THE UNIVERSITY OF CALGARY

4D Light Field Processing and its Application to Computer Vision

by

Donald G. Dansereau

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER
ENGINEERING

CALGARY, ALBERTA
October, 2003

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a thesis entitled "4D Light Field Processing and its Application to Computer Vision" submitted by Donald G. Dansereau in partial fulfillment of the requirements for the degree of Master of Science.

Supervisor, Dr. Len Bruton
Department of Electrical and Computer Engineering

Dr. John Nielsen
Department of Electrical and Computer Engineering

Dr. Brian Wyvill
Department of Computer Science

Date

Abstract

Light fields have been explored extensively as a means of quickly rendering images of 3-dimensional scenes from novel camera positions. Because a light field models the light rays permeating a scene, rather than modelling the geometry of that scene, the process of rendering images from a light field is fast, with a speed which is independent of scene complexity.

The light field itself is a 4-dimensional data structure, representing the values of the light rays permeating a scene as a function of their positions and directions. Because a light field can be used to model a real-world scene, and because the resulting model contains a wealth of information about that scene, simple and robust techniques may be applied to light fields to accomplish complex tasks.

This work develops methods of extracting useful information from light field models of real-world scenes. In particular, techniques are developed for filtering light fields based on depth, and for estimating the geometry of the scenes that they model. Two classes of depth filters are explored: the first selectively transmits objects which lie at a single prescribed depth in the scene, while the second selectively transmits objects which lie within a range of prescribed depths. Three classes of shape estimation algorithms are explored, all of which estimate the geometry of a scene based on the characteristics of the corresponding light field model.

The techniques explored here accomplish complex tasks using robust and simple methods, and might therefore be useful in a range of computer vision applications as diverse as robot navigation, scene modelling, and object recognition.

Acknowledgements

Throughout the course of this work, I have been helped along by a number of people. First off, my supervisor, Dr. Len Bruton, has given me invaluable technical support and guidance. His ability to provide insight, even in new and unfamiliar territory, has guided me through several difficult problems.

Funding for this project has come from the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Alberta Informatics Circle of Research Excellence (iCORE).

Thanks to my father for help in the construction of the camera mounting bracket used with the camera gantry.

Thanks to the Dominion Radio Astrophysical Observatory (DRAO), Guinness®, and the Kobe Beef eatery for providing the three scene elements which make up the light fields featured in this work.

Thanks to the following present and past members of the MDSP group: Ben Anderson, Leila Khademi, Santosh Singh, Arjuna Madanayake and Bernhard Kuenzle, for sometimes enlightening, and always entertaining, discussions. Likewise to table 200 at the Kilkenny Irish Pub.

Finally, a special thanks to my family and friends for their constant support, and especially to Linda for her nearly unwavering patience.

Guinness® is a registered trademark of Guinness Ltd.

To my parents

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

∇^{su}	$[\nabla_s^{su}, \nabla_u^{su}]$: 2D gradient operator, applied in the s and u dimensions of a light field; $\nabla^{su} \in \mathbb{R}^2$.
∇^{tv}	$[\nabla_t^{tv}, \nabla_v^{tv}]$: 2D gradient operator, applied in the t and v dimensions of a light field; $\nabla^{tv} \in \mathbb{R}^2$.
$\varepsilon(\mathbf{p})$	The error associated with the point $\mathbf{p}$, used in estimating p_z.
$[\theta, \phi]$	Angles used to define the direction of a ray, as in spherical coordinates.
θ_c	Angle of the central axis of a fan shape with respect to the Ω_u and Ω_v axes.
θ_z	Angle of the hyperplane corresponding to the depth p_z, with respect to the Ω_u and Ω_v axes.
κ	Memory increase factor used in zero-phase filtering.
ω	$[\omega_s, \omega_t, \omega_u, \omega_v]$: single point in discrete 4D frequency space; $\omega \in \mathbb{R}^4$.
$\mathbf{\Omega}$	$[\Omega_s, \Omega_t, \Omega_u, \Omega_v]$: single point in continuous 4D frequency space; $\mathbf{\Omega} \in \mathbb{R}^4$.
2PP	Two-plane parameterization.
<n>D	n-dimensional.
$\mathbf{a}$	$[a_x, a_y]$: discrete index into a 2D map; $\mathbf{a} \in \mathbb{N}^2$.

$\mathbf{a}'$	$[a'_x, a'_y]$: continuous-domain index into a 2D map, used in interpolation; $\mathbf{a}' \in \mathbb{R}^2$.
$A(\mathbf{a})$	2D map.
$\tilde{A}(\mathbf{a}')$	Value interpolated from a 2D map at the location corresponding to the continuous-domain index $\mathbf{a}'$.
ASIC	Application-specific integrated circuit.
B	The -3dB bandwidth of a filter.
B_k	The -3dB bandwidth of the k^{th} sub-band processing filter in a 4D-DFFB.
BIBO	Bounded-input bounded-output.
c	Constant bandwidth offset, used with the 4D-DFFB.
$\mathbf{C}$	$[C_x, C_y, C_z]$: the camera's position in 3D space; $\mathbf{C} \in \mathbb{R}^3$.
CMOS	Complimentary metal-oxide semiconductor.
d	Distance between the two reference planes.
$\hat{\mathbf{d}}$	$[d_s, d_t, d_u, d_v]$: normal of a hyperplane in 4D; $\hat{\mathbf{d}} \in \mathbb{R}^4$.
$\hat{\mathbf{d}}'$	$[d'_s, d'_t, d'_u, d'_v]$: normal of a hyperplane in 4D, corrected for differing sample rates in the four dimensions; $\hat{\mathbf{d}}' \in \mathbb{R}^4$.
DFFB	Dual-fan filter bank.
$\hat{\mathbf{d}}^{su}$	$[d_s^{su}, d_u^{su}]$: normal of a hyperplane, where the normal lies entirely in the s and u dimensions; $\hat{\mathbf{d}}^{su} \in \mathbb{R}^2$.

$\hat{\mathbf{d}}^{tv}$	$[d_t^{tv}, d_v^{tv}]$: normal of a hyperplane, where the normal lies entirely in the t and v dimensions; $\hat{\mathbf{d}}^{tv} \in \mathbb{R}^2$.
d_z	Target depth in a depth-based filtering operation.
$\mathbf{D}$	$[D_s, D_t, D_u, D_v]$: physical dimensions of the reference planes, in meters; $\mathbf{D} \in \mathbb{R}^4$.
$\mathbf{D'}$	$[D'_x, D'_y]$: physical dimensions of a 2D image plane, in meters; $\mathbf{D'} \in \mathbb{R}^2$.
$\mathbf{F}$	$[F_x, F_y]$: field of view in the x and y directions, in radians; $\mathbf{F} \in \mathbb{R}^2$.
FIR	Finite impulse response.
FOV	Field of view.
FPGA	Field programmable gate array.
$h_{BP}^k(n)$	Impulse response of the 1D bandpass filter which separates the k^{th} sub-band in a filter bank.
$h_{LP}(n)$	Impulse response of the 1D lowpass filter from which $h_{BP}^k(n)$ are built.
$\mathbf{I}$	$[I_x, I_y]$: number of samples in a 2D map; $\mathbf{I} \in \mathbb{N}^2$.
IIR	Infinite impulse response.
L_i	Values of the inductors used in constructing an analog prototype filter.
$\bar{L}_{P_p}$	Average value of the light field samples which lie in the plane P_p.
$L(\mathbf{n})$	Sampled light field.

$L_{cont}(\mathbf{r})$	Continuous-domain light field.
$\tilde{L}_{cont}(\mathbf{n}')$	Iterpolated value of the light field at the location corresponding to the continuous-domain index $\mathbf{n}'$.
$\tilde{L}_{cont}(\mathbf{r})$	Estimate of the value of the ray $\mathbf{r}$ obtained through interpolation at the corresponding continuous-domain index $\mathbf{n}'$.
$L_{freq}(\mathbf{\Omega})$	4D continuous-domain Fourier transform of the light field.
$L^k(\mathbf{n})$	The k^{th} sub-band signal in a filter bank.
$\mathbf{M}$	$[M_x, M_y]$: extent of the u, v reference plane contained within the FOV of the camera; $\mathbf{M} \in \mathbb{R}^2$.
m_∇	Slope of a 2D gradient vector in s and u, or in t and v.
m_i	Displacement of a tracked neighbourhood in the i^{th} frame along s – used in feature tracking-based depth estimation.
m_j	Displacement of a tracked neighbourhood in the j^{th} frame along t – used in feature tracking-based depth estimation.
m_{plane}	Slope of a 4D plane in s and u, or equivalently t and v.
MSE	Mean squared error.
n	Discrete 1D index; $n \in \mathbb{N}$.
$\mathbf{n}$	$[n_s, n_t, n_u, n_v]$: discrete 4D index; $\mathbf{n} \in \mathbb{N}^4$.

$\mathbf{n}'$	$[n'_s, n'_t, n'_u, n'_v]$: continuous-domain index into a 4D light field, used in interpolation; $\mathbf{n}' \in \mathbb{R}^4$.
n_0	Delay introduced by the sub-band separation filters of a fan filter bank.
$\mathbf{n}_0$	$[n_{0_s}, n_{0_t}, n_{0_u}, n_{0_v}]$: index of the sample at the center of a neighbourhood to be tracked – used in feature tracking-based depth estimation.
$\mathbf{N}$	$[N_s, N_t, N_u, N_v]$: number of samples in the light field in each dimension; $\mathbf{N} \in \mathbb{N}^4$.
N_b	Number of sub-bands used by a 4D-DFFB.
N_i	Number of frames through which a feature is tracked along the s dimension, in feature tracking-based depth estimation.
N_j	Number of frames through which a feature is tracked along the t dimension, in feature tracking-based depth estimation.
N_z	Number of candidate depths used in plane variance minimization.
O	Order of an FIR filter.
$\mathbf{p}$	$[p_x, p_y, p_z]$: single point in 3D space; $\mathbf{p} \in \mathbb{R}^3$.
P_p	The plane in a 4D light field which corresponds to the point $\mathbf{p}$.
$P(\ldots)$	$P(\theta, \phi, \lambda, t, p_x, p_y, p_z)$: the 7D plenoptic function.

$P_{rgb}(\ldots)$	$P_{rgb}(\theta, \phi, t, p_x, p_y, p_z)$: the colour plenoptic function: a subset of the plenoptic function with a value given by the colour triplet, $[r, g, b]$.
q	Number of lobes of the sinc function to be included in $h_{LP}(n)$; $q \in \mathbb{N}$.
$\mathbf{r}$	$[r_s, r_t, r_u, r_v]$ or $[s, t, u, v]$: single point in 4D ray space; $\mathbf{r} \in \mathbb{R}^4$.
R	Value of the resistor used in constructing an analog prototype filter.
ROS	Region of support.
$\hat{\mathbf{v}}$	$[v_x, v_y, v_z]$: normalized direction vector in 3D space; $\hat{\mathbf{v}} \in \mathbb{R}^3$.
$\mathbf{v}'$	$[v'_x, v'_y, v'_z]$: un-normalized direction vector parallel with $\hat{\mathbf{v}}$; $\mathbf{v}' \in \mathbb{R}^3$.
$\hat{\mathbf{V}}_f$	$[v_{f_x}, v_{f_y}, v_{f_z}]$: normalized direction vector pointing along the optical axis of the camera; $\hat{\mathbf{V}}_f \in \mathbb{R}^3$.
$\hat{\mathbf{V}}_r$	$[v_{r_x}, v_{r_y}, v_{r_z}]$: normalized direction vector orthogonal to the optical axis of the camera, pointing to the right in the camera's frame of reference; $\hat{\mathbf{V}}_r \in \mathbb{R}^3$.
$\hat{\mathbf{V}}_u$	$[v_{u_x}, v_{u_y}, v_{u_z}]$: normalized direction vector orthogonal to the optical axis of the camera, pointing up in the camera's frame of reference; $\hat{\mathbf{V}}_u \in \mathbb{R}^3$.

$\mathbf{W}$	$[W_x, W_y]$: size of the window in a 2D moving average filter; $\mathbf{W} \in \mathbb{N}^2$.
$w(n)$	Rectangular window.
$x(\mathbf{n})$	Generic input signal.
$y(\mathbf{n})$	Generic output signal.
z_{max}	Maximum depth.
z_{min}	Minimum depth.

Chapter 1

Introduction

We exist in a sea of light. Our world is filled with a seemingly endless number of photons – infinitesimal packets of light that are constantly reflecting, refracting, interfering, and sometimes passing straight through us, all at the speed limit of the universe. Given that they exhibit such a huge range of behaviours, it seems odd that photons are the primary means by which we perceive our reality. As it turns out, the value of the information that light conveys outweighs the computational cost associated with perceiving it, and so at some point in history our sense of sight evolved. It has since become so strong as to become our primary sense, to the point where the visual appearance of reality – that is, the model of reality that we form based on the photons striking our eyes – and reality itself, are sometimes confused. Some animals fail to recognize their own image in a mirror, for example, because of the assumption that the *image* of an entity behind the mirror is equivalent to *presence* of that entity behind the mirror.

Given the complexity of the process of visual perception, which has taken billions of years to evolve, it seems an insurmountable task to fully understand it. And yet the desire to recreate this sense in automated systems is undeniable. Since the first experiments in computing carried out by Charles Babbage, Alan Turing and John von Neumann, we have seen computing devices revolutionize almost every aspect of our existence. Starting with pure math, progressing into thermodynamic and other physical modelling tasks, then into the worlds of audio and video processing,

and most recently into the realm of wireless communication, we have steadily been endowing computers with the senses which they use to interact with our world. And yet, when it comes to analyzing visual data, today's technology falls short of what we know to be possible.

Perhaps this shortcoming is a symptom of the fact that the most sophisticated modern visual processing tasks are not dedicated to the *analysis* of images, but rather to their *synthesis*, mostly in the form of computer-generated imagery for movies and video games. Historically, as the sophistication of tasks in graphical synthesis evolved, the computing power necessitated by them increased proportionally. It took some time for the available computing power to become sufficient for tasks in visual analysis, and so this field has historically lagged that of synthesis. Coupled with the tremendous economic drive associated with synthesized graphics in movies and video games, the gap between visual analysis and synthesis has only continued to grow. Modern examples of analysis in face recognition and automated scene modelling, for example, have begun to yield encouraging results, yet fail to even approach the performance and generality of the human visual system. Meanwhile, the more mature field of visual synthesis has progressed to the point of completely fooling the human visual system with synthetic images.

Given the sophistication of the more mature field of visual synthesis, it seems possible that techniques from that field might be adopted in order to advance the field of visual analysis. That is the premise of this thesis: to adapt the light field [1], a concept which first appeared in the context of synthesis, to performing tasks in visual analysis.

1.1 Modelling an Image or Imaging a Model

When the field of computer graphics was first being explored, 2D images were typically stored using a *vector* representation, in which the image was defined in terms of a set of 2D primitives such as lines, triangles, circles, and so on. This contrasts with most modern techniques, which utilize a *raster* representation in which an image is defined in terms of its individual pixels. Raster representations define complex images more efficiently, with the tradeoff of representing simple images less efficiently. These two different approaches to the representation of 2D images can been seen as being, respectively, *model*-based, requiring a model of the shapes that are to make up the image, and *image*-based, requiring a representation of each individual pixel of the image, but containing no explicit knowledge of the shapes contained within the image.

Several recent advancements in the realm of audio-visual processing have involved the exploration of model-based alternatives to traditionally image-based techniques, and vice-versa. One example of this is the model-based encoding of human heads [2] for video conferencing. Video conferencing is classically achieved using a purely image-based technique, in which images of the speaker are transmitted in sequence. The model-based approach differs in that a model of the head is first established, and then only the information required to manipulate that model is sent over the communication channel. The results are similar to those obtained with image-based video-conferencing, but at a small fraction of the bandwidth.

More subtle than the video-conferencing example is its audio processing equivalent: voice encoding technology that uses a model of the human speech system to

more efficiently represent voice signals [3]. It is so effective that most users whose cell phones utilize this technology are entirely unaware of it.

Some image-based alternatives to classically model-based solutions have also recently emerged. In the field of computer graphics, the process of bump mapping is essentially one of using an image-based model to represent the fine details of a surface's shape, rather than attempting to form a geometric model of those details. Taking this kind of approach to its extreme, an entirely image-based model of a scene may be formed, in which no aspect of the scene's geometry is explicitly represented. The advantage of such a representation is that, like bump mapping and raster graphics, it represents complex details as readily as simple ones, and so the process of rendering images from such a model will be equally fast for any scene complexity. The light field [1] is an example of this kind of image-based approach.

Although it was first explored in the context of synthesizing images, the possibility of using the light field in visual analysis is attractive. It is a fairly simple matter to generate a light field model of a *real-world* scene, and the resulting model contains a vast amount of information about the scene that it represents. By applying simple algorithms to a light field, extensive information about the scene that it represents can be extracted. This presents an appealing alternative to classical approaches to computer vision, in which a geometric model of a scene is generated on-the-fly – that is, using images of the scene as they become available – resulting in an accumulation of errors in the model over time. Because the light field approach forgoes the formation of a geometric model in favour of directly recording image-based information, no errors are accumulated during the formation of the model – a light field model is always guaranteed to accurately reflect the contents of the scene

that it models.

The light field model presents some unique advantages over classical geometric representations in computer vision, and it is therefore a good choice for migration from tasks in visual synthesis towards tasks in visual analysis.

1.2 Contributions of this Thesis

This thesis introduces new approaches to machine vision which utilize the light field model as an intermediary between the real world and the computer's analysis of a scene. More specifically, algorithms useful for the extraction of objects and the estimation of shape from light fields of real world scenes are explored. An emphasis is placed on *simple* and *robust* techniques which, when coupled with an appropriate light field measurement system, might be implemented in real-time.

The light field structure is described in detail in Chapter 2. Because we use light to perceive objects in 3D space, it is most natural to think of light in terms of photons traveling through the three spatial dimensions. This chapter introduces another way of representing light, in which *rays* of light are described in terms of their positions and directions. The result is a representation of light rays in 4D *ray space*. Though this space is less intuitive than the 3D geometric space we are used to, it effectively represents a huge amount of information about the light permeating a scene, and allows for the simple, robust techniques which are developed in later chapters.

Chapter 3 deals with the practical aspects of working with light fields. Methods for measuring, storing, and rendering images from light fields are presented.

Chapter 4 deals with light field characteristics, starting with some useful ways of visualizing light fields before delving into the specifics of the point-plane correspondence and its ramifications in the frequency domain. The observations made in this chapter form the basis for the rest of the thesis.

Chapters 5 and 6 present methods for extracting objects from a light field model based on their depth in the scene that it models. The first of the two chapters presents both a 4D plane averaging filter and a 4D frequency-planar filter [4], both of which extract objects at a single depth in a scene. The second presents a 4D dual-fan filter bank (4D-DFFB), which extracts objects within a range of depths.

Chapter 7 is concerned with estimating the shapes of scenes. The basic premise is that the shape of a scene can be found by estimating the depth, and thus the 3D location, of every visible point in the scene. Three techniques are presented for accomplishing depth estimation, based on plane variance, 2D gradients, and feature tracking, respectively.

Chapter 8 concludes the thesis with a brief discussion of how the techniques described might be used in real-world applications, and an indication of possible directions for future work.

Chapter 2

Representing Light

It is interesting that the behaviour of electromagnetic waves varies so widely. At low frequencies, a wave might diffract around entire buildings as though they were pebbles in a pond, while at higher frequencies reflecting off them as though they were perfect, monolithic mirrors. It is fortunate, in a sense, that humans only perceive light in a narrow range of frequencies, over which its behaviour tends towards the reflective behaviours of rays, and away from the refractive behaviours of waves. This fact allows us to model light in terms of rays, and neglect many of the more complex, wave-like behaviours associated with electromagnetic radiation in general.

The first section of this chapter introduces the plenoptic function, which describes the set of all light rays passing through space-time. Subsequent sections deal with the process of representing subsets of the plenoptic function, including the particular subset referred to as the light field.

2.1 The Plenoptic Function

In order to describe a single ray of light, one must account not only for its spectral content, which determines what we perceive as its colour, but also its other characteristics, such as its position in space, the direction in which it is traveling, and how it changes over time. Most completely, one may describe a single, monochromatic ray of light in terms of seven quantities: three for position: p_x, p_y and p_z; two for di-

rection: θ and ϕ; one for wavelength: λ; and one for time: t. The plenoptic function is the function $P(\theta, \phi, \lambda, t, p_x, p_y, p_z)$ which describes the intensity of light rays as a function of these seven quantities [5]. This function can be thought of as describing a 7D ray space, where each point in this space corresponds to a single light ray.

The process of visual perception is one of measuring and extracting information from a subset of the plenoptic function. Because it would be impractical to experience the entire spectrum of light, over all space, in all directions, and over all time, we make do with a subset of this information. A colour photograph taken with a pinhole camera, for example, measures the plenoptic function over the range of frequencies occupied by visible light, for a fixed position and time, and over a finite range of directions. Otherwise stated, the photograph is a recording of $P(\theta, \phi, \lambda, t_0, p_{x_0}, p_{y_0}, p_{z_0})$, where the '0' subscripts indicate that t_0, p_{x_0}, p_{y_0} and p_{z_0} are each fixed at a single value corresponding to the location of the pinhole in space-time, while λ varies over the range of visible light, and θ and ϕ vary over a range corresponding to the field of view (FOV) of the camera. Because only three of the parameters may vary, and the others are fixed, the colour photograph may be considered to represent a 3D slice of the plenoptic function.

The human visual system may be very roughly modelled as two pinhole cameras, at separate locations in space, and recording a sequence of images over time. Assuming the two eyes are separated along the x axis, this system essentially measures a 5D slice of the plenoptic function, $P(\theta, \phi, n_\lambda, t, n_x, p_{y_0}, p_{z_0})$. Note that the p_x and λ dimensions are sampled at discrete points, as denoted by the use of the index variables n_x and n_λ. In p_x only two samples are taken, at the points in space corresponding to the two eye positions. Similarly, the λ dimension may be thought

of as being sampled at three points, corresponding roughly to the wavelengths of red, green and blue light. This latter observation is a broad simplification of the way that the human eye perceives colour – in reality we perceive colour in three extended and overlapping frequency bands, with centroids which coincide roughly with the frequencies of red, green and blue light. The representation of colour as a triplet, given as 'red', 'green' and 'blue' components, is widely utilized in digital media due to its simplicity. Adopting this scheme, we may approximate the subset of the plenoptic function that the human visual system perceives using the *colour* plenoptic function $P_{rgb}(\theta, \phi, t, n_x, p_{y_0}, p_{z_0})$, where the 'rgb' subscript on the function indicates that its value is given by a colour triplet, $[r, g, b]$.

There are obviously some very good reasons, from an evolutionary point of view, for two eyes, as this number seems to have been settled on by most forms of life on Earth. Most probably, this is because two is the fewest number of eyes that can be used to establish depth from a scene while remaining stationary. There is no reason, however, to remain constrained to this number of eyes in computer vision. It is possible to add a third or fourth eye – or camera, in the case of computer vision – along the x axis, for example. Taking this a step further, one might add cameras separated along another axis – say, the y axis. In the limit, one might imagine a purely theoretical camera array in the x, y plane, consisting of an infinite number of cameras to represent the continuous domain, and measuring a 5D subset of the colour plenoptic function. At a single instant in time, this array of cameras would measure the 4D slice of the colour plenoptic function given by $P_{rgb}(\theta, \phi, t_0, p_x, p_y, p_{z_0})$. This 4D slice of the colour plenoptic function is that subset which is represented in a light field. As we will see in following sections, it has the distinct advantage of

representing just enough information about the light in a scene to allow a rendering system to generate 2D images of the scene from novel camera positions.

2.2 Image-Based Modelling

Classical techniques in computer graphics seek to generate novel views of 3D scenes based on geometric models of those scenes. This process has a complexity which depends on the complexity of the geometry of the scene. It is also limited in the extent to which it accurately represents the ways in which light interacts with a surface, as it makes use of lighting and surface models which are necessarily a simplification of reality. These two factors limit the effectiveness of geometry-based modelling, particularly in real-time applications.

Image-based modelling came about as an alternative to geometry-based modelling, with the realization that the process of rendering an image of a scene is essentially one of calculating the value of an appropriate subset of the plenoptic function for that scene. This leads to the idea that if one could somehow represent the plenoptic function itself, or some appropriate subset of that function, one could render novel views of a scene without ever being concerned with geometric, lighting, or surface models. The goal of image-based rendering can be summarized as the generation of a continuous representation of the plenoptic function given an incomplete set of discrete samples of that function [6]. The image-based model, then, is a set of samples of some subset of the plenoptic function.

The speed of image-based rendering is the strength which initially drove its development. Novel views can be rendered quickly, independent of the complexity of the

geometry, lighting, and surface properties of a scene. For scenes with geometry that is sufficiently complex to prevent classical rendering techniques from being practical – say, for scenes containing hair or foliage – image-based rendering is particularly well suited.

It wasn't until after being introduced, in the context of rendering, that image-based techniques were explored as a means of performing tasks in analysis. The process of sampling a subset of the plenoptic function for even the most complex real-world scenes can be extremely simple – much simpler than that of forming a geometric model of such scenes, as anyone in that area of research will confirm. Because image-based models contain such a vast amount of information about the scenes that they represent, and because of the ease with which they may be measured, they are well suited to tasks in visual analysis.

Image-based techniques are not without their shortcomings, however. Scenes with complex geometry will typically require many samples of the plenoptic function in order to be accurately represented. Even the 4D subset of the colour plenoptic function with which light fields are concerned may require a huge amount of memory – on the order of hundreds of megabytes for a typical light field – to be represented. Because of their size, processing and analysis of image-based models can be prohibitively time-consuming, necessitating simple techniques for most practical applications.

2.3 Parameterization Methods

Depending on the amount of information required by a particular application, image-based models may represent different subsets of the plenoptic function. For an interactive rendering system, for example, the freedom with which the camera may move through the virtual environment is determined by the dimensionality of the model. If the model represents the 3D sampled subset of the colour plenoptic function given by $P_{rgb}(n_\theta, n_\phi, t_0, n_x, p_{y_0}, p_{z_0})$, the modelled scene must be static, and the camera must be constrained to move along the x axis – note that n_θ and n_ϕ correspond to multiple, discrete sample points, while t_0, p_{y_0} and p_{z_0} are each fixed at a single value. Because t_0, p_{y_0} and p_{z_0} are all fixed, there is no information in the model with regards to the behaviour of the light rays in those dimensions.

In the case of a 5D slice of the colour plenoptic function, $P_{rgb}(n_\theta, n_\phi, t_0, n_x, n_y, n_z)$, there is enough information to allow completely free camera motion – that is, translation and rotation in all possible directions. One might conclude that this is the best subset to utilize for a generic rendering system, though there is significant redundancy associated with it. There is a simplification which yields a 4D subset without significant penalty [1] [7] – it is the subset represented by the light field. By imposing the constraint that the light rays in a scene must be of constant value everywhere along their directions of propagation, the dimensionality may be reduced by one. This is essentially exploiting the similarity of all light rays propagating along any single line: the rays must all have the same value along the line, so there is no need to represent them at all points on that line – it suffices to represent them at one point on the line. This does, however, impose two restrictions on the types of

scenes which may be represented. First, the light rays must propagate through a clear medium, such as air, and not an attenuating medium such as fog. Second, the scene of interest must be entirely contained within some bounded area into which the camera may not enter – this way the change in a light ray's value at its point of intersection with a surface need not be represented.

The 4D subset of the colour plenoptic function represented by a light field may be expressed as $P_{rgb}(n_\theta, n_\phi, t_0, n_x, n_y, p_{z_0})$. This represents each ray in terms of its direction of propagation, as two angles, and its position on the x, y plane, as two distances. There are other ways to represent a light field – defining position in terms of y and z, rather than x and y, for example. The remainder of this chapter discusses some of the different ways of parameterizing light fields.

2.3.1 Spherical-Cartesian Parameterization

Perhaps the most obvious way to represent the light field is using the combination of Cartesian coordinates and angles discussed above. Figure 2.1 depicts this parameterization. The position of each ray is defined along x and y, and the direction is defined as two angles, ϕ within the x, y plane, and θ with respect to that plane. Only two values are needed to represent position, rather than three, because of the constancy of each ray along its direction of propagation. This amounts to defining the position of each ray in terms of its point of intersection with a reference plane, given by $z = p_{z_0}$. Note that the z direction is horizontal by convention, corresponding to depth in the scene, while y is vertical.

A sampled light field parameterized using the spherical-Cartesian parameterization is denoted as $L_{rgb}(n_\theta, n_\phi, n_x, n_y)$. The value of this function is assumed to be

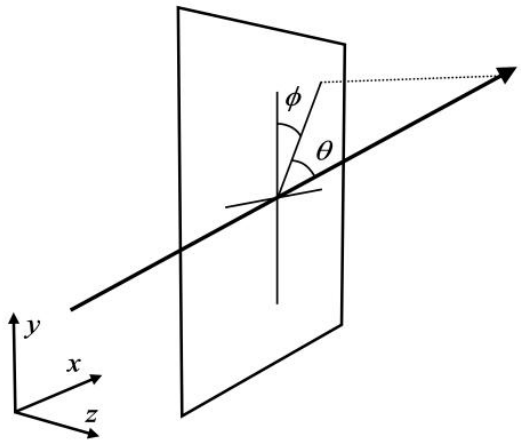


Figure 2.1: Spherical-Cartesian parameterization of light rays.

a colour triplet, and so the 'rgb' subscript is usually omitted. This representation has the advantage of representing light rays traveling in all directions with equal resolution. That is, the resolution as a function of the direction of propagation of each ray is constant. This parameterization has some distinct disadvantages, however. Because it deals explicitly with the angles associated with each ray, performing even the simplest operations with this representation requires the use of trigonometric functions – for other parameterizations, such as the two-plane parameterization discussed in the following section, operations are typically much simpler.

2.3.2 Two-Plane Parameterization

In the spherical-Cartesian parameterization, the direction of each light ray is represented in terms of two angles. This is essentially a matter of finding the point of intersection of each ray with a reference sphere, centered at the point corresponding to n_x, n_y. Rather than using a reference sphere for this purpose, we may record

the point of intersection of each ray with a second reference plane. This will require the use of two sets of local coordinates to avoid confusing the points of intersection – the first plane will use the s and t coordinates, while the second will use the u and v coordinates, as depicted in Figure 2.2. The two reference planes are ideally of infinite extent, though in practical applications they are sampled over a finite region of support. The reference planes are separated by a distance d, and light rays are assumed to travel from the u, v reference plane towards the s, t reference plane.

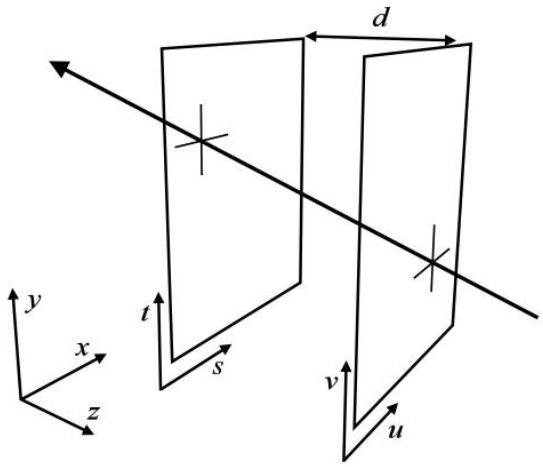


Figure 2.2: Two-plane parameterization of light rays.

The sampled function $L_{rgb}(n_s, n_t, n_u, n_v)$ resulting from this two-plane parameterization (2PP) is the 4D light field parameterization as it was first introduced in [1]. It is assumed to have a value which is a colour triplet, and so again the 'rgb' subscript is usually omitted.

Most operations turn out to be simpler with this parameterization than for the spherical-Cartesian parameterization. The 2PP does have some drawbacks, however. Most notably, it represents a smaller subset of the plenoptic function than

the spherical-Cartesian parameterization, because it only includes those light rays propagating from the u, v plane towards the s, t plane. Furthermore, the resolution with which light rays are represented in the 2PP varies with their direction of propagation. This is most clear in the case of light rays traveling nearly parallel with the reference planes, for which the resolution approaches zero. The spherical-Cartesian parameterization, on the other hand, represents light rays traveling in all directions with equal resolution.

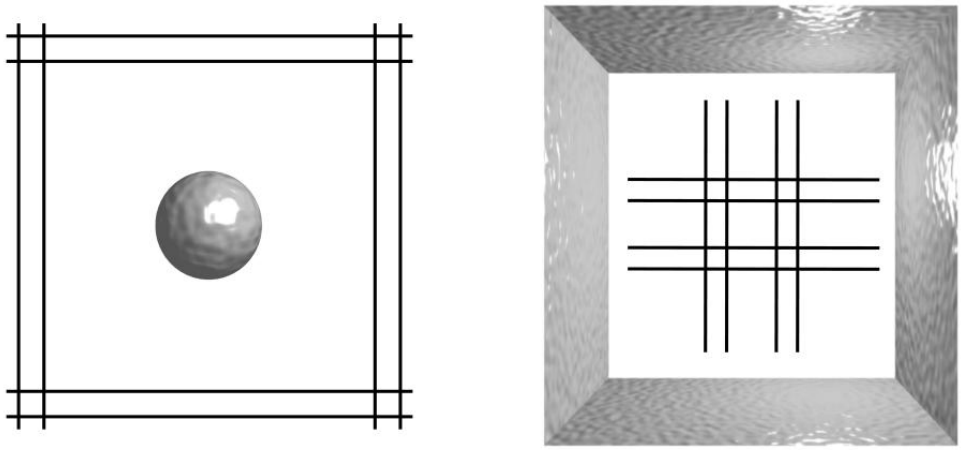
Regardless of its shortcomings, the 2PP yields some unique and interesting characteristics, and as a result it is the parameterization which will be used throughout this thesis.

2.3.3 Multiple Reference Systems

As mentioned earlier, the 2PP yields a resolution which changes as a function of the direction of propagation of the light rays. More specifically, the maximum resolution corresponds to rays perpendicular to the two reference planes, and the resolution drops off to zero as the rays approach an orientation parallel with the reference planes. To deal with this, the original light field proposal included a scheme in which multiple sets of reference planes are used [1]. Each set of reference planes is called a light slab, and a collection of multiple light slabs can be used to create a nearly uniform sampling of the plenoptic function.

The way in which the light slabs are oriented depends on the type of scene to be modelled. Figure 2.3 depicts the two common cases: in a) the scene consists of an object, or collection of objects, that the camera moves around while facing inward; in b) the scene consists of a room inside which the camera moves while facing outward.

For each case, a collection of light slabs is shown which will represent the scene on all sides. This effectively solves the problem of varying resolution associated with the 2PP, and allows the representation of more of the light rays permeating a scene.



(a) (b)

Figure 2.3: Multiple light slabs in the case of a) an object around which the camera moves and b) a room inside which the camera moves.

2.3.4 Freeform Light Fields

The concept behind the freeform, or unstructured, light field is to forego the use of a regular parameterization in favour of a more ad-hoc structure [8]. In this scheme, images taken at arbitrary locations in a scene are stored without further processing, along with the parameters of the camera, such as its position and orientation, corresponding to each image. Because each image represents a 2D slice of the colour plenoptic function, the net result is of describing a subset of that function without the use of a formalized structure. This kind of approach can yield good rendering results with a fraction of the memory requirements, because it avoids some of

the redundancy associated with other parameterizations. It has the disadvantage, however, of requiring more complex algorithms for rendering and analysis.

2.4 The Light Field

For the remainder of this thesis, a *light field* [1] will refer to a two-plane parameterized, single-light slab representation of a 4D subset of the colour plenoptic function. The 2PP is chosen because it yields simple math, allowing for faster, simpler algorithms. Only a single light slab is considered for the sake of simplicity, though some of the techniques developed may be extended do deal with multiple-light-slab light fields.

Both the continuous form, $L(s,t,u,v)$, and the sampled form, $L(n_s,n_t,n_u,n_v)$ of the light field will be used. For the sake of brevity, the vectors $\mathbf{r} = [r_s,r_t,r_u,r_v] = [s,t,u,v]$ and $\mathbf{n} = [n_s,n_t,n_u,n_v]$ will be used, as in $L(\mathbf{r})$ and $L(\mathbf{n})$. In cases where it is not clear whether the continuous or sampled form of the light field is being used, a subscript will be used, as in $L_{cont}(\mathbf{r})$, to indicate the continuous version. The 4D Fourier transform of the continuous-domain light field will be denoted as $L_{freq}(\mathbf{\Omega})$, $\mathbf{\Omega} \in \mathbb{R}^4$.

A Note on Colour

The light fields explored in this work are all colour light fields, with values which are $[r,g,b]$ triplets. The 'rgb' subscript on the light field function will be omitted for the sake of brevity. In general, operations performed on the light field will be applied to each colour channel independently. The Fourier transform, for example, is applied to

[1] The term 'light field' was actually first introduced in the world of physics by Arun Gershun's 1936 paper on the vector irradiance field [9]. The scalar irradiance field, which more closely resembles the concept of a light field used in this work, was later introduced as the photic field by Parry Moon in 1981 [10]. These concepts were first applied to visual processing in 1996 [1] [7].

each colour channel of the light field in turn, yielding a colour 4D Fourier transform, with a value which is also a triplet. Modification of the techniques developed here to deal with grayscale light fields is a simple matter of representing a single colour channel in the light field, rather than three. In general, sampled light fields will be quantized to 8 bits per colour channel – that is, 24 bits per colour triplet.

A Note on 4D Geometry

A hyperplane in 4D behaves similarly to a plane in 3D: it is entirely described by a single equation [11] given by

$$d_s s + d_t t + d_u u + d_v v = k, \tag{2.1}$$

and its orientation is described by a single 4D normal vector, $\hat{\mathbf{d}}$. When a point is constrained to lie within a plane in 3D, the point has two degrees of freedom. If constrained to a hyperplane in 4D, the point has three degrees of freedom.

Similarly, a plane in 4D behaves much like a line in 3D. It is described by two equations, not one. In the case of a line in 3D space, each of the two equations describes a plane, and the intersection of the two planes describes the line. In the case of a plane in 4D space, each of the two equations describes a hyperplane, and the intersection of the two hyperplanes describes a plane [11], as in

$$d_s^1 s + d_t^1 t + d_u^1 u + d_v^1 v = k_1, \tag{2.2}$$

$$d_s^2 s + d_t^2 t + d_u^2 u + d_v^2 v = k_2. \tag{2.3}$$

These two equations can be expressed as a single system, as in

$$\begin{bmatrix} d_s^1 & d_t^1 & d_u^1 & d_v^1 \\ d_s^2 & d_t^2 & d_u^2 & d_v^2 \end{bmatrix} \begin{bmatrix} s \\ t \\ u \\ v \end{bmatrix} = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} \tag{2.4}$$

A point constrained to exist on a line in 3D has only one degree of freedom. A point constrained to exist on a plane in 4D has two degrees of freedom.

It does not make sense to refer to the normal of a line in 3D, nor does it make sense to discuss the normal of a plane in 4D. One might, however, refer to the orientation of a line in 3D, or a plane in 4D. One might also refer to vectors or planes which are orthogonal to a line in 3D, and in the same way refer to vectors, planes, or even hyperplanes that are orthogonal to a plane in 4D.

Chapter 3

Working with Light Fields

In Chapter 2 the concept of the light field was introduced, in the context of the plenoptic function, as a way of generating novel rendered views of a 3D scene using a 4D data structure. This chapter deals with the practical issues associated with working with light fields, including measuring light fields from virtual and real scenes, storing light fields, and rendering novel views of a scene from a light field model.

3.1 Measuring a Light Field

3.1.1 Virtual Scenes: the Raytracer

A light field may represent a virtual scene as readily as a real one, and because doing so will require no special hardware, it seems simplest to begin an examination of light field measurement with a purely virtual setup. Generating a light field model of a virtual scene will also highlight the differences between light field models and geometric models.

The process of measuring a light field is essentially one of building a database of light rays emanating from a scene – the set of rays represented by the database is determined by the parameters of the two reference planes. The measurement process may be carried out by tracing the path of each ray in the database back into the scene, and finding the value of the scene where the ray intersects it using the geometric model. This process bears a striking resemblance to that carried out by

a raytracer [12], which uses a geometric model of a scene in exactly the same way as described above, but for the set of rays corresponding to a photograph taken by a single camera. It is a relatively simple task to modify a raytracer so that, rather than finding the value of each ray entering a virtual camera, it finds the value of each ray in the light field database.

The first step in measuring the light field is to determine the parameters of the two reference planes. The resolution with which each is sampled determines the amount of scene detail that will be modelled. Figure 3.1 shows a typical configuration for the reference planes – it is typical for the virtual camera to be placed near the s, t plane during the rendering process, and for the u, v plane to coincide with the area of interest in the scene. Because of this placement, the s and t dimensions mostly represent the motion of the camera – one might imagine each of the samples along s and t as corresponding to a specific camera position for which the camera is on the reference plane. Similarly, one might imagine each of the samples in the u, v plane as being a single pixel in a still image taken from a specific location on the s, t plane. Because the subjective image quality depends more strongly on the resolution of the still images than it does on the resolution with respect to camera motion, a typical light field will have fewer samples along s and t than along u and v – as few as one eighth the samples. For moderate scene complexity, a resolution corresponding to 256 samples along u and v, and 32 samples along s and t is typical.

The size and separation of the reference planes effectively determine the subset of light rays that will be represented in the light field, and thus determine the range of motion that the virtual camera is allowed when rendering. Larger reference planes will cover a larger area, and thus allow more camera motion, though at the cost of

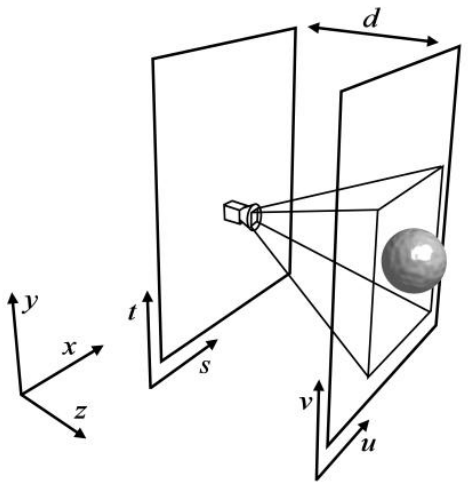


Figure 3.1: The reference planes in relation to the virtual camera and the scene.

decreased spatial resolution for a fixed number of samples. The plane separation, on the other hand, will affect a quantity related to the field of view of the light field – that is, the range of ray directions that are represented. A smaller plane separation represents a broader range of ray directions, though at the cost of a lower angular resolution. Finally, the two planes need not be of the same size, and the ratio of sizes will affect both the spatial and angular ranges of the light field. A larger s, t plane allows for more camera motion near that plane, while a larger u, v plane allows more angular freedom. In the extreme case of a single sample in the s, t plane, the camera has lost all freedom to translate, but it is free to rotate.

The reference plane parameters discussed above are summarized in Table 3.1. The resolution in each dimension, in samples per meter, can be found by dividing the number of samples by the total length in each direction, as in N_s/D_s samples/m along s. For the sake of simplicity, the position and orientation of the reference planes are chosen such that the s, t plane is given by $z = 0$, and the u, v plane is given by

$z = d$. Furthermore, the center of both reference planes is placed at $x = y = 0$.

Table 3.1: Reference plane parameters.

Symbol	**Meaning**
$\mathbf{D} = [D_s, D_t, D_u, D_v]$	Physical dimensions of the reference planes, in meters.
$\mathbf{N} = [N_s, N_t, N_u, N_v]$	Total number of samples along each dimension.
d	Plane separation

With the reference planes defined, we may find the characteristics of the light rays passing through them. For each index $\mathbf{n}$ in the database, there is a corresponding ray $\mathbf{r}$, given by its points of intersection with the two reference planes as

$$\begin{bmatrix} r_s \\ r_t \\ r_u \\ r_v \end{bmatrix} = \begin{bmatrix} D_s \left(\frac{n_s}{N_s - 1} - 0.5\right) \\ D_t \left(\frac{n_t}{N_t - 1} - 0.5\right) \\ D_u \left(\frac{n_u}{N_u - 1} - 0.5\right) \\ D_v \left(\frac{n_v}{N_v - 1} - 0.5\right) \end{bmatrix}. \tag{3.1}$$

In order to trace each ray back into the scene, it must be expressed in terms accepted by a standard raytracing algorithm. Most raytracing algorithms work with rays defined in terms of a point through which the ray passes, and a normalized direction vector, given by $\mathbf{p} = [p_x, p_y, p_z]$, and $\hat{\mathbf{v}} = [v_x, v_y, v_z]$ respectively. We may define $\mathbf{p}$ as the point of intersection of the ray with the s, t plane, as in

$$\mathbf{p} = [r_s, r_t, 0], \tag{3.2}$$

and define the direction vector as

$$\hat{\mathbf{v}} = \frac{[r_u - r_s, r_v - r_t, d]}{\|r_u - r_s, r_v - r_t, d\|}. \tag{3.3}$$

With the light ray indexed by $\mathbf{n}$ described in terms of $\mathbf{p}$ and $\hat{\mathbf{v}}$, a standard raytracing algorithm may be used to determine its colour. By iterating through the entire light field, the value of $L(\mathbf{n})$ is determined for every index $\mathbf{n}$, completing the light field measurement process.

The raytracing algorithm works on a single ray at a time, returning its value as a colour triplet. The first step in raytracing is finding the point of intersection of the light ray with each primitive in the scene, if it exists. The colour of the surface at the closest point of intersection is determined using lighting and surface models. For a moderately complex raytracer, these models may incorporate surface properties, positions of lights in the scene, shadows cast by other primitives, reflections of other primitives and light sources, and perhaps the refraction of light rays through transparent surfaces [12]. Some raytracers will also find the values of a neighbourhood of rays surrounding each ray of interest, taking the mean value of the rays in the neighbourhood in order to perform antialiasing.

Alternative approach

Rather than implementing a raytracer for the purposes of measuring the values of the rays in the light field, it is possible to employ an existing raytracer. By placing virtual cameras at locations corresponding to the sample points on the s, t plane, oriented so as to face the u, v plane, the light field can be measured. This approach is equivalent to that taken for real-world scenes, described in more detail in Section 3.1.2.

Implementation Details

A specialized raytracer was implemented in C++, and is incorporated into a software package called *Lightbench*, which incorporates elements from the rest of this thesis. The main loop of the virtual light field measurement system cycles through all the sample indices $\mathbf{n}$ of the light field, calculating the corresponding values of $\mathbf{p}$ and $\hat{\mathbf{v}}$, and calling the raytracing procedure with these as parameters. The output of that procedure is a colour triplet which is recorded in the appropriate location in a 4D light field database. An optional antialiasing function causes the raytracer to find the values of four light rays surrounding the input light ray, taking the mean value of these four rays in order to reduce the effects of aliasing.

The raytracing procedure uses a geometric scene model which describes the scene in terms of light sources and geometric primitives. The scene descriptions are stored as text files, and primitives are described in terms of their position, size, orientation and surface characteristics. Spheres and polygons are the only implemented primitives, and the omni-directional point light source is the only implemented light source. Surface characteristics may include any or all of the following [13] [14]: diffuse colour, specular colour, and specular power. Diffuse colour is the colour of the light that scatters from a surface, as in the light reflecting from a perfectly matte surface. Specular colour is the colour of the light that reflects coherently off a surface, as in the light reflecting off a mirror or polished metal. Specular power affects the behaviour of the specular reflection – a high specular power corresponds to a glossy surface with a sharp specular highlight. The specular and diffuse components of the surface colour can be described either as a single colour or as a bitmap, for which the contents of a 2D bitmap file are mapped onto the surface of the primitive.

The speed with which the raytracer measures a light field depends on the complexity of the scene. As the number of primitives and light sources increases, the raytracing process becomes slower. For a light field with 256 samples in u and v, and 32 samples in s and t, of a scene containing about 20 primitives, the rendering process takes about 10 minutes on a Pentium IV running at 1 GHz.

3.1.2 Real Scenes: the Camera Gantry

In the preceding section, a method for measuring a light field of a virtual scene was described. This makes sense as a first step in measuring light fields, as it requires no specialized hardware. However, the true strength of light fields lies in their ability to represent real-world scenes, and so it is desirable to build an apparatus capable of measuring light fields of such scenes. Because the light field represents rays with varying positions as well as angles, it is necessary to measure images at multiple positions relative to the scene. There are several ways of accomplishing this, such as the use of multiple cameras, discussed in Section 3.1.3, a single camera with a robotic gantry, which is the solution pursued for this thesis, or various combinations of moving scene elements and a moving camera, as discussed in [1].

By moving a single camera relative to a stationary scene, one may measure all the light rays necessary for the construction of a light field. There are several ways of doing this, but the simplest is to move the camera to the positions corresponding to sample points in the s, t plane, while facing the u, v plane. Figure 3.1 depicts a camera at one of the s, t sample points. The image measured at each camera position represents the set of rays passing through the u, v plane for that position in the s, t plane. The gantry moves the camera to each sample point on the s, t plane, recording

an image at each location. By aligning the camera such that the image's horizontal and vertical axes are parallel with the light field's u and v axes, respectively, the process of incorporating the measured images into a light field is simplified.

The camera gantry described above has only two degrees of freedom, corresponding to the s and t dimensions, and is therefore the simplest possible gantry which can be used to measure a complete light field. This method is not without its drawbacks, however. Of particular concern is the fact that a camera with a limited FOV may not be able to measure the entire u, v plane at every location on the s, t plane – particularly near the extremities of the s, t plane, and for parameterizations with a small plane separation. One way around this limitation is to utilize a camera with pan and tilt capabilities, essentially adding two degrees of freedom to the gantry, and allowing the camera to cover a wider field of view. Other alternatives include utilizing a wide-angled lens, or an alternate form of gantry.

Another concern in the construction of the gantry is the condition that the lighting in the scene remain static. If light is allowed to pass through the camera gantry and into the scene, the shadow cast by the gantry will change as the camera moves, violating the condition that the lighting be static. For a planar camera gantry, it is a simple matter to prevent light from flowing through the gantry – a simple wall or board blocking the flow of light will suffice. For other forms of gantry, the solution is not always so simple, and in some cases it is simplest to fix the position of the lighting relative to the scene, and manipulate the entire scene rather than the camera.

The gantry is instructed to collect images at locations corresponding to the sample locations on the s, t plane. As a result, no resampling is required in these dimensions. However, each recorded image consists of pixels which do not necessarily correspond

to the samples on the u, v plane. The gantry images must therefore be resampled such that the ray corresponding to each value of n_u and n_v may be found. This process begins with an appropriately selected lowpass filter, designed to eliminate aliasing, and proceeds by finding which pixels in the gantry image correspond to each value of n_u, and n_v. In both steps, the parameters of the camera which recorded the gantry images are required – that is, the camera must be calibrated.

The calibration of a camera is a mapping between the location of a pixel in a camera image and the direction of the ray corresponding to that pixel, where the direction of the ray is expressed relative to the optical axis of the camera. The process of generating calibration data can be quite involved, though for a high-quality camera this mapping may be approximated by the ideal projective mapping corresponding to a pinhole camera. Assuming this mapping, only the FOV of the camera is required to perform the calibration. Note that this FOV can differ for the horizontal and vertical axes, and so the vector $\mathbf{F} = [F_x, F_y]$ is adopted to represent the field of view of the camera in each direction, where both components are measured in radians.

Figure 3.2 shows a top view of the camera at some sample point on the s, t plane. The image plane is shown between the two reference planes. It contains I_x by I_y pixels, and the pixels are indexed by the vector $\mathbf{a} = [a_x, a_y],\ \mathbf{a} \in \mathbb{N}^2$, where $0 \leq a_x < I_x$ and $0 \leq a_y < I_y$. The vector $\mathbf{M}$ represents the extent, in each direction, of the u, v reference plane that is contained within the FOV of the camera, and can be expressed as

$$\begin{bmatrix} M_x \\ M_y \end{bmatrix} = 2d \begin{bmatrix} \tan(F_x/2) \\ \tan(F_y/2) \end{bmatrix}. \tag{3.4}$$

Recognizing the similar triangles in Figure 3.2, we may find the index of the

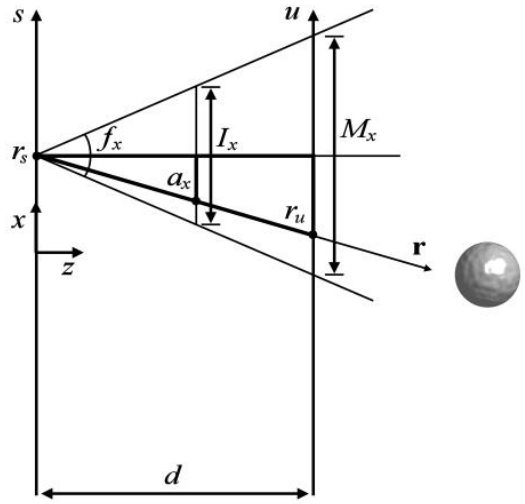


Figure 3.2: Finding a given ray in the recorded images.

image pixel, **a**, corresponding to each vector **r** as

$$\begin{bmatrix} a_x/(I_x-1) \\ a_y/(I_y-1) \end{bmatrix} - 0.5 = \begin{bmatrix} (r_u-r_s)/M_x \\ (r_v-r_t)/M_y \end{bmatrix}, \tag{3.5}$$

which is easily solved for **a**.

The process of generating the light field from the measured images, then, is to first apply an anti-aliasing filter to each image, then iterate through all the values of **n**, indexing the appropriate image using n_s and n_t, and finding the pixel corresponding to each value of n_u and n_v using Equations (3.1) and (3.5). The value of the corresponding pixel is assigned to the light field sample $L(\mathbf{n})$.

Implementation Details

A planar camera gantry, as described above, was designed and constructed by the author. Motion in the s,t plane is accomplished using a belt-driven XY positioning table, controlled by two stepper motors. The gantry has a range of about 47 cm in s and t, and is oriented in the horizontal plane such that the camera faces up. In

this configuration, the motors never have to work against gravity, and there is no possibility of light passing through the gantry, as it lies flat on the workbench. The disadvantage of a horizontal gantry is that the scene needs to be suspended above it – this may be difficult or impossible to arrange for some types of scene. Figure 3.3 shows the gantry in action.

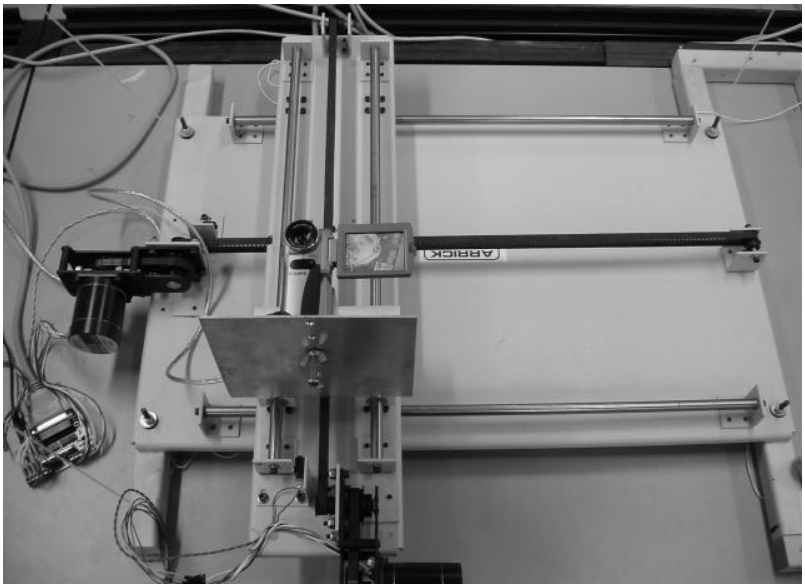


Figure 3.3: Planar camera gantry in action.

The motors used to control the gantry are unipolar stepper motors with 400 steps per revolution. Pulley boxes are in place between the motors and gantry, providing a mechanical advantage and higher precision. Alternatively, these boxes can be reversed in order to increase the speed of the measurement system, at the cost of decreased precision. A set of motor drivers control the motors via four control lines per motor. Each motor is actuated by activating each of its four control lines in sequence. Each step in the sequence corresponds to a single step of the motor, and reversing the sequence reverses the direction of the motor. The motor drivers are

controlled via a parallel interface with the controlling PC.

The camera used by the gantry may be any camera that is compatible with Windows®-based video capture, including cameras that utilize USB® and Firewire® interfaces. The camera used for measuring the light fields in this thesis is a Canon® Elura 10, interfaced via the Firewire® port. This camera has a progressive scan video mode, which makes it particularly suitable for high-quality single-frame imaging. Note that using a DV® camera has some drawbacks: the DV® video format encodes colour in terms of hue, saturation, and luminance, and uses only half the resolution in the hue and saturation channels. This means that while the resolution is 720×480 in the luminance channel, it is only half this in the other two channels. Fortunately, because the resolution of the camera far exceeds the resolution of a typical light field, the anti-aliasing filter removes most of the artifacts that would normally result from this shortcoming.

A camera mounting bracket was fabricated to mount the camera to the gantry. Because the mounting bracket vibrates slightly while the gantry is in motion, the gantry is completely stopped for a second before each image is measured. This has the further advantage that if the gantry is ever operated with a camera with no progressive scan mode, the frame will be totally still during measurement, and so no interlacing artifacts will occur. The process of measuring a light field which has an s, t plane which occupies most of the gantry's range, and which has 32 samples along each of s and t, takes about half an hour.

Windows® is a registered trademark of Microsoft Corporation.
USB® is a registered trademark of USB-IF, Inc.
Firewire® is a registered trademark of Apple Computer, Inc.
Canon® is a registered trademark of Canon Inc.
DV® is a registered trademark of Sony Corporation

The software which controls the gantry, called *Lightscribe*, was written in C++, and is dedicated to the task – i.e. it is not incorporated into the Lightbench package. It actuates the camera gantry via the parallel port, and captures images from the camera via the Firewire® port, saving the images on the hard drive as bitmaps. The software which generates a light field from the resulting images is incorporated into the Lightbench package. It simply iterates through all the indices **n**, finding the values of the corresponding light field samples as described above. The anti-aliasing filter is implemented as a moving average filter, with a window size **W** determined as the ratio of the spatial resolution of the gantry images to the spatial resolution of the light field, as in

$$\begin{bmatrix} W_x \\ W_y \end{bmatrix} = \begin{bmatrix} \frac{I_x}{2d\tan(F_x/2)} / \frac{N_u}{D_u} \\ \frac{I_y}{2d\tan(F_y/2)} / \frac{N_v}{D_v} \end{bmatrix}. \tag{3.6}$$

The process of building a typical light field from gantry images takes about 5 minutes on a Pentium IV running at 1 GHz.

3.1.3 In Real-Time: Multiple-Camera Arrays

By appropriately arranging multiple cameras into an array, an entire light field may be measured at once. This approach is taken in [15], in which a planar array of 128 cameras is proposed. At a cost per camera of around $50 at the time of its construction, this kind of approach is surprisingly affordable, due mostly to the decreasing cost of CMOS imaging technology. A photograph of the 8 by 16 camera array is shown in Figure 3.4.

Because this kind of array can image an entire light field at once, it can be used to measure a still light field of a dynamic scene. The ability to instantaneously measure a light field opens up the possibility of real-time analysis of dynamic scenes, in which

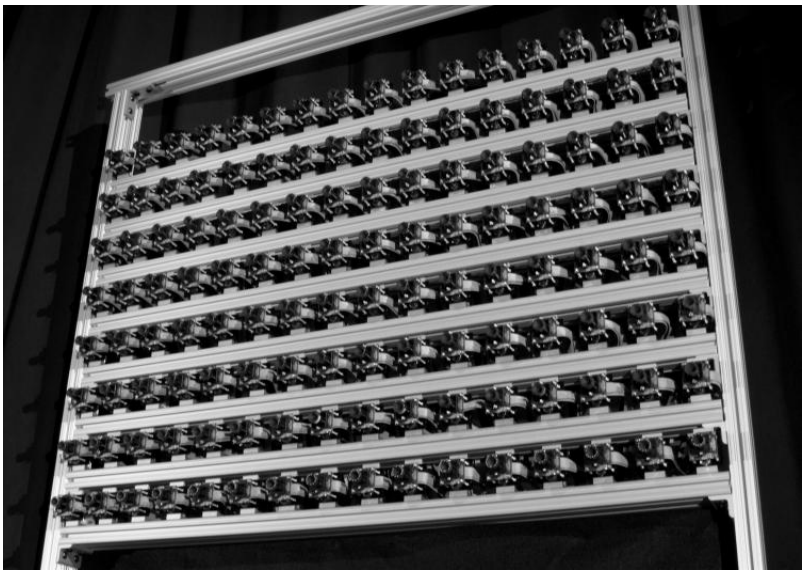


Figure 3.4: An array of CMOS cameras; *image courtesy the Stanford University Computer Graphics Laboratory.*

several light fields are measured and analyzed over time. In addition to applications in scene analysis, the resulting 'light field video' might be useful in virtual presence applications such as telemedicine.

3.1.4 From Video

Camera gantries and multiple-camera arrays both require specialized hardware for light field measurement. A technique for measuring light fields using only a consumer-grade hand-held video camera would open up a wide range of applications. The basic idea behind such a system is to move a single video camera through a scene, usually sweeping it back and forth in a zigzag pattern, and apply some form of camera tracking in order to construct a light field from the images in the video sequence. This is the approach taken in [16], [17], [18], [19], and [20]. In these contributions, feature tracking is utilized to determine the camera's position for any given frame.

Feature tracking utilizes the apparent motion of features in the scene to determine the trajectory, and sometimes the internal parameters such as the FOV, of the camera. This information is then used to find the values of the light field samples from the video sequence. Feature tracking works well for scenes with easily identifiable features that are always visible and are therefore easily tracked. This is not always the case, however, and so the feature tracking algorithm needs to take into account issues such as occlusion, and the possibility that features may not be within the camera's FOV for the entire sequence. Furthermore, feature tracking can be extremely difficult for some kinds of scene, such as scenes containing periodic or scattered but similar structures which may easily be confused. In order to deal with these issues, some methods for enhancing basic feature tracking have been proposed, such as the use of dense stereo matching as in [16] and [19].

3.2 Storing a Light Field

Once a light field has been measured, a method must be devised for storing it. The most obvious method of doing this is to generate a single binary file containing all the data in the 4D light field array. Alternatively, it may be desirable to store the light field as a 2D array, in s and t, of 2D images, in u and v – this is conceptually similar to storing the gantry images used to generate the light field, except that here the image pixels exactly correspond to the light field samples. As an example, consider a light field with 32 samples in s and t, and 256 samples in u and v. This light field would be stored as $32 \times 32 = 1024$ images, each containing 256×256 pixels. This technique ultimately occupies more space on disk, because of the overhead in

the operating system associated with storing such a large number of files, but it has the distinct advantage of allowing the use of commonly available software – indeed, almost any image-processing software, including freely available paint programs – for viewing and editing the resulting files.

Note that a typical light field, with 256 samples in u and v, 32 samples in s and t, and three colour channels, occupies 192 MB of memory. This is prohibitively large for some applications, and so a method for compressing light fields is desirable. This has been the subject of extensive research, most of which is concerned with utilizing disparity-based encoding [21], [22] to minimize the entropy of the light field by exploiting the similarity of the images that it contains. More complex methods seek to utilize the geometry of the scene modeled by light field as a starting point, using the light field only to refine the geometric model, yielding a lower overall entropy [23], [24], [25].

Implementation Details

For the sake of simplicity, no compression techniques have been incorporated into the Lightbench package. Light fields are stored as a sequence of images in s and t, as described above. Each image is stored as a bitmap file – this format has no compression, and so the light field occupies its full size on the hard drive. The bitmap format was chosen for compatibility – virtually every image processing tool can deal with a bitmap file. This format also avoids the artifacts associated with lossy compression schemes, and avoids the decrease in colour depth associated with some formats.

3.3 Rendering from a Light Field

Rendering an arbitrary view of a scene from a light field model is extremely simple – the speed of this process is why image-based modelling first came about. As in a classical raytracing approach, a model of the virtual camera is used to determine the set of rays corresponding to the image pixels. Rather than tracing each ray back into a geometric model of the scene, as in raytracing, each ray is simply parameterized in terms of its points of intersection with the two reference planes. This parameterization is used to interpolate the ray's value from the light field database.

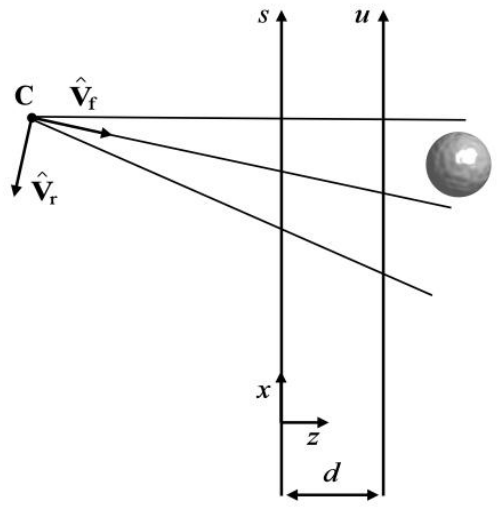


Figure 3.5: Rendering from a light field.

The virtual camera is most simply modelled as a pinhole camera. A position vector $\mathbf{C}$ defines the position of the camera, and its orientation is defined in terms of two normalized direction vectors, $\hat{\mathbf{V}}_f$ and $\hat{\mathbf{V}}_r$, which point forward along the optical axis, and towards the right orthogonal to the optical axis, as shown in Figure 3.5. A third direction vector pointing up, $\hat{\mathbf{V}}_u$, may be found as the cross product $\hat{\mathbf{V}}_u =$

$\hat{\mathbf{V}}_r \times \hat{\mathbf{V}}_f$. The field of view of the camera is defined by the vector $\mathbf{F} = [F_x, F_y]$, and the rendered image contains I_x by I_y pixels.

For each pixel index $\mathbf{a} = [a_x, a_y]$, $0 \leq a_x < I_x$, $0 \leq a_y < I_y$, we define the ray to which it corresponds in terms of a point that it passes through, $\mathbf{p}$, and a normalized direction vector $\hat{\mathbf{v}}$. Because each ray must pass through the center of projection of the camera, we may choose $\mathbf{p} = \mathbf{C}$. Exploiting the geometry of the situation, or referring to a good reference on raytracing [12] [13] [26], we may define the un-normalized direction vector of the ray, v', as

$$\begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix} = \begin{bmatrix} V_{f_x} & V_{r_x} & V_{u_x} \\ V_{f_y} & V_{r_y} & V_{u_y} \\ V_{f_z} & V_{r_z} & V_{u_z} \end{bmatrix} \begin{bmatrix} 1 \\ \left(\frac{2a_x}{I_x - 1} - 1\right) \tan\left(\frac{F_x}{2}\right) \\ \left(\frac{2a_y}{I_y - 1} - 1\right) \tan\left(\frac{F_y}{2}\right) \end{bmatrix}, \tag{3.7}$$

which may be normalized as $\hat{\mathbf{v}} = [v'_x, v'_y, v'_z]/\|v'\|$.

Examining the geometry of the ray passing through $\mathbf{C}$ with direction $\hat{\mathbf{v}}$, we may find the point of intersection of this ray with the reference planes as

$$\begin{bmatrix} r_s \\ r_t \\ r_u \\ r_v \end{bmatrix} = \begin{bmatrix} C_x - \left(C_z \cdot \frac{v_x}{v_z}\right) \\ C_y - \left(C_z \cdot \frac{v_y}{v_z}\right) \\ C_x + \left((d - C_z) \cdot \frac{v_x}{v_z}\right) \\ C_y + \left((d - C_z) \cdot \frac{v_y}{v_z}\right) \end{bmatrix}. \tag{3.8}$$

At this point, all that remains is to determine the value of the ray given by the vector $\mathbf{r}$ by finding its value in the light field database. The simplest way to do this is to determine the index of the closest sample to the ray by rearranging Equation (3.1),

as in

$$\begin{bmatrix} n'_s \\ n'_t \\ n'_u \\ n'_v \end{bmatrix} = \begin{bmatrix} \left(\frac{r_s}{D_s} + 0.5\right)(N_s - 1) \\ \left(\frac{r_t}{D_t} + 0.5\right)(N_t - 1) \\ \left(\frac{r_u}{D_u} + 0.5\right)(N_u - 1) \\ \left(\frac{r_v}{D_v} + 0.5\right)(N_v - 1) \end{bmatrix}. \tag{3.9}$$

Note that this yields a continuous-domain index – that is, the value of each element of the index is real, and not an integer. By rounding each index element to the nearest integer, the nearest sample to the ray, $\mathbf{n}$, may be found.

The nearest-ray approach requires very few calculations, but the resulting rendered images suffer from severe distortion, most notably the form of discontinuities where transitions are made between light field samples. This is a manifestation of the aliasing associated with improperly resampling the light field data. A more ideal result is obtained by interpolating from the light field database using quadrilinear interpolation [1] [7], a simple extension of the bilinear interpolation commonly used in 2D image processing (Appendix A). For a typical light field, and for a typical rendering system, the rendered image will be at a higher resolution than the light field model, and so no pre-filtering is required prior to interpolation. For the case when the light field is at a higher resolution than the rendered image, an antialiasing filter must be applied prior to interpolation. The notation $\tilde{L}_{cont}(\mathbf{r})$ will be used to denote the estimate of the value of the ray $\mathbf{r}$, obtained through interpolation of the sampled light field.

Implementation Details

A light field rendering system capable of real-time operation was implemented, and incorporated into the Lightbench package. For each rendered image, the software

calculates the direction of the ray corresponding to each image pixel using Equation (3.7), then the point of intersection of that ray with the reference planes using Equation (3.8). The corresponding continuous-domain index is found using Equation (3.9), and is utilized in quadrilinear interpolation to estimate the value of the corresponding ray using the light field database. For rays that are not within the bounds of the reference planes, a value of $[0, 0, 0]$ – i.e. a black ray – is returned. The assumption is made that the rendered image will sample the plenoptic function more densely than the light field database, and so no pre-filtering is required prior to interpolation. An option is in place to force the use of the nearest-sample approach, rather than the quadrilinear interpolation scheme, for the purposes of experimentation.

The rendering system is part of a greater interactive system which allows the virtual camera's position and orientation to be manipulated via the mouse and keyboard. The rendered image responds to the camera movement in real-time in the case of the light field rendered images. This same interactive system can be used to visualize raytraced scenes, though for scenes with significant geometric complexity, raytracing is too time consuming to allow real-time operation.

3.4 Aliasing

While it is true that a light field may generate rendered images of a scene quickly, independent of the complexity of that scene, it is not true that a light field with a fixed number of samples will represent all scenes equally well. In fact, a light field essentially *samples* the continuous-domain plenoptic function, and so the resulting

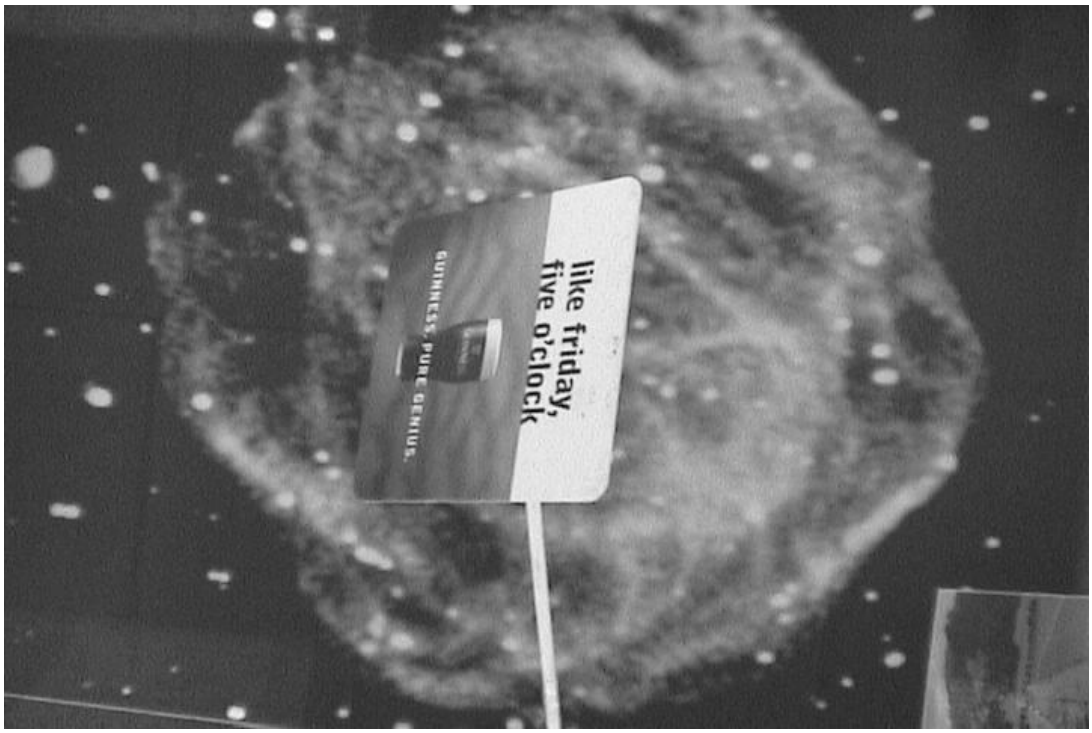
sampled data must be of a sufficient resolution so as not to violate the Nyquist criterion. In practicality, this criterion is satisfied through the use of an antialiasing filter in the light field measurement system, such as the filter applied to each gantry image in Section 3.1.2, and the antialiasing function of the raytracer in Section 3.1.1.

The antialiasing in the raytracer is effective along all four dimensions, because the values of extra light rays are taken along s, t, u and v. In the case of real-world scenes, however, only the minimum number of samples in s and t are taken, and so some aliasing will always remain in those dimensions. This manifests itself as ghosting in the light field – the appearance of multiple instances of the same object. The ghosting is most apparent for objects which are distant from the u, v plane because, as will be seen in the following chapter, these objects have the most energy along s and t, and so are more susceptible to aliasing along these dimensions.

3.5 An Example

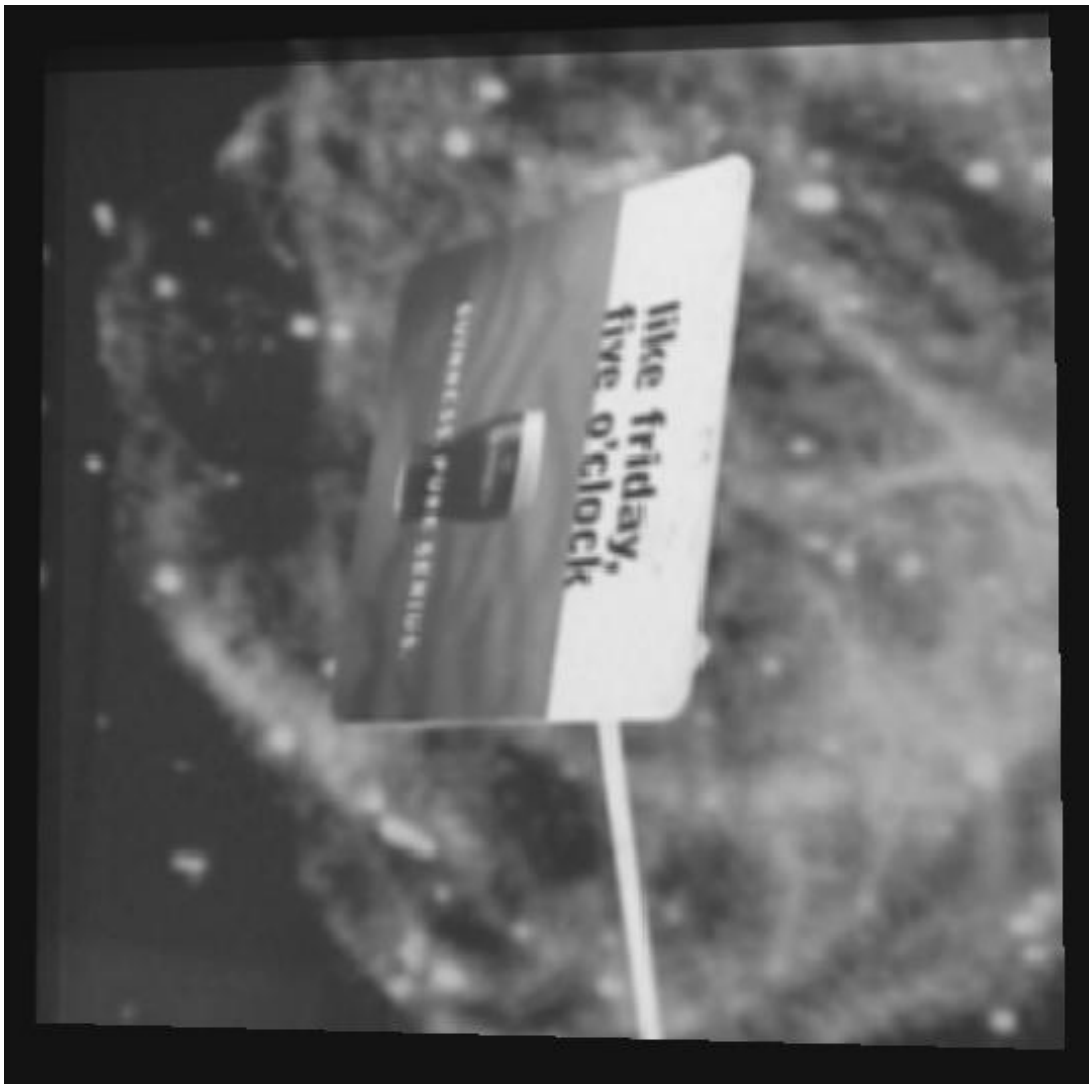
Figure 3.6 shows an example of a real-world scene being modelled using a light field. The light field was measured using the camera gantry described in Section 3.1.2 – one of the gantry images is shown in a). An array of 32 $\times$ 32 such gantry images, each with a resolution of 720 $\times$ 480 pixels, was measured. The images were used to form a light field model, with 256 samples along u and v, and 32 samples along s and t. A rendered view of the scene is shown in b), from a novel camera angle – that is, the camera was never in this position during the measurement process. The rendering system used quadrilinear interpolation, and generated an image with a resolution of 512 $\times$ 512 pixels. The effect of using the nearest ray, rather than

quadrilinear interpolation, is shown in c) – note the appearance of discontinuities – and the effect of undersampling in s and t is shown in d) – note the appearance of ghosting, particularly in the background.

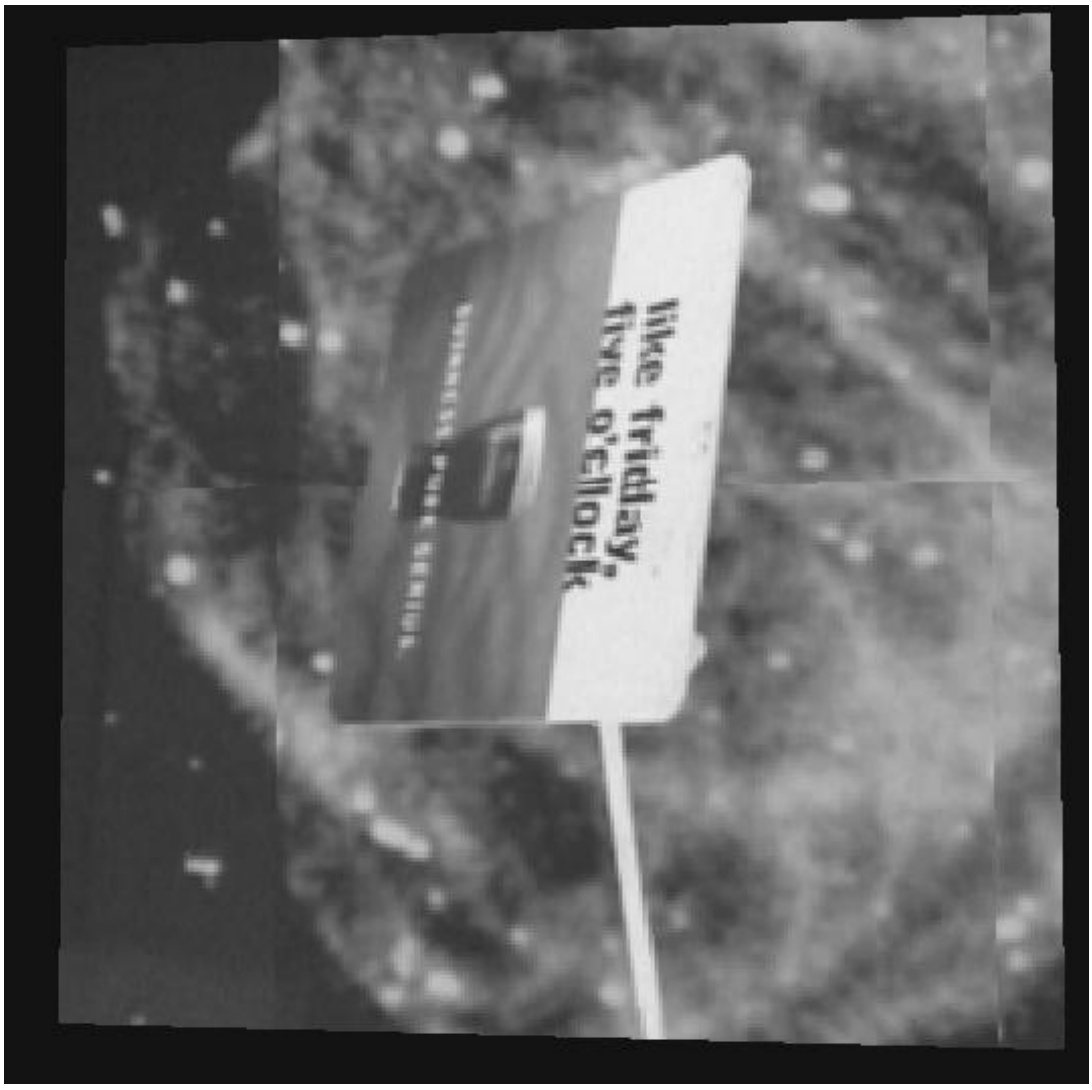


(a)

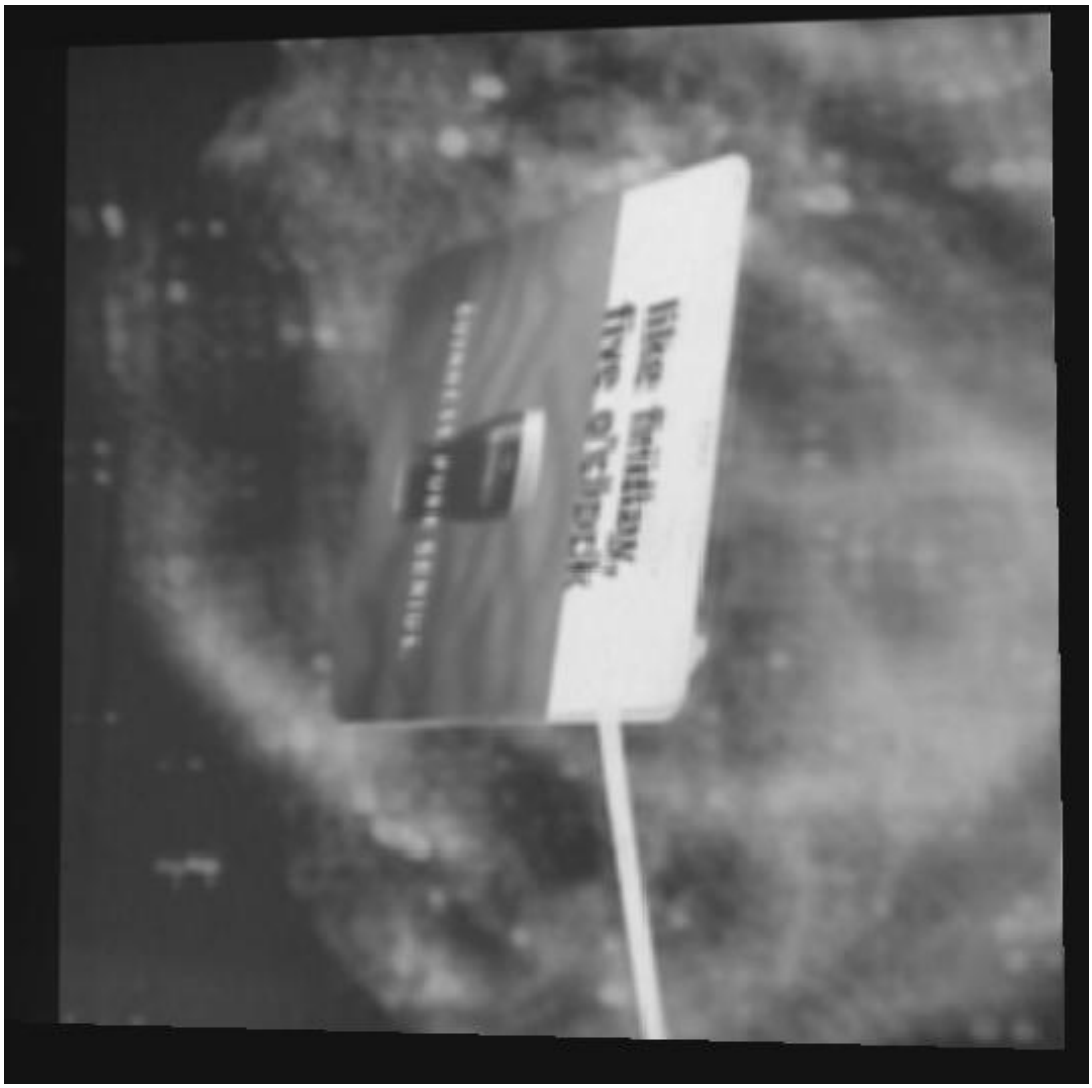
Figure 3.6: An example of light field rendering: a) one of the gantry images, b) an image rendered using quadrilinear interpolation, c) an image rendered using the nearest-ray method, and d) the effects of undersampling in s and t.



(b)



(c)



(d)

Chapter 4

Light Field Properties

This chapter explores some of the properties of light fields, including those that will be exploited in later chapters to perform useful operations.

4.1 Visualizing Light Fields

When working with a 4D data structure it is important to be able to visualize the contents of the structure in a way that that is intuitive. Given that humans have difficulties visualizing data directly in 4D, it is desirable to work with lower-dimensional subsets of the 4D light field. Taking a cue from the world of video processing, in which 3D data is often visualized as a stack of 2D slices, a light field may also be expressed in terms of 2D slices. The difference between video and light fields is that, rather than being stacked in one direction as in video, the slices of the light field must be stacked in two orthogonal directions. The result is a visualization of the light field as a 2D array of slices. There are several ways to orient these slices within the light field, though four stand out as being particularly useful, in that they convey the structures within the light field in the most intuitive fashion.

Arrays of Slices in u and v

Perhaps the most intuitive of all the representations is a 2D array of slices taken in the u and v dimensions, as shown in Figure 4.1. Each slice in the array corresponds to a single point in the s, t plane – in this sense each slice resembles a single image

taken with the planar camera gantry described in the previous chapter. Moving from slice to slice along s and t is similar to changing perspective via camera translation in the horizontal and vertical directions, respectively. Note that each slice covers the same range in u and v – the extent of the u, v reference plane – and in this sense each slice is not strictly identical to a gantry image, which would cover a different range in u and v depending on its position in s and t. This visualization lends itself well to direct analysis of the scene's contents.

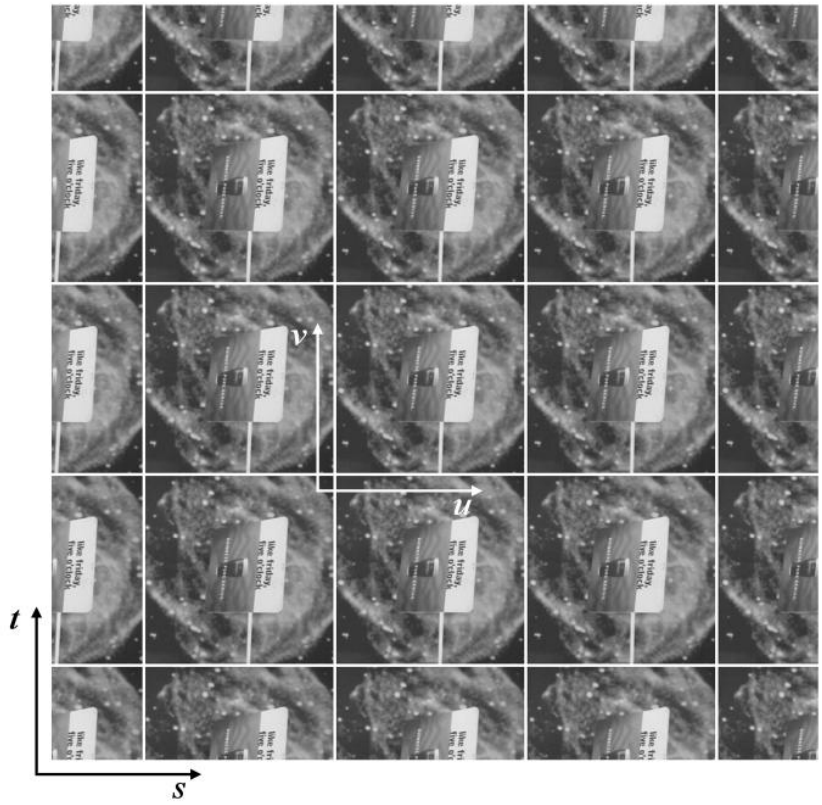


Figure 4.1: Visualizing the light field as an array of slices in u and v.

Arrays of Slices in s and t

The visualization scheme described above yields images in which the scene contents can be easily identified. Taking slices in s and t does not yield such intuitive results. Each slice corresponds to a single point on the u, v plane, and so the resulting image is a map of the rays passing through that point. For a scene containing surfaces which are distant from the two reference planes, the resulting slices resemble narrow-FOV images of the scene. For a scene containing surfaces near the u, v plane, however, the images resemble maps of rays reflected from points on those surfaces. Figure 4.2 shows an example of a light field visualized in this way – note that this scene contains surfaces behind the u, v plane, and so the image resembles a narrow-FOV view of the scene. Note also that a light field containing 256 samples in u and v, and 32 samples in s and t was used to generate these images, resulting in slices in s and t with fewer samples than those taken in u and v in the previous visualization.

Arrays of Slices in s and u

Figure 4.3 shows an example of a light field visualized as an array of slices in s and u – the light field used to generate this figure has fewer samples s and t than in u and v, as reflected by the narrowness of each slice along s. This representation is perhaps the least intuitive in terms of representing the contents of a scene, though it most clearly represents some of the important properties of light fields. In this representation, both t and v are fixed for each slice. Because the two remaining variables, s and u, represent the behaviour of the light field in the horizontal spatial direction – that is, in the x, z plane – each slice in this visualization represents a horizontal slit of the scene. One might imagine each s, u slice as showing a horizontal slit of a gantry

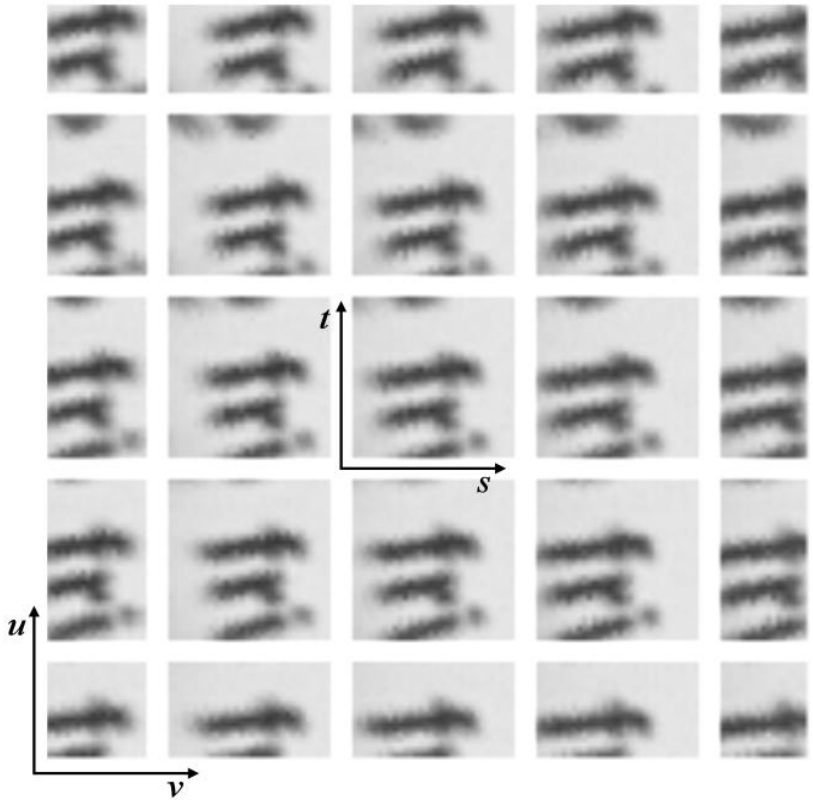


Figure 4.2: Visualizing the light field as an array of slices in s and t.

camera image, along the u direction, changing as a function of the camera position, along the s direction. Moving from slice to slice in the t and v directions is similar to changing perspective via camera translation and rotation, respectively, in the vertical direction.

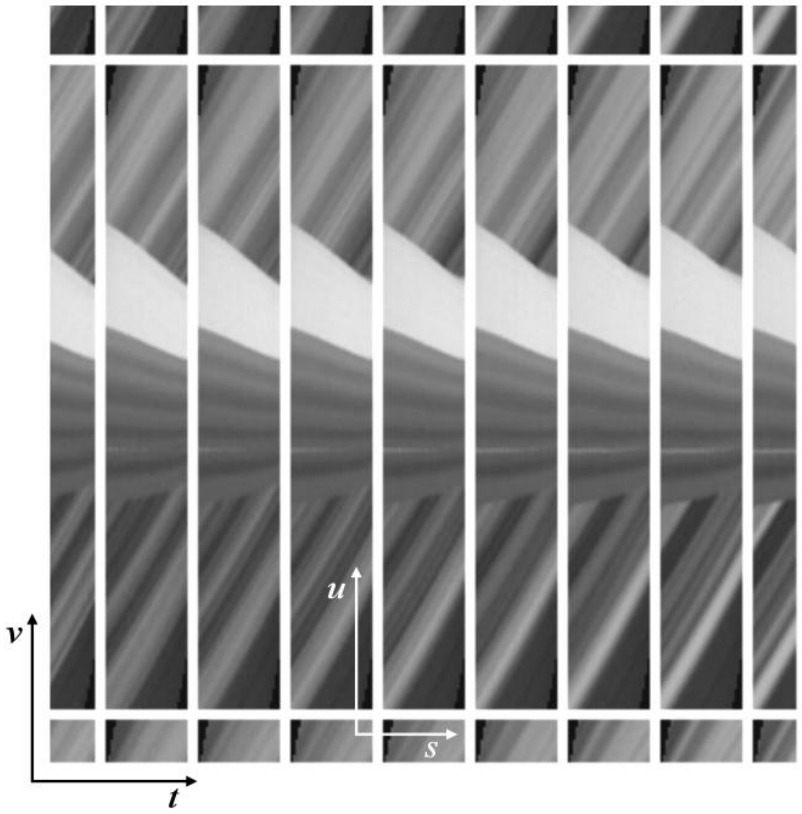


Figure 4.3: Visualizing the light field as an array of slices in s and u.

Arrays of Slices in t and v

Arrays of slices in t and v closely resemble slices in s and u, except that rather than visualizing a slit in the horizontal direction as a function of the horizontal camera

position, each slice in this representation shows a slit in the vertical direction, as a function of the vertical camera position. Moving from slice to slice in the s and u directions corresponds to camera translation and rotation, respectively, in the horizontal direction. Figure 4.4 shows an example of this representation.

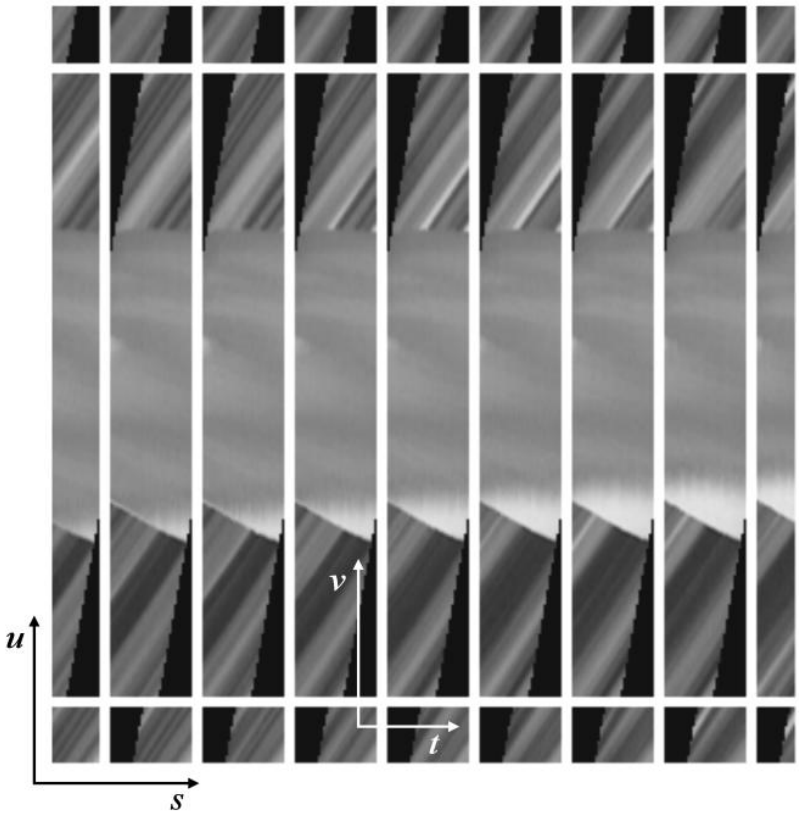


Figure 4.4: Visualizing the light field as an array of slices in t and v.

4.2 The Point-Plane Correspondence

An exploration of light fields in the context of scene analysis will require an understanding of the behaviours of different types of scenes and scene elements within the light field. A good starting point for this analysis is a single point in space, because any geometry may be modelled, by superposition, as a collection of points. Figure 4.5 is a top view of a single point in space, at the position $\mathbf{p} = [p_x, p_y, p_z]$, and one of the rays emanating from that point through the two reference planes.

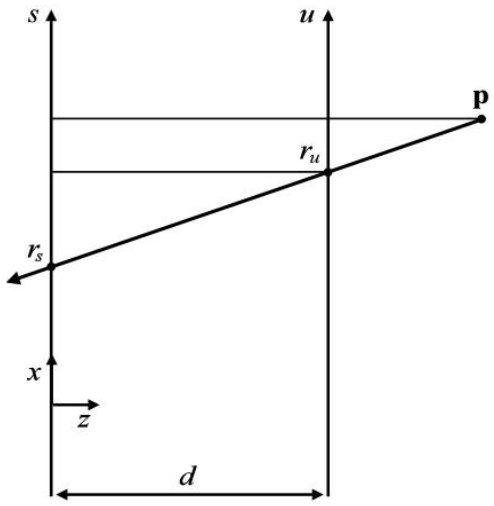


Figure 4.5: Top view of a point source of light, shown with the two reference planes.

It is clear that only a subset of the rays represented by the light field will intersect with the point $\mathbf{p}$. Less obvious is the relationship followed by all those rays intersecting the point, which is most easily derived by noting the similar triangles in the figure. Based on these similar triangles, the relationship between s and u for any ray intersecting $\mathbf{p}$ can be expressed as

$$\frac{r_u - r_s}{d} = \frac{p_x - r_s}{p_z}. \tag{4.1}$$

A similar expression can be written for t and v, as in

$$\frac{r_v - r_t}{d} = \frac{p_y - r_t}{p_z}. \tag{4.2}$$

Note that Equations (4.1) and (4.2) each describe a hyperplane in the 4D light field [11]. Both these equations need to be satisfied simultaneously in order for a ray to intersect the point in space. We may restate these two equations as a single equation, describing the subset of 4D ray space occupied by the point $\mathbf{p}$ as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} s & p_x \\ t & p_y \end{bmatrix} \begin{bmatrix} 1 - d/p_z \\ d/p_z \end{bmatrix}. \tag{4.3}$$

Equation (4.3) represents the intersection of the two hyperplanes described by Equations (4.1) and (4.2). Just as the intersection of two planes in 3D is a line, so is the intersection of two hyperplanes in 4D a plane. Thus, Equation (4.3) is the equation of a plane in 4D. Otherwise stated, a point in space exists as a plane in a light field. Evidence of this relationship can be seen in the straight lines which appear in the s, u and t, v slices of the light field shown in Section 4.1.

For the sake of simplicity, the notation $[s, t, u, v] \in P_p$ will be adopted as shorthand for the statement that the ray $[s, t, u, v]$ is a member of the plane corresponding to the point $\mathbf{p}$ – that is, it satisfies Equation (4.3).

There are a few observations worth noting with regards to the point-plane correspondence derived above. First, the hyperplanes described by Equations (4.1) and (4.2) have normals that depend only on the z coordinate of the point $\mathbf{p}$, and not on its x or y coordinates. Second, the normal is similar for the two hyperplanes – u varies with s as $\frac{1-d}{p_z}$, and v varies with t by the same amount. Finally, the point-plane correspondence describes those rays that intersect the point in space, but makes no

attempt to describe what the values of those rays might be. In this sense, the correspondence describes a mapping of those rays emanating from the point **p** into a plane in the light field, without dealing explicitly with the values of the rays.

4.2.1 Frequency-Domain ROS of an Omni-Directional Point Light Source

In order to describe the values of the rays within the plane described by the point-plane correspondence, we must make some assumptions with regards to what lies in the scene at the point **p**. The simplest assumption is that an omni-directional point light source lies at this point. By the definition of an omni-directional point light source, all the rays emanating from the point will have the same value, and so all the rays within the light field plane corresponding to the point will also have this same value.

The continuous-domain 4D Fourier transform of a light field is given by

$$L_{freq}(\mathbf{\Omega}) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L_{cont}(s,t,u,v)e^{-j(\Omega_s s+\Omega_t t+\Omega_u u+\Omega_v v)}dvdudtds. \tag{4.4}$$

For a light field containing only a point **p** which lies on the u, v reference plane – that is, for which $p_z = d$ – the light field contains only a single plane, given by Equation (4.3) as $u = p_x, v = p_y$. Note that this plane is aligned so as to be parallel with s and t, and perpendicular to u and v. If there is an omni-directional point light source at **p** then the light field plane will be of constant value, and so the entire light field will be of constant value along s and t. This means that the integrations along s and t will both evaluate to zero everywhere except at a frequency of zero

along either axis, and so the Fourier transform may be re-written as

$$L_{freq}(\mathbf{\Omega}) = \delta(\Omega_s)\delta(\Omega_t) \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L_{cont}(\Omega_s, \Omega_t, u, v)e^{-j(\Omega_u u + \Omega_v v)}dvdu. \tag{4.5}$$

This simplification is equivalent to making the observation that the signal is constant in s and t, and its ROS must therefore exist at a frequency of zero along those axes. The resulting frequency-domain signal has a region of support (ROS) defined by the two delta functions in Equation (4.5). Each delta function defines a hyperplane which passes through the origin, and the product of the two defines a plane at the intersection of the two hyperplanes. Thus, the frequency-domain ROS of an omni-directional point light source at the point $\mathbf{p}$, where $p_z = d$, is a plane through the origin.

The above observations may be generalized for a point at any depth in the scene by noting that any omni-directional point light source exists as a plane in the light field, regardless of the depth of that light source. Only the orientation of the plane changes with the depth of the light source, and so the light field remains constant along two orthogonal vectors. As a consequence, the frequency-domain ROS of the light field should be expected to be a rotated version of the plane through the origin described above. More formally, we may view the new light field as a rotated version of the case where $p_z = d$, as in

$$L'_{cont}(s, t, u, v) = L_{cont}(\mathbf{R}[s, t, u, v]^T), \tag{4.6}$$

where $\mathbf{R}$ is a 4D rotation matrix which depends on the depth p_z of the light source. It is a property of the Fourier transform that a rotation in the spatial domain corre-

sponds to a rotation in the frequency domain, and so we may write

$$L'_{freq}(\mathbf{\Omega}) = L_{freq}(\mathbf{R}\mathbf{\Omega}^T). \tag{4.7}$$

Thus, the rotated frequency-domain light field will have an ROS which is a rotated version of the plane through the origin.

We may avoid working directly with the rotation matrix $\mathbf{R}$ by noting the relationship between the orientation of the planar frequency-domain ROS and the orientation of the plane in the spatial-domain light field. In the s, u plane, the frequency-domain ROS is orthogonal to the spatial-domain plane – that is, it's rotated by 90 degrees. The same observation holds in the t, v plane, and the two observations together completely describe the orientation of the frequency-domain ROS. From Equation (4.3), the slope of the spatial-domain plane is found as $1 - d/p_z$ in both s, u and in t, v. Rotating this plane 90 degrees in s, u and in t, v in order to define the frequency-domain ROS is a matter of taking the negative of the inverse of the slope in both cases, as in

$$\begin{bmatrix} \Omega_s \\ \Omega_t \end{bmatrix} = \left(\frac{d}{p_z} - 1\right) \begin{bmatrix} \Omega_u \\ \Omega_v \end{bmatrix}. \tag{4.8}$$

This expression describes the frequency-domain ROS of an omni-directional point light source at any depth in the scene. Note that the orientation of this planar ROS depends only on the depth of the point in the scene, p_z.

4.2.2 Frequency-Domain ROS of a Lambertian Surface

The point-plane correspondence deals with a single point in space. Extension of this correspondence to deal with surfaces is a matter of superposition. This requires representing the surface as a set of surface elements, such as polygons. For a very

large number of surface elements, each one becomes infinitesimally small, and its behaviour approaches that of a single point in space, allowing us to apply the point-plane correspondence.

Pursuing this approach for a perfectly matte surface yields a particularly simple result. According to Lambert's cosine law, the luminous intensity $I(\theta)$ of a light ray reflecting from a single point on such a surface, as shown in Figure 4.6, varies as the cosine of the angle between the ray and the normal of the surface – i.e. $I(\theta) \propto \cos(\theta)$. Because of the way we perceive light, it is more useful to discuss the luminance, or luminous intensity per unit projected area, of the surface. This is because each cone or rod in our eye, much like each pixel in an image, actually measures the light emanating from a projected area of a surface, and not from a single point on that surface. The luminance may be found by integrating the luminous intensity over the projected area of a surface element. Doing this yields the observation that the luminance of a perfectly matte surface is independent of viewing angle. Because this observation derives from Lambert's cosine law, a perfectly matte surface is often referred to as a Lambertian surface.

At a Single Depth

Because of the above observations, a Lambertian surface may be broken down into infinitesimally small elements, and each element replaced with an omni-directional point light source. This process is not perfect, because it fails to correctly model occlusions, and so the initial assumption of a Lambertian surface in a scene with no occlusions is made. Under this assumption, the frequency-domain ROS of a Lambertian surface in the light field may be derived as the superposition of the

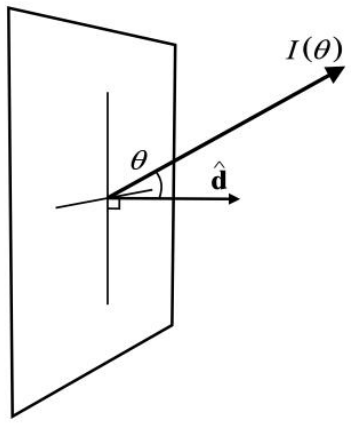


Figure 4.6: A ray of light reflecting from a point on a Lambertian surface.

regions of support of the point light sources that make up its model.

It is simplest to begin with a surface that exists at a single depth in the scene – that is, one that is parallel with the reference planes, say at the depth p_z. We may make the observation that all the point light sources in the corresponding surface model have the same depth, and therefore the same planar frequency-domain ROS, given by Equation (4.8). The ROS formed by the superposition of these planes is itself a plane [27] [28]. Note that this observation holds for a scene containing any number of Lambertian surfaces, so long as they are all at the same depth, p_z. Figure 4.7 depicts the planar ROS of an object at a single depth in the scene – a) is a slice in the Ω_s, Ω_u plane, and b) is a slice in the Ω_t, Ω_v plane. Observing Equation (4.8), the orientation of this planar ROS implies that the object is at a depth which is near the u, v reference plane, though not on it – the angle between the plane and the Ω_u and Ω_v axes is small, but not zero.

An alternative way of obtaining this result is by observing the light field in the

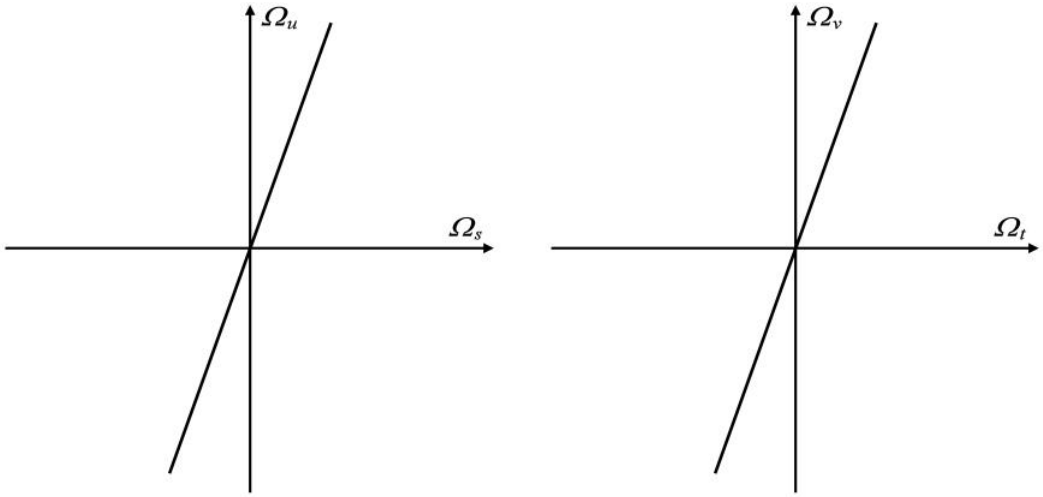


Figure 4.7: Planar frequency-domain ROS shown as a slice in a) Ω_s, Ω_u and b) Ω_t, Ω_v.

spatial domain. Modelling the scene as omni-directional point light sources, the light field contains only constant-valued planes, because the light sources in the model are all omni-directional. Furthermore, because the point light sources are all at the same depth, the planes are all parallel. As a consequence, the light field is again constant-valued along two orthogonal vectors, and so the frequency-domain ROS must exist at a frequency of zero along those two vectors.

Over a Range of Depths

The case of a Lambertian surface which exists over a range of depths becomes significantly more complex. Again, because modelling the scene as a collection of point light sources fails to correctly model occlusions, the assumption of a scene with no occlusions is made. Under this assumption, the principle of superposition may be employed. Starting with a scene containing a number of surfaces, each of which is parallel with the reference planes and lies at one of N discrete depths, we observe

that the frequency-domain ROS will consist of N planes through the origin. The orientation of each of these planes is different, and depends on the depth of the corresponding surface according to Equation (4.8).

Moving to the more general case in which the scene contains surfaces which are not parallel with the reference planes, we may consider the point light sources in the models of these surfaces as varying continuously in depth between the extents, in z, of the surfaces. Thus, the frequency-domain ROS may be obtained by allowing the orientation of the planar ROS from Equation (4.8) to vary continuously between the angles corresponding to the surfaces' extents in z – that is, by allowing p_z to vary between the minimum depth of the surface, z_{min}, and its maximum depth, z_{max}. The resulting equation,

$$\begin{bmatrix} \Omega_s \\ \Omega_t \end{bmatrix} = \left(\frac{d}{p_z} - 1 \right) \begin{bmatrix} \Omega_u \\ \Omega_v \end{bmatrix}, \qquad z_{min} \leq p_z \leq z_{max}, \tag{4.9}$$

describes the intersection of two fan shapes: one in the Ω_s, Ω_u plane, and another in the Ω_t, Ω_v plane. The intersection of the two fans will be referred to as a dual-fan.

The dual-fan is the ROS of any Lambertian surface, or collection of Lambertian surfaces, which lie between the depths z_{min} and z_{max}, and which are not occluded within the light field [28]. The dual-fan ROS of a scene containing a surface near the u, v reference plane is shown in Figure 4.8.

4.2.3 Effects of Specular Reflection and Occlusion

The above derivations only hold for scenes containing Lambertian surfaces and no occlusions. Real-world scenes contain both non-Lambertian surfaces and occlusions, and so an examination of both these types of scene behaviours on the frequency-

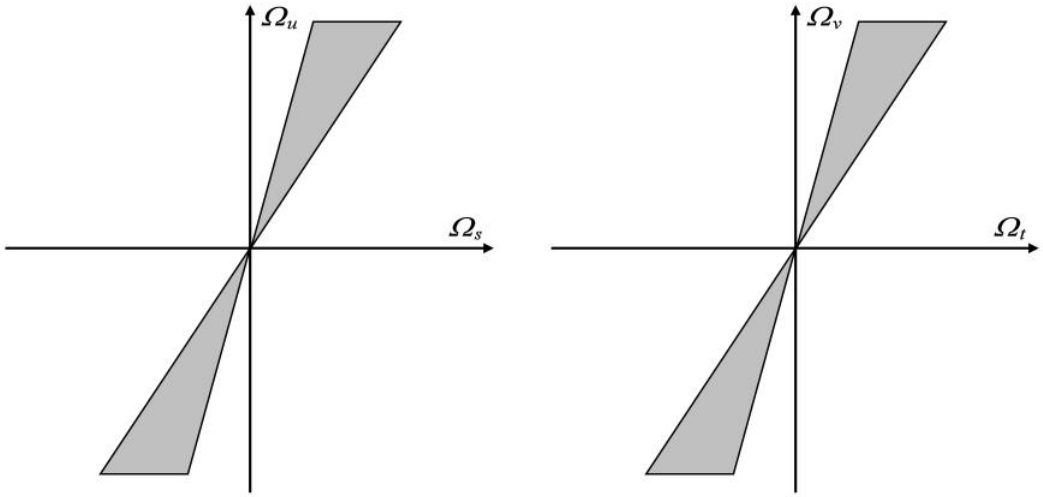


Figure 4.8: Dual-fan frequency-domain ROS shown as a slice in a) Ω_s, Ω_u and b) Ω_t, Ω_v.

domain ROS of the light field is in order.

The rays reflected from a point on a non-Lambertian surface have values which depend on the angle from which the surface is viewed. This dependence typically comes about as a result of specular reflection [14] – that is, light that reflects coherently from the surface, as it does from a mirror. Because the light is reflected coherently, the apparent colour of a point on the surface depends on the angle from which the surface is viewed. As a result, a point on a surface with specular reflections will correspond to a plane in the light field with non-constant value. For polished metal objects, the contents of the plane in the light field will typically resemble a 2D Gaussian pulse, as shown in Figure 4.9. The position of the center of this pulse coincides with the direction of maximum coherent reflection from the surface, and the rate at which it falls off depends on the specular power, or glossiness, of the surface. In the case of a scene containing only surfaces at a single depth, this Gaussian

characteristic will cause a deviation from the ideal planar frequency-domain ROS. Rather than being perfectly flat, the planar ROS will take on a finite width. A similar behaviour occurs for the case of a dual-fan ROS, for which a bowtie-like shape surrounds the fan in both the Ω_s, Ω_u and Ω_t, Ω_v planes.

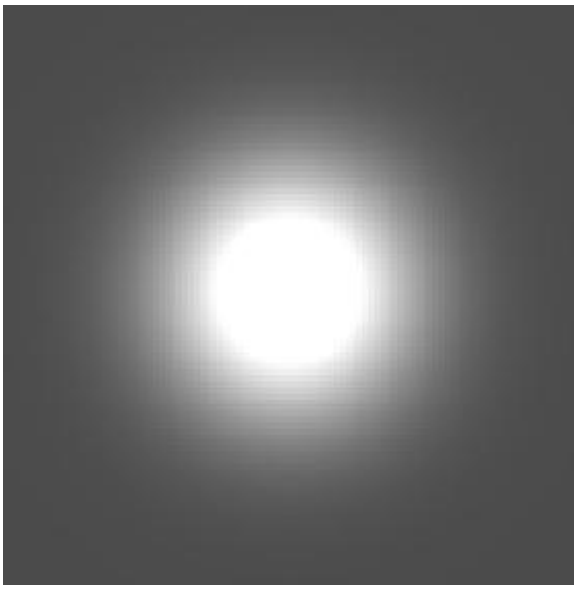


Figure 4.9: The contents of a light field plane which corresponds to a point on a polished metal surface.

It is fortunate that most of the energy in typical real-world scenes is contained in diffuse reflection, and not in specular reflection. Even shiny objects exhibit some diffuse reflection – specular highlights account for a small portion of the energy in most practical scenes. As a result, the planar ROS corresponding to the Lambertian portions of a scene tend to dominate the overall frequency-domain ROS. There are obvious exceptions – a hall of mirrors, for example – but for practical applications we might assume that this kind of situation is to be avoided, as even the human visual system is fooled in these circumstances.

Occlusion manifests itself, in the frequency domain, in much the same way as specular reflection. Planes in the spatial-domain light field intersect with each other where occlusions occur, resulting in discontinuities in what would otherwise be constant-valued planes. These sharp discontinuities have an effect similar to that of specular reflections on the frequency-domain ROS of a light field, though the effect tends to be more drastic. Planar regions of support are thickened, and dual-fan regions of support tend towards bowtie shapes surrounding the fans.

Again, for most practical scenes, the amount of energy in occluded regions tends to be small compared with the energy in non-occluded regions. As a result, the planar and dual-fan frequency-domain regions of support derived in this chapter tend to dominate.

Chapter 5

Filtering Light Fields for Depth

This chapter deals with the extraction of objects from a light field based on the depth of the objects in the scene that it models. Specifically, two methods are presented for extracting planar objects which are parallel with the reference planes at a single, prescribed depth. The proposed methods attenuate and blur all scene elements that are not at the prescribed depth, while selectively transmitting those elements which are at the prescribed depth.

The first of the methods proposed in this chapter is a plane averaging filter which utilizes the point-plane correspondence in the spatial domain to select surfaces at the desired depth. The second is a recursive filter with a passband which surrounds the planar frequency-domain ROS associated with surfaces at the desired depth. Both filters operate optimally in the absence of occlusions, and with scenes containing only Lambertian surfaces.

5.1 The Plane Averaging Filter

In Chapter 4, it was demonstrated that a Lambertian scene containing surfaces at a single depth yields a light field which consists entirely of parallel, constant-valued planes. Similarly, a scene containing surfaces at several discrete depths can be shown to yield a light field containing collections of parallel planes, with each collection having an orientation which depends on the depth of the corresponding surface.

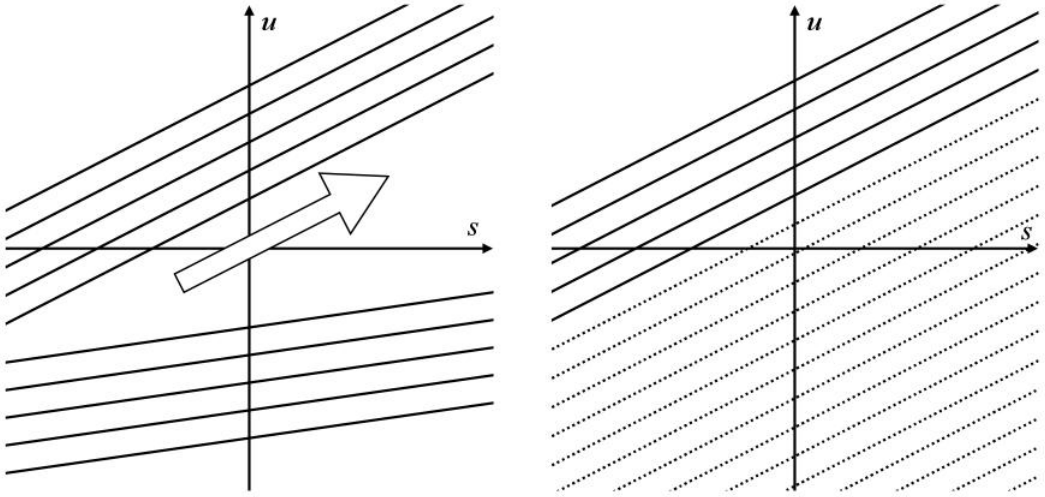
This latter observation can be used to create a filter which selectively transmits only those objects which lie at a desired depth, d_z, while removing or blurring all other scene elements.

The basic approach proposed for this filter is to synthesize a new light field out of parallel, constant-valued planes with orientations corresponding to the desired depth, d_z. The value of each of these synthesized planes is found as the average of the rays belonging to the corresponding plane in the input light field. For points in the scene at the depth d_z, this will result in little or no distortion, since for these points all rays in the averaged plane should have the same value. For points not at the depth d_z, however, the rays in the averaged plane will not have the same value, and so blurring will occur.

Figure 5.1 depicts the process of plane averaging. Part a) of this figure is a slice, in s and u, of the input light field, which models a scene containing surfaces at two depths. Each surface corresponds to a set of parallel, constant-valued planes – the depth of each surface determines the orientation of the corresponding set of planes. To extract the more distant of the two surfaces, corresponding to the set of planes at the top of the figure, a new light field is synthesized consisting entirely of constant-valued planes. The synthesized planes are parallel with the planes corresponding to the desired surface. The value of each of the synthesized planes is found by finding the average value of the corresponding plane in the input light field – this process is depicted by the arrow in Figure 5.1a).

For the surface at the desired depth, the synthesized planes are exact copies of the corresponding planes in the input light field. This is because the input planes are constant-valued in the direction in which the averaging occurs. This is depicted

by the solid lines in Figure 5.1b). For the undesired surface, however, the averaging process crosses the constant-valued planes in the input light field, and so the resulting averages are a blurred version of the input. This manifests itself as parallel, blurred planes in the output light field, as depicted by the dotted lines in Figure 5.1b). The process of plane averaging, then, selectively transmits objects at the desired depth, and blurs all the other elements of the scene.



(a) (b)

Figure 5.1: The process of plane averaging visualized in s and u: a) the input light field and b) the synthesized output light field.

In order to minimize memory requirements, plane averages may be found and stored in a 2D map $A(a_x, a_y)$ prior to synthesis of the new light field, which may replace the input light field in memory. This method eliminates the need to store both input and output light fields in memory simultaneously.

Modelling the scene as a collection of omni-directional point light sources, a single sample of the map $A(\mathbf{a})$ may be seen as an estimate of the value of the light source

at the point $\mathbf{p}$, where p_x and p_y correspond to a_x and a_y, and $p_z = d_z$, the target depth of the filter which remains constant across the map.

5.1.1 Finding Plane Averages

The value of each sample of $A(\mathbf{a})$ may be found by taking the average of all the light rays that lie in the plane corresponding to that sample. If the sample indexed by $\mathbf{a}$ corresponds to the point $\mathbf{p}$, then the corresponding light field plane can be found using the point-plane correspondence given by Equation (4.3).

Note that the resolution of $A(\mathbf{a})$ and the range of p_x and p_y that it covers are arbitrary. For good results, the total visible area at the depth d_z should be covered. To avoid aliasing, at least the total number of samples along u and v – assuming the light field has more samples in those dimensions – should be used in the map.

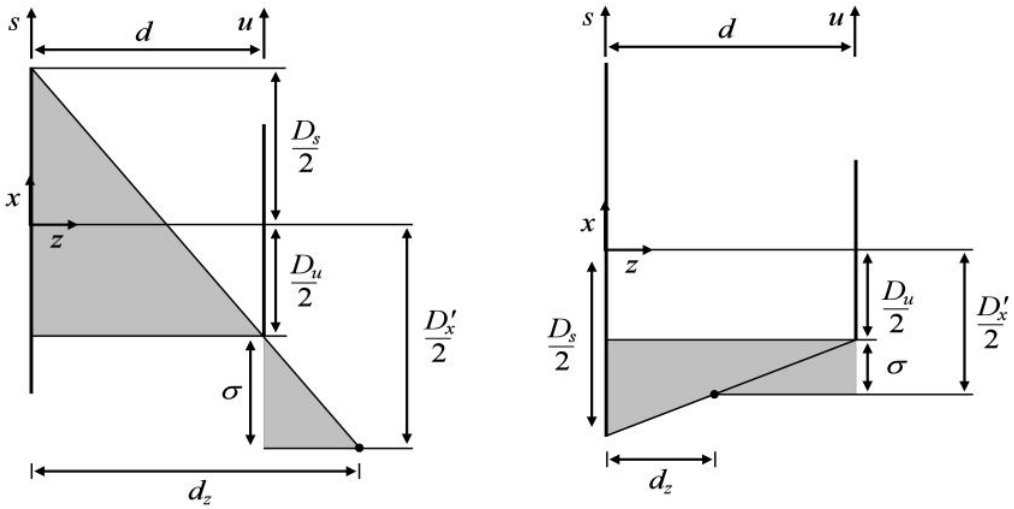
Figure 5.2 shows the geometry associated with finding the physical extents of the visible area at the depth d_z – these are the extents of the scene which will be represented by the map $A(\mathbf{a})$. These physical extents, given by $\mathbf{D'} = [D'_x, D'_y]$, are determined differently depending on the relative magnitudes of the target depth and reference plane separation. For a target depth which lies beyond the u, v reference plane – i.e. for $d_z > d$ – the total area covered by $A(\mathbf{a})$ is determined by the geometry shown in a). For a target depth which is closer than the u, v reference plane – i.e. for $d_z \leq d$ – the total area is determined as shown in b). In both cases, the length σ is found using the similar triangles which have been shaded in these figures. Note that the reference planes, shown as heavy lines, are assumed to be of different sizes to assure the generality of the resulting equations.

The expression for $\mathbf{D'}$ derived from the geometry shown in these figures is given

as

$$\begin{bmatrix} D'_x \\ D'_y \end{bmatrix} = \begin{bmatrix} D_u \frac{d_z}{d} \pm D_s \left(1 - \frac{d_z}{d}\right) \\ D_v \frac{d_z}{d} \pm D_t \left(1 - \frac{d_z}{d}\right) \end{bmatrix}, \tag{5.1}$$

where the positive right term of each equation is used for $d_z > d$, and the negative is used for $d_z \leq d$.



(a) (b)

Figure 5.2: Finding the physical extents of the map $A(\mathbf{a})$ when a) $d_z > d$ and b) $d_z \leq d$.

Assuming more samples in u and v than in s and t, the total number of samples in the map $A(\mathbf{a})$ can be taken as $\mathbf{I} = [N_u, N_v]$ – i.e. $0 \leq a_x < N_u$ and $0 \leq a_y < N_v$. Then, the physical extents represented by the map, along with the number of samples in the map, can be used to find the point $\mathbf{p}$ in the scene corresponding to the sample $A(\mathbf{a})$, as in

$$\begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} \left(\frac{a_x}{I_x - 1} - 0.5\right) D'_x \\ \left(\frac{a_y}{I_y - 1} - 0.5\right) D'_y \\ d_z \end{bmatrix}. \tag{5.2}$$

Having found the point $\mathbf{p}$ corresponding to each sample in the map, the next step is to estimate the average value of the rays belonging to the plane corresponding to that point. This is most quickly accomplished by iterating through every sample in u and v, finding the corresponding values of s and t using Equation (4.3). In the case where there are more samples in s and t than in u and v, the process should be reversed, finding the values of u and v corresponding to each sample in s and t.

The values of s, t, u and v resulting from this process describe the rays belonging to the plane corresponding to the point $\mathbf{p}$ – that is, $[s,t,u,v] \in P_p$. Plane averaging is effected by finding the average of the values of those rays, with the values of the rays found through interpolation from the light field. For the case where there are more samples in u and v than in s and t, this can be expressed as

$$A(\mathbf{a}) = \bar{L}_{P_p} = \frac{\sum_{n_u=0}^{N_u-1} \sum_{n_v=0}^{N_v-1} \tilde{L}_{cont}(s,t,u,v) \Big|_{[s,t,u,v]\in P_p}}{N_u N_v}, \tag{5.3}$$

where $\bar{L}_{P_p}$ denotes the average value of the light field samples which lie in the plane P_p. In this equation, u and v are found from n_u and n_v using Equation (3.1), the point $\mathbf{p}$ corresponding to the sample $\mathbf{a}$ is found using Equation (5.2), and the values of s and t corresponding to u and v are found using Equation (4.3).

In order to operate correctly near the edges of the light field, rays which lie outside the light field should be ignored when finding the average. This only affects the denominator of Equation (5.3), which is reduced by one for each sample which lies outside the light field. Because the interpolation process returns a value of zero for rays outside the light field, no adjustment of the numerator is necessary.

In [27], a process similar to plane averaging is described which simulates finite depth of field in a camera. By applying a lowpass filter to rays in a neighbourhood

surrounding each input ray, finite depth of field is simulated. Plane averaging differs from this technique in that it uses all available light field samples, thus yielding the maximum depth selectivity possible using this kind of approach.

5.1.2 Synthesizing the New Light Field

Depending on the application, the map $A(\mathbf{a})$ may be a sufficient form of output. To perform object detection, for example, one might search for edges in the map in order to determine the presence of an object at the specified depth. For applications requiring interactive viewing, however, $A(\mathbf{a})$ can be used to synthesize a new light field which consists of constant-valued parallel planes. This could be accomplished by iterating through each plane in turn, as was done in finding $A(\mathbf{a})$, though it is quicker and more effective to iterate through the whole light field and operate on each sample in turn.

This approach begins by finding the point in space corresponding to each light field sample – the resulting points in space are assumed to be at the target depth d_z. The value of each light field sample is then interpolated from the map $A(\mathbf{a})$ using the resulting points in space to find the appropriate index $\mathbf{a}$. Because the depth d_z is assumed for every point in space, the resulting light field will consist entirely of constant-valued, parallel planes.

For each sample $L(\mathbf{n})$, the corresponding ray given by $[s, t, u, v]$ is found using Equation (3.1). Next, the values of p_x and p_y are found by rearranging Equation (4.3) using the condition $p_z = d_z$, as in

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} s & u \\ t & v \end{bmatrix} \begin{bmatrix} 1 - p_z/d \\ p_z/d \end{bmatrix}. \tag{5.4}$$

The corresponding index into the map is found from $\mathbf{p}$ by solving Equation (5.2) for $\mathbf{a}$, as in

$$\begin{bmatrix} a'_x \\ a'_y \end{bmatrix} = \begin{bmatrix} \left(\frac{p_x}{D'_x} + 0.5\right)(I_x - 1) \\ \left(\frac{p_y}{D'_y} + 0.5\right)(I_y - 1) \end{bmatrix}. \tag{5.5}$$

Note that this yields a continuous-domain value, $\mathbf{a}'$, which is used to interpolate from the map $A(\mathbf{a})$. The resulting interpolated value is assigned to the light field at the index $\mathbf{n}$. This entire process can be summarized as

$$L(\mathbf{n}) = \tilde{A}(\mathbf{a}') \Big|_{[s,t,u,v] \in P_p}, \tag{5.6}$$

where $[s, t, u, v]$ is found from $\mathbf{n}$ using Equation (3.1), $\mathbf{p}$ is found from $[s, t, u, v]$ using Equation (5.4), and $\mathbf{a}'$ is found from $\mathbf{p}$ using Equation (5.5).

5.1.3 Improvements

For densely sampled light fields, it may not be necessary to utilize all available samples when calculating the average value of each plane. A significant speed improvement could be effected, for example, by skipping every other sample in each of the two directions of iteration. Note that it is important to utilize samples from all areas of the light field, however, rather than taking samples from some small region, in order to achieve maximum depth selectivity.

This filter can also be made to deal with incomplete light fields by ignoring missing samples during the averaging process. If a light field sample is empty (black), then it is simply ignored when estimating the average value of the rays in a plane. This has the effect of reducing the denominator of Equation (5.3) by one for each empty sample. Because this process already occurs for ignoring samples that lie outside the light field, this operation is essentially free. Furthermore, empty samples in the

input light field are effectively filled in the output light field, because the synthesis process always generates complete planes.

5.1.4 Results

The plane averaging filter was implemented and incorporated into the Lightbench package. It was tested over a range of target depths and on a number of different light fields, some of which contained occlusions and specular reflections. Results were positive in all cases. Results will be shown here for a light field containing occlusions – the light field is shown in Figure 5.3, and has parameters as summarized in Table 5.1. The scene that it models contains a beer mat in the foreground, and a poster in the background. The beer mat is parallel with the reference planes at a depth of 48 cm, and is held in place by a wooden dowel, which can be seen near the bottom of the figure. The poster is of a supernova, imaged by the Dominion Radio Astrophysical Observatory in British Columbia, and is also parallel with the reference planes at a depth of 66 cm.

Table 5.1: Light field parameters.

colour channels	3
s, t samples	32
u, v samples	256
s, t size (cm)	45
u, v size (cm)	36
separation d (cm)	57

The plane averaging filter was applied to the input light field with target depths

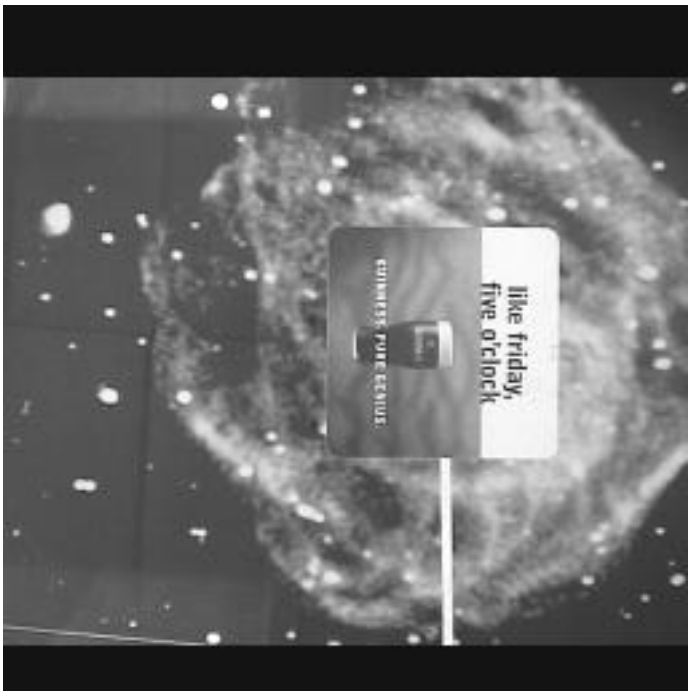
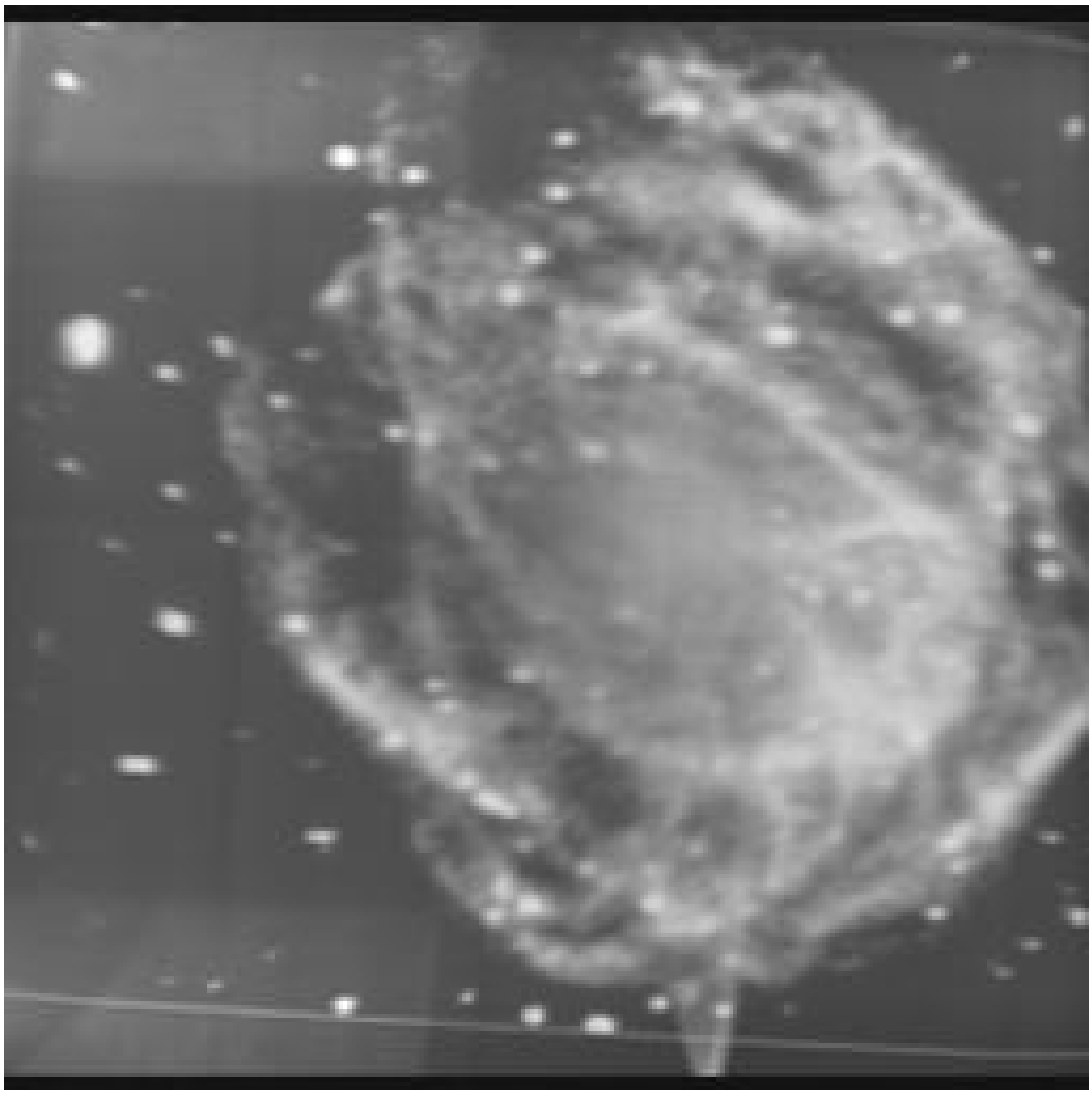


Figure 5.3: Input light field modelling a scene containing a beer mat and a poster.

of 48 and 66 cm – the results are shown in Figure 5.4a) and b), respectively. Each filtering operation took about 10 minutes on a Pentium IV running at 1 GHz. A speed-enhanced version of the algorithm was produced by reducing the samples used in the plane averaging process by a factor of 8. The speed-enhanced algorithm took just over 1 minute to run – the results are shown in c) and d).

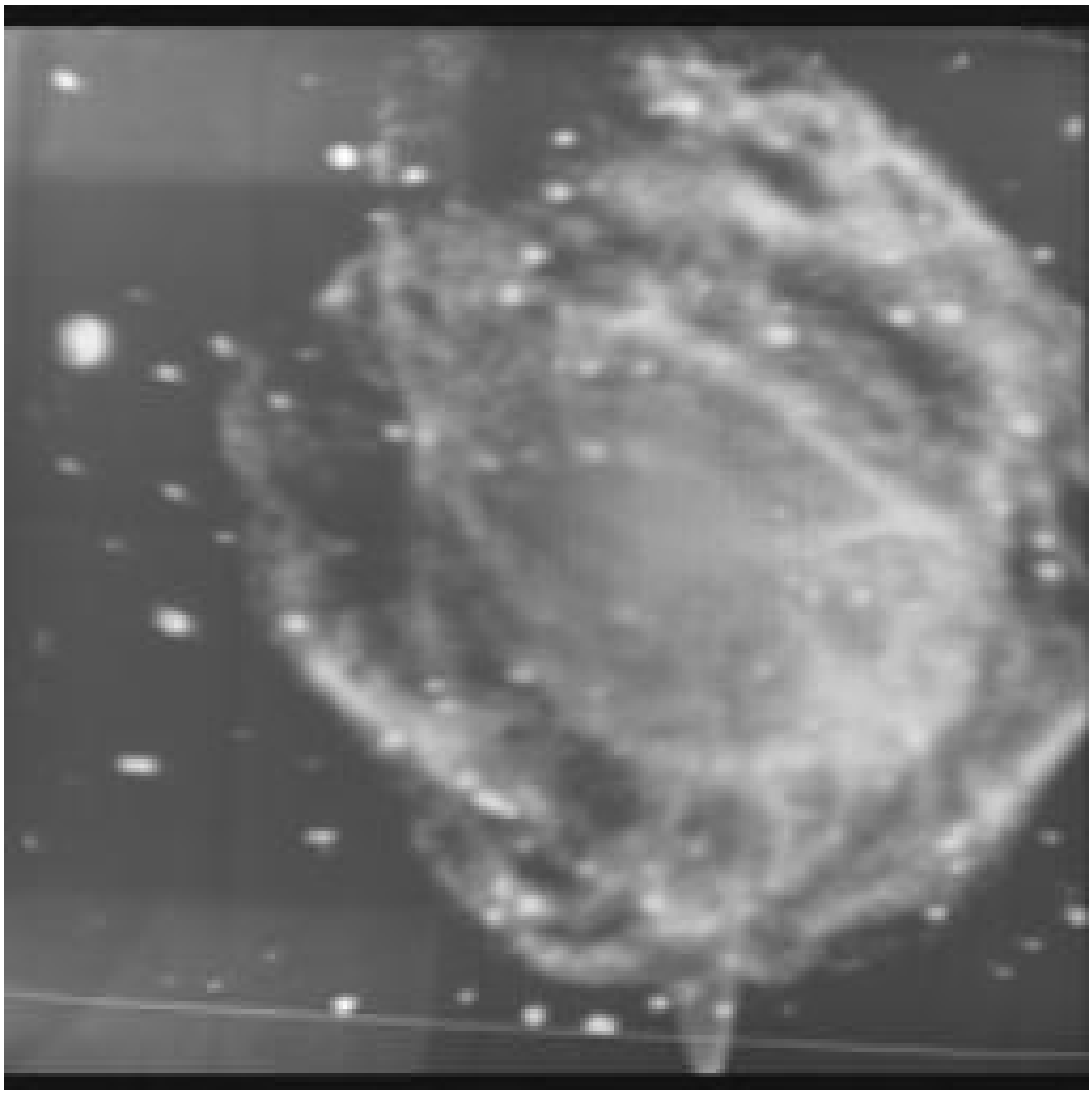
(a)

Figure 5.4: Results of plane averaging: a) applied at 48 cm and b) 66 cm; c) the speed-enhanced version applied at 48 cm and d) 66 cm.



(b)

(c)



(d)

Clearly, the plane averaging filter successfully isolated foreground and background portions of the light field. The effects of occlusion in this case are clear – the occluded portions of the poster were not successfully reconstructed. This is not surprising, as some portions near the center of the poster were completely occluded across the light field – that is, they were not represented in any portion of the light field.

Note also that some blurring of the desired signal did occur, though this is most likely a symptom of error in the light field measurement process associated with the assumption that the camera behaves as an ideal pinhole camera.

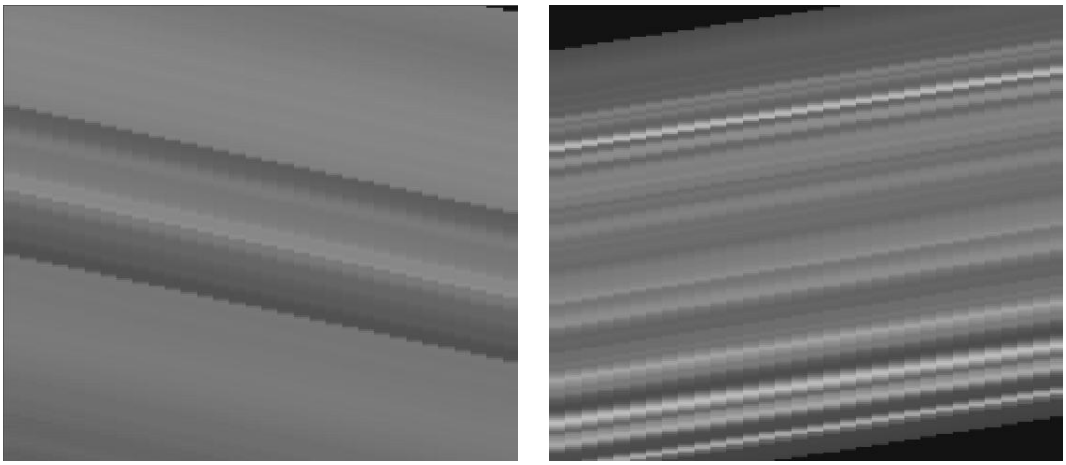
When applied to scenes containing specular reflections, the plane averaging filter removes specular highlights from surfaces at the desired depth. This is because specular reflections on objects at the desired depth correspond to planes in the light field which are not of constant value, and which have orientations not parallel with the averaged planes. As a result, specular reflections are blurred by plane averaging.

The speed-enhanced version of the plane averaging algorithm produces results nearly indistinguishable from those produced by the original version, though at a significantly higher speed. Some artifacts arose due to the lower certainty associated with the plane average estimates.

A visualization of the output light field as a slices in s and u, as shown in Figure 5.5, reveals that the behaviour of the synthesized light field is ideal. That is, all the planes in the light field are parallel, and constant-valued. Results from this section will be contrasted with those obtained using the frequency-planar filter in the following section.

5.2 The Frequency-Planar Filter

The plane averaging technique presented in the previous section exploited the point-plane correspondence by working directly in the spatial domain. This section presents an alternative approach in which linear filters are utilized to extract the appropriate passband in the frequency domain [4].



(a) (b)

Figure 5.5: Results of plane averaging at a) 48 cm and b) 66 cm, visualized as slices in s and u.

In Chapter 4, the point-plane correspondence was used to show that a Lambertian surface parallel with the reference planes at a given depth in the scene has a frequency-domain ROS which is a plane through the origin. The orientation of that planar ROS was shown to depend only on the depth of the surface in the scene. By superposition, a scene containing several surfaces at different depths will have a frequency-domain ROS which is a set of planes, with each one passing through the origin, and each one having a different orientation. By extracting one of those planes in the frequency domain, it should be possible to create a depth-selective filter similar to the plane averaging filter from the previous section.

In order to minimize computational complexity, recursive filters will be utilized to form the 4D planar passband. Previous work in the design of recursive 3D frequency-planar filters [29] forms a good starting point for the design of this filter. In fact, the 3D frequency-planar filter presented in [29] can be directly extended to a 4D

frequency-hyperplanar filter. Furthermore, just as a beam-shaped passband can be formed in 3D by intersecting two planar passbands [29], so too can a planar passband be formed in 4D by intersecting two hyperplanar passbands. The basic approach to designing the planar filter, then, is to intersect two appropriately oriented hyperplanar passbands.

5.2.1 Review of the 3D Frequency-Planar Recursive Filter

In [29] a digital, recursive 3D frequency-planar filter is proposed based on an analog 3D prototype filter. The prototype filter, shown in Figure 5.6, is guaranteed to be stable. The formal proof for this stability is beyond the scope of this thesis [30], though it can be seen intuitively through the realization that the three inductors in the filter are passive, and when combined with the output load R are guaranteed to form a bounded output for any bounded input. Thus, this filter prototype is said to be bounded-input bounded-output (BIBO) stable.

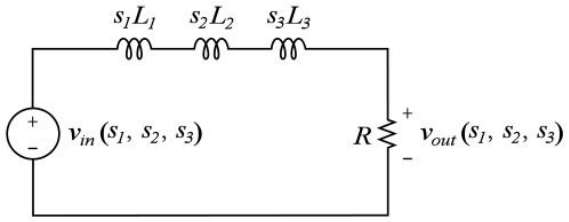


Figure 5.6: The analog 3D prototype filter.

The continuous-domain Laplace transform transfer function of this prototype filter is given by

$$T(s_1, s_2, s_3) = \frac{1}{1 + (s_1L_1 + s_2L_2 + s_3L_3)/R}. \tag{5.7}$$

Substituting $s_i = j\Omega_i$, the frequency response of this prototype is given by

$$T(j\Omega_1, j\Omega_2, j\Omega_3) = \frac{1}{1 + j(\Omega_1 L_1 + \Omega_2 L_2 + \Omega_3 L_3)/R}. \tag{5.8}$$

The magnitude of the frequency response, given by

$$M(\Omega_1, \Omega_2, \Omega_3) = \frac{1}{\sqrt{1 + (\Omega_1 L_1 + \Omega_2 L_2 + \Omega_3 L_3)^2/R^2}}, \tag{5.9}$$

is constant-valued in planes given by

$$\Omega_1 L_1 + \Omega_2 L_2 + \Omega_3 L_3 = k, \quad k \in \mathbb{R}. \tag{5.10}$$

Note that these planes all have the same normal, in the direction $[L_1, L_2, L_3]$. Also note that the magnitude response has a maximum value of unity, on the plane defined by $k = 0$. Moving away from this resonant plane, the value of the magnitude response falls away from unity. The rate at which it falls off is determined by R. Thus, this prototype forms a planar passband, with a bandwidth selected by R, and a normal defined by $[L_1, L_2, L_3]$. From Equation (5.9), the -3dB planes – that is, the planes for which the magnitude of the filter is $1/\sqrt{2}$ – are given by

$$\Omega_1 L_1 + \Omega_2 L_2 + \Omega_3 L_3 = \pm R. \tag{5.11}$$

A digital form of the analog prototype filter is obtained by applying the bilinear transformation [29] [31]. Substituting $s_i = \frac{z_i - 1}{z_i + 1}$ in the 3D case results in the z-transform transfer function

$$T(\mathbf{z}) = \frac{\sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} z_1^{-i} z_2^{-j} z_3^{-k}}{\sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1} b_{i,j,k} z_1^{-i} z_2^{-j} z_3^{-k}}, \tag{5.12}$$

where $b_{i,j,k}$ are given as the 8 sign configurations of

$$b_{i,j,k} = 1 + (\pm L_1 \pm L_2 \pm L_3)/R, \tag{5.13}$$

as summarized in Table 5.4. The difference equation used to implement this 3D digital filter [29] is given by

$$y(n_1, n_2, n_3) = \frac{1}{b_{0,0,0}} \Bigg[\sum_{i=0}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} x(n_1 - i, n_2 - j, n_3 - k) - \sum_{\substack{i=0 \\ i+j+k \neq 0}}^{1} \sum_{j=0}^{1} \sum_{k=0}^{1} b_{i,j,k}\, y(n_1 - i, n_2 - j, n_3 - k) \Bigg], \tag{5.14}$$

where x($\mathbf{n}$) and y($\mathbf{n}$) are the input and output signals, respectively.

Table 5.4: Coefficients of the 3D frequency-planar filter.

$b_{0,0,0} = 1 + (L_1 + L_2 + L_3)/R$	$b_{1,0,0} = 1 + (-L_1 + L_2 + L_3)/R$
$b_{0,0,1} = 1 + (L_1 + L_2 - L_3)/R$	$b_{1,0,1} = 1 + (-L_1 + L_2 - L_3)/R$
$b_{0,1,0} = 1 + (L_1 - L_2 + L_3)/R$	$b_{1,1,0} = 1 + (-L_1 - L_2 + L_3)/R$
$b_{0,1,1} = 1 + (L_1 - L_2 - L_3)/R$	$b_{1,1,1} = 1 + (-L_1 - L_2 - L_3)/R$

Stability

The digital form of the recursive 3D frequency-planar filter is practical-BIBO stable [29], but only when the normal of the resonant plane is in the first octant. Normals with negative components correspond to negative inductances in the prototype filter – this leads to instability. In order to ensure stability, then, the normal is forced to be non-negative along all three dimensions – that is, it is flipped in all dimensions along which it is initially negative. Reversing the normal in a dimension has the

effect of reversing the frequency-domain behaviour of the filter in that dimension. This can be counteracted by reversing the direction of iteration of the filter along all those dimensions for which the normal is flipped [29]. The overall effect is of a stable filter with the desired passband.

5.2.2 The 4D Frequency-Hyperplanar Recursive Filter

Extension of the 3D frequency-planar filter to operate in 4D is a simple matter of adding an extra spatial variable. The analog prototype gains an inductor, operating in the fourth dimension, resulting in a continuous-domain Laplace transform transfer function given by

$$T(s_1, s_2, s_3, s_4) = \frac{1}{1 + (s_1 L_1 + s_2 L_2 + s_3 L_3 + s_4 L_4)/R}. \tag{5.15}$$

Following the same derivation as for the 3D case, we reach the conclusion that this prototype yields a magnitude frequency response which is resonant in a *hyperplane*, given by

$$\Omega_1 L_1 + \Omega_2 L_2 + \Omega_3 L_3 + \Omega_4 L_4 = k. \tag{5.16}$$

As with the 3D version of this filter, the normal is determined by the values of the inductors, L_i, and the bandwidth is determined by the resistance, R.

For the sake of consistency with the rest of this thesis, the symbols $\hat{\mathbf{d}} \in \mathbb{R}^4$ and $B \in \mathbb{R}$ will be adopted to represent the normal of the resonant hyperplane, and the -3dB bandwidth of the passband, respectively. Also, the numbered subscripts on 4D vectors will be replaced with s, t, u and v, to avoid confusion.

At this point, it is worth mentioning that differing sample rates in the four light field dimensions will require adjustment of the normal used with the hyperplanar

filter. A higher sample rate in any of the dimensions will cause compression in the corresponding dimension in the frequency domain, and so the normal will need to be adjusted appropriately. The adjusted normal, $\hat{\mathbf{d}}'$, is found as

$$\hat{\mathbf{d}}' = \frac{\left[d_s\frac{N_s}{D_s}, d_t\frac{N_t}{D_t}, d_u\frac{N_u}{D_u}, d_v\frac{N_v}{D_v}\right]}{\left\|d_s\frac{N_s}{D_s}, d_t\frac{N_t}{D_t}, d_u\frac{N_u}{D_u}, d_v\frac{N_v}{D_v}\right\|} \tag{5.17}$$

Using the corrected normal, the input-output difference equation for the 4D frequency-hyperplanar filter is given by

$$\begin{aligned} y(\mathbf{n}) = \frac{1}{b_{0,0,0,0}}\Bigg[&\sum_{i=0}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}\sum_{l=0}^{1} L(n_s-i, n_t-j, n_u-k, n_v-l) \\ &- \sum_{\substack{i=0\\ i+j+k+l\neq 0}}^{1}\sum_{j=0}^{1}\sum_{k=0}^{1}\sum_{l=0}^{1} b_{i,j,k,l}\, y(n_s-i, n_t-j, n_u-k, n_v-l)\Bigg], \end{aligned} \tag{5.18}$$

where L($\mathbf{n}$) and y($\mathbf{n}$) are the 4D input and output signals, respectively, and $b_{i,j,k,l}$ are found as the 16 sign configurations of

$$b_{i,j,k,l} = 1 + (\pm d'_s \pm d'_t \pm d'_u \pm d'_v)/B. \tag{5.19}$$

As with the 3D frequency-planar filter, this filter becomes unstable if the sign on any of the components of the normal is negative. In order to filter outside the first hexadecimant, the same approach is followed as in the 3D case for filtering outside the first octant.

5.2.3 Forming a Frequency-Planar Passband

Intersecting Planes in 3D

In [29], a scheme is proposed in which two frequency-planar filters are cascaded in order to form a beam-shaped passband. The overall transfer function of the cascaded

filters is unity only where the resonant planes intersect – that is, on the line defined by the intersection of the two planes. The process of orienting the beam-shaped passband, then, is one of choosing two planar passbands which intersect along the desired beam. It turns out there are an infinite number of ways to do this – Figure 5.7 shows a 2D slice of one of the possible arrangements, in terms of -3dB surfaces.

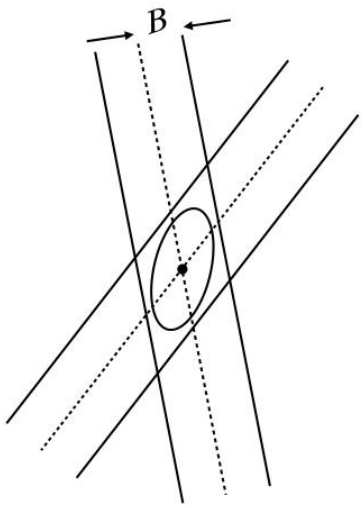


Figure 5.7: 2D slice of two 3D frequency-planar passbands intersecting in a beam, shown as -3dB surfaces.

From Figure 5.7, it is clear that the shape of the beam passband distorts somewhat from an ideally round cylinder due to the finite bandwidth of the two planar passbands that make it up. There is a set of planar orientations which yields the optimally shaped passband – that is, the passband which deviates the least from the ideally round cylinder. This set is described as all those for which the two planar passbands are orthogonal. This condition minimizes the stretching of the cylindrical passband associated with those regions where the angle between the two planes is small.

Intersecting Hyperplanes in 4D

The beam-forming technique explored in 3D can easily be extended to plane-forming in 4D. By cascading two appropriately oriented frequency-hyperplanar filters, a resonant plane can be formed at the intersection of the two resonant hyperplanes.

In Chapter 4, the frequency-domain ROS of a Lambertian surface parallel with the reference planes at a depth p_z in the scene was derived as in Equation (4.8). This expression can be rearranged as

$$\begin{bmatrix} 1 & 0 & 1-d/p_z & 0 \\ 0 & 1 & 0 & 1-d/p_z \end{bmatrix} \begin{bmatrix} \Omega_s \\ \Omega_t \\ \Omega_u \\ \Omega_v \end{bmatrix} = 0, \tag{5.20}$$

from which it is clear that each half of this equation describes a hyperplane, with the normals of the two hyperplanes given by

$$\hat{\mathbf{d}}^{su} = \frac{[1, 0, 1-d/p_z, 0]}{\sqrt{1+(1-d/p_z)^2}}, \tag{5.21}$$

and

$$\hat{\mathbf{d}}^{tv} = \frac{[0, 1, 0, 1-d/p_z]}{\sqrt{1+(1-d/p_z)^2}}. \tag{5.22}$$

Using the observations made above, the frequency-planar filter may be formed by cascading two frequency-hyperplanar filters with the normals $\hat{\mathbf{d}}^{su}$ and $\hat{\mathbf{d}}^{tv}$. Note that there are an infinite number of ways to select the normals of the hyperplanes, though there are two good reasons to use the two proposed here. First, because each of these two normals is oriented so as to present selectivity along only two of the frequency axes, a simplification of the 4D hyperplanar filters will be possible, as explored in the following section. Second, as with the 3D case, there is a set of optimal orientations associated with the hyperplanes, for which the distortion of the

passband is minimized. These orientations are those for which the hyperplanes are orthogonal. Because the dot-product of the normals proposed here is always zero, the two hyperplanes are always orthogonal, satisfying this criterion and yielding the optimally shaped planar passband.

Each filter is implemented using the difference equation described by Equation (5.18), and each has a continuous-domain Laplace transform transfer function defined by Equation (5.15). The two filters are cascaded, resulting in the overall transfer function given by

$$T(s_s, s_t, s_u, s_v) = T_{su}(s_s, s_t, s_u, s_v)T_{tv}(s_s, s_t, s_u, s_v), \tag{5.23}$$

which is only unity in the resonant plane which coincides with the planar ROS defined by Equation (4.8). If the sample rates in the four dimensions differ, corrected normals need to be found using Equation (5.17).

To summarize, the process of designing a filter to extract objects at a depth d_z begins by calculating the normals $\hat{\mathbf{d}}^{su}$ and $\hat{\mathbf{d}}^{tv}$ corresponding to the target depth using Equations (5.21) and (5.22). If necessary, these normals are corrected for differing sample rates using Equation (5.17). Then, the filter coefficients $b_{i,j,k,l}$ are calculated for each of the two cascaded filters, using Equation (5.19), with an appropriately selected bandwidth B. Finally, the coefficients are used with the difference equation given by Equation (5.18) to create each of the two filters.

A Simpler Hyperplanar Filter

Because of the way the normals of the frequency-hyperplanar filters are aligned to form the planar passband, a simplification of these two filters is possible. The normals of both hyperplanes, $\hat{\mathbf{d}}^{su}$ and $\hat{\mathbf{d}}^{tv}$, are aligned so as to produce selectivity along only

two of the frequency axes. In the case of $\hat{\mathbf{d}}^{su}$, for example, there is only selectivity along Ω_s and Ω_u, and none along Ω_t or Ω_v. This fact can be used to form a 2D beam filter which, when applied throughout the light field, will exactly reproduce the passband of the 4D hyperplanar filter. In the case of the filter with selectivity along Ω_s and Ω_u, the beam filter is applied to slices, in Ω_s and Ω_u, of the light field.

In order to maintain a low computational complexity, the 2D beam filter may also be designed as a recursive filter. This is accomplished by removing a single dimension from the 3D frequency-planar filter described in Section 5.2.1. In the case of the first of the two cascaded filters, the input-output difference equation becomes

$$y(n_s, n_u) = \frac{1}{b^{su}_{0,0}} \left[\sum_{i=0}^{1} \sum_{j=0}^{1} L(n_s - i, n_u - j) - \sum_{\substack{i=0 \\ i+j \neq 0}}^{1} \sum_{j=0}^{1} b^{su}_{i,j}\, y(n_s - i, n_u - j) \right], \quad (5.24)$$

where the $b_{i,j}$ coefficients are found as the 4 sign configurations of

$$b^{su}_{i,j} = 1 + (\pm d^{su}_s \pm d^{su}_u)/B. \quad (5.25)$$

This filter has a magnitude frequency response which is resonant in a line, given by

$$\Omega_s d^{su}_s + \Omega_u d^{su}_u = k. \quad (5.26)$$

By iterating through all combinations of n_t and n_v, applying the 2D beam filter to the corresponding slice in n_s and n_u, the desired 4D hyperplanar passband is formed. Creating the second filter, with selectivity in Ω_t and Ω_v is a matter of iterating through all combinations of n_s and n_u, applying the beam filter in the n_t and n_v directions. Modifying Equations (5.24) and (5.25) to operate in the Ω_t and Ω_v dimensions is trivial. Again, for differing sample rates in the four dimensions, Equation (5.17) needs to be used to find corrected normals.

5.2.4 Zero-Phase Filtering

Recursive filters are used because they lead to implementations which are less computationally expensive than their non-recursive counterparts. Unfortunately, high-selectivity recursive filters tend to have long impulse responses, and thus long transient responses. A long transient response manifests itself, in the case of a light field, as darkening near the edges of the light field, and as a smearing effect in the direction of iteration of the filter. These effects are tolerable for input signals with many samples – a 2D image, for example, might have hundreds of samples in both dimensions, and so a startup transient of 8 samples is hardly noticeable. For a light field, on the other hand, there are often very few samples along the s and t dimensions – as few as 32 or even 16 – and so a startup transient that lasts 8 samples represents a significant loss of information in the light field. A method of counteracting the effects of a long transient response is therefore desirable.

The proposed method is to repeat the filtering operation, in a second pass, with the directions of iteration of the filters reversed. This can be seen as a reversal of all spatial, and therefore all frequency, axes in the second-pass filters. Repeating the derivation shown in Section 5.2.1 with reversed spatial dimensions, the 4D hyperplanar filter with reversed directions of iteration has the frequency response given by

$$T(j\Omega_s, j\Omega_t, j\Omega_u, j\Omega_v) = \frac{1}{1 - j(\Omega_s L_s + \Omega_t L_t + \Omega_u L_u + \Omega_v L_v)/R}, \tag{5.27}$$

which differs from the original only in the sign of the imaginary component of the denominator. Note that the magnitude of the frequency response is identical to that of the original, despite the reversal of axes. This observation relates to the symmetry

of the passband, and applies equally well to the 3D frequency-planar and 2D beam filters described earlier. It can be shown intuitively, in the case of the 2D beam filter, by imagining the beam-shaped passband being mirrored along both axes in turn, ultimately yielding the exact same shape.

Although the sign change on the axes has no effect on the magnitude response of the filters, the phase response is reversed. This can also be seen intuitively, by imagining each filter as *smearing* the input along its directions of iteration. If the first-pass filters smear the signal in the positive s, t, u, and v directions, then the second-pass filters will smear the result back, in the negative s, t, u, and v directions. As a result of this reversal of phase, the net phase of the cascade of filters is zero. Furthermore, because the magnitude response of the two sets of filters is the same, given as $M(j\Omega_s, j\Omega_t, j\Omega_u, j\Omega_v)$, the magnitude response of the cascade of filters is given as the product of the two, $(M(j\Omega_s, j\Omega_t, j\Omega_u, j\Omega_v))^2$. Thus, zero-phase filtering has the added bonus of increasing the selectivity of the filter.

In order to maximize the benefit of using zero-phase filtering, extra memory needs to be allocated. As the input is smeared in the direction of iteration by the first-pass filters, the values that are smeared off the edges of the light field need to be saved. These saved values are then used, essentially as initial conditions, by the second-pass filter in order to negate the effects of the startup transient. The alternative is to have the second-pass filter darken the edges of the light field, yielding a startup transient as severe as in the original case, except at the opposite side of the light field. The storing of extra output values will require both more memory and more processing time.

5.2.5 Implementation Details

Four types of filter were implemented and incorporated into the Lightbench package: the frequency-planar filter consisting of two cascaded frequency-hyperplanar filters, the simplified version which consists of two beam filters, and the zero-phase versions of both. The user interface accepts a target depth, d_z, and bandwidth, B, as input. To the author's knowledge, this is the first known implementation and application of a recursive 4D filter.

The input light field is stored as an array of colour triplets, with each component of the triplet represented as a single byte. As a result, each of the colour components takes on one of only 256 possible states. Although this is an acceptable resolution for the human visual system, it is somewhat limiting from a filtering point of view. Undesired artifacts and even instability can result from utilizing only 8 bits of precision with the filters described above. The problem arises in the feedback component of the filter, in which the previous outputs determine the current output. A small error due to quantization in one step is compounded as the filter iterates. In order to combat this effect, the output of the filter is stored as 32-bit floating-point numbers, rather than 8-bit integers.

In order to simplify computation, the filters operate on a single colour channel at a time. The first hyperplanar filter is applied to the first colour channel, initially storing the results in an array of floating-point numbers, and then overwriting the input array with the results. The resulting partially-filtered light field is then processed by the second hyperplanar filter, which operates on the same colour channel, again storing the results as floating-point numbers, then overwriting the input with the resulting

output. In the case of zero-phase filtering, the second-pass filters are then applied to this same colour channel. Once completed, this process is repeated for the two remaining colour channels.

One way of implementing each hyperplanar filter is to compute and store the entire array of 32-bit floating-point outputs at once, then convert the contents of this array back to 8-bit integers, overwriting the input light field with the output values. For a light field containing $N_{tot} = N_s N_t N_u N_v$ samples, this scheme would require $7N_{tot}$ bytes of memory: 3 bytes per input colour triplet, and 4 bytes per floating-point output.

However, rather than storing the entire output array, it is possible to set up a more efficient memory-swapping scheme. The light field can be processed in *frames* in s, t and u – one might imagine each frame as being a single frame of a video signal (although containing a third dimension), with the v axis behaving as the time axis. Because all the filters used are first-order, only one previous output frame is required to calculate the current output frame. The new scheme, then, is to store only two output frames as 32-bit floating-point numbers at once, rather than all of them. Each of the two stored frames takes turns representing the new output frame and the previous output frame. As each new output frame is completed, it is used to overwrite the previous input frame. Under this scheme, only $N_s N_t N_u (8 + 3N_v)$ bytes of memory are needed. For a light field with 256 samples in u and v, and 32 in s and t, this amounts to 194 MB of memory, rather than the 448 MB required by the scheme in which the entire array is stored as floating-point numbers. Under this swapping scheme, there are only two extra megabytes of memory associated with the storage of intermediate results.

Because it operates on 2D slices, the faster hyperplanar filter consisting of 2D beam filters requires even less memory than in the scheme discussed above. Rather than operating on 3D frames in s, t and u, the beam filter may operate on 1D frames in s or t. The memory savings associated with taking this approach are minimal, however, and so the Lightbench implementation always works on 3D frames.

The zero-phase filtering technique was implemented by allowing the direction of iteration of each hyperplanar filter to be reversed. By repeating the frequency-planar filtering operation with the directions of iteration reversed, a zero-phase filtering operation is carried out. Extra memory is allocated to store samples as they are smeared off the edges of the light field. This is implemented by growing all the dimensions of the input and output light fields by a factor κ. Because the memory-swapping technique is used, the memory required by the zero-phase filtering technique is calculated as $\kappa^3 N_s N_t N_u (8 + 3\kappa N_v)$. For a factor of $\kappa = 1.2$, this amounts to about twice the memory for the example used above. It also amounts to about four times the processing time, because the filters are applied twice, and they operate on an input array with twice the samples.

5.2.6 Results

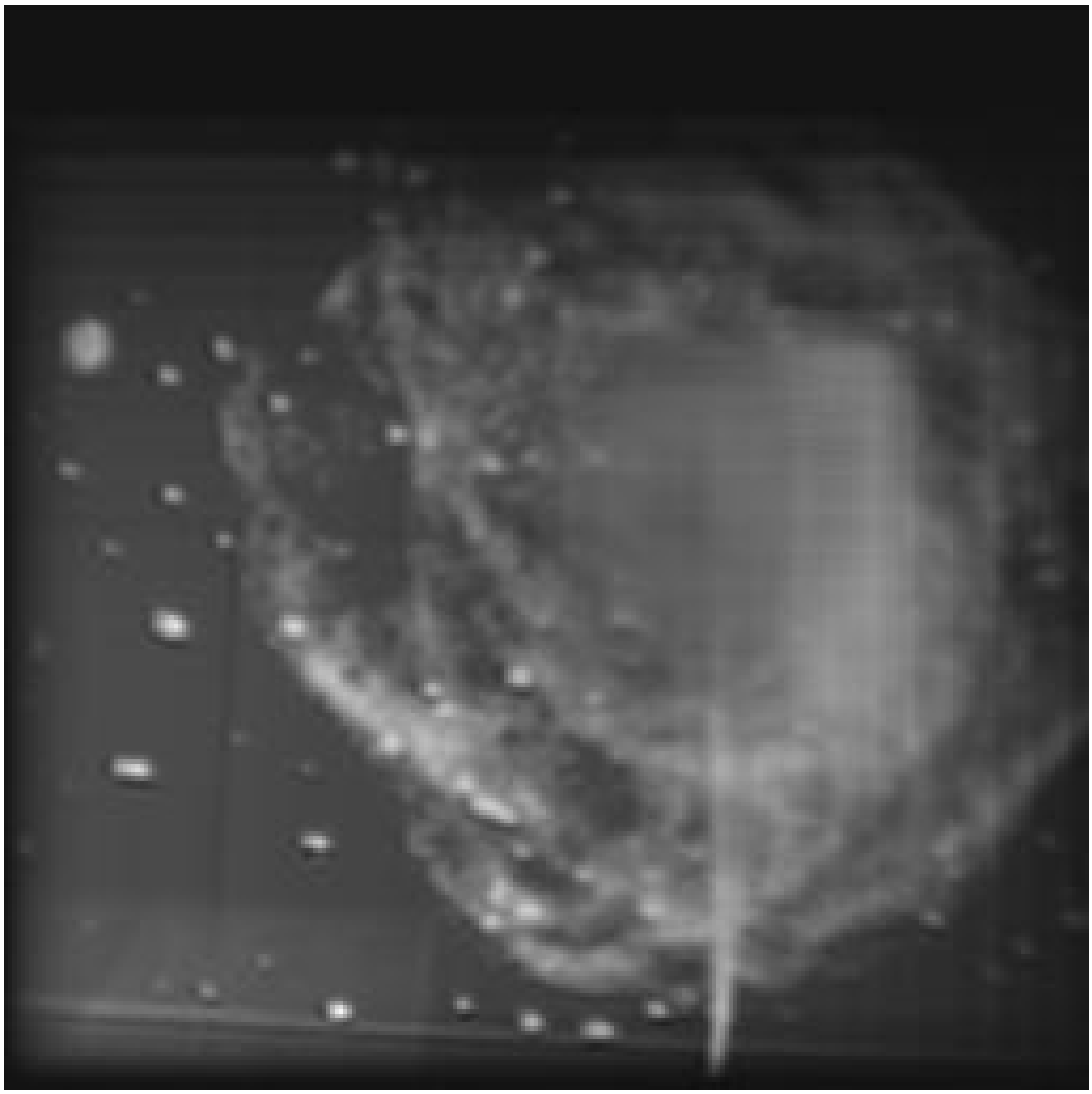
The four types of filter were tested at a series of target depths and on a wide range of light fields, including some containing both occlusions and specular reflections. Results were positive in all cases. Figure 5.8 shows the results for one of the light fields which contains occlusions – this is the same light field used to test the plane averaging filter.

Each filter was applied to the input light field with target depths of 48 and 66 cm,

corresponding to the planar foreground and background objects, respectively. The bandwidth was selected, through trial and error, to be 0.05. The results for the filter which does not utilize the beam filter simplification are shown in a),b),c) and d) respectively. Each filter took about 10 minutes on a Pentium IV running at 1 GHz. The zero-phase version of each filter, utilizing a memory increase factor of $\kappa = 1.1$, took about three times as long, as well as occupying 1.5 times the memory.

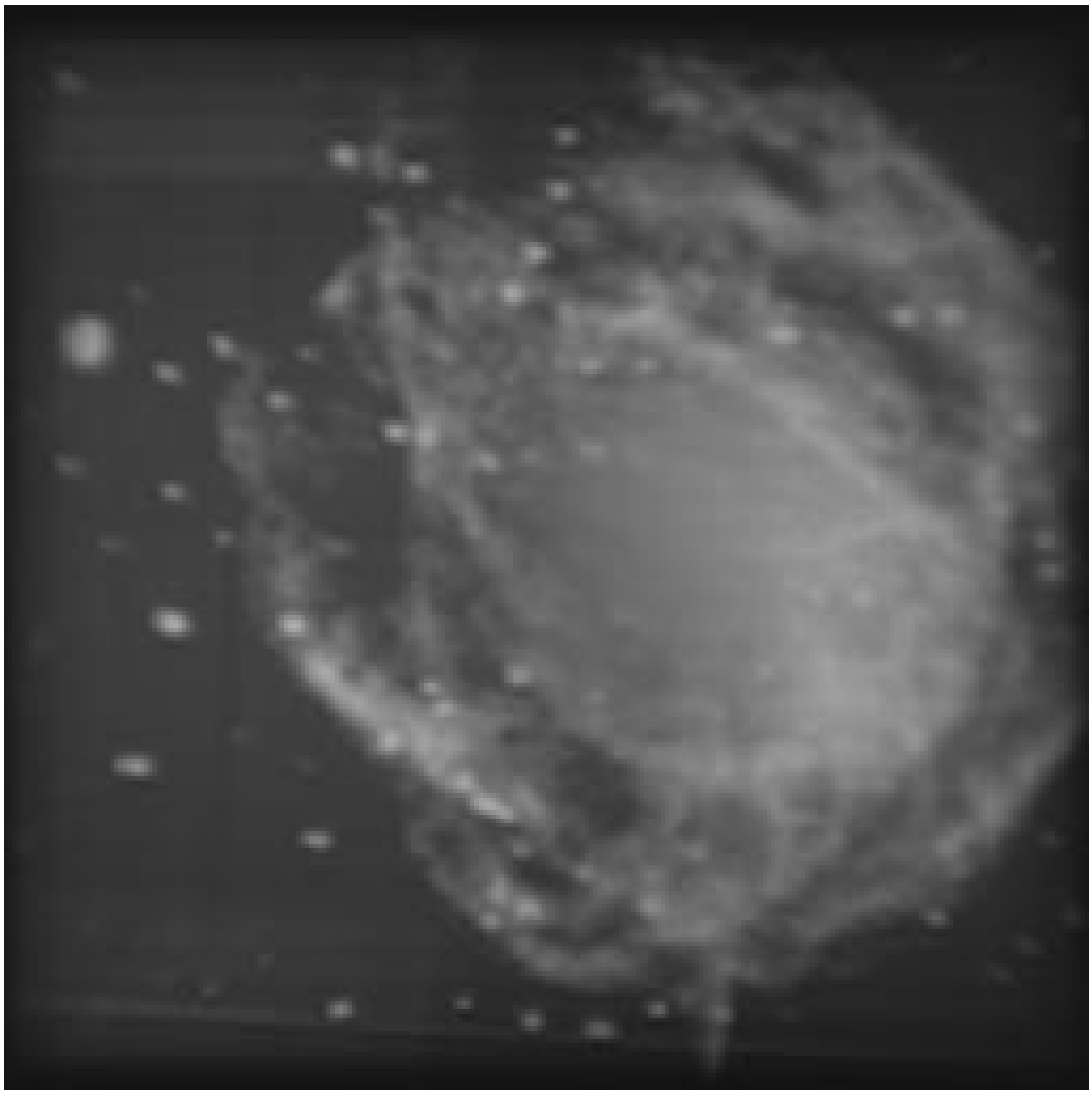
(a)

Figure 5.8: Results of frequency-planar filtering: a) applied at 48 cm and b) 66 cm; c) the zero-phase version applied at 48 cm and d) 66 cm.



(b)

(c)



(d)

The beam-filter simplification yielded identical results, but with a speed increase of about a factor of 1.7. The results are indiscernible from those shown in Figure 5.8, and are therefore omitted for the sake of brevity.

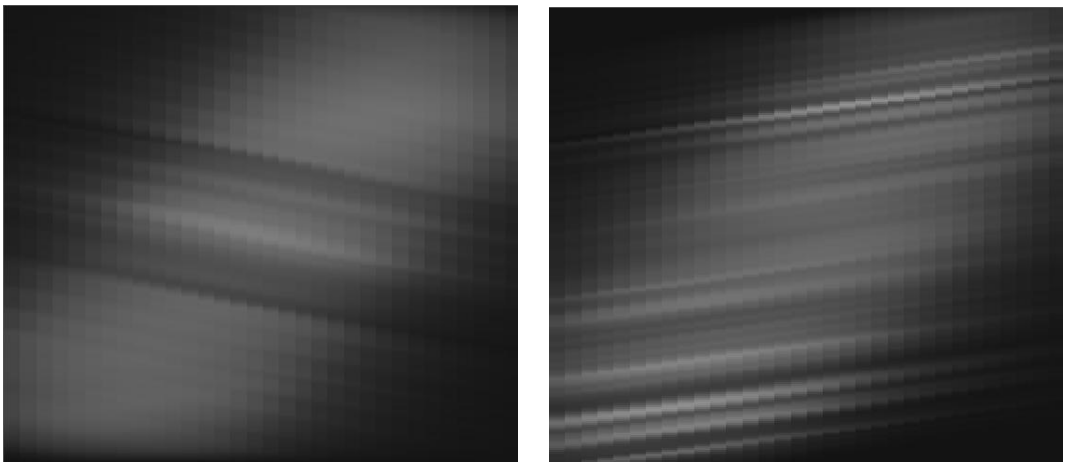
Clearly, the frequency-planar filters have successfully isolated the foreground and background portions of the light field. The results are similar to those obtained using

the plane averaging technique, though they're somewhat blurrier. The increased blurriness is most likely due to warping associated with the bilinear transformation [31] [32], which is most prominent at higher frequencies.

The zero-phase filtering technique clearly yielded a shorter transient response, as evidenced by the decrease in darkening near the edges of the light field, and higher selectivity, as seen in the blurrier stopband signals. As with the plane averaging filter, occluded regions of the light field were not all successfully reconstructed, although this is to be expected given that some portions of the background are completely occluded across the light field – that is, they are not represented in any portion of the light field.

The frequency-planar filtering technique yielded less ideal results than the plane averaging technique in terms of the dynamic behaviour of the light field. A visualization of the output light field as a slice in s and u, as shown in Figure 5.9, reveals that the behaviour of the synthesized light field is non-ideal. That is, all the planes in the light field are not constant-valued. This is most clear near the edges of the light field, where some samples are missing due to the limited FOV of the gantry camera, and near the center of the s, u slice of the light field, where the poster is occluded by the coaster. When viewing the light field interactively, this latter deviation from the ideal manifests itself as a floating, blurry blob in the space that the coaster originally occupied. Although neither plane averaging nor frequency-planar filters were able to correctly reconstruct occluded regions of the poster, the plane averaging technique at least generated an ideally flat scene as its output.

The behaviour of the frequency-planar filter in the presence of specular reflections is similar to that of the plane averaging filter. Specular reflections on surfaces at



(a) (b)

Figure 5.9: Results of frequency-planar filtering applied at a) 48 cm and b) 66 cm, visualized as slices in s and u.

passband depths become severely blurred, while the diffuse components of the surface reflections remain intact. In some circumstances, this type of behaviour might be beneficial. In robot navigation, for example, the presence of specular reflections might confuse the robot, and so a filter which removes specular reflections might be desirable.

Although the frequency-planar filter generated results which were in may ways inferior to those generated by the plane averaging filter, it remains interesting for several reasons. Because it is entirely linear, and first-order, this filter is an excellent candidate for optimization through parallelization, pipelining, or the use of systolic architectures. More importantly, this filter lays the groundwork for more complex linear filters, such as the 4D-DFFB explored in the following chapter.

Chapter 6

Extracting a Range of Depths

Chapter 5 focussed on extracting planar objects parallel with the reference planes, at a prescribed depth in the scene. This chapter presents a more generally useful type of filter, which extracts non-planar objects occupying a range of depths in the scene.

6.1 The Dual-Fan Filter Bank

In Chapter 4, the frequency-domain ROS of a surface occupying a range of depths was derived to be the dual-fan described by

$$\begin{bmatrix} \Omega_s \\ \Omega_t \end{bmatrix} = \left(\frac{d}{p_z} - 1 \right) \begin{bmatrix} \Omega_u \\ \Omega_v \end{bmatrix}, \quad z_{min} \leq p_z \leq z_{max}. \tag{6.1}$$

The basic approach presented in this chapter is to design a 4D dual-fan filter bank (4D-DFFB) with a passband which surrounds this dual-fan shape, in order to extract objects over a range of depths in a scene. The dual-fan passband is formed as the intersection of two fan passbands, just as a planar passband was formed in the previous chapter as the intersection of two hyperplanar passbands. The intersection is effected by cascading two frequency-fan filters. The first of the two fan filters has selectivity in Ω_s and Ω_u, while the second has selectivity in Ω_t and Ω_v.

Each of the two fan-shaped passbands is formed using a filter bank. Filter banks are a recently proven method, in the realm of multidimensional signal processing, for the formation of complex passband shapes [33] [32]. The filter banks first separate

the input signal into sub-bands, then each sub-band is filtered using a frequency-hyperplanar filter with an appropriately selected orientation and bandwidth. This is similar to the approach taken in [33] to form a 3D conical passband.

6.1.1 The Fan Filter Banks

Figure 6.1 is a representation of an N_b-band filter bank. The input signal is broken up into sub-bands by a series of sub-band separation filters, then each sub-band is operated on by a sub-band processing filter. The resulting filtered signals are then recombined to form the final output. This section will focus on design of the first of the two filter banks, which operates in the Ω_s and Ω_u dimensions – extension of this design to function as the second filter bank, which operates in Ω_t and Ω_v, is trivial.

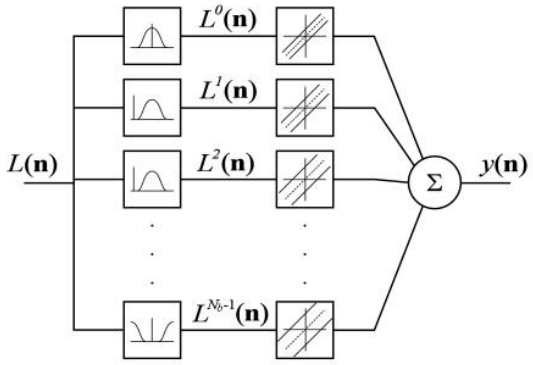


Figure 6.1: The Fan Filter Bank.

Fan Geometry

In order to best approximate a fan-shaped passband, the sub-band separation filters must separate the input signal along the dimension which is closest to the central axis of the fan. This requires finding the orientation of the central axis, which we

will define in terms of the angle, θ_c, between that axis and the Ω_u axis. Finding θ_c requires a definition of the range of depths to be extracted by the filter. If the minimum and maximum depths to be extracted are given by z_{min} and z_{max}, then the angles of the hyperplanes corresponding to these depths may be found, and used to find θ_c. Using Equation (6.1), and observing the geometry of the situation, the angle formed between the hyperplane corresponding to the depth p_z and the Ω_u axis is given by

$$\theta_z = \tan^{-1}(d/p_z - 1). \tag{6.2}$$

Note that the angle θ_z is positive in the clockwise direction. In the case where the sample rates in the four dimensions differ, this equation needs to be corrected, as in

$$\theta_z = \tan^{-1}\left((d/p_z - 1)\frac{N_u/D_u}{N_s/D_s}\right). \tag{6.3}$$

The angle formed between the central axis of the fan and the Ω_u axis can be found as the mean of the angles corresponding to the minimum and maximum depths, as in

$$\theta_c = \frac{\theta_{z_{min}} + \theta_{z_{max}}}{2}. \tag{6.4}$$

If the central axis of the fan is within 45 degrees of the Ω_u axis, then the input signal should be separated into sub-bands along that axis – otherwise, the signal should be separated into sub-bands along Ω_s. From Equation (6.3), the central axis of the fan will be closer to the Ω_u axis for most practical scenes – that is, for scenes in which the object to be extracted is near the u, v reference plane – and so the assumption is made that the input signal is always to be separated into sub-bands along Ω_u. Extension to operate in the general case is a matter of separating sub-bands along the Ω_s axis when appropriate.

Following [33], Figure 6.2 shows the passband formed by a four-band filter bank for an object occupying a range of depths near the u, v reference plane. This figure shows the ideal passband as a dashed line, and the approximated passband as a solid line. Note that the sub-bands, labelled from 0 to 3, are real – that is, they include both positive and negative frequencies, so that the spatial-domain signal is entirely real-valued.

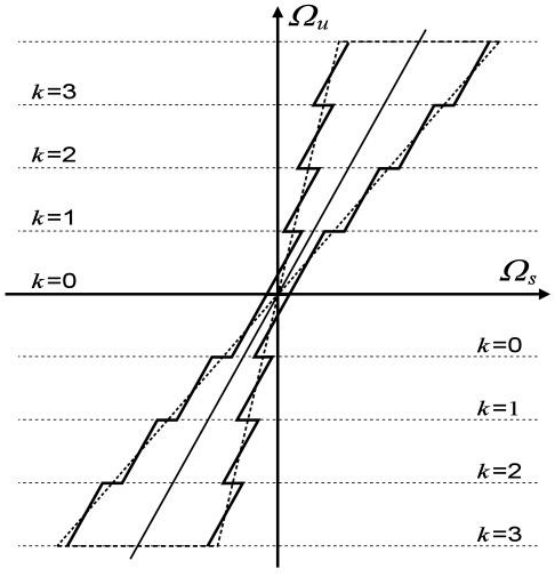


Figure 6.2: Approximating the fan-shaped passband using four sub-bands.

Exact-Perfect-Reconstruction Band Separation Filters

The input signal is separated into sub-bands by N_b sub-band separation filters based on the exact-perfect-reconstruction filters described in [33] and [34]. The filters are not recursive, are of order O, and have real coefficients. They are constructed from the lowpass filter given by

$$h_{LP}(n_u) = \frac{1}{2(N_b - 1)} \text{sinc}\left(\frac{n_u - (O-1)/2}{2(N_b - 1)}\right). \tag{6.5}$$

This filter is time-scaled by the factor $2(N_b-1)$ to obtain the correct bandwidth for N_b real sub-bands. In order to maintain causality, it introduces a delay of $n_0 = (O-1)/2$ samples.

Each of the sub-band separation filters is constructed from the lowpass filter by modulating it up to the correct center frequency, and windowing the result to the length O. The k^{th} bandpass filter is given by

$$h_{BP}^{k}(n_u) = \begin{cases} w(n_u)h_{LP}(n_u)\cos\left(\pi n_u \frac{k}{N_b-1}\right), & k = 0, N_b-1 \\ 2w(n_u)h_{LP}(n_u)\cos\left(\pi n_u \frac{k}{N_b-1}\right), & k = 1, 2, ..., N_b-2, \end{cases} \tag{6.6}$$

where $w(n_u)$ is a rectangular window of length O.

The conditions for perfect reconstruction [33], in the case of real signals and filter coefficients, can be derived from the criterion

$$h_{bank} = \sum_{k=0}^{N_b-1} h_{BP}^{k}(n_u) = \delta(n_u - n_0). \tag{6.7}$$

Substituting Equation (6.6) into Equation (6.7) yields the expression

$$h_{bank} = w(n_u)h_{LP}(n_u)h_c(n_u) = \delta(n_u - n_0), \tag{6.8}$$

where

$$h_c(n_u) = \left[\cos(0) + \cos(\pi n_u) + 2\sum_{k=1}^{N_b-2}\cos\left(\pi n_u \frac{k}{N_b-1}\right)\right]. \tag{6.9}$$

It is a simple matter to show that $h_c(n_u)$ has a value given by

$$h_c(n_u) = \begin{cases} 2(N_b-1), & n_u = 2r(N_b-1), \quad r \in \mathbb{Z}, \\ 0, & \text{otherwise}, \end{cases} \tag{6.10}$$

In order to satisfy Equation (6.7), then, three conditions for perfect reconstruction may be deduced:

$$\frac{n_0}{2(N_b-1)} \in \mathbb{Z}, \tag{6.11}$$

$$h_{LP}(n_0) = \frac{1}{2(N_b - 1)}, \tag{6.12}$$

and

$$h_{LP}(n_u) = 0 \text{ when } n_u = 2r(N_b - 1) \text{ and } n_u \neq n_0,\ r \in \mathbb{Z}. \tag{6.13}$$

The first and last of these three conditions is met by setting O such that an integer number of lobes of the sinc function are contained within the function's ROS – setting $O = 4(N_b - 1)q + 1$, where $q \in \mathbb{N}$ is the number of lobes, satisfies this condition. Low values of q yield less selectivity, but a less jagged approximation to the fan shape. The second of the three conditions has already been met because the magnitude of $h_{LP}(n_u)$ is scaled by the appropriate factor.

The N_b sub-band signals, denoted as $L^k(\mathbf{n})$, are obtained through convolution of the input signal with the bandpass filters, as in

$$L^k(n_s, n_t, n_u - n_0, n_v) = L(\mathbf{n}) * h^k_{BP}(\mathbf{n}). \tag{6.14}$$

Note that the delay introduced by the bandpass filters is eliminated by shifting the output light field back along the u axis by n_0 samples. This is possible because the delay n_0 is always a known integer number of samples.

6.1.2 The Frequency-Hyperplanar Filters

With the input signal separated into N_b sub-bands, the next step is to filter each sub-band signal with the appropriately designed hyperplanar filter. Only the hyperplanar filters corresponding to the first of the two fan filter banks will be designed – extension to the second fan filter bank is straightforward.

Because the fan filter bank only has selectivity in Ω_s and Ω_u, the simplified hyperplanar filters from Section 5.2.3 can be used. Each of the hyperplanar filters

requires a normal and a bandwidth. Observing Figure 6.2, it is clear that the normals of all the hyperplanar filters should be identical, set such that each hyperplane is aligned with the central axis of the fan-shaped passband. This is accomplished by setting

$$\mathbf{d}^{su} = \frac{[1, 0, \tan(\theta_c), 0]}{\|1, 0, \tan(\theta_c), 0\|}. \tag{6.15}$$

The bandwidth of each hyperplanar filter depends on the difference of the slopes of the hyperplanes corresponding to z_{min} and z_{max}. There are many ways to calculate the bandwidth – one way is to approximate it as the distance, at the center of each sub-band along Ω_u, between the minimum- and maximum- depth hyperplanes, along Ω_s. The center of the k^{th} sub-band along Ω_u is found as $(k+0.5)/N_b$, and the distance between the hyperplanes along Ω_s for that sub-band is found geometrically as

$$B_k = \frac{N_u/D_u}{N_s/D_s} \frac{d(k+0.5)}{2N_b} \left[\frac{1}{z_{min}} - \frac{1}{z_{max}}\right] + c, \tag{6.16}$$

where the first term corrects for differing sample rates in s and u. The constant bandwidth offset c allows the passband to *surround* the fan in a bowtie shape, leaving room for some deviation from the ideal fan shape – such deviation might be caused by occlusions or specular reflections, for example.

6.1.3 Recombining the Sub-Bands

Each fan filter bank begins by separating the input signal into sub-band signals using N_b sub-band separation filters. Each sub-band signal is then filtered using a first-order recursive frequency-hyperplanar filter with an appropriately selected normal and bandwidth. The filtered sub-band signals are then recombined, as depicted in

Figure 6.1, through summation. This process can be expressed as

$$L_{out}(\mathbf{n}) = \sum_{k=0}^{N_b-1} L_{out}^{k}(\mathbf{n}), \tag{6.17}$$

where $L_{out}^{k}(\mathbf{n})$ is the filtered version of the k^{th} sub-band signal, and $L_{out}(\mathbf{n})$ is the overall output of each fan filter bank.

6.1.4 Intersecting two Fan Filters

The fan filter bank with selectivity in Ω_s and Ω_u is cascaded with a second fan filter bank, designed along the same lines as the first, except with selectivity in Ω_t and Ω_v. Each of the two filter banks has a fan-shaped passband, and the cascade of the two forms a dual-fan passband. The Laplace transform transfer function of the system can be expressed as

$$T(s_s, s_t, s_u, s_v) = T_{su}(s_s, s_t, s_u, s_v)T_{tv}(s_s, s_t, s_u, s_v), \tag{6.18}$$

where $T_{su}(\mathbf{s})$ and $T_{tv}(\mathbf{s})$ are the Laplace transform transfer functions corresponding to the two fan filter banks.

6.1.5 Zero-Phase Filtering

Each of the two fan filter banks can have its transient response significantly reduced using the zero-phase filtering technique described in the previous chapter. This is particularly relevant given that the bandwidths of the hyperplanar filters applied to the sub-bands nearest to the origin can be extremely narrow, resulting in long transient responses. By repeating, in a second pass, the two fan filter banks with their directions of iteration reversed, the total phase of the system is forced towards zero.

6.1.6 Implementation details

Four types of filter were implemented and incorporated into the Lightbench package: the DFFB which makes use of the original frequency-hyperplanar filters, the version which makes use of the simplified hyperplanar filters, and the zero-phase versions of both. Most of the implementation details match those for the frequency-planar filter from the previous chapter. The user interface accepts a minimum and maximum depth as input, along with a constant bandwidth offset, the number of sub-bands to use, and the number of lobes of the sinc function to use.

As with the frequency-planar filter, each colour channel of the input signal is operated on individually in order to minimize the memory requirements associated with intermediate signals. Because the two fan filter banks are significantly more complex than the frequency-planar filter, special care had to be taken to minimize computational complexity, and in particular memory requirements.

As with the frequency-planar filter, the colour channels are processed in sequence. In order to conserve memory, the sub-bands are also processed sequentially. In this scheme, each sub-band of the first colour channel is separated, filtered, and incorporated into an intermediate output signal prior to moving on to the next sub-band. This scheme requires only three floating-point arrays for storing intermediate values: one which stores each newly separated sub-band, one which is used by the hyperplanar filter in finding the corresponding filtered signal, and one into which the resulting filtered sub-band signal is added. Once the final sub-band has been added to the intermediate output array, it is converted to 8-bit integers and used to replace the first colour channel of the input light field, at which point the entire process

begins again for the second and third colour channels. With those colour channels completed, the second fan filter bank follows the same process with the resulting light field.

All the intermediate arrays in this scheme use a 32-bit floating-point representation, to reduce quantization error. The total memory occupied is $N_s N_t N_u (8+11N_v)$. Note that this is still quite a lot of memory – 706 MB for a light field with 32 samples in s and t and 256 samples in u and v. It should be possible to further reduce the memory requirements of this filter by pipelining the calculation process further – finding the values of small subsets of the intermediate output array, and incorporating them back into portions of the input light field that are no longer needed. This further pipelining of the DFFB was not implemented.

6.1.7 Results

The four types of filter were tested on a wide range of light fields, including some containing occlusions. Results were positive in all cases. Results will be shown here for a light field containing occlusions – the light field is shown in Figure 6.3, and has parameters as summarized in Table 6.1. The scene modelled by this light field contains the same elements as the one used in the previous chapter, but with the coaster mounted at approximately 45 degrees to the reference planes, causing it to occupy a range of depths from about 40 cm to 50 cm. The geometry of this light field was chosen to emphasize the coaster, and thus the effects of extracting a range of depths on the passband signal.

The DFFB and its zero-phase version were applied to the input to extract the coaster. The versions which utilized the simplified frequency-hyperplanar filters

Table 6.1: Light field parameters.

colour channels	3
s, t samples	32
u, v samples	256
s, t size (cm)	21
u, v size (cm)	15
separation d (cm)	45

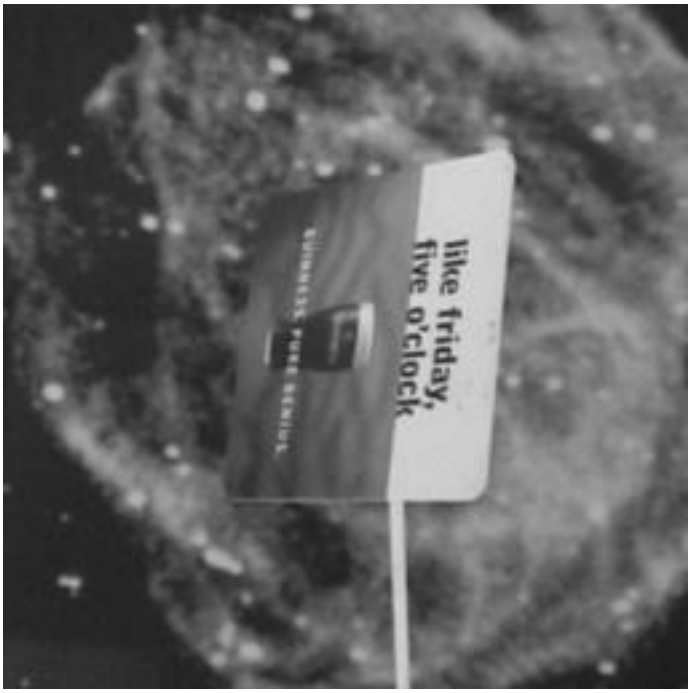
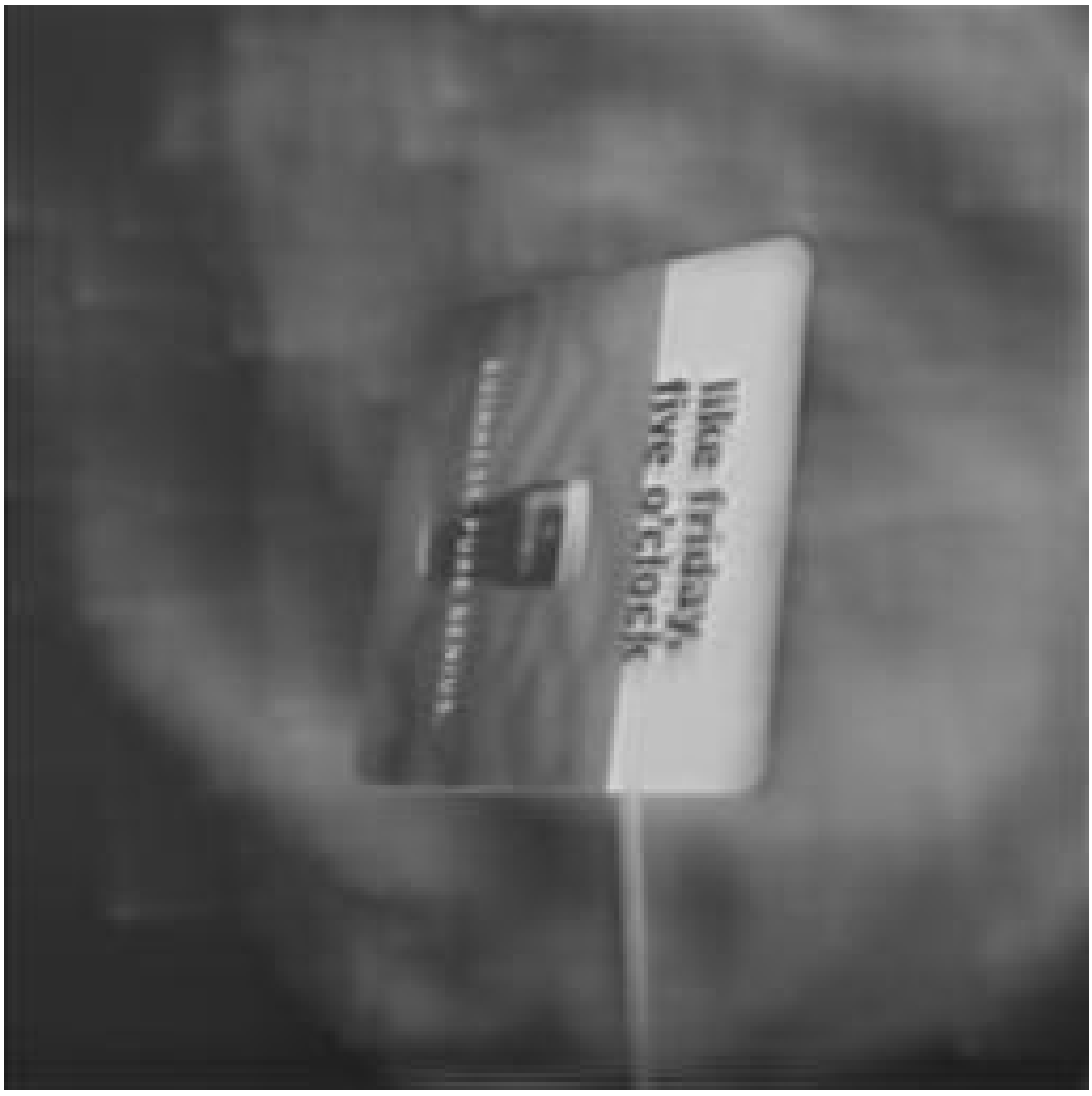


Figure 6.3: Input light field modelling a scene containing a beer mat, mounted at 45°, and a poster.

yielded identical results to those which did not, and so only the results for the simplified version will be shown. The DFFB was set to extract objects in the range 40 cm to 50 cm using four sub-bands, first without a bandwidth offset, and then with a bandwidth offset of $c = 0.05$. In all cases, only the first lobe of the sinc function was used by the sub-band separation filters – i.e. $q = 1$.

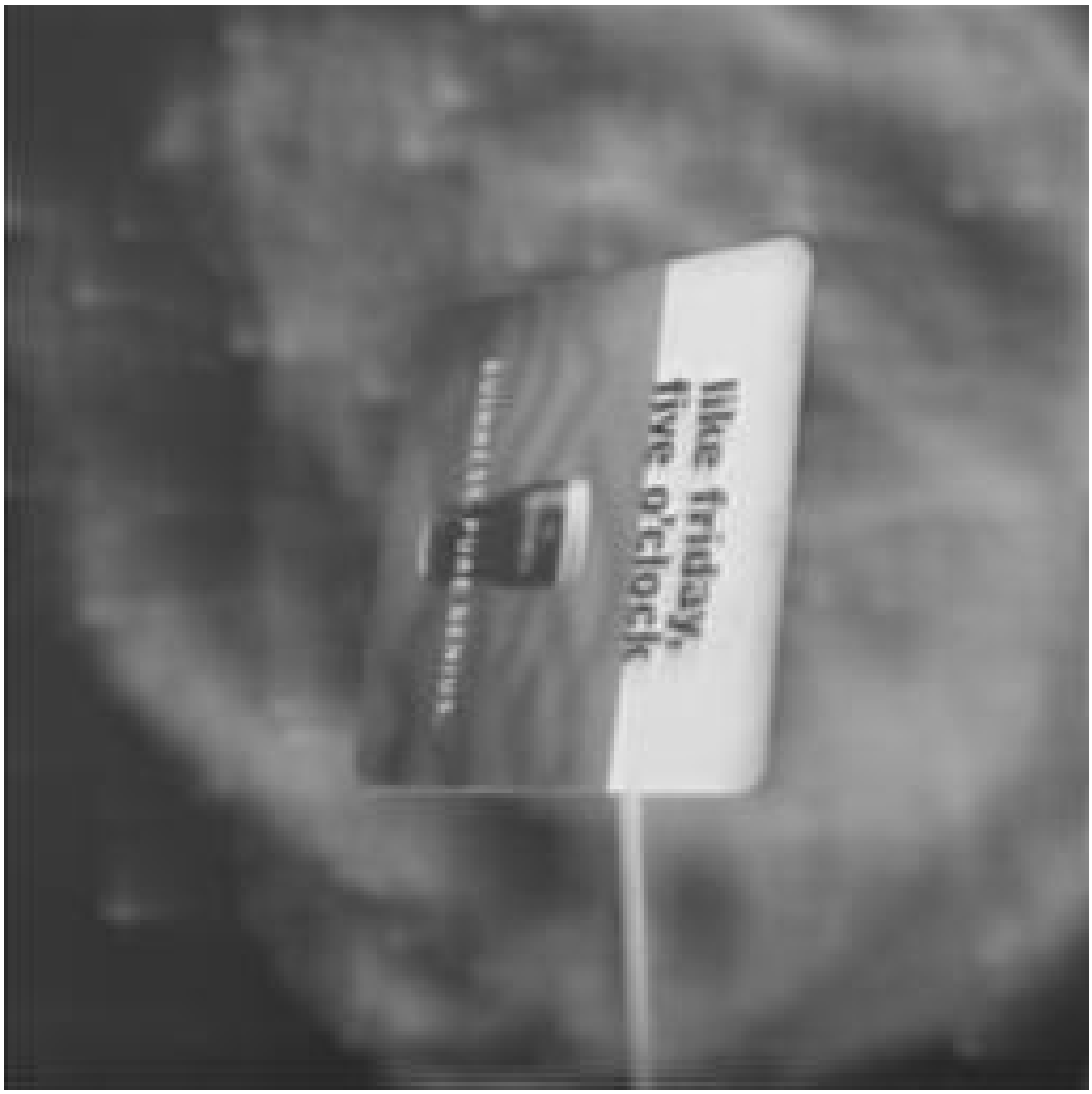
As a reference, the zero-phase frequency-planar filter designed in the previous chapter was also applied to this input light field, with a depth centered on the coaster, and with bandwidths of 0.3 and 0.1 – note that a bandwidth of 0.3 corresponds to the best-fit of the frequency-planar filter to the dual-fan passband.

The results of these operations can be seen in Figures 6.4 and 6.5. All the filters successfully attenuated the background, though clearly the zero-phase filters displayed more selectivity and a shorter transient response, as evidenced by an increase in the blurring of the stop-band signal, and a decrease in the darkening near the light field edges.



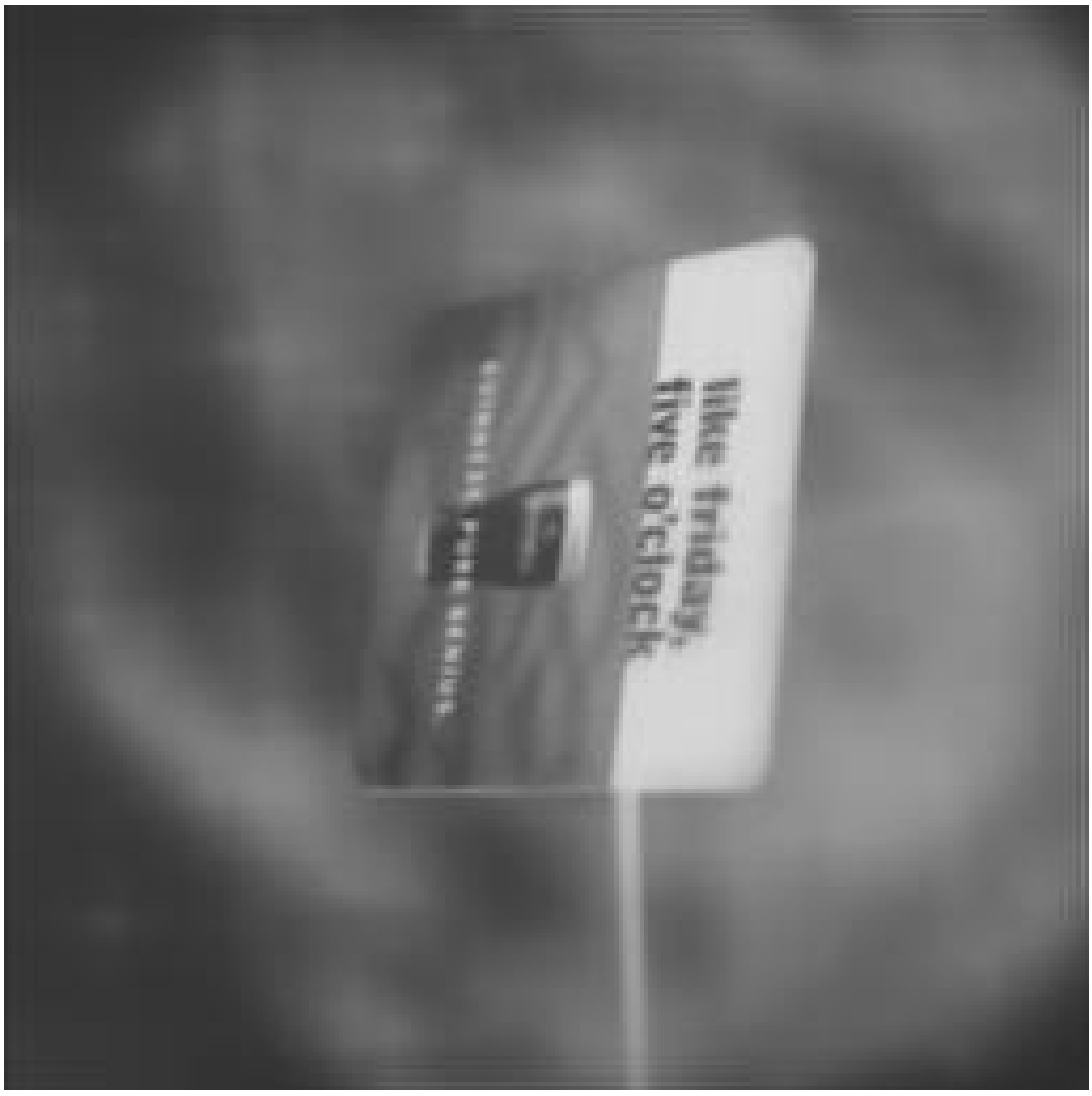
(a)

Figure 6.4: Results of applying the DFFB with offsets, c, of a) 0.00 and b) 0.05; the zero-phase DFFB with offsets of c) 0.00 and d) 0.05.



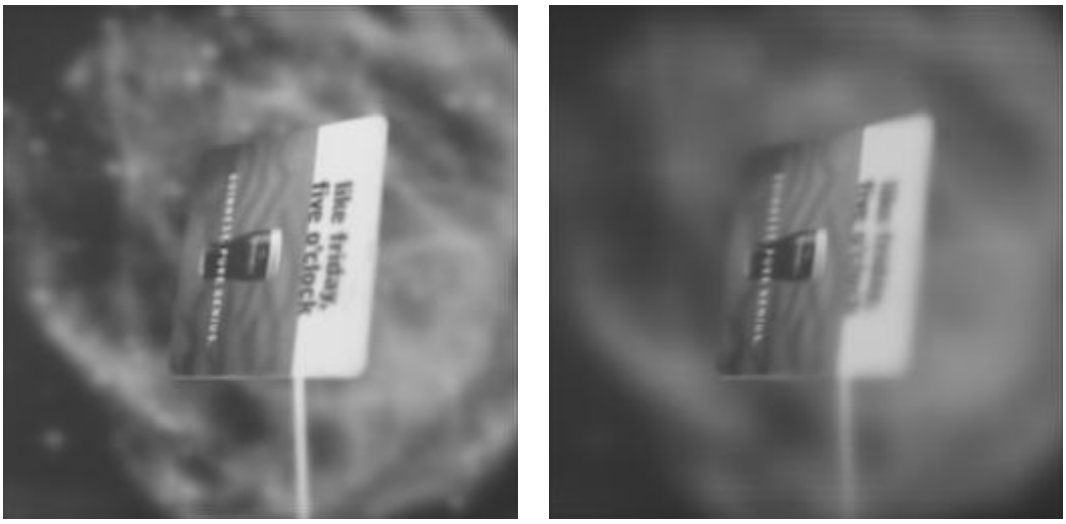
(b)

(c)



(d)

The filters with the constant bandwidth offset showed improved performance near the extremities of the coaster. This may be due to deviation from the ideal dual-fan frequency-domain ROS associated with the occlusions near these locations. Likely more prominent is the warping of the passband associated with the bilinear transformation [31] [32]. Though less sensitive to these effects, the filters with a



(a) (b)

Figure 6.5: Results of frequency-planar filtering, with bandwidths of a) 0.3 and b) 0.1.

constant bandwidth offset had the disadvantage of lower stop-band attenuation.

It can be seen in Figure 6.5 that the frequency-planar filters either caused less blurring of the poster, as in a), or more blurring of the coaster, as in b), than the DFFB. Clearly, the DFFB represents a significant improvement over the planar filter for extracting objects occupying a range of depths.

Chapter 7

Estimating Shape

Previous chapters have focussed on filtering light fields to extract objects based on their depth. This chapter moves away from filtering light fields and towards the analysis of a light field's contents. Perhaps the most fundamental piece of information one might want to know about a light field is the geometry of the scene that it models. The goal of this chapter, then, is to introduce methods for estimating the geometry of a scene from its light field representation.

Uniqueness of Shape

A recently proposed theorem of 3D vision [35] states that the only circumstance under which the geometry of a scene is *inherently* ambiguous is the one in which extended regions of constant value exist. The theorem deals only with Lambertian scenes, and the ambiguity to which it refers is deemed to be *inherent* because the assumption is made that the entire plenoptic function is available – that is, the ambiguity does not arise from an inadequate sampling or parameterization of the plenoptic function.

The techniques developed in this chapter work within this limitation by assuming the absence of extended regions of constant value in the scene. The further assumption of a Lambertian scene is made, with the understanding that specular reflections can sometimes fool even the human visual system, and so attempting to deal with them is beyond the scope of this thesis. Generalization of the techniques derived

here to work with scenes with specular reflections would require an added level of intelligence, designed to ignore such reflections.

7.1 Plane Variance Minimization

The point-plane correspondence derived in Chapter 4 states that a point in space corresponds to a plane in the light field, where the orientation of the plane depends only on the depth of the point in the scene. In the case of a Lambertian scene with no occlusions, the further observation was made that the plane corresponding to an infinitesimally small surface element has a constant value. The plane variance minimization technique exploits these two observations by finding the plane, corresponding to each point in space, which comes closest to having a constant value.

Making the assumption of no occlusions, the shape of the scene can be expressed as a 2D map $A(a_x, a_y)$ of depths. For a map with extents given by $[D'_x, D'_y]$ and containing I_x by I_y samples, the point in space, $\mathbf{p}$, corresponding to the index $\mathbf{a}$, must have the x and y coordinates given by Equation (5.2).

The remaining coordinate, p_z, is most simply found through an iterative process. Assuming the scene lies within some depth range z_{min} to z_{max}, one might iterate through N_z candidate depths which lie between z_{min} and z_{max}. For each candidate depth, the point $\mathbf{p}$ corresponding to each sample in the map $A(\mathbf{a})$ is uniquely defined, and so the plane to which it corresponds can be found using the point-plane correspondence.

Following the approach described in Chapter 5 for plane averaging, the light field samples corresponding to the plane can be found by iterating through n_u and n_v,

finding the corresponding values of n_s and n_t using Equations (4.3) and (3.1). The interpolated light field values should be equal across the plane for the correct value of p_z. The variance of the values in the plane can be used as an error estimate – a higher variance corresponds to more deviation from the ideally constant-valued plane. The process of finding the error, $\varepsilon(\mathbf{p})$, can be expressed as

$$\varepsilon(\mathbf{p}) = \frac{\sum_{n_u=0}^{N_u-1} \sum_{n_v=0}^{N_v-1} \| \tilde{L}_{cont}(s,t,u,v) - \bar{L}_{P_p} \|^2 \Big|_{[s,t,u,v] \in P_p}}{N_u N_v}, \tag{7.1}$$

where $\bar{L}_{P_p}$ is the mean value of the samples in the plane, found using the plane averaging technique from Chapter 5.

Theoretically, any error minimization scheme might be used to minimize the error associated with the depth of a point, though the simplest to implement is the scheme which simply iterates through the N_z candidate values of p_z, finding the lowest corresponding error. Because the error function grows quickly to a point of saturation as p_z deviates from the correct depth, error minimization schemes are unlikely to succeed unless given a very close initial approximation to p_z. A good approach to finding the plane of minimum variance, then, is to initially iterate through N_z candidate depths, finding the depth corresponding to the minimum error, and using this depth as an initial estimate in some error minimization scheme.

Note that plane variance minimization should function in the presence of some specular reflection, because the lowest plane variance will always correspond to the correct depth if the energy contained in diffuse reflections outweighs the energy in specular reflections. Note, however, that occlusions are not treated correctly by this scheme because only one depth is estimated for each x, y coordinate, disallowing the representation of both occluding and occluded surfaces.

Improvements

Plane variance minimization can be sensitive to aliasing and noise, and so a method for improving its performance is desirable. By finding the mean of the variance corresponding to a neighbourhood of points surrounding each point, $\mathbf{p}$, and using that as an error estimate, the sensitivity of the technique to noise is reduced.

7.2 Gradient-Based Depth Estimation

Plane variance minimization is extremely time-consuming when the scene is not guaranteed to lie within a relatively narrow range of depths, requiring a large number of candidate depths to be tested. Gradient-based depth estimation attempts to determine the orientations of the planes more directly, by exploiting their constant value under the assumption of a Lambertian scene – the use of gradients to estimate depth is mentioned briefly in [35].

The orientation of a plane is difficult to estimate in 4D. In the general case, two 4D vectors are required to do so. Thankfully, the planes in a light field are constrained to have the same orientation in the s, u directions as in the t, v directions. This observation will allow the use of 2D techniques in estimating a plane's orientation, and it will introduce redundancy which can be used to validate the results. Estimating the shape of a scene begins by finding the 2D gradient vectors corresponding to each sample in the light field. Those gradient vectors are then used to estimate the orientation of the plane passing through each sample, and thus the depth of the corresponding scene element.

This method differs from plane variance minimization in that it forms an estimate

of depth for every light field sample, based on the properties of the light field in the vicinity of the sample. Because a depth estimate is made for every light field sample, this technique is more suitable for scenes with occlusions, as both occluded and occluding surfaces are represented in the output.

Observing an s, u slice of a typical light field, such as the one shown in Section 4.1, it is clear that a 2D gradient operator, applied at some point in the slice, will yield a gradient vector which points orthogonal to the plane passing through that point. The basic approach associated with gradient-based depth estimation, then, is to use a 2D gradient operator in the s and u dimensions to estimate the orientation of the plane passing through each light field sample. Gradient operators applied in t and v should yield the same results, introducing a redundancy of a factor of two. In fact, because the 2D gradient operator can be applied to each colour channel of the light field independently, a total redundancy of a factor of six is introduced.

The 2D gradient operator, applied to a single colour channel in the s and u directions, is defined as

$$\nabla^{su} L(\mathbf{n}) = \left[\frac{\partial L(\mathbf{n})}{\partial s}, \frac{\partial L(\mathbf{n})}{\partial u} \right]. \tag{7.2}$$

The values of the partial derivatives can be estimated using a number of methods. One of the simpler methods, which has a relatively high immunity to noise, utilizes 2D convolution in s and u, as in

$$\frac{\partial L(\mathbf{n})}{\partial s} \simeq L(n_s, n_u) * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \tag{7.3}$$

and

$$\frac{\partial L(\mathbf{n})}{\partial u} \simeq L(n_s, n_u) * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}. \tag{7.4}$$

Extension of both the gradient operator and partial derivatives to operate on slices in t and v is trivial.

Knowing the gradient vector, the slope of the plane passing through each sample in the light field can be found, and from this slope the depth of the corresponding point in the scene can be found. From the point-plane correspondence, the slope of the plane corresponding to a point in the scene is given, in the s and u directions, by

$$m_{plane} = 1 - \frac{d}{p_z}. \tag{7.5}$$

The direction of the gradient vector can also be expressed as a slope, as in

$$m_{\nabla} = \frac{\nabla_u^{su} L(\mathbf{n})}{\nabla_s^{su} L(\mathbf{n})} = \frac{\partial L(\mathbf{n})/\partial u}{\partial L(\mathbf{n})/\partial s}. \tag{7.6}$$

Assuming no regions of constant value, for which the magnitude of the gradient vector is zero, the gradient vector will always point orthogonal to the plane. This means that the relationship between the slope of the plane and the slope of the gradient vector can be expressed as $m_{plane} = -m_{\nabla}^{-1}$. Rearranging to solve for p_z, and generalizing for differing sample rates in the four dimensions, yields the equation

$$p_z = \frac{d}{1 + \frac{\partial L(\mathbf{n})/\partial s}{\partial L(\mathbf{n})/\partial u} \frac{N_s/D_s}{N_u/D_u}}, \tag{7.7}$$

which is easily generalized to the t and v dimensions.

Estimating the shape of a scene using this method, then, begins by finding the partial derivative estimates, defined by Equations (7.3) and (7.4), corresponding to each sample in the light field. Those derivatives are then used with Equation (7.7) to estimate the depth of the scene element corresponding to that light field sample.

Note that throughout this process, only the direction of the gradient vector is used. The magnitude of that vector, $\|\nabla\|$, is essentially an indication of the contrast

in the light field at each sample. Areas with low contrast will have short gradient vectors, whereas areas of high contrast will have long gradient vectors. It is well known that high-contrast areas give more information about a scene's geometry than low-contrast areas – the most extreme case being an extended area of constant value, which represents no information about the scene's geometry. Because of this observation, the magnitude of the gradient vector may be used as an indication of the certainty of the corresponding depth estimate.

Because of the redundancy associated with having three independent colour channels, with two independent depth estimates per channel, some method of optimally combining the estimates is in order. The magnitude of the gradient is an indication of certainty, and so the six depth estimates may be weighted with the magnitudes of the corresponding gradients. This can be expressed mathematically as

$$\bar{p}_z = \frac{\sum_{i=0}^{5} p_{z,i} \|\nabla_{\mathbf{i}}\|}{\sum_{i=0}^{5} \|\nabla_{\mathbf{i}}\|}, \tag{7.8}$$

where the i^{th} gradient and depth estimate correspond to one of the six unique combinations of colour channel and direction. Because the denominator of this equation is an indication of overall certainty, it can be used as a threshold – allowing depth estimates with inadequate support to be ignored altogether.

7.3 Feature Tracking-Based Depth Estimation

In classical computer vision techniques, scene geometry is estimated from a very sparse sampling of the plenoptic function, corresponding to as few as two camera positions, rather than the $N_s \times N_t$ camera positions associated with light field-based

techniques. The basic approach of the classical techniques is to find corresponding features in the available images of the scene, computing depth based on the relative positions of the features in the images. In the simplest possible setup, two cameras are used, and the process of matching scene features across the two resulting images is called stereo matching [36].

Because the light field can be seen as consisting of many pairs of stereo images – an array of $N_s \times N_t$ such images, to be exact – stereo matching can be applied to a light field. The only difference between a light field slice in u and v and a camera image of the scene is the offset which forces the same section of the u, v reference plane to be represented in each slice. By accounting for this offset, any classical computer vision algorithm can be applied to a light field's contents.

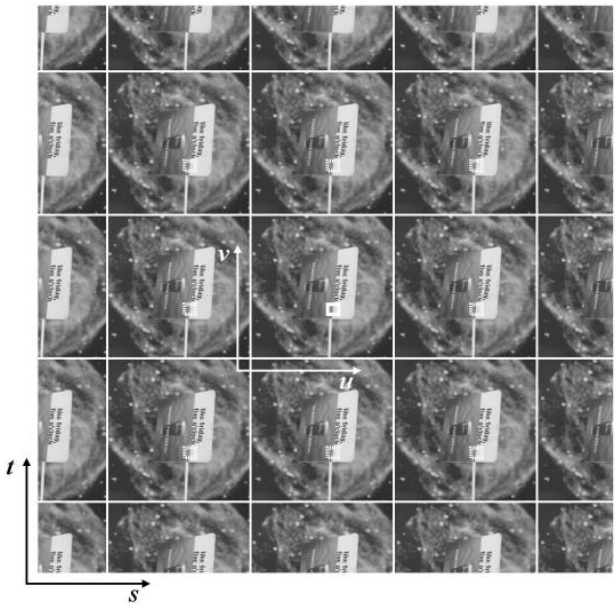
From the array of light field images, there are $N_s N_t (N_s N_t - 1)/2$ ways to choose two images – this corresponds to 523776 ways in the case of a light field with 32 samples in s and t. This means that a classical computer vision technique will either use only a small subset of the information available in the light field, or it will generate a huge amount of information which is extremely difficult to consolidate. As a result, most computer vision approaches need to be modified somewhat in order to yield more optimal results with light fields.

Feature tracking-based depth estimation is an extension of stereo matching which follows features through several u, v slices of the light field, rather than working on pairs of images. For the sake of clarity, the slices of the light field in u and v will be referred to as frames in the context of feature tracking-based depth estimation. This technique resembles the gradient-based method in that it estimates a depth for every sample in the light field.

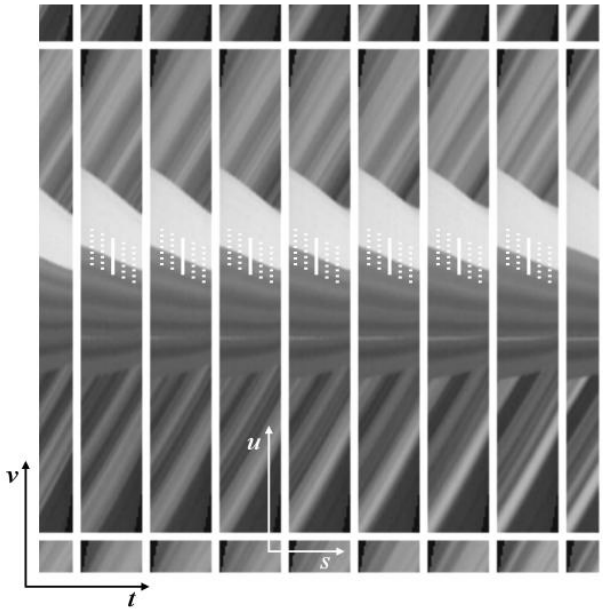
Feature tracking works on neighbourhoods of samples in the u, v frames. For each sample in the light field, the surrounding neighbourhood is tracked through s and t. By tracking the neighbourhood through all the u, v frames – that is, through all values of n_s and n_t – all the information available in the light field is used. This process is represented in Figure 7.1, as slices in u, v, and as slices in s, u. The solid lines represent the neighbourhood to be tracked, while the dotted lines represent the resulting tracked locations along s and u.

For each frame into which the neighbourhood is tracked, a new depth estimate may be calculated. The mean of the resulting estimates is the final depth estimate for the sample at the center of the original neighbourhood. Tracking a feature through all the u, v frames would be excessively time-consuming, and so a simplified approach might track the neighbourhood through a subset of the available u, v frames. A simple subset of u, v frames that exploits the full dimensionality of the light field, while yielding simple calculations, is the one that tracks a neighbourhood of samples for some number of frames, N_i, along s, and then for some number of frames, N_j, along t. This subset of frames is shown surrounding the center u, v frame in Figure 7.2. In this figure, $N_i = N_j = 4$ – note that the 0^{th} frame is ignored, because it is the original neighbourhood and so does not require tracking.

In this scheme, frames are only ever analyzed along one dimension at a time, and so the resulting calculations are extremely simple. Only the depth estimates found by iterating through the frames in the s direction will be derived here, with generalization to operate in the t direction being trivial. Each sample in the light field belongs to a plane with a slope, in s and u, given by Equation (7.5). This slope can be used with the tracked neighbourhoods to form a set of depth estimates.



(a)



(b)

Figure 7.1: Feature tracking, visualized as slices in a) u, v, and b) s, u.

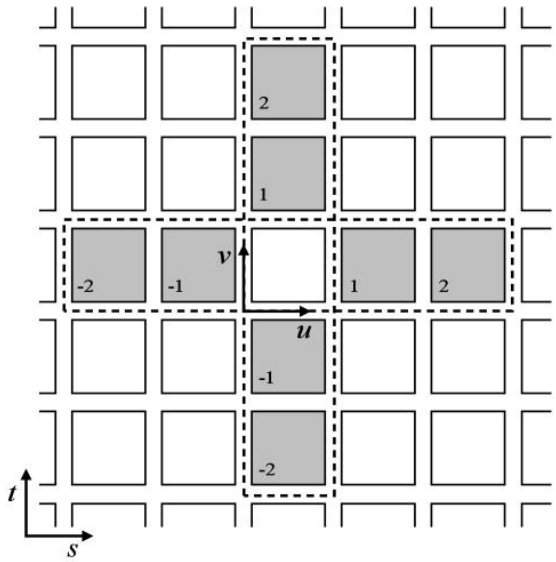


Figure 7.2: An optimized frame subset for feature tracking.

Because movement along s will only yield a change in the neighbourhood's position along u, the search for a corresponding neighbourhood only needs to operate along that dimension. If the original neighbourhood is centered on a sample at $\mathbf{n_0} = [n_{0_s}, n_{0_t}, n_{0_u}, n_{0_v}]$, for example, then the position of the neighbourhood in the adjoining frame along s is given by $[n_{0_s} + 1, n_{0_t}, n_{0_u} + m_1, n_{0_v}]$, where m_1 is the distance, in samples along u, that the neighbourhood has moved between the original frame and the first frame along s.

The values of m_i over the N_i frames for which the neighbourhood is tracked are most easily found by iterating through a search range in u, for each frame along s, minimizing an error function. The matched position in each new frame is that position within the search range for which the mean squared error (MSE) between the original neighbourhood, and the new candidate neighbourhood, is a minimum.

This can be seen as the minimization of the error function given by

$$\varepsilon(m_i) = \frac{\sqrt{\sum_{k=-I_u/2}^{I_u/2} \sum_{l=-I_v/2}^{I_v/2} \|L(n_{0_s}+i, n_{0_t}, n_{0_u}+k+m_i, n_{0_v}+l) - L(n_{0_s}, n_{0_t}, n_{0_u}+k, n_{0_v}+l)\|^2}}{(I_u+1)(I_v+1)}, \tag{7.9}$$

where $(I_u + 1)$ and $(I_v + 1)$ define the size of the neighbourhood along u and v, respectively.

N_i different values of m_i are found using this technique – note that $i = 0$ is excluded from this count because it corresponds to $\mathbf{n_0}$, the original neighbourhood position. For each value of m_i, the corresponding depth estimate is found using the relationship between m_i and the slope of the plane in s and u, as in

$$m_{plane} = 1 - \frac{d}{p_{z,i}} = \frac{m_i}{i}. \tag{7.10}$$

Solving this equation for p_z, and introducing terms to correct for differing sample rates in the four dimensions, yields the equation

$$p_{z,i} = \frac{d}{1 - \frac{m_i}{i}\frac{N_s/D_s}{N_u/D_u}}. \tag{7.11}$$

Note the similarity to Equation (7.7) – the only difference is the way in which the slope of the plane is being estimated.

Having found N_i depth estimates along s, and following the simple extension to find N_j additional estimates along t, a total of $N_i + N_j$ depth estimates are formed. The mean of these estimates forms the overall depth estimate for each sample in the light field. Alternatively, a weighting scheme similar to that used with the gradient-based method – as in Equation (7.8) – could be used. The certainty associated with each estimate is most simply found in this case as the inverse of the error $\varepsilon(m_i)$ or

$\varepsilon(m_j)$ corresponding to each estimate. Alternatively, the magnitude of the gradient operator, applied to the center of the original neighbourhood, might be used. These approaches again have the advantage of allowing a thresholding operation, in which regions of low certainty are ignored.

7.4 Results

The three shape estimation techniques discussed in this chapter were implemented and incorporated into the Lightbench package. The techniques were tested on a down-sampled version of the light field used in Chapter 6 to test the DFFB. That light field was reduced to 128 samples in u and v, maintaining 32 samples in s and t, to allow the algorithms to run more quickly, and to reduce the regions of constant value. An image of this light field is shown for reference in Figure 7.3.

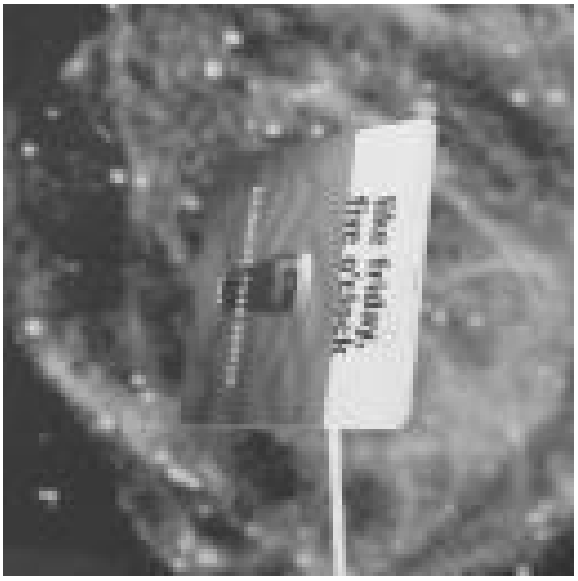
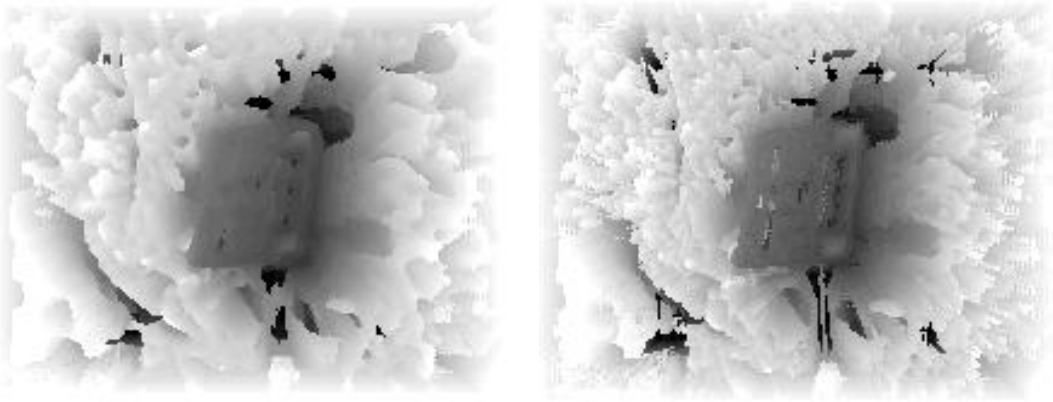


Figure 7.3: Image of the light field used to test the shape estimation algorithms.

Plane Variance Minimization

Plane variance minimization was implemented without an additional error minimization scheme – only the initial iterative depth search was carried out. The neighbourhood-based improvement was implemented, and the technique was tested both with and without this improvement. A depth range of 30 cm $\geq p_z \geq$ 70 cm was assumed, and $N_z = 40$ candidate depths were tested for each point. The map $A(a)$ had 256 samples in each direction. The results can be seen in Figure 7.4. This figure visualizes the depth estimates as intensity – pure black represents a depth of 30 cm, while pure white represents a depth of 70 cm. A neighbourhood size of 4 was used with the neighbourhood-based improvement. As is evident from the improved quality of the corresponding figure, the use of neighbourhoods reduced the sensitivity of the plane variance technique to noise and aliasing.

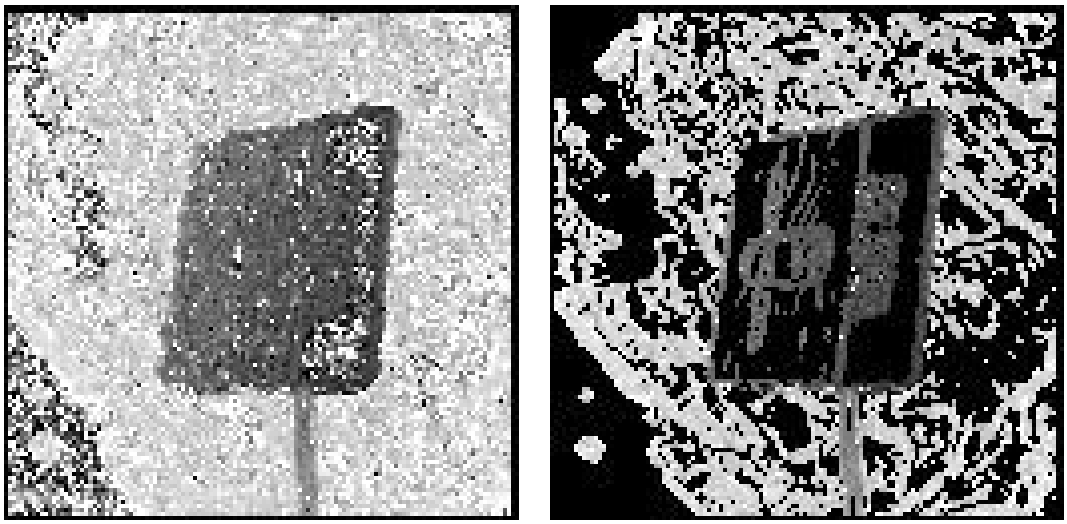


(a) (b)

Figure 7.4: Results of plane variance minimization a) with and b) without the neighbourhood-based improvement.

Gradient-Based Depth Estimation

The results of gradient-based depth estimation are shown in Figure 7.5. The poor estimates associated with areas of constant value can be ignored through thresholding based on certainty. This technique was applied, with a threshold of 1.0 – that is, the average magnitude of the gradient vectors had to exceed 1.0 before the corresponding depth estimate was used. Areas where this threshold was not met were assigned a value of zero, for easy identification. The results of the thresholding are shown in Figure 7.5b) – regions that have been ignored due to thresholding appear black in this figure. Clearly the thresholded estimate is more desirable, having the advantage of allowing regions of low certainty to be ignored.

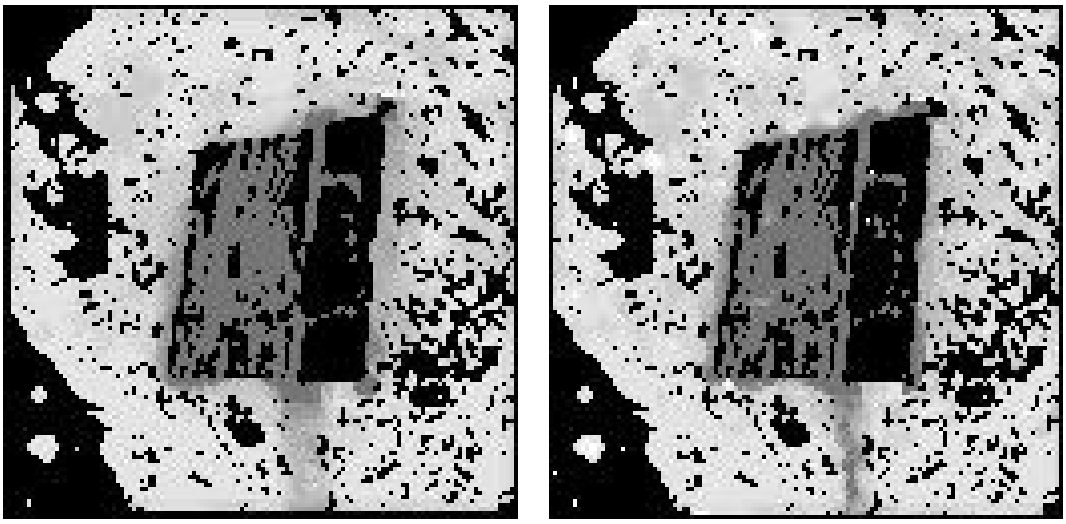


(a) (b)

Figure 7.5: Results of gradient-based depth estimation a) without thresholding, and b) with thresholding.

Feature Tracking-Based Depth Estimation

Feature tracking-based depth estimation yielded the results shown in Figure 7.6 – a) and b) correspond to neighbourhood sizes of 9 and 5 samples, respectively. In both cases, a search window of 17 samples along s and t was used, and a gradient threshold of 0.5 was applied – that is, the magnitude of the gradient operator applied at the center of the neighbourhood had to exceed 0.5 in order for the result to be utilized. Features were tracked through a total of $N_i = 9$ frames along s and $N_j = 9$ frames along t. Both versions of this algorithm produced reasonable results. Regions that have been ignored due to thresholding appear black in the figures.



(a) (b)

Figure 7.6: Results of feature tracking-based depth estimation with a neighbourhood size of a) 9 and b) 5.

All three of the techniques described here produced useful results. The plane variance minimization technique took the most time, requiring 9 minutes to generate a 2D map of depths. The gradient-based method was the fastest, taking about 35 seconds in total. Because it generated an estimate for every light field sample, rather

than generating a 2D map of estimates, it would be more equitable to compare the speed of the gradient-based method applied to a single 2D slice of the light field. This resulted in a running time of about 35 ms. The feature tracking method rated somewhere between the two, taking about 30 seconds per 2D slice.

7.5 Improvements

Based on both quality of results, and processing time required, the gradient-based method seems to be the strongest of the three methods described here. Although the results that it generates look noisy, they may easily be enhanced through the use of a 4D lowpass filter. This is particularly appropriate, as it goes some way towards consolidating the large amount of information associated with having a depth estimate for every light field sample.

The depth estimates generated by the gradient-based method might be further enhanced by filling the gaps which are a result of the thresholding operation. This can be carried out using a simple region growing algorithm. By iteratively assigning each empty sample the average value of all of its nearest neighbours along s, t, u and v, for example, the gaps in the estimates can be filled.

Applying the region growing algorithm, followed by a simple 4D moving average filter, to the output of the gradient-based depth estimation yields the result shown in Figure 7.7 – in this case, a moving average window size of 4 samples was used. These two techniques have significantly improved the output.

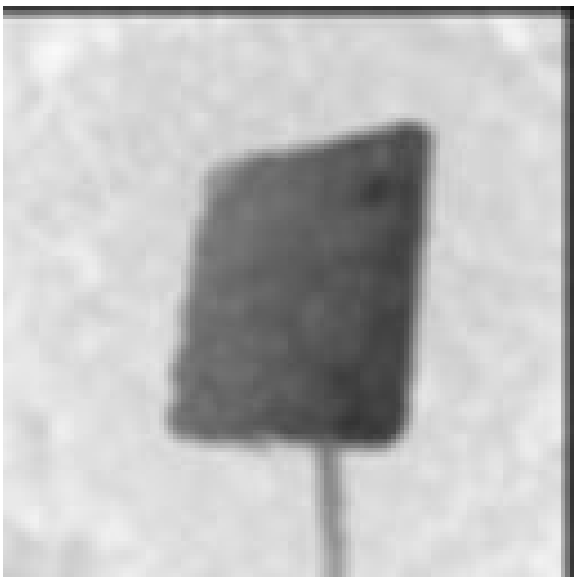


Figure 7.7: Results of applying region growing and a 4D moving average filter to the output of gradient-based depth estimation.

Chapter 8

Conclusions and Directions for Future Work

This work has explored a variety of techniques for performing useful tasks on visual data. Employing a light field representation of the light permeating a scene has raised the possibility of performing complex tasks with simple techniques. The frequency-planar filter, DFFB, plane averaging filter, and depth estimation techniques all exploit the characteristics of the light field representation to produce useful results with a minimal amount of algorithmic complexity.

The frequency-planar filter, DFFB, and gradient-based depth estimation technique are all particularly simple, in that they all apply the same, simple rules throughout the light field to generate a useful result. This makes them robust as well as making them good candidates for techniques such as parallelization and pipelining.

Plane variance minimization and feature tracking-based depth estimation, on the other hand, are comparatively complex algorithms. They make complex decisions based on the minimization of error functions. Although these algorithms produce useful results, their use of complex decision making processes makes them less robust than simpler techniques such as the gradient-based method. The implication is that unless a complex algorithm accounts for all possible eventualities, simpler, more robust algorithms will yield superior results.

8.1 Real-Time Applications

Hardware Considerations

This thesis contains some simple techniques for performing tasks in computer vision. Filtering light fields to extract objects at a single depth, or over a range of depths, and estimating the geometry of a scene are all examples of techniques which might find use in real-world applications. The main point which has not been addressed in this thesis is how to incorporate these techniques into a useful system. A first-order 4D filter is the simplest kind of 4D filter, but how does one go about using one in a real-world application?

The obvious stumbling block comes about with the inherent size associated with light fields. A real-time light-field processing system will require a large amount of memory. The first step towards implementing a useful system, then, is establishing exactly how many light field samples are required, based on the complexity of the scene. Throughout this thesis, light fields with 256 samples in u and v, and 32 samples in s and t were utilized. It is possible to model many scenes with fewer samples than this, though it seems doubtful that any real-world application could operate with fewer than 128 samples in u and v, and 16 in s and t – note that even this reduction of a factor of 2 in each dimension corresponds to a total reduction of a factor of 16 in the size of the light field.

A further reduction of the memory required by a light field can be effected by utilizing a monochrome light field, rather than three colour channels, resulting in a memory savings of a factor of three. The amount of memory required by a monochrome light field with 128 samples in u and v, and 16 in s and t is only 4 MB, compared

with the 192 MB required by the light fields used throughout this thesis. Even at the full size of $256 \times 256 \times 32 \times 32$ samples, a monochrome light field occupies only 64 MB of memory. With the ever-decreasing cost of memory, even simple embedded systems can be found with more than enough memory to deal with light fields.

The next stumbling block in implementing a real-world application is processing time. The simplest linear filter presented in this thesis took 6 minutes to run on a Pentium IV running at 1 GHz. The code used to perform the filtering was not heavily optimized, however, and a general-purpose processor such as the Pentium does not represent the optimal architecture for the implementation of this kind of filter. Through the use of parallelization or pipelining, the filters described here might be implemented for operation in real-time in an ASIC (application-specific integrated circuit) or FPGA (field-programmable gate array), for example. A further optimization, through the use of alternative architectures, such as wave-digital [37], or systolic architectures, might yield even faster implementations.

Having addressed the memory and processing-time issues, the remaining hardware issue is the input to the system. Real-time light field processing can be accomplished using a camera array, such as that described in Chapter 3 – with the decreasing cost of digital camera technology, this kind of array is becoming accessible to a much broader market. The alternative to using a camera array is to use a camera gantry, or a moving video camera. One might expect this to prevent the application from performing in real-time, though techniques in dynamic light field construction from moving camera sequences are well-established [16] [17] [18] [19] [20]. In this type of scheme, a light field is constructed as the camera images become available. Once an adequately complete light field has been constructed, the

techniques described in this thesis can be applied.

Applications

Possibly the most obvious potential application for real-time light field processing is in robot navigation [38]. By constructing a light field on-the-fly using one or more onboard cameras, a robot might use real-time depth estimates to navigate. It might also use depth-based filtering to isolate scene elements of relevance – e.g. to extract objects that are close, and are therefore an immediate hazard. The DFFB is especially well-suited to this application, as it allows a range of depths – say, the three feet immediately in front of the robot – to be extracted.

Another potential application is in object or face recognition [39] [40]. In this kind of application, the object or face to be recognized often passes along a known path, or within a fixed space – this is the case in an airport security check, for example. The DFFB could be applied, in this scenario, to extract the region of interest from the rest of the scene, simplifying the recognition task. The depth estimation techniques from Chapter 7 might also be applied, and the resulting depth estimate used as the basis for scene, object or face recognition.

The most general application of light field processing is arguably in scene modelling [20]. As discussed in Chapter 3, a single camera may be moved through a scene, or a multiple-camera array placed in the scene, to form a light field model. From that light field model, geometric, lighting, and surface models of the scene can be formed. Because the light field contains a wealth of visual information associated with the scene, the resulting models can be extremely accurate. The most obvious application of the techniques described here to scene modelling is in the depth esti-

mation techniques introduced in Chapter 7, from which the geometry of the scene can be derived. A less obvious application might be in applying the planar depth filters over a range of distances, and using edge detection to effect depth-from-focus [27] to obtain a geometric model of the scene. The depth filters might also be used to segment the scene or to remove specular reflections.

8.2 Future Work

Hopefully this thesis has laid the groundwork for a novel approach to computer vision, placing an emphasis on simple techniques, and opening up a wide range of questions for further investigation. Some of the specific challenges that come out of the work presented here are as follows.

In Chapter 6, the observation was made that most of the blurring associated with high-frequency components in the passband of the DFFB likely comes about as a result of the warping introduced by the bilinear transformation. In an approach similar to that taken in [33] and [32], each of the sub-band filters might be modified to correct for this warping, yielding a more optimally-shaped passband.

The use of multi-rate filtering [41] might improve the speed of the DFFB presented in Chapter 6. Because the sub-bands of the light field signal can be re-sampled at a lower rate without loss of information, lower-rate filters can be implemented to process each of them, resulting in an implementation which utilizes less memory and requires less processing time.

Though the DFFB was introduced in this work as a means of extracting objects from scenes, it might find another use as an anti-aliasing filter. Because the overall

frequency-domain ROS of a scene is described by the dual-fan shape, the DFFB is the optimally shaped anti-aliasing filter. This observation might allow the storage and transmission of light fields at significantly lower sampling rates.

The effects of specular reflections and occlusions were briefly mentioned in Chapter 4. A more in-depth analysis of the effects that these phenomena have on a light field, both in the spatial and frequency domains, might yield useful information. For example, a new passband shape designed specifically to deal with specular reflections might be devised, or the depth estimation schemes from Chapter 7 might be optimized for operation in the presence of occlusions.

The gradient-based and feature tracking-based depth estimation techniques from Chapter 7 both generated a depth estimate for every light field sample. This represents a significant amount of redundancy, and some way of consolidating the results into a single 3D model might be interesting. The hybridization of the depth-estimation techniques presented here with voxel-based techniques [42], which directly generate 3D models, might also be interesting.

Finally, the techniques described here were all designed to work with the two-plane-parameterized, single-light slab light field. Generalization of these techniques to other forms of light field, including multiple light slabs, and perhaps even freeform light fields, may be possible.

Appendix A: Quadrilinear Interpolation

In bilinear interpolation, the value of a 2D map $A(\mathbf{a})$ corresponding to a 2D point $\mathbf{p}$ is estimated from the four samples nearest to that point. Assuming the point $\mathbf{p}$ corresponds to a continuous-domain index given by $\mathbf{a}'$, and the nearest four samples are indexed by the discrete indices $\mathbf{a}^0$, $\mathbf{a}^1$, $\mathbf{a}^2$ and $\mathbf{a}^3$, then the interpolated value $\tilde{A}(\mathbf{a}')$ depends on the distances along x and y given by $d_x^i = |a'_x - a_x^i|$ and $d_y^i = |a'_y - a_y^i|$ – note that the maximum value of each of these distances is unity, because $\mathbf{a}^i$ are the nearest four samples to $\mathbf{a}'$. The interpolated value is found as

$$\tilde{A}(\mathbf{a}') = \sum_{i=0}^{3} (1 - d_x^i)(1 - d_y^i) A(\mathbf{a}^i). \tag{A-1}$$

Extension of this technique to 4D *quadrilinear* interpolation [1] [7] is a matter of taking the sixteen samples closest to the 4D point $\mathbf{r}$. The distances between the continuous-domain index $\mathbf{n}'$ corresponding to $\mathbf{r}$, and these sixteen samples, are found as $d_s^i = |n'_s - n_s^i|$, $d_t^i = |n'_t - n_t^i|$, $d_u^i = |n'_u - n_u^i|$, and $d_v^i = |n'_v - n_v^i|$. The interpolated light field value is given by

$$\tilde{L}(\mathbf{n}') = \sum_{i=0}^{15} (1 - d_s^i)(1 - d_t^i)(1 - d_u^i)(1 - d_v^i) L(\mathbf{n}^i). \tag{A-2}$$

Bibliography

[1] M. Levoy and P. Hanrahan. Light field rendering. In *Proceedings of SIGGRAPH*, pages 31–42, 1996.

[2] J.A. Provine and L.T. Bruton. 3-D model based coding - a very low bit rate coding scheme for video-conferencing. In *IEEE International Symposium on Circuits and Systems*, volume 2, pages 798–801, 1996.

[3] G. Baudoin, F. Capman, J. Cernocky, F. El Chami, M. Charbit, G. Chollet, and D. Petrovska-Delacretaz. Advances in very low bit-rate speech coding using recognition and synthesis techniques. In *5th International Conference on Text, Speech and Dialogue*, pages 269–276, Berlin, DE, 2002. Springer.

[4] D. Dansereau and L.T. Bruton. A 4D frequency-planar IIR filter and its application to light field processing. In *Proceedings of ISCAS*, volume 4, pages 476–479, Bangkok, Thailand, May 2003.

[5] E.H. Adelson and J.R. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing*, pages 3–20, 1991.

[6] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Proceedings of SIGGRAPH*, pages 39–46, Los Angeles, CA, August 1995.

[7] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Proceedings of SIGGRAPH*, pages 31–42, August 1996.

[8] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Proceedings of SIGGRAPH*, pages 425–432, August 2001.

[9] A. Gershun. The light field. *Translated by P. Moon and G. Timoshenko in Journal of Mathematics and Physics, MIT, 1939*, XVIII:51–151, 1936.

[10] P. Moon and D.E. Spencer. *The Photic Field*. MIT Press, 1981.

[11] S. Lang. *Calculus of Several Variables*, page 38. Springer Verlag, 3^{nd} edition, January 1995.

[12] T. Akenine-Moller and E. Haines. *Real-Time Rendering*. A K Peters Ltd., 2^{nd} edition, July 2002.

[13] W. Heidrich, P. Eisert, M. Stamminger, and A. Scheel. *Principles of 3D Image Analysis and Synthesis*, pages 204–213, 26–30. Kluwer Academic Publishers, Boston, 2000.

[14] Young and Freedman. *University Physics*, pages 1055–1056. Addyson-Wesley Publishing Company, Inc., 9^{th} edition, June 1996.

[15] B. Wilburn, M. Smulski, and H. Lee. The light field video camera. In *Proceedings of Media Processors 2002, SPIE Electronic Imaging 2002*, 2002.

[16] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. Van Gool. Plenoptic modeling and rendering from image sequences taken by a hand-held camera. In *Proceedings 21. Symposium für Mustererkennung (DAGM '99)*, pages 94–101, September 1999.

[17] B. Heigl and H. Niemann. Camera calibration from extended image sequences for lighfield reconstruction. In *Workshop on Vision, Modeling, and Visualization*, pages 43–50, Erlangen, Germany, November 1999.

[18] R. Koch, M. Pollefeys, B. Heigl, L. van Gool, and H. Niemann. Calibration of hand-held camera sequences for plenoptic modeling. In *Proceedings of the 7th International Conference on Computer Vision (ICCV)*, pages 585–591, Corfu, September 1999. IEEE Computer Society Press.

[19] R. Koch, B. Heigl, M. Pollefeys, L. Van Gool, and H. Niemann. A geometric approach to lightfield calibration. In *Computer Analysis of Images and Patterns, 8th International Conference*, pages 596–603, September 1999.

[20] C. Vogelgsang, B. Heigl, Greiner G., and H. Niemann. Automatic image-based scene model acquisition and visualization. In *Workshop Vision, Modeling and Visualization*, pages 189–198, Saarbrücken, Germany, November 2000.

[21] M. Magnor and B. Girod. Data compression for light-field rendering. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(3):338–343, April 2000.

[22] B. Girod, C.-L. Chang, P. Ramanathan, and X. Zhu. Light field compression using disparity-compensated lifting. In *(submitted) IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, 2003.

[23] M. Magnor, P. Eisert, and B. Girod. Model-aided coding of multi-viewpoint image data. In *Proceedings of the IEEE International Conference on Image*

Processing ICIP-2000, volume 2, pages 919–922, Vancouver, Canada, September 2000.

[24] P. Ramanathan, E. Steinbach, P. Eisert, and B. Girod. Geometry refinement for light field compression. In *IEEE International Conference on Image Processing, ICIP-02*, volume 2, pages 225–228, Rochester, NY, September 2002.

[25] C.-L. Chang, X. Zhu, P. Ramanathan, and B. Girod. Shape adaptation for light field compression. In *IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003.

[26] E. Angel. *Interactive Computer Graphics: A Top-Down Approach with OpenGL.* Addison-Wesley Publishing, 2^{nd} edition, 2000.

[27] A. Isaksen, L. McMillan, and S. Gortler. Dynamically reparameterized light fields. In *Proceedings of SIGGRAPH*, pages 297–306, 2000.

[28] S.C. Chan and H.Y.Shum. A spectral analysis for light field rendering. In *Proceedings 2000 International Conference on Image Processing*, volume 2, pages 25–28, 2000.

[29] L.T. Bruton and N.R. Bartley. Three-dimensional image processing using the concept of network resonance. *IEEE Transactions on Circuits and Systems*, CAS-32(7):664–672, July 1985.

[30] P. Agathoklis and L.T. Bruton. Practical BIBO stability of n-dimensional discrete systems. In *Proceedings of ISCAS*, pages 923–926, Newport Beach, CA, May 1983.

[31] S.K. Mitra and J.F. Kaiser. *Handbook for Digital Signal Processing*, chapter 5. John Wiley and Sons, NY, 1993.

[32] Ben Anderson. Design of a non-uniform bandwidth frequency-planar filter bank. Master's project, University of Calgary, Department of Electrical and Computer Engineering, November 2002.

[33] L.T. Bruton. Selective filtering of spatio-temporal plane waves using 3D cone filter banks. In *Proceedings of PACRIM*, pages 60–70, Victoria, B.C., August 2001.

[34] B. Anderson and L.T. Bruton. Non-uniform bandwidth frequency-planar (NUB-FP) filter banks. In *Proceedings of ISCAS*, pages 809–812, 2002.

[35] S. Baker, T. Sim, and T. Kanade. When is the shape of a scene unique given its light-field: A fundamental theorem of 3D vision? *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 25(1):100–109, January 2003.

[36] I.J. Cox, S.L. Hingorani, and S.B. Rao. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, May 1996.

[37] Q. Liu and L.T. Bruton. Design of 3-D planar and beam recursive digital filters using spectral transformations. *IEEE Transactions on Circuits and Systems*, 36(3), March 1998.

[38] B. Heigl, J. Denzler, and H. Niemann. Combining computer graphics and computer vision for probabilistic visual robot navigation. In *Proceedings of SPIE's*

14th Annual International Symposium on Aerospace/Defense Sensing, Simulation, and Controls, Orlando, Florida, April 2000.

[39] B. Heigl, J. Denzler, and H. Niemann. On the application of light field reconstruction for statistical object recognition. In *European Signal Processing Conference (EUSIPCO)*, pages 1101–1105, 1998.

[40] R. Gross, I. Matthews, and S. Baker. Appearance-based face recognition and light-fields. Technical report, Robotics Institute, Carnegie Mellon University, CMU-RI-TR-02-20, August 2002.

[41] P.P. Vaidyanathan. *Multirate Systems and Filter Banks*, chapter 4. Prentice Hall, NJ, 1^{st} edition, 1993.

[42] P. Eisert, E. Steinbach, and B. Girod. 3-D shape reconstruction from light fields using voxel back-projection. In *Vision, Modeling, and Visualization Workshop*, Erlangen, November 1999.