } \Delta \\ 0 & \text{else} \end{cases} . \tag{7.5}$$

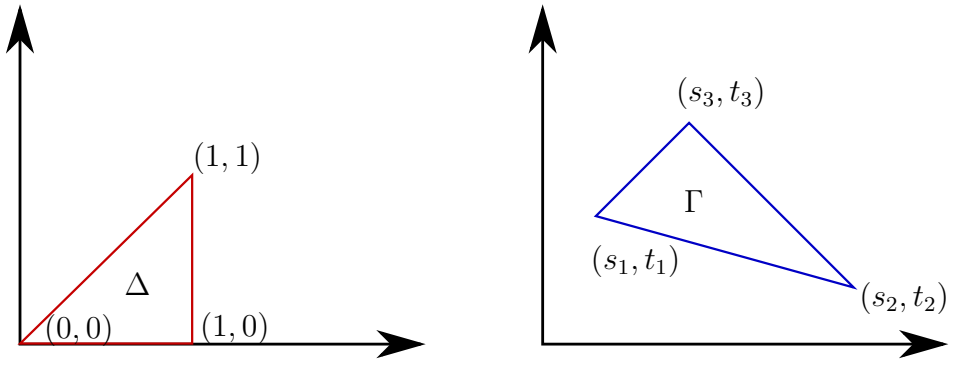


Figure 7.3: Two triangles. (a) A simple triangle with vertices in $(0,0),(1,0),(1,1)$. (b) A general triangle with vertices in $(s_1,t_1),(s_2,t_2),(s_3,t_3)$

To acquire the Fourier spectra of Δ, we compute the 2D Fourier transform of f_Δ Thus, let F_Δ be the Fourier transform of f_Δ. By definition

$$F_\Delta(u,v) = \int\!\!\int_{-\infty}^{\infty} f_\Delta(x,y) \exp(-2\pi i(xv+yv))\, dydx. \tag{7.6}$$

As f_Δ is constant valued 1 on Δ and 0 everywhere else, we can rewrite this formula as

$$\begin{aligned} F_\Delta(u,v) &= \int\!\!\int_{\Delta} \exp(-2\pi i(xu+yv))\, dydx \\ &= \int_0^1 \int_0^x \exp(-2\pi i(xu+yv))\, dydx. \end{aligned} \tag{7.7}$$

Solving the integral we arrive at the following expression

$$F_\Delta^{uv}(u,v) = \frac{\exp(-2\pi iu) - 1}{(2\pi)^2 uv} + \frac{1 - \exp(-2\pi i(u+v))}{(2\pi)^2 v(u+v)}. \tag{7.8}$$

However, this function is not properly defined at the point $(0,0)$ as well as on the lines $u = 0$, $v = 0$ and $u = -v$. We solve the integral especially for the critical values in order to see how F_Δ behaves at these regions.

Case 1: $u = 0, v \neq 0$ Inserting $u = 0$ in Eq. 7.7 we have

$$\begin{aligned} F_\Delta &= \int_0^1 \int_0^x \exp(-2\pi iyv)\, dydx \\ &= \frac{1 - \exp(-2\pi iv)}{(2\pi v)^2} - \frac{i}{2\pi v}. \end{aligned} \tag{7.9}$$

Case 2: $u \neq 0, v = 0$ Similar to case 1, we integrate with $v = 0$:

$$\begin{aligned} F_\Delta &= \int_0^1 \int_0^x \exp(-2\pi ixu)\, dydx \\ &= \frac{\exp(-2\pi iu) - 1}{(2\pi u)^2} + \frac{i\exp(-2\pi iu)}{2\pi u}. \end{aligned} \tag{7.10}$$

Case 3: $u = -v, v \neq 0$ Inserting $u = -v$ in Eq. 7.7

$$\begin{aligned} F_\Delta &= \int_0^1 \int_0^x \exp(-2\pi iv(y-x))\, dydx \\ &= \frac{1 - \exp(2\pi iv)}{(2\pi v)^2} + \frac{i}{2\pi v}. \end{aligned} \tag{7.11}$$

Case 4: $u = 0, v = 0$ The integral at the origin is simply the area of the triangle

$$F_\Delta = \int_0^1 \int_0^x \exp(0)\, dydx = \frac{1}{2}. \tag{7.12}$$

Thus, the Fourier transform of f_Δ can be expressed analytically as

$$F_\Delta(u,v) = \begin{cases} \frac{1}{2} & u = v = 0, \\ \frac{1-\exp(-2\pi iv)}{(2\pi v)^2} - \frac{i}{2\pi v} & u = 0, v \neq 0, \\ \frac{\exp(-2\pi iu)-1}{(2\pi u)^2} + \frac{i\exp(-2\pi iu)}{2\pi u} & u \neq 0, v = 0, \\ \frac{1-\exp(2\pi iv)}{(2\pi v)^2} + \frac{i}{2\pi v} & u = -v, v \neq 0, \\ \frac{\exp(-2\pi iu)-1}{(2\pi)^2 uv} + \frac{1-\exp(-2\pi i(u+v))}{(2\pi)^2 v(u+v)} & \text{else} \end{cases}. \tag{7.13}$$

7.3.2 The Fourier spectra of a general triangle

The next step is to determine the Fourier spectra of a general triangle. For this we make use of the fact that an affine transform relating two triangles also can be use to relate their respective Fourier spectra.

Figure 7.3(b) shows an example of a general triangle, Γ, with vertices at $(s_1, t_1), (s_2, t_2), (s_3, t_3)$. We can relate the vertices of Δ with those of Γ by an affine transform

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} (s_2 - s_1) & (s_3 - s_2) & s_1 \\ (t_2 - t_1) & (t_3 - t_2) & t_1 \\ 0 & 0 & 1 \end{bmatrix}, \tag{7.14}$$

so that

$$\begin{bmatrix} s \\ t \\ 1 \end{bmatrix} = A \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7.15}$$

As long as Γ is a well behaved triangle there is of course also some inverse mapping

$$B = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ 0 & 0 & 1 \end{bmatrix}. \tag{7.16}$$

That relates

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = B \begin{bmatrix} s \\ t \\ 1 \end{bmatrix}. \tag{7.17}$$

Thus, if we let $f_\Gamma : \mathbb{R}^2 \to \mathbb{R}$ be the function describing Γ analog to Eq. 7.5 we can relate

$$f_\Gamma(x, y) = f_\Delta(s, t). \tag{7.18}$$

The Fourier transform of f_Γ is

$$F_\Gamma(u, v) = \iint_{-\infty}^{\infty} f_\Gamma(x, y) \exp(-2\pi i(su + tv))\, dxdy. \tag{7.19}$$

By substituting the function according to Eq. 7.18 and using the affine transform in Eq. 7.16 to relate the variables, we have

$$F_\Gamma(u, v) = \iint_{-\infty}^{\infty} f_\Delta(s, t) \exp(-2\pi i(u(b_{11}s + b_{12}t + b_{13}) + v(b_{21}s + b_{22}t + b_{23})))J\, dsdt. \tag{7.20}$$

Where $J = (b_{22}b_{11} - b_{12}b_{21})$ is the Jacobian determinant of B.

Now, by rearranging the exponential we have

$$F_\Gamma(u,v) = \iint_{-\infty}^{\infty} f_\Delta(d,t)\exp(-2\pi i(ub_{13}+vb_{23})) \exp(-2\pi i(s(ub_{11}+vb_{21})+t(ub_{12}+vb_{22})))J\,dsdt. \quad (7.21)$$

Finally, moving the terms not dependant on s,t outside the integral we note that the rest corresponds to F_Δ under a coordinate transform, and arrive to

$$F_\Gamma(u,v) = (b_{22}b_{11}-b_{12}b_{21})\exp(-2\pi i(ub_{13}+vb_{23})) F_\Delta(ub_{11}+vb_{21}, ub_{12}+vb_{22}). \quad (7.22)$$

Thus, from the results in Equations 7.13 and 7.22 we can compute the Fourier transform of a general 2D triangle analytically. This is a very important result, as it will allow us to perform rendering in Fourier space directly instead of first rendering the triangle and then sampling and transforming it.

7.3.3 Light propagation in the angular domain

Earlier in this thesis we have described how the propagation of wave fields can be achieved by directly implementing the Rayleigh-Sommerfeld integral. This strategy can however be rather time consuming in general cases. Here we will describe how to perform the propagation of a Fourier transformed wave front.

Propagation between parallel planes

Wave front propagation between two planes Π_1 and Π_2 can be described using the Rayleigh-Sommerfeld integral as described in Eq. 2.14. If we assume $U_1(x,y)$ to be the known wave field on Π_1 we can compute the distribution, $U_2(s,t)$ in a parallel plane at a distance r as

$$U_2(s,t) = -\frac{r}{\lambda}\iint_{-\infty}^{\infty} U_1(x,y)\frac{\exp\left(\frac{2\pi i}{\lambda}\sqrt{(x-s)^2+(y-t)^2+r^2}\right)}{\sqrt{(x-s)^2+(y-t)^2+r^2}}\,dxdy. \quad (7.23)$$

In this formulation of the integral we have assumed the normal of the planes to be $\mathbf{n}=[0,0,1]$, thus the scalar product in Eq. 2.14 reduces to r.

This equation however, can be written as a convolution

$$U_2(s,t) = -\frac{r}{\lambda}(U_1(x,y) * \frac{\exp\left(\frac{2\pi i}{\lambda}\sqrt{x^2+y^2+r^2}\right)}{\sqrt{x^2+y^2+r^2}}). \quad (7.24)$$

Using the convolution theorem we can express this as a multiplication in the Fourier domain

$$U_2 = -\frac{r}{\lambda}\mathcal{F}^{-1}\{A_1 K_r\}. \tag{7.25}$$

$A_1 = \mathcal{F}\{U_1\}$ is called the *angular spectrum* of U_1 and K_r is called the transfer function. Chapter 5 covers the properties of the angular spectrum more in depth.

According to [53] the transfer function, K_r can be expressed as

$$K_r(s,t) = \exp\left(\frac{2\pi i}{\lambda} r\sqrt{1-(s\lambda)^2-(t\lambda)^2}\right). \tag{7.26}$$

So, multiplication of the angular spectrum of a wave field by K_r and an inverse Fourier transform will yield the wave field in a plane at distance r from the first one.

Propagation between rotated planes

Another problem is related to calculating the light propagation between non parallel planes. The general problem is shown in Figure 7.4: planes Π_1 and Π_2 are related by a rotation matrix R. Given the wave field, U_1 of Π_1 compute U_2 of Π_2. Direct propagation by the Rayleigh-Sommerfeld integral is not possible as the planes are not parallel. One solution to this problem can be expressed using the angular spectrum [114, 115, 80, 81]. We will use the approach of Matsushima et al. [80, 81] and briefly follow their argumentation.

As discussed in chapter 5, the angular spectrum in a point (u, v) can be interpreted as the complex amplitude of a plane wave. The wave vector can be expressed as

$$\mathbf{k} = \frac{2\pi}{\lambda}\left[u\lambda, v\lambda, \sqrt{1-u^2\lambda^2-v^2\lambda^2}\right]^T. \tag{7.27}$$

From the definition of the planar waves in Section 2.1.2 we know that the wave vector denotes the propagation direction.

Matsushima et al. show that the wave vectors defined by the angular spectra in the two planes can be related through the inverse rotation matrix R^{-1}. Their arguments are similar to the ones we use in Section 7.3.2, and they arrive at the expression

$$U_2(s,t) = \mathcal{F}^{-1}\{A_1(r_{11}s + r_{12}t + r_{13}d(s,t), r_{21}s + r_{22}t + r_{23}d(s,t))J(s,t)\}. \tag{7.28}$$

Where, $A_1 = \mathcal{F}\{U_1\}$ denotes the angular spectrum and r_{ij} are the elements of the rotation matrix.

$$d(s,t) = \sqrt{\frac{1}{\lambda^2} - u^2 - v^2} \tag{7.29}$$

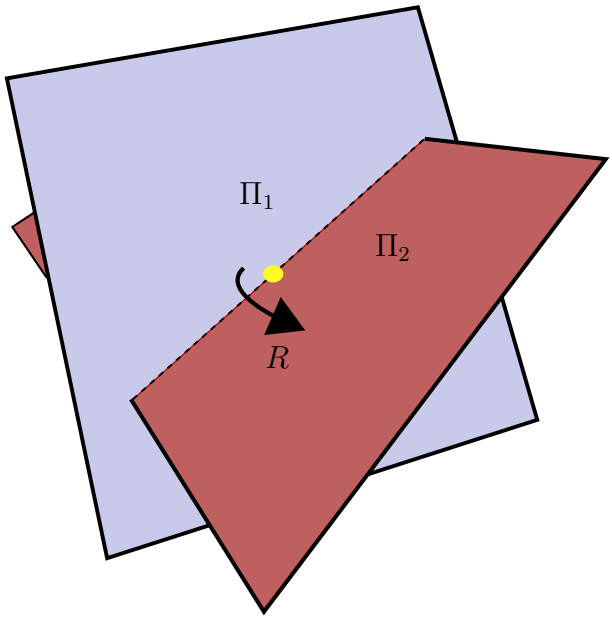


Figure 7.4: The angular spectrum in Π_1 can be related to the one in Π_2 using the rotation R.

is the distance transform, derived from Eq. 7.27 and

$$J(s,t) = (r_{12}r_{23} - r_{13}r_{22})\frac{s}{d(s,t)} + (r_{13}r_{22} - r_{11}r_{23})\frac{t}{d(s,t)} + (r_{11}r_{22} - r_{12}r_{21}) \quad (7.30)$$

is the Jacobian determinant.

Now, using Eq. 7.28 it is possible to compute the wave distribution in a plane rotated with respect to the source plane. Together with the transport equation in 7.25 this can be used to propagate wave fields between arbitrary planes located along an optical axis.

7.4 Algorithm and implementation

From the results in the previous section we can now start to design an algorithm suitable for implementation.

By the use of Eqs. 7.13 and 7.22 the angular spectrum of a general triangle can be computed. Thereafter by applying methods as discussed in Section 7.3.3 it can be propagated to the hologram plane for recording. The steps described by these equations are illustrated in Figure 7.5. The angular

spectrum is computed in the triangle plane Π_t. This spectrum is then related to the spectrum on an intermediate plane Π'_t by the transform $\mathcal{R}$ (see Eq. 7.28). Once we know the values in Π'_t which is parallel to the hologram plane Π_H we can use the method for propagation of the angular spectrum as described in Section 7.3.3. In the algorithm outline below these transform steps are combined into a single operation. For understanding, however it helps to visualize this chain of operations.

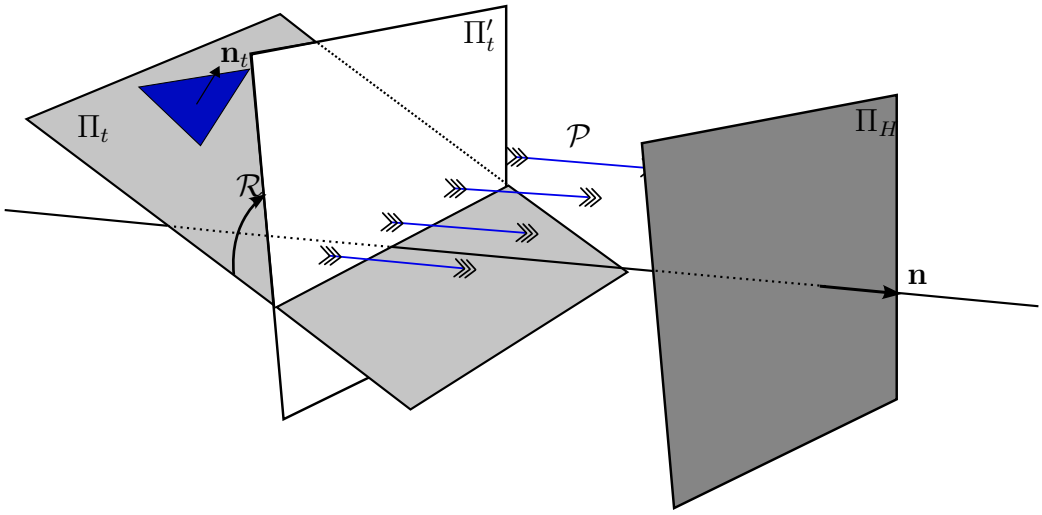


Figure 7.5: Stepwise transformation and rendering of the angular spectrum from a triangle. The angular spectrum of the triangle is computed in the triangle plane Π_t, having normal $\mathbf{n}_t$. Thereafter the angular spectrum is transformed using the rotation transform $\mathcal{R}$ to a plane, Π'_t normal to the optical axis, $\mathbf{n}$. In the last step a propagation transform $\mathcal{P}$ is used to propagate the light to the hologram plane Π_H.

Now, by repeatedly adding the propagated angular spectra of all planes and then performing an inverse Fourier transform we will compute the wave field in the hologram plane, as expressed in Eq. 7.3. This suggests a loop over all triangles and pixels in the hologram. We can however reduce the number of triangles that needs to be processed by using so called backface culling. This involves computing

$$s = \mathbf{n}_t \cdot \mathbf{n}. \tag{7.31}$$

where $\mathbf{n}_t$, as before, is the normal of the current triangle and $\mathbf{n}$ is the normal of the hologram plane. If the scalar product $s \leq 0$ we know that the planes are orthogonal or facing away from each other. In this case we will assume

that no light contribution from the plane can reach the hologram plane and we do not have to process the triangle. This factor can be used even further, by taking into consideration the maximum angle of diffraction as limited by the hologram sampling. In this proof of concept implementation we let the user define a threshold value in order to avoid aliasing artifacts.

It should also be mentioned that at this stage we do not consider true hidden surface removal. This means that for some complex concave objects light propagating from one triangle occluded by another one, closer to the hologram plane may still be visible. This is a rare case for many objects as the backface algorithm takes care of the major problems. However, it is still an issue that needs to be considered in the future. Simple depth occlusion algorithms do not work directly in the case of wave propagation, as a triangle can very well be fully covered by another and still contribute light to the hologram plane. Therefore, occlusion methods that are similar to soft shadow algorithms in computer graphics may be a solution. As we are mainly considering a proof of concept of our analytic rendering algorithm at this stage, we will ignore full hidden surface removal for now.

7.4.1 Overview of the algorithm

Algorithm 7.1 shows the main steps in our proof of concept algorithm. The basic idea is, for each triangle in the model to check if it is facing the hologram plane. In that case, the angular spectrum of the light distribution from the current triangle is calculated in the hologram plane using methods presented in Section 7.3. The wave field is sampled at each hologram pixel position and accumulated in a complex valued image. After all triangles have been processed the image is inverse Fourier transformed to create the wave field in the hologram plane.

To create a hologram intensity distribution from this, all that needs to be done is to add a reference wave and measure the magnitude of the resulting field.

We have implemented an un-optimized proof of concept version of the algorithm in C++. The software renders holographic interference patterns from triangular mesh models. Some results and a discussion can be found in the next section.

7.5 Discussion and results

In this section we will present some visual results from our proof of concept implementation. The current implementation is non-optimized and thus not

Algorithm 7.1: The basic steps of our proof of concept methods.

Data: A set of triangles $\{t_j\}$. A hologram plane H.
Result: The complex valued wave field distribution from all triangles in the mesh.

```
for Each triangle t_j do
    Let T be the plane containing all three triangle vertices, t_v1, t_v2, t_v3 ;
    Let I be a complex valued image defined on H ;
    Compute the triangle flat shading value using Eq. 7.4 and store the result in s ;
    if normal(T) · normal(H) > 0 then
        Find the mapping F : H → T that maps from hologram plane coordinates to triangle plane coordinates ;
        Compute the affine mapping B from t to the right triangle ;
        Concatenate the mappings; G = B{F} now maps from image plane Fourier space to triangle plane Fourier space ;
        for Each pixel p ∈ I do
            Find the coordinate of p in the right triangle Fourier space. q = G(p) ;
            Compute the complex amplitude a_q at q using Eq. 7.13 ;
            Let a_p be a_q transported to the image plane using Eqs. 7.22, 7.28 and 7.25 ;
            Let I_p = I_p + a_p s ;
        end
    end
end
Compute the wave field of I by I = IFFT{I} ;
```

fast enough for real-time performance. The models presented here contain a few thousand triangles and render in a few minutes. An optimized version should be on par with the point based method described in Chapter 6 however.

We have been using 3D models consisting of a few thousand triangles. For all renderings, the distance between hologram plane and object has been 0.1 meters and wavelength has been kept at 633 nm. For the tests a wave field resolution of 1220×1220 and a pixel size of $8.1\mu m$ has been used. The complex wave field was saved and numerically reconstructed at a distance of 0.1 meter.

Figure 7.6 shows input, wave field and reconstruction. In 7.6(a) we see a

mesh model of the input as viewed from the hologram plane. After rendering, the complex valued wave field was saved. Figure 7.6(b) shows the magnitude of the wave field. A numerical simulation of the reconstruction is shown in 7.6(c). The result looks solid and the different features of the model are clearly visible. There are also a few artifacts visible which we will discuss now.

First, there are some brighter areas visible. These are due to the lack of global hidden surface removal as discussed earlier in this chapter. Light from overlapping or partially overlapping surfaces will interfere and some areas will be perceived as brighter. We believe that this may be less of a problem in real world cases where the dynamic range of the laser lit displays may well even out these effects.

Second, aliasing artifacts are visible along some edges of the reconstruction. These are most clearly visible at the tail feathers and the head of the bird. The source of these artifacts are the triangles at the visual edge of the object. These will have a very steep angle to the hologram plane. As was discussed in Chapter 2, the resolution and pixel size of a sampled wave field limits the diffraction angle and thus can not faithfully represent the angle of the incoming light. The effect is also visible in the magnitude image, 7.6(b), as jagged artifacts in the outer regions of the pattern.

One direct way of limiting the aliasing artifacts is to raise the threshold used to determine if a triangle is back-facing or not. As discussed in Section 7.4, a triangle can be considered back-facing if the scalar product of the triangle plane normal and the hologram plane normal (Eq. 7.31) is less than zero. In order to avoid aliasing the threshold can be raised somewhat to discard the triangles that are the source of the problem. Ideally, the threshold should be set to correspond to the diffracting angle of the hologram plane. However, in most practical cases this angle is so small that most edge triangles will be clipped. This in turn, removes all sense of object curvature and may also erode or create holes in the mesh. Thus, a balance between aliasing artifacts and model quality has to be kept. Figure 7.7 shows the reconstructed wave field where the backfacing threshold were set to 0.2 while rendering. I.e. if the scalar product of the triangle plane normal and the hologram plane normal was less or equal to 0.2 the triangle was discarded. As can bee seen the amount of aliasing has been visibly reduced, and the edges of the model are perceived to be sharper.

Finally, Figure 7.8 shows a model of an abstract "eight" shape. This is an ideal case without any self occlusion and the algorithm produces a high quality model with clearly visible shading.

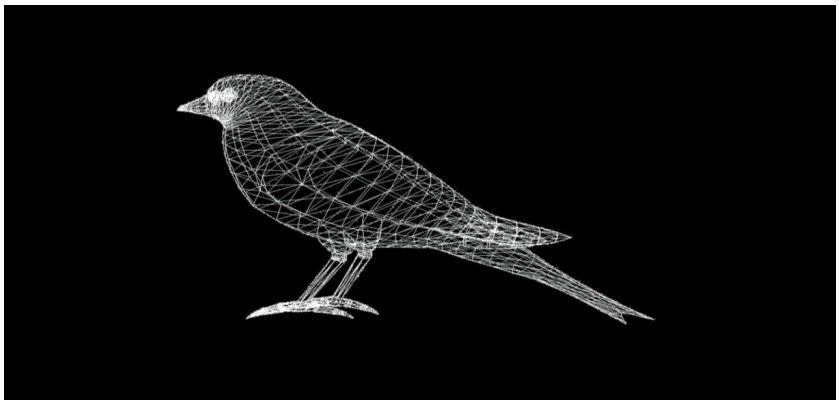
7.6 Conclusions

We have presented a novel method for generating wave fronts from 3D triangular mesh models. Our approach has many advantages over previous methods. By formulating an analytic expression for the plane wave spectra there is no need to sample the 3D model surface, removing a potential source of aliasing. The complexity of the algorithm is only dependent on the resolution of the target image and the number of triangles in the 3D model. Previous methods either had to sample the 3D object or perform a FFT for each triangular patch. Thus, for interactive applications the only choice was to represent the object as a low-sampled point cloud.

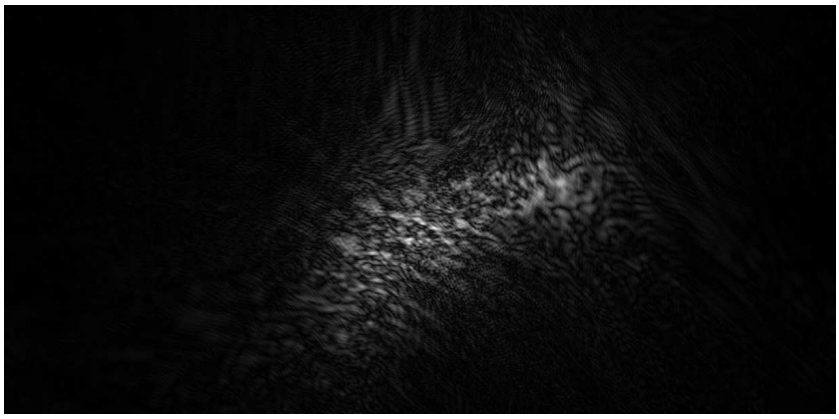
The theory presented in this chapter has been supported by a proof of concept implementation that can render wave fields and holographic patterns from triangular mesh models. The current implementation is fairly straight forward, but it may well be heavily optimized. For instance, using a GPU-based implementation. This leads to a worst case scenario of one pass per triangle, comparing to one pass per point using the techniques discussed in Chapter 6.

In the future, we would like to focus work on exploring how different material properties can be incorporated in our method. Currently we assume a homogeneous material, and perform simple flat shading of the triangle. However, the incorporation of material and texture properties is not straightforward. Texturing the triangle will unavoidably lead to a convolution of the angular spectrum which may slow the method down. However, it may be possible to work with locally varying materials for instance.

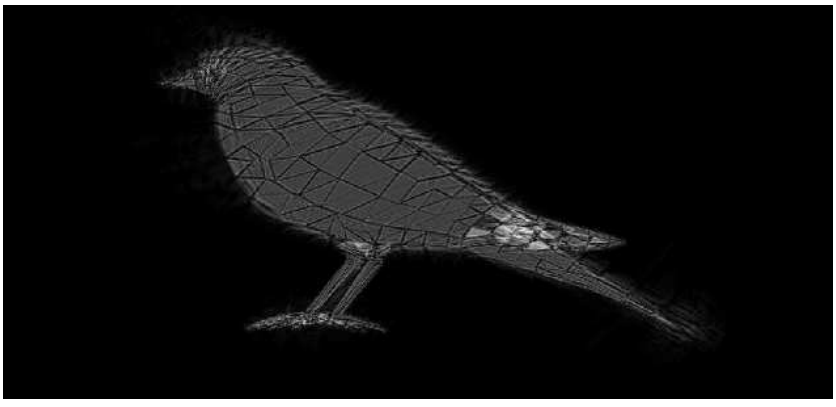
The theory presented in this chapter has been exemplified mostly using holographic applications, however it is of course also applicable in other areas. This could include computer graphics rendering where the wave model of light is more applicable, optical tweezers and microscopy.



(a)



(b)



(c)

Figure 7.6: Rendering a bird using a wave field resolution of 1220×1220 samples. (a) Original mesh model. 2246 triangles. (b) Magnitude of the resulting wave field. (c) Numerical reconstruction from the wave field.

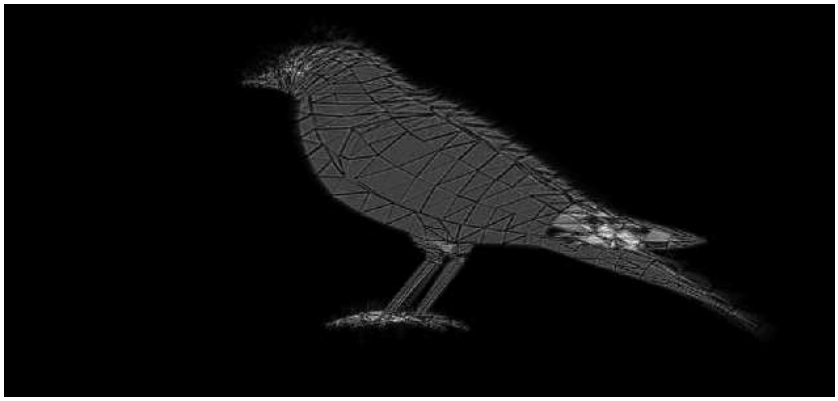


Figure 7.7: Numerical reconstruction using a backface-threshold of 0.2. Triangles with $\mathbf{n}_t \cdot \mathbf{n} \leq 0.2$ gets culled thus removing some of the angular aliasing. Wave field resolution: 1220×1220 pixels.

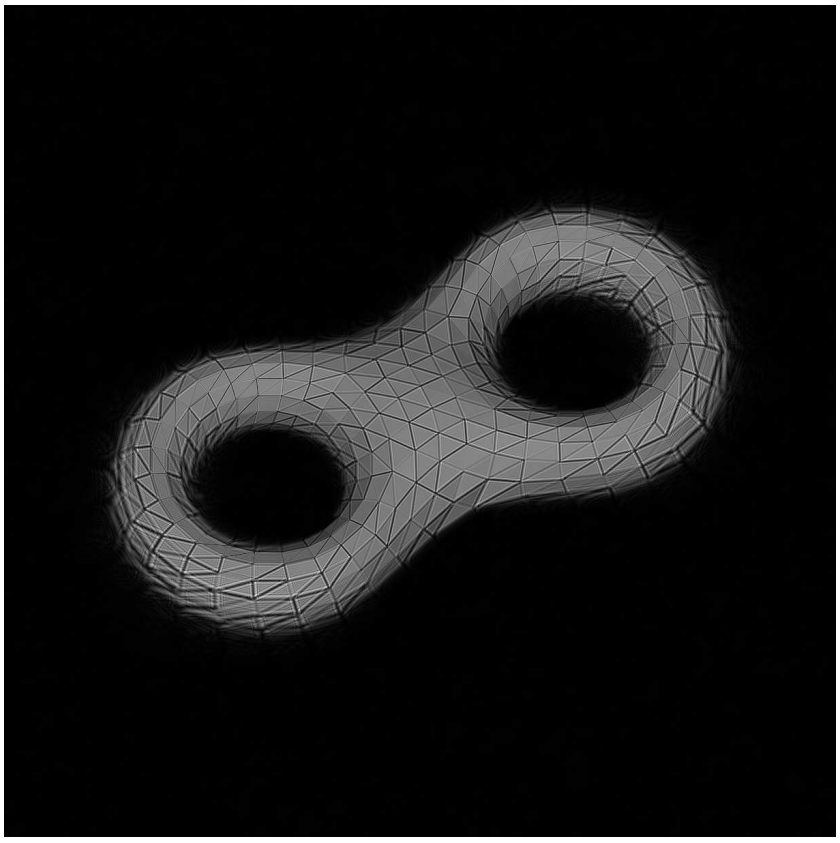


Figure 7.8: Numerical reconstruction of a mesh containing a few hundred triangles. The directional shading of the triangles is clearly visible. Wave field resolution: 1220×1220 samples.

Chapter 8
Conclusions

In this Chapter we will conclude the work presented in this dissertation. We give a brief summary and discuss possible future work.

8.1 Summary

We have presented techniques at the border between optics and computer graphics research. The work in this thesis deals with what we call full light representations. That is, the complete visual information of a scene is stored and rendered. We have used two well known representations, the light field and the hologram.

In Chapter 4 we presented work adopting linear operators from ray optics to a light field framework. We showed how to approximate rotation, propagation and interface operators using matrices. For a chain of operators each describing a linear transform, the matrices for the individual elements are multiplied, thus creating a very efficient one-step transform. We also showed how wavelet compression of light fields fit into our framework.

In Chapter 5, we demostrate the duality of the light field and wave field representations. We discussed the principle of each representation and also presented a time-frequency approach and exemplified it by the short-term Fourier transform. The method can be seen as assuming the object wave front to be piecewise linear over the hologram plane.

The two final Chapters in this dissertation both considered hologram synthesis. While the main focus is on rendering for holographic displays, the techniques can be used for general wave field construction. Chapter 6 showed how holographic interference patterns could be generated from 3D point objects using programmable graphics hardware. We showed how generating

the fragment program in run-time using unrolled loops, allowed us to accelerate the rendering and also use larger 3D objects then previous techniques. Chapter 7 presented a new strategy for computer generated holograms from polygonal models. While previous approaches have rasterized each polygon and Fourier transformed this 2D image, we compute the Fourier transform of a general triangle analytically. This has several advantages as the wave field is not sampled until it is propagated all the way to the hologram plane. Thus, our technique does not suffer from the need to rotate and filter the Fourier coefficients like previous approaches.

8.2 Future work

While the contributions in this thesis all improve on previous methods and algorithms, they are by no means final. There is always room for improvements, and while working on a project one always has new ideas on future work. In this section we will discuss some of these ideas, and how we would like to continue the work started in this dissertation.

8.2.1 Matrix optics for light fields

While our matrix optics formulation can not take second order effects into account, it may still have interesting uses as it allows for very fast transformation of the ray space. As discussed in Chapter 4 an interesting future application of this would be to incorporate it into a Fourier slice framework as the one proposed by Ng [89].

In Ng's work, images with different focal depth can be rendered from a light field by applying the Fourier slice theorem. In the paper the author showed how a certain optical configuration corresponded to a specific 2D slice in the four dimensional Fourier space of a light field. The optical configuration does however only correspond to a fixed transportation using our operator notation. It would be interesting to try to incorporate our matrix optics transforms into the Fourier Slice framework. This would allow for more general rendering applications.

8.2.2 Hologram to light field transform

There are two interesting venues of future work to pursue regarding work relating holograms and light fields. The first is to improve methods for depth and phase reconstruction from light fields and holograms. This includes

plenoptic depth reconstruction, depth from defocus and interferometric methods. Could this be properly performed for general scenes it would be a great step forward for hologram recording. General holograms of real scenes could then be recorded in natural lighting situations using for instance plenoptic cameras. Work in this direction has already been performed in [127].

The second possibility is to continue work on signal analysis. As discussed in Chapter 5, the Fourier method presented there approximated the wave front by planar segments. However, there are other methods available in time-frequency analysis that may suite the problem better. If some higher degree basis function can be found it may be possible to approximate the local wave front curvature and thus depth. One possibility is using wavelets. Liebling et al. [64] have proposed a family of wavelet basis functions that we would like to have a closer look at for this purpose. Another possible transform that we would like to implement is the Wigner distribution [9].

Hologram to light field transform may have many applications in computer graphics and image-based rendering. Holography has a long history in measurement and material scanning. One example where computer graphics techniques may benefit from holographic recording is in BRDF measurements. In order to capture a complete anisotropic BRDF using standard camera techniques samples have to be taken for each light and viewing position. However, as holographic recording captures both light intensity and direction, such a setup would only require one parameter to be changed.

8.2.3 Computer generated holograms

Wave field synthesis is still in its early stages, and the two different contributions to the field presented in this thesis are a good base for future work.

In Chapter 6 we developed a method for accelerating hologram generation from point models using graphics hardware. GPU technology is one of the most innovative and rapidly advancing fields in computer hardware today. A new generation technology has already been introduced while writing this thesis and future improvements can be seen at the horizon. Already, there are possibilities and new programming models available, for instance using the recently introduced CUDA [90] from NVIDIA. The pipelined structure of the graphics cards are still what makes up for most of the speed however. We believe that our basic strategy, i.e. generating custom programs on the fly, will be valid for yet some time.

We believe that our analytic approach to rendering triangle models may be the most promising when looking for future perspectives. There are several interesting possibilities to pursue in order to make the method more general.

Currently we do not address the problem of hidden surface removal in our

approach. For future high resolution displays, this would be a requirement however. The general problem for full light hidden surface removal is that it is view dependent. This means that an image space method would have to perform visibility tests for every polygon per pixel, which is very expensive. In [77] Matsushima presented a method for hidden surface removal in CGH rendering. However, it requires two Fourier transforms per polygonal surface, which may be too expensive for real time graphics. We would like to further investigate the problem, and believe that it is very similar to computer graphics techniques for soft shadowing.

Right now we are only considering perfectly diffuse, unified materials, however in order to be able to render a more interesting class of 3D models we would like to investigate how to incorporate other material properties. This would of course involve describing the BRDF analytically, but given the results of Stam [107] we are optimistic about this possibility. In recent years there has also been a lot of important work in the computational imaging community related to frequency analysis of light transport and shading [93, 8, 22, 94]. We would like to investigate if these results could be used to further extend our Fourier rendering approach.

Our proof of concept implementation is currently too slow for real time display, however this is a naive implementation and the method can be accelerated in several ways. First, we have the possibility of GPU acceleration which looks very attractive for this solution. The per pixel frequency distribution could be computed in a fragment shader and the transforms are all affine which is very fast to perform using graphics hardware. Second, it would be interesting to find a better form of Eq. 7.13. It may be possible to find an iterative approximation, or express it as a separable function. This would allow for much faster CPU implementations. Third, it would be very interesting to construct a hardware architecture implementing our method. While we have mostly regarded custom made hardware as too expensive and time consuming in this dissertation, and thus an argument for using other approaches to achieve acceleration, the lowered prices and general availability of Field Programmable Gate Arrays may make this an attractive venue for further research.

Finally, all light transport performed in this thesis is done from surface to hologram. However, due to the duality of light transport, the inverse calculation is possible. Thus, we would like to investigate a ray-tracing solution. Instead of tracing single rays, it may be possible to use volumetric primitives such as cones or frustums. An analytic solution for the intersection of these entities and the 3D model corresponding to the one presented in Chapter 7 could probably be derived. To further accelerate the method, some sort of hierarchical solution may be pursued.

8.3 Final thoughts

The main contributions of this thesis has been in the area of wave field synthesis. One main application is rendering for holographic displays, and we have presented two different techniques, based on point and triangle models respectively. Although different experimental holographic display systems have been available for over 10 years, it is only recently that commercial manufacturing attempts have been made. We believe that these displays will open up for new challenges in rendering and computer graphics.

In some aspect, the bottom line may be that one of the basic principles in almost all computer graphics related rendering, the camera projection, disappears. In addition to the technical challenges presented by higher memory bandwidth and processing, this also changes the rendering problem. Instead of a camera, the rendering target will be a window peering into a virtual world. This requires full light information all the way through the scene and to the display. Thus, standard methods building on the projection principle may not be valid, and alternative strategies must be found. We believe that wave front construction and propagation will be an integral part of this.

Bibliography

[1] David Abookasis and Joseph Rosen. Computer-generated holograms of three-dimensional objects synthesized from their multiple angular viewpoints. *JOSA A*, 20:1537–1545, August 2003.

[2] 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*. The MIT Press, 1991.

[3] E. H. Adelson and J. Y. A. Wang. Single lens stereo with a plenoptic camera. *IEEE Transactions on Pattern Analysis and machine Intelligence*, 14(2):99–106, 1992.

[4] Lukas Ahrenberg, Philip Benzie, Marcus Magnor, and John Watson. Computer generated holography using parallel commodity graphics hardware. *Optics Express*, 14(17):7636–7641, August 2006.

[5] Lukas Ahrenberg, Ivo Ihrke, and Marcus Magnor. A mobile system for multi-video recording. In *1st European Conference on Visual Media Production (CVMP)*, pages 127–132, Savoy Place, London, UK, March 2004. Institution of Electrical Engineers.

[6] Lukas Ahrenberg, Ivo Ihrke, and Marcus Magnor. Volumetric reconstruction, compression and rendering of natural phenomena from multi-video data. In Eduard Gröller, I. Fujishiro, Klaus Müller, and T. Ertl, editors, *Volume graphics 2005 : Eurographics/IEEE VGTC workshop proceedings ; Fourth International Workshop on Volume Graphics*, pages 83–90, Stony Brook, New York, USA, 2005. Eurographics.

[7] Lukas Ahrenberg and Marcus Magnor. Light field rendering using matrix optics. *Journal of WSCG*, 14(1-3):177–184, 2006. 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, January 2006.

[8] Ronen Basri and David W. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):218–233, 2003.

[9] Martin J. Bastiaans. The wigner distribution function applied to optical signals and systems. *Optics Communications*, 25(1):26–30, April 1978.

[10] Stephen A. Benton. Survey of holographic stereograms. In *SPIE Vol. 367 Processing and Display of Three-Dimensional Data*, pages 15–18, 1982.

[11] Max Born and Emil Wolf. *Principles of Optics*. Cambridge University Press, seventh edition, 1999.

[12] Michael Bove, Wendy Plesniak, Tyler Quentmeyer, and James Barabas. Real-time holographic video images with commodity pc hardware. In *Proc. SPIE Stereoscopic Displays and Applications*, volume 5664, pages 255–262. SPIE, March 2005.

[13] B. R. Brown and A. W. Lohmann. Complex spatial filtering with binary masks. *Appl. Opt.*, 5:967–969, 1966.

[14] O. Bryngdahl and F. Wyrowski. Digital holography - computer–generated holograms. In Emil Wolf, editor, *Progress in Optics*, volume 28, pages 1–86. North Holland, 1990.

[15] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 425–432. ACM Press, 2001.

[16] Emilio Camahort. *4D Light-Field Modeling and Rendering*. PhD thesis, The University of Texas at Austin, 2001.

[17] Emilio Camahort, Apostolos Lerios, and Donald S. Fussell. Uniformly sampled light fields. In *Rendering Techniques*, pages 117–130, 1998.

[18] Joel Carranza, Christian Theobalt, Marcus Magnor, and Hans-Peter Seidel. Free-viewpoint video of human actors. *ACM Trans. on Computer Graphics*, 22(3), July 2003.

[19] Billy Chen, Eyal Ofek, Heung-Yeung Shum, and Marc Levoy. Interactive deformation of light fields. In *SI3D '05: Proceedings of the 2005*

symposium on Interactive 3D graphics and games, pages 139–146, New York, NY, USA, 2005. ACM Press.

[20] Wei-Chao Chen, Jean-Yves Bouguet, Michael H. Chu, and Radek Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 447–456. ACM Press, 2002.

[21] Ingrid Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, 1992.

[22] Frédo Durand, Nicolas Holzschuch, Cyril Soler, Eric Chan, and François X. Sillion. A frequency analysis of light transport. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pages 1115–1126, New York, NY, USA, 2005. ACM Press.

[23] Janus Egholm and Niels J. Christensen. Rendering compact discs and other diffractive surfaces illuminated by linear light sources. In *GRAPHITE '06: Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia*, pages 329–332, New York, NY, USA, 2006. ACM Press.

[24] Patrick Flandrin. *Time-frequency/time-scale analysis*, volume 10 of *Wavelet Analysis and its Applications*. Academic Press Inc., San Diego, CA, 1999. With a preface by Yves Meyer, Translated from the French by Joachim Stöckler.

[25] Grant R. Fowles. *Introduction to Modern Optics*, chapter 10. Dover Publications, second edition, 1975.

[26] Yann Frauel, Thomas J. Naughton, Osamu Matoba, Enrique Tajahuerce, , and Bahram Javidi. Three-dimensional imaging and processing using computational holographic imaging. *Proceedings of the IEEE, Special Issue on 3-D Technologies for Imaging and Display*, 94(3):636–653, March 2006.

[27] E. R. Freniere, G. G. Gregory, and R. A. Hassler. Edge diffraction in Monte Carlo ray tracing. In R. C. Juergens, editor, *Proc. SPIE Vol. 3780, p. 151-157, Optical Design and Analysis Software, Richard C. Juergens; Ed.*, volume 3780 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 151–157, September 1999.

[28] M. Frigo and S.G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93:216–231, February 2005.

[29] Dennis Gabor. A new microscopic principle. *Nature*, 161:777–778, 1948.

[30] Dennis Gabor. Microscopy by reconstructed wavefronts. In *Proceedings of the Royal Society*, volume 197, pages 454–487, 1949.

[31] Dennis Gabor. Microscopy by reconstructed wavefronts: 2. In *Proceedings of the Physics Society B*, volume 64, pages 449–469, 1951.

[32] A. Gerrard and J. M. Burch. *Introduction to Matrix Methods in Optics*. Dover Publications, 1994.

[33] Arun Gershun. The light field. *Journal of Mathematics and Physics*, 18, 1939. Translated by P. Moon and G. Timoshenko.

[34] Bastian Goldlücke and Marcus Magnor. Space-time isosurface evolution for temporally coherent 3d reconstruction. *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04),* Washington, USA, I:350–355, June 2004.

[35] Bastian Goldlücke, Marcus Magnor, and Bennett Wilburn. Hardware-accelerated dynamic light field rendering. In G. Greiner, H. Niemann, T. Ertl, B. Girod, and H.-P. Seidel, editors, *Proceedings Vision, Modeling and Visualization VMV 2002*, pages 455–462, Erlangen, Germany, November 2002.

[36] Joseph W. Goodman. *Introduction to Fourier Optics*. McGraw-Hill, second edition edition, 1996.

[37] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Computer Graphics (SIGGGRAPH'96 Conf. Proc.)*, pages 43–54. ACM SIGGRAPH, August 1996.

[38] Tobias Haist, Marcus Reicherter, Min Wu, and Lars Seifert. Using graphics boards to compute holograms. *Computing in Science and Engg.*, 8(1):8–13, 2006.

[39] Michael Halle. Holographic stereograms as discrete imaging systems. In *SPIE Proceedings 2176 Practical Holography VIII*, 1994.

[40] Michael Halle. Multiple viewpoint rendering. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 243–254, New York, NY, USA, 1998. ACM Press.

[41] Michael Halle. Autostereoscopic displays and computer graphics. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Courses*, page 104, New York, NY, USA, 2005. ACM Press.

[42] P. Hariharan. *Basics of Holography*. Cambridge University Press, 2002.

[43] Jon Hasselgren and Tomas Akenine-Möller. An efficient multi-view rasterization architecture. In *Proceedings of the EGSR 2006*, pages 67–72, 2006.

[44] Wolfgang Heidrich, Philipp Slusallek, and Hans-Peter Seidel. An image-based model for realistic lens systems in interactive computer graphics. In *Proceedings of the conference on Graphics interface '97*, pages 68–75, Toronto, Ont., Canada, Canada, 1997. Canadian Information Processing Society.

[45] Xianyou Hou, Li-Yi Wei, Heung-Yeung Shum, and Baining Guo. Real-time multi-perspective rendering on graphics hardware. In *Proceedings of the EGSR 2006*, pages 93–102, 2006.

[46] Ivo Ihrke, Lukas Ahrenberg, and Marcus Magnor. External camera calibration for synchronized multi-video systems. In *WSCG '2004 : the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2004 ; short communication papers proceedings*, volume 12 of *Journal of WSCG*, pages 537–544, Plzen, Czech Republic, February 2004. UNION Agency.

[47] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 297–306, New York, NY, USA, 2000. ACM Press/Addison-Wesley Publishing Co.

[48] T. Ito, T. Yabe, M. Okazaki, and M. Yanagi. Special-purpose computer HORN-1 for reconstruction of virtual image in three dimensions. *Computer Physics Communications*, 82:104–110, September 1994.

[49] Tomoyoshi Ito, Hesham Eldeib, Kenji Yoshida, Shinya Takahashi, Takashi Yabe, and Tomoaki Kunugi. Special-purpose computer for

holography horn-2. *Computer Physics Communications*, 93:13–20, January 1996.

[50] Tomoyoshi Ito, Nobuyuki Masuda, Kotaro Yoshimura, Atsushi Shiraki, Tomoyoshi Shimobaba, and Takashige Sugie. Special-purpose computer horn-5 for a real-time electroholography. *Optics Express*, 13:1923–1932, 2005.

[51] Daniel Aaron Kartch. *Efficient rendering and compression for full-parallax computer-generated holographic stereograms.* PhD thesis, Cornell University, 2000. Adviser-Donald P. Greenberg.

[52] Joseph B. Keller. Geometrical theory of diffraction. *J. Opt. Soc. Am.*, 52:116–130, 1962.

[53] Thomas Kreis. *Handbook of Holographic Interfereometry.* Wiley-VCH, 2005.

[54] Paul Lalonde and Alain Fournier. Interactive rendering of wavelet projected light fields. In *Proceedings of the 1999 conference on Graphics interface '99*, pages 107–114. Morgan Kaufmann Publishers Inc., 1999.

[55] Wai-Hon Lee. Computer-generated holograms: Techniques and applications. In Emil Wolf, editor, *Progress in Optics*, volume 16, pages 119–232. North Holland Publishing Company, 1978.

[56] Emmet N. Leith and Juris Upatnieks. Reconstructed waefronts and communications theory. *JOSA*, 52:1123–1130, 1962.

[57] Emmet N. Leith and Juris Upatnieks. Wavefront reconstruction with continuous-tone objects. *JOSA*, 53:1377–1381, 1963.

[58] Emmet N. Leith and Juris Upatnieks. Wavefront reconstruction with diffused illumination and three-dimensional objects. *JOSA*, 54:1295–1301, 1964.

[59] Dan Lelescu and Frank Bossen. Representation and coding of light field data. *Graph. Models*, 66(4):203–225, 2004.

[60] D. Leseberg and C. Frère. Computer-generated holograms of 3-d objects composed of tilted planar segments. *Applied Optics*, 27:3020–3024, July 1988.

[61] Marc Levoy and Pat Hanrahan. Light field rendering. In *Computer Graphics (SIGGGRAPH'96 Conf. Proc.)*, pages 31–42. ACM SIGGRAPH, August 1996.

[62] Marc Levoy, Ren Ng, Andrew Adams, Matthew Footer, and Mark Horowitz. Light field microscopy. *ACM Trans. Graph.*, 25(3):924–934, 2006.

[63] Ming Li, Marcus Magnor, and Hans-Peter Seidel. Hardware-accelerated rendering of photo hulls. *Proc. Eurographics (EG'04),* Grenoble, France, pages 635–642, September 2004.

[64] Michael Liebling, Thierry Blu, and Michael Unser. Fresnelets: New multi-resolution wavelet bases for digital holography. *IEEE Transactions on Image Processing*, 12:29–43, January 2003.

[65] A. W. Lohmann and D. P. Paris. Binary Fraunhofer holograms, generated by computer. *Applied Optics*, 6:1739–1748, 1967.

[66] Mark Lucente. Optimization of hologram computation for real-time display. In S. A. Benton, editor, *Proc. SPIE Vol. 1667, p. 32-43, Practical Holography VI, Stephen A. Benton; Ed.*, volume 1667 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 32–43, May 1992.

[67] Mark Lucente. Interactive computation of holograms using a look-up table. *Journal of Electronic Imaging*, 2(1):28–34, January 1993.

[68] Mark Lucente. *Diffraction-Specific Fringe Computation for Electro-Holography.* PhD thesis, Massachusetts Institute of Technology, September 1994.

[69] Mark Lucente. Interactive three-dimensional holographic displays: seeing the future in depth. *SIGGRAPH Comput. Graph.*, 31:63–67, 1997.

[70] Mark Lucente and Tinsley A. Galyean. Rendering interactive holographic images. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 387–394, New York, NY, USA, 1995. ACM Press.

[71] Reto Lütolf, Bernt Schiele, and Markus H. Gross. The light field oracle. In *Pacific Conference on Computer Graphics and Applications*, pages 116–126, 2002.

[72] Lihong Ma, Hui Wang, Yong Li, and Hongzhen Jin. Numerical reconstruction of digital holograms for three-dimensional shape measurement. *Journal of Optics A: Pure and Applied Optics*, 6:396–400, 2004.

[73] Marcus Magnor and Bernd Girod. Data compression for light field rendering. *IEEE Trans. Circuits and Systems for Video Technology*, 10(3):338–343, April 2000.

[74] Marcus A. Magnor. *Video-Based Rendering*. A. K. Peters, 2005.

[75] Nobuyuki Masuda, Tomoyoshi Ito, Takashi Tanaka, Atsushi Shiraki, and Takashige Sugie. Computer generated holography using a graphics processing unit. *Optics Express*, 14:603–608, 2006.

[76] Kyoji Matsushima. Computer-generated holograms for three-dimensional surface objects with shade and texture. *Applied Optics*, 44(22):4607–4614, August 2005.

[77] Kyoji Matsushima. Exact hidden-surface removal in digitally synthetic full-parallax holograms. In *Practical Holography XIX and Holographic Materials XI*, 2005.

[78] Kyoji Matsushima. Performance of the polygon-source method for creating computer-generated holograms of surface objects. In *Proceedings of ICO Topical Meeting on Optoinfomatics/Information Photonics 2006*, pages 99–100,, 2006.

[79] Kyoji Matsushima and Akinobu Kondoh. Wave optical algorithm for creating digitally synthetic holograms of three-dimensional surface objects. In Sylvia H. Stevenson Tung H. Jeong, editor, *Proceedings of SPIE – Volume 5005 Practical Holography XVII and Holographic Materials IX*, pages 190–197, May 2003.

[80] Kyoji Matsushima, Hagen Schimmel, and Frank Wyrowski. New creation algorithm for digitally synthesized holograms in surface model by diffraction from tilted planes. In S. A. Benton, S. H. Stevenson, and T. J. Trout, editors, *Proc. SPIE Vol. 4659, p. 53-60, Practical Holography XVI and Holographic Materials VIII, Stephen A. Benton; Sylvia H. Stevenson; T. John Trout; Eds.*, volume 4659 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 53–60, June 2002.

[81] Kyoji Matsushima, Hagen Schimmel, and Frank Wyrowski. Fast calculation method for optical diffraction on tilted planes by use of the angular spectrum of plane waves. *JOSA A*, 20(9), 2003.

[82] Kyoji Matsushima and Masahiro Takai. Recurrence formulas for fast creation of synthetic three-dimensional holograms. *Appl. Opt.*, 39:6587–6594, 2000.

[83] Wojciech Matusik, Chris Buehler, and Leonard McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 115–126, London, UK, 2001. Springer-Verlag.

[84] Wojciech Matusik and Hanspeter Pfister. 3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *SIGGRAPH '04: ACM SIGGRAPH 2004 Papers*, pages 814–824, New York, NY, USA, 2004. ACM Press.

[85] Gavin S. P. Miller, Steven M. Rubin, and Dulce B. Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *Rendering Techniques*, pages 281–292, 1998.

[86] Hans P. Moravec. 3d graphics and the wave theory. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques*, pages 289–296, New York, NY, USA, 1981. ACM Press.

[87] P. J. Narayanan, Peter Rander, and Takeo Kanade. Synchronous capture of image sequences from multiple cameras. Technical Report CMU-RI-TR-95-25, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1995.

[88] R. Ng, M. Levoy, M. Bredif, G. Duval, M. Horowitz, and P. Hanrahan. Light field photography with a hand-held plenoptic camera. Technical Report CSTR 2005–02, Stanford Computer Science, 2005.

[89] Ren Ng. Fourier slice photography. *ACM Trans. Graph.*, 24(3):735–744, 2005.

[90] Nvidia. *NVIDIA CUDA Compute Unified Device Architecture Programming Guide*, 0.82 edition, April 2007. http://developer.nvidia.com/object/cuda.html.

[91] Ingmar Peter and Wolfgang Straßer. The wavelet stream: Interactive multi resolution light field rendering. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 127–138. Springer-Verlag, 2001.

[92] Christoph Petz and Marcus Magnor. Fast hologram synthesis for 3d geometry models using graphics hardware. In *Practical Holography XVII and Holographic Materials IX*, pages 266–275. SPIE, 2003.

[93] Ravi Ramamoorthi and Pat Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 117–128, New York, NY, USA, 2001. ACM Press.

[94] Ravi Ramamoorthi, Dhruv Mahajan, and Peter Belhumeur. A first-order analysis of lighting, shading, and shadows. *ACM Trans. Graph.*, 26(1):2, 2007.

[95] Alf Ritter, Joachim Böttger, Oliver Deussen, Matthias König, and Thomas Strothotte. Hardware-based rendering of full-parallax synthetic holograms. *Applied Optics*, pages 1364–1369, 1999.

[96] G. L. Rogers. The production of a lenticular stereogram from a hologram. *J. Phys. E: Sci. Instrum.*, 1:473–474, 1968.

[97] Randi J. Roost. *OpenGL Shading Language*. Addison-Wesley Professional, second edition, 2006.

[98] Amir Said and William A. Pearlman. A new fast and efficient image codec based on set partitioning in hierarchical trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6:243–250, 1996.

[99] Ulf Schnars and Werner Jueptner. *Digital Holography*. Springer, 2005.

[100] Ulf Schnars and Werner P O Jüptner. Digital recording and numerical reconstruction of holograms. *Measurement Science and Technology*, 13(9):R85–R101, 2002.

[101] T. Shimobaba, N. Masuda, T. Sugie, S. Hosono, S. Tsukui, and T. Ito. Special-purpose computer for holography horn-3 with pld technology. *Computer Physics Communications*, 130:75–82, July 2000.

[102] T. Shimobaba, Hishinuma S., and Ito T. Special-purpose computer for holography horn-4 with recurrence algorithm. *Computer Physics Communications*, 148:160–170, October 2002.

[103] Tomoyoshi Shimobaba and Tomoyoshi Ito. An efficient computational method suitable for hardware of computer-generated hologram with phase computation by addition. *Computer Physics Communications*, 138(1):44–52, July 2001.

[104] Cris Slinger, Colin Cameron, and Maurice Stanley. Computer-generated holography as a generic display technology. *Computer*, 38:46–53, August 2005.

[105] Peter-Pike Sloan and Charles Hansen. Parallel lumigraph reconstruction. In *PVGS '99: Proceedings of the 1999 IEEE symposium on Parallel visualization and graphics*, pages 7–14. ACM Press, 1999.

[106] P. St.-Hilaire, S. A. Benton, M. E. Lucente, M. L. Jepsen, J. Kollin, H. Yoshikawa, and J. S. Underkoffler. Electronic display system for computational holography. In S. A. Benton, editor, *Proc. SPIE Vol. 1212, p. 174-182, Practical Holography IV, Stephen A. Benton; Ed.*, volume 1212 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pages 174–182, May 1990.

[107] Jos Stam. Diffraction shaders. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 101–110, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[108] A. D. Stein, Z. Wang, and J. S. Leigh, Jr. Computer-generated holograms: A simplified ray-tracing approach. *Computers in Physics*, 6:389–392, July 1992.

[109] Timo Stich, Art Tevs, and Marcus Magnor. Global depth from epipolar volumes - a general framework for reconstructing non-lambertian surfaces. In *Third International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 1–8. IEEE Computer Society, 2006.

[110] Eric J. Stollnitz, Tony D. Derose, and David H. Salesin. *Wavelets for computer graphics: theory and applications*. Morgan Kaufmann Publishers Inc., 1996.

[111] Yinlong Sun, F. David Fracchia, Mark S. Drew, and Thomas W. Calvert. Rendering iridescent colors of optical disks. In *Proceedings of the Eurographics Workshop on Rendering Techniques 2000*, pages 341–352, London, UK, 2000. Springer-Verlag.

[112] Marek Sutkowski and MaImagegorzata KujawiImageska. Application of liquid crystal (lc) devices for optoelectronic reconstruction of digitally stored holograms. *Optics and Lasers in Engineering*, 33:191–201, March 2000.

[113] Christian Theobalt, Ming Li, Marcus Magnor, and Hans-Peter Seidel. A flexible and versatile studio for synchronized multi-view video recording. *Proc. IMA Vision, Video, and Graphics 2003 (VVG'03),* Bath, UK, July 2003.

[114] Tullio Tommasi and Bruno Bianco. Frequency analysis of light diffraction between rotated planes. *Optics Letters*, 17:556–558, April 1992.

[115] Tullio Tommasi and Bruno Bianco. Computer-generated holograms of tilted planes by a spatial frequency approach. *JOSA A*, 10(2):299–305, February 1993.

[116] G. Tricoles. Computer generated holograms: an historical review. *Appl. Opt.*, 26:4351–4360, 1987.

[117] Nicolas Tsingos. A geometrical approach to modeling reflectance functions of diffracting surfaces. Technical report, Bell LabsMarch, March 2000.

[118] J. Watlington, M. Lucente, C. Sparrell, V. Bove, and J. Tamitani. A hardware architecture for rapid generation of electro-holographic fringe patterns. In *SPIE Practical Holography IX*, volume 2406, pages 172–183, 1995.

[119] Bennet Wilburn, Michael Smulski, Hsiao-Heng Kelin Lee, and Mark Horowitz. The light field video camera. In *Proceedings of Media Processors 2002, SPIE Electronic Imaging 2002*, 2002.

[120] Tien-Tsin Wong, Pheng-Ann Heng, Siu-Hang Or, and Wai-Yin Ng. Image-based rendering with controllable illumination. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97*, pages 13–22. Springer-Verlag, 1997.

[121] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Tom Duchamp, David H. Salesin, and Werner 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. ACM Press/Addison-Wesley Publishing Co., 2000.

[122] Stephan Würmlin, Edouard Lamboray, Oliver G. Staadt, and Markus H. Gross. 3d video recorder: a system for recording and playing free-viewpoint video. *Comput. Graph. Forum*, 22(2):181–194, 2003.

[123] I. Yamaguchi and T. Zhang. Phase-shifting digital holography. *Opt. Lett*, 22:1268–1270, 1997.

[124] Jason C. Yang, Matthew Everett, Chris Buehler, and Leonard McMillan. A real-time distributed light field camera. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 77–86, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[125] Ruigang Yang, Xinyu Huang, and Shunnan Chen. Efficient rendering of integral images. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Posters*, page 44, New York, NY, USA, 2005. ACM Press.

[126] Zhunping Zhang, Lifeng Wang, Baining Guo, and Heung-Yeung Shum. Feature-based light field morphing. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 457–464, New York, NY, USA, 2002. ACM Press.

[127] Remo Ziegler, Simon Bucheli, Lukas Ahrenberg, Marcus Magnor, and Markus Gross. A bidirectional light field - hologram transform. In *Proceedings of Eurographics 2007 – to appear*, 2007.

[128] Remo Ziegler, Peter Kaufmann, and Markus Gross. A framework for holographic scene representation and image synthesis. Technical report, Swiss Federal Institute of Technology Zurich, 2006.

Data sources

We would like to acknowledge the following persons and institutions for kindly allowing us to use data and resoures:

The Buddha light field used in Chapter 4 was downloaded from the (old) Stanford Light Fields Archive[1], while the Bunny model used in Chapter 6 was downloaded from the Stanford 3D Scanning Repository[2]. Both these resources are provided online by Stanford University Computer Graphics Laboratory.

The chess knight hologram used in Chapter 5 was kindly provided by Ervin Kolenovic and Jan Mueller at Bremer Institut für angewandte Strahltechnik.

The Bird and "Eight" models used in Chaper 7 are courtesy of DISI and INRIA respectively. The Max Planck bust model used in some of the illustrations in Chapter 2 is courtesy of Max-Planck-Institut für Informatik. All three models were kindly provided online by the Aim@Shape project[3].

[1] http://graphics.stanford.edu/software/lightpack/lifs.html
[2] http://graphics.stanford.edu/data/3Dscanrep/
[3] http://www.aimatshape.net/

Curriculum Vitæ

1976	Born at Baldersnäs, Steneby, Sweden
1983 – 1989	Ekhagsskolan, Dals Långed, Sweden (Primary School)
1989 – 1992	Bengstgården, Bengtsfors, Sweden (Primary School)
1992 – 1995	Karbergsskolans Gymnasium, Åmal, Sweden (High School)
1995 – 2000	Study of Computer Science, Uppsala University, Sweden
2002 – 2007	Ph.D. Student at the Max-Planck-Institut für Informatik, Saarbrücken, Germany

Lebenslauf

1976	Geboren in Baldersnäs, Steneby, Schweden
1983 – 1989	Ekhagsskolan, Dals Långed, Schweden (Grundskola)
1989 – 1992	Bengstgården, Bengtsfors, Schweden (Grundskola)
1992 – 1995	Karbergsskolans Gymnasium, Åmal, Schweden
1995 – 2000	Studium der Informatik, Universität Uppsala, Schweden
2002 – 2007	Doktorand am Max-Planck-Institut für Informatik, Saarbrücken, Deutschland