Parallax Photography: Creating 3D Motions from Stills

Ke Colin Zheng

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2008

Program Authorized to Offer Degree:
Computer Science & Engineering

University of Washington
Graduate School

This is to certify that I have examined this copy of a doctoral dissertation by

Ke Colin Zheng

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Co-Chairs of the Supervisory Committee:

_____

Brian Curless

_____

David H. Salesin

Reading Committee:

_____

Michael F. Cohen

_____

Brian Curless

_____

David H. Salesin

Date: _____

In presenting this dissertation in partial fulfillment of the requirements for the doctoral degree at the University of Washington, I agree that the Library shall make its copies freely available for inspection. I further agree that extensive copying of this dissertation is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for copying or reproduction of this dissertation may be referred to Proquest Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 1-800-521-0600, to whom the author has granted "the right to reproduce and sell (a) copies of the manuscript in microform and/or (b) printed copies of the manuscript made from microform."

Signature_____

Date_____

University of Washington

Abstract

Parallax Photography: Creating 3D Motions from Stills

Ke Colin Zheng

Co-Chairs of the Supervisory Committee:
Professor Brian Curless
Computer Science & Engineering

Professor David H. Salesin
Computer Science & Engineering

Photography is the process of making pictures by literally *'drawing with light'*, albeit in a limited and restricted way. The recent digitization of photography offers unprecedented opportunities to move beyond the traditional constraints of analog photography, and lead to the enhancement and enrichment of visual media. In this thesis, we explore one approach to addressing this challenge, namely, *parallax photography*. Our approach begins by capturing multiple digital photographs of a scene from different viewpoints exhibiting parallax. We present algorithms that find dense correspondences among the captured imagery, as well as algorithms for seamless interpolation. The approach also embodies methods and interfaces that identify and highlight the interesting pieces of the captured data, and data representations for efficient storing and rendering. As we show in this thesis, the results, which allow reexperiencing subtle parallax, create a more visceral sense of immersion in the scene than traditional photographs.

We apply this approach to three projects in particular. We introduce the "layered depth panorama," a representation that allows the user to experience 3D by off-axis panning. The system asks little more of the user than capturing a simple panorama from a sparse set of images with a hand-held camera. We describe an algorithm that constructs the LDP by sequentially determining the disparity and color of each layer using multi-view stereo. Geometry visible through the cracks at depth discontinuities in the frontmost layer is determined and assigned to layers behind the frontmost layer.

All layers are then used to render novel panoramic views with parallax. In the next project, we exploit the spatial-angular tradeoff for lightfield cameras based on Georgiev's new (integral) camera design with a bundle of lenses and prisms attached externally to the camera. This optical design can be treated as a planar camera array with all views being parallel to each other. A sparse set of views can be captured at a single exposure, and be densified via tri-view morphing. The interpolated set of rays, or light field can be used to produce synthetic aperture effects, new view synthesis and refocusing, all of which are impossible to do with conventional cameras. In the final project, we extend our approach to synthesize a small portion of a lightfield from a few off-plane views, with an application to create cinematic effects with simulated, smooth camera motions that exhibit a sense of 3D parallax. We present a small-baseline multi-view stereo algorithm that computes dense view-dependent depthmaps for interpolating new views, thus generating a small lightfield from a handful of input images. We also describe the cinematic conventions of these effects by presenting a taxonomy of camera moves and other details that were distilled from observation of many hours of documentary film footage; the taxonomy is organized by the number of subjects of interest in the scene. We then present an automatic, content-aware approach to applying these cinematic conventions to an input lightfield. Finally, we evaluate and demonstrate the approach on a wide variety of scenes, and present a user study that compares the 3D cinematic effects to their 2D counterparts.

Throughout this thesis, we demonstrate how these novel techniques can be used to create expressive visual media that provides rich 3D experiences.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to first thank my numerous collaborators for the projects described in chapters 4-6, which include Professors David Salesin, Brian Curless, Shree Nayar, Maneesh Agrawala, and Chris Pal, UW students Alex Colburn, Microsoft Research collaborators Michael Cohen, Richard Szeliski, and Sing Bing Kang, and Adobe Research collaborators Aseem Agarwala and Todor Georgiev. For other projects I completed during my time at UW that were not included as chapters in this thesis, my collaborators were Yung-Yu Chuang, Dan Goldman, Noah Snavely and Pravin Bhat.

Numerous people beyond the author lists of my papers helped me immeasurably. Rachel Zhu, Harlan Hile, and Chris Gonterman appear in my photographs and videos. Larry Zitnick gave great comments and suggestions on extending his consistent segmentation-based flow algorithm to stereo used in chapter 5 and chapter 6. Noah Snavely wrote the structure-from-motion system used throughout this thesis.

Thanks to all the members of the UW Grail lab for providing a great environment to work and collaborate, and providing a hand at those critical moments. Thanks also to Andrea Lingenfelter for her feedback on an early draft of this dissertation.

Finally, I would like to thank my family and friends, for their inspiration and support throughout the ups and downs of graduate school.

DEDICATION

I dedicate this dissertation to papa and mama, for their endless love.

Chapter 1

INTRODUCTION

When I'm ready to make a photograph, I think I quite obviously see in my mind's eye something that is not literally there in the true meaning of the word. I'm interested in something which is built up from within, rather than just extracted from without.

—Ansel Adams

When we view a scene, even a static scene, we do not assemble a full mental image in an instant. Instead, we may move slightly from side to side, refocus our eyes from near to far, and/or scan over various objects in the scene to form a mental image of the moment.

And when we capture a scene using our camera, we act similarly before pressing the shutter button. We may interact with the scene through the viewfinder by moving slightly around, fixating on various parts of the scene.

We take a snapshot of the scene, hoping the photograph reflects our own interpretation of the scene, or more specifically, our visual consciousness of that moment. Despite the fact that we faithfully capture the scene by freezing it, the whole interaction and dynamic viewing experience through our mind's eye has somehow been lost. Rather than a mere photograph, a more effective depiction is needed to match our visual memory of the scene, providing a closer way to look into another's mind visually.

This main limitation of traditional photography is related to the principle underlying the mechanism of the camera, a principle which was discovered in the Middle Ages, yet hasn't changed since its invention over centuries ago, namely, the *camera obscura*, Latin for "dark room". Cameras only capture the light rays that pass through their centers of projection. In other words, traditional photography makes pictures by drawing with a special and restrictive sampling of the complete set of light rays that resides in a real scene.

The recent digitization of photography offers an unprecedented opportunity for computer scientists to move beyond the constraints of traditional photograph, and towards the creation of more expressive visual media. Digital photography embodies the convergence of the camera and the computer. The camera uses new optics and digital sensors to capture light in the world, and the computer employs computational algorithms to model and manipulate the digitized light. The challenge then is to develop novel hardware devices that can capture more information, and software algorithms that can process this information to communicate powerful and expressive visual information.

In this thesis, we explore one approach that can address this grand challenge, namely, *parallax photography*. The process begins with the capture of multiple digital photographs of a scene, taken from slightly different viewpoints and exhibiting parallax. We present algorithms that reconstruct dense 3D geometry of the scene based on parallax, as well as algorithms for seamless interpolation among the captured imagery. This approach also includes interfaces and methods that identify and highlight the most interesting features of the captured data, and data representations for efficient storing and rendering. As we show in this thesis, the results, which allow the viewer to re-experience subtle parallax, create a more visceral sense of immersion in the scene than traditional photographs.

In the rest of this chapter, we will first describe how to parameterize all of the light that exists in a scene. We will then use this parameterization to describe how cameras sample light to form photographs, and how humans sample light to form visual consciousness. Finally, we will demonstrate how the large gap between the two motivates new optics and computational methods to move beyond the limitations of traditional cameras in ways that enrich the visual experience. In the subsequent chapter we describe an overall framework for techniques that create better viewing experiences by combining multiple samples of the light in a scene, as well as related work and an overview of our own research and contributions within the context of this framework. Chapters 3 - 5 provide technical details and results for each of the three projects that form the bulk of this thesis. Finally, we conclude this thesis by summarizing our contributions and offering ideas for future work.

1.1 Parameterizing light

Before discussing the possibility of creating any visual media, it is useful to precisely describe and quantify what visual reality is.

One possible solution is to record all the objects in the world and their interactions. The traditional model-based rendering approach adopts such a description: the shapes of objects are represented by certain geometric models; properties of the objects' surfaces are described by texture maps and light reflection models; lighting and shading are the results of interaction between the light sources and objects, etc. Such a description is often compact and insightful. However, this description is not always available. Moreover, deriving such a description from what we observe with our eyes or cameras, is by no means trivial. This has been a goal of computer vision for more than twenty years and has proven quite challenging.

As an alternative, we describe the world through light that reflects from objects in our environment. This light can be modeled as rays that travel through three-dimensional space without interfering with each other. Imagine a parameterized model of all the rays of light that exist throughout three-dimensional space and time. Such a model would describe the entire visual world and contain enough information to create any possible photograph or video ever taken. Adelson and Bergen [2] showed that this space, which they call the *plenoptic function*, can be parameterized by seven parameters: the three-dimensional location of the pinhole (V_x, V_y, V_z), time (t), the wavelength of the light (λ), and the direction of the light entering the pinhole (parameterized by spherical coordinates (θ, φ)). Given these seven parameters, the plenoptic function returns the intensity of light (radiance). Thus, such a 7D plenoptic function: $P_7(V_x, V_y, V_z, \theta, \varphi, \lambda, t)$ provides a precise notion of the visual world. The image-based rendering (IBR) approach adopts such a description and uses photographs rather than geometry as the main primitives for rendering.

1.1.1 From cameras

Any photograph is formed by sampling the plenoptic function, and it is useful to understand the nature of this sampling. An ideal pinhole camera with an infinitesimal aperture will select a pencil[1] of rays and project a subset of them onto a two-dimensional surface to create an image. Assuming

[1] The set of rays passing through any single point is referred to as a *pencil*.

the surface is a black and white photosensitive film, the final image will provide samples of radiance along a certain interval over two axes of the plenoptic function: θ and ϕ (the interval depends on the size of the film, and the resolution depends on the size of the film grain). The pinhole location V_x, V_y, V_z is fixed, and variations over a short range of time and wavelength are integrated into one intensity.

Real cameras, however, do not strictly conform to this idealized model. In a practical setting, the infinitesimal aperture of an ideal pinhole camera would not allow enough light to reach the film plane to form an image. Instead, cameras use a lens to capture and focus a wider range of rays. Thus, modern cameras do not measure a pencil of light rays, but a subset of the rays impinging upon their lenses. Also, the film will only have a certain dynamic range; that is, if the intensity returned by the plenoptic function is too great or too small, it will be clipped to the limits of the film. Films (and digital sensors) are not perfect at recording light, and thus will exhibit noise; film has a signal-to-noise ratio that expresses how grainy the output will be given a certain amount of incoming light. Typically, increasing the light that impinges on the film will increase the signal-to-noise ratio.

Some cameras sample the plenoptic function more widely. A full-view panorama extends the sampling of a photograph to encompass light from all directions θ, ϕ. Color film samples across a range of wavelengths. A video from a stationary camera samples over a range of time. Stereoscopic and holographic images also sample along additional parameters of the plenoptic function.

1.1.2 From eyes

The human eye can be thought of as an optical device similar to a camera; light travels through a pinhole (iris) and projects onto a two-dimensional surface, the retina. The light stimulates sensors (cones and rods) on this surface, the retina, and signals from these sensors are then transmitted to the brain. Human eyes can sample the plenoptic function simultaneously along five axes. Our two eyes provide two samples along a single spatial dimension (say, V_x) over a range of time. Information along the wavelength axis is extracted using three types of cones, which respond to a much larger dynamic range than film or digital sensors. Our eyes adjust dynamically to illumination conditions, in a process called *adaptation*. Finally, we sample along a range of incoming directions θ, ϕ (though the resolution of that sampling varies spatially across the retina). These samples across five dimen-

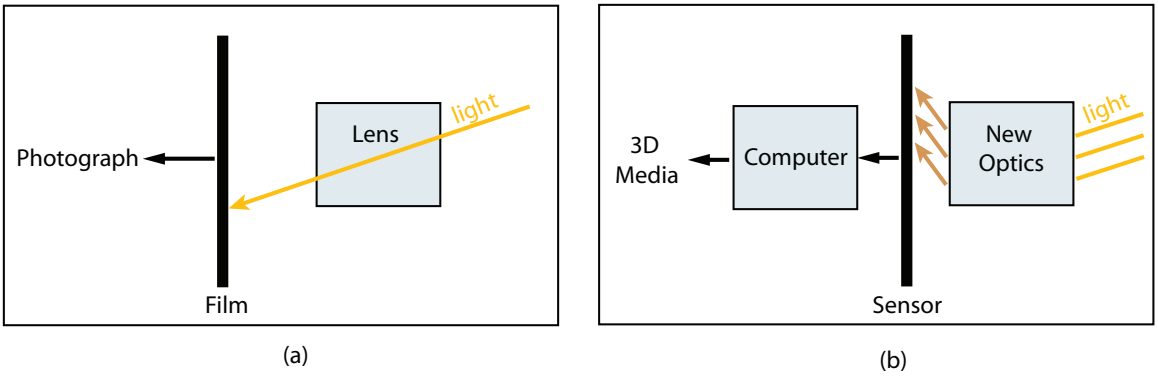


Figure 1.1: (a) Traditional photography is based on the camera obscura principle and produces a linear perspective image. (b) Parallax photography uses novel optics to capture a new set of light rays and use computational module to process and combine the data to produce new types of visual information.

sions serve as the input to the human visual system, which further gets processed to construct an interpretation of the light in the scene. However, this fairly complex process of construction is, for the most part, unconscious. The visual representation created through our eyes is very rich, given its high dynamic range, wide field of view, good depth of field, and more importantly, its ability to reflect the 3D nature of our world.

Cameras and humans capture the plenoptic function in different ways, and the large gap between the two interpretations motivates an obvious question: can we use digital technology to bridge this divide, and move beyond the limitations of traditional cameras to produce a more effective visual experience? In the next chapter we will describe one way of addressing this challenge. As illustrated in Figure 1.1, the approach begins by capturing a wider sampling of the plenoptic function than a camera would and then going on to process and combine this wider sampling into visual media that are more comprehensive than traditional photographs.

Chapter 2

A FRAMEWORK

In photography there is a reality so subtle that it becomes more real than reality.

—Alfred Stieglitz

Photography, since its birth over a century ago, has been used extensively to capture moments for documentation or artistic creation. Despite being photo-realistic by nature, photographs can be further enhanced to better match what we see in our mind's eye. This observation, coupled with photography's ease of use and the recent explosion of digital cameras, have made the creation of photographic visual media a new and exciting area of research.

In this thesis, we propose *parallax photography*, which captures information beyond just a pencil of rays and makes the representation of the recorded scene a rich 3D visual experience. It follows a generalized framework depicted in Figure 2.1 that resembles much of the innovation currently taking place in the digital photography research. The process for parallax photography begins by having the user take multiple samples of the plenoptic function with a camera. The use of multiple samples is well-motivated by digital photography's ease and economy of scale, as well as by the observation that human visual perception similarly integrates multiple samples. The next stage is some sort of alignment or registration process that is applied to the samples in order to position them within a single frame of reference. Next, the aligned samples are integrated or merged into one entity. The final output of the process is some sort of visual media. This media could be a photograph or a movie, or something more complex that requires a specialized viewer.

In the rest of this chapter, we will describe how this general framework applies to a large body of research in digital photography, which we will summarize and categorize in a taxonomy. We will narrow our focus to parallax photography, paying special attention to how this framework can be customized to leverage parallax from multiple samples and create an immersive viewing experience exhibiting 3D motion. We will highlight the main components of this framework for parallax

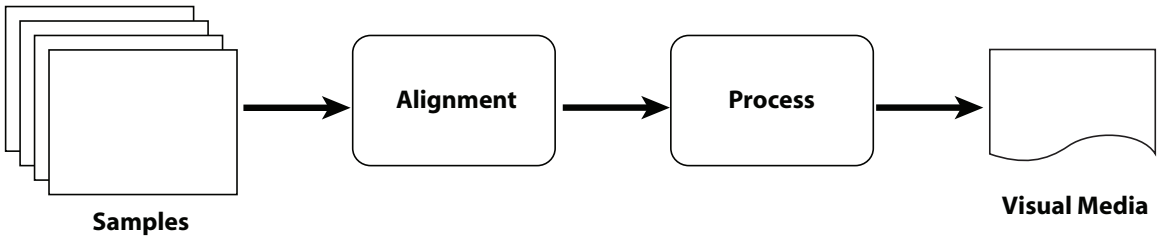


Figure 2.1: A diagram of the general process used to create expressive visual media by combining multiple samples of the plenoptic function. Multiple photographs are captured, aligned, and processed to create some sort of visual media.

photography, most importantly capture, representation and processing, and authoring and rendering.

2.1 A taxonomy of creating effective visual media by fusing multiple samples of the plenoptic function

A framework that fuses multiple samples of a plenoptic function into a visually more pleasing medium has been applied extensively in digital photography research for the last decade. We present a taxonomy of techniques used to create effective visual representations. We organize the taxonomy according to the appearance of the output: from photo-realistic to even beyond.

2.1.1 Enhancement to traditional photography

A large body of digital photography research works by simply recording the scene via multiple samples at a fixed camera location, each captured with slight variation of the camera parameters. Successive images (or neighboring pixels) may have different settings for parameters such as *exposure*, *focus*, *aperture*, *view*, *shutter*, *illumination*, or *instant of capture*. Each setting allows recording of partial information about the scene and the final output is constructed from these multiple observations by taking desired features from all samples. The final output is usually an enhanced version of the input samples which overcome the limitations on the sensors and optics, thus better matches what humans perceive.

- Field of view: A wide field of view panorama is achieved by stitching and mosaicing photographs taken by panning a camera around a common center of projection [93, 16]. This has also been extended to create panoramic videos that loop infinitely [6].

- Dynamic range: A high dynamic range image is created by merging photographs at a series of exposure values [25, 66]. Kang *et al.*applied a similar technique to videos under different exposures to create high dynamic range video [50].

- Depth of field: An all-in-focus image is reconstructed from a succession of photographs taken with varying planes of focus [5]. This is particularly useful under low lighting or in macro mode when small aperture is not achievable.

- Spatial resolution: Higher resolution is achieved by tiling multiple cameras (and mosaicing individual photographs) [100] or by jittering a single camera [54].

- Temporal resolution: High speed imaging is achieved by staggering the exposure time of multiple low frame-rate cameras. The exposure durations of individual cameras can be non-overlapping [100] or overlapping [82].

- Illumination: A well-lit image is synthesized by combining details from a sharp photograph taken under flash with colors from a blurry/noisy photograph taken under normal lighting [72]. Multi-spectral fusion has also been introduced to combine samples each from a distinctive spectrum to create enhanced results not visible to the human eye [12].

2.1.2 Beyond traditional photography

What we perceive in our mind's eye is usually a subjective abstraction of reality that does not exist alone at any instance in reality. More specifically, our visual consciousness is the result of the integration of a large set of rays over a certain period of time. Rather than enhancing input samples along certain dimensions, this type of non-traditional photography research takes pieces from different samples corresponding to different set of rays, and creates a visual representation that goes beyond the photo-realistic and more closely matches our visual consciousness.

- Temporal selection: the creation of a good group shot involves taking good portions (e.g., smiling open-eyed faces) from a sequence of images taken at different times, and merging them together seamlessly [5]. Similarly, the fusion of time-lapse information into a single entity [13] or the creation of stroboscopic painterly effects for a motion sequence communicates

information effectively by extracting and presenting information originating from different time instances. Shape-time photography is another example of the hybridization of different temporal elements [30].

- Illumination selection: to emphasize geometric edges, different parts of photographs taken under different illuminations can be merged to highlight discontinuities in shape [75, 28] or increase contrast by showing more details [28].

- Spatial selection: multi-perspective images are another example in this category where rays are selected spatially with respect to continuously varying perspectives to create effective illustrations of large scale scenes [101, 4].

2.1.3 Summary

Besides various capturing devices and methods, the different techniques listed in the above taxonomy vary in the details of the procedures for alignment and processing, but all still fall under the same general framework. Most of these techniques assume samples are either from fixed cameras or under planar projective transformation. All more or less assume that none of the individual input samples alone are sufficient to serve as the final result that the user wishes to create; instead, parts of these samples exhibit the qualities that the user desires. Through a combination of algorithms and user interfaces, these techniques identify those desirable qualities and further formulate the qualities in a numerically measurable fashion. They construct output from pieces of the aligned samples such that the result both contains the specified qualities and also appears natural. This construction is performed through optimization that maximizes the desired qualities while simultaneously minimizing the appearance of artifacts in the result.

2.2 Our approach

As shown from the taxonomy, a large body of work fits within the framework in Figure 2.1. Our thesis projects, while all sharing the same goal of creating rich 3D viewing experiences, nonetheless apply a single, basic approach albeit with some variations, which we will now describe. We will first summarize how the human visual system perceives 3D. This process is the basis for all

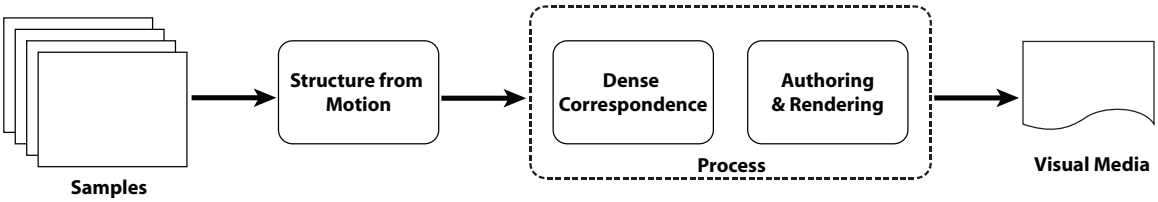


Figure 2.2: A diagram of the process used in all three parallax photography projects. Similar to Figure 2.1, multiple photographs are captured, aligned, and processed to create some 3D visual media exhibiting parallax.

our projects. We will then discuss the process of creating 3D motion from multiple samples in Figure 2.2, followed by detailed explanations of the three projects.

The human visual system perceives the three-dimensional (3-D) world as retinal images in our eyes through a process called projection. The availability of physiological and psychological cues gives the human visual system the ability to perceive depth. Some examples of these cues include binocular parallax, monocular movement parallax, accommodation, convergence, linear perspective, shading and shadows, and so on. Binocular parallax is the arguably most important depth cue in our visual system. It refers to the slightly different images sensed by the left and right eyes because of their slight difference in location. The biological visual system combines these two images to reconstruct a 3-D description of the world that it sees. In a similar manner, our visual system exploits the monocular motion parallax by fusing together slightly different images taken from slightly different locations in our surroundings. Even though these two cues are physiologically different, binocular parallax can be regarded as a special case of monocular motion parallax from a computational perspective. Therefore, we will loosely refer to both as motion parallax throughout this thesis.

Given how important motion parallax cues are to depth perception in the human visual system, our approach aims to exploit motion parallax to deliver a rich 3D viewing experience from a few photographs. Moreover, the motion parallax needs to be exhibited in a smooth and continuous way to better resemble the visual memory. Hence associating rays from the sparse input samples and constructing a dense representation of all rays has been one of the key challenges. Authoring effective viewing experiences that are scene-dependent and user-controllable is another part of the

puzzle to be solved. Through a combination of algorithms and user interfaces, we typically apply a four step process that combines multiple samples to create visual media exhibiting parallax:

1. Multiple photographs from slightly different locations are taken with either specialized optics or normal hand-held cameras. All three projects use multiple samples exhibiting small amounts of parallax.

2. The alignment puts all samples into a common coordinate space. This is achieved by Structure-from-Motion (SFM). The coordinate space varies from global cylindrical space to view-dependent planar representation.

3. In order to combine all samples to form better visual representations, all samples somehow need to be connected for every single pixel as a first step. This component varies for each project. In the first project, we use a plane sweep approach to construct a global geometric model, while for the second project, we compute dense pixel correspondences between image pairs. In the last project, we construct view-dependent depthmaps.

4. Finally, we render the scene into visual media based on different design goals. Different pieces from different samples are extracted and blended together according to the geometry of the scene or the correspondences among input samples.

We apply this approach to three projects in particular.

In the first project [105], we introduce the "layered depth panorama," (LDP), a representation that allows the user to experience 3D through off-axis panning. The system asks little more of the user than capturing a simple panorama from a sparse set of images with a hand-held camera. We describe an algorithm that constructs the LDP by sequentially determining the disparity and color of each layer using multi-view stereo. All layers are then fused together to render novel panoramic views with parallax.

In the next project [34], we exploit the spatial-angular tradeoff for lightfield cameras based on Georgiev's new (integral) camera design with a bundle of lenses and prisms attached externally to the camera. This optical design can be treated as a planar camera array with all views being parallel

to each other. A sparse set of views can be captured at a single exposure, and be densified via tri-view morphing. The interpolated set of rays, or light field can be used to produce synthetic aperture effects, new view synthesis and refocusing, all of which are impossible to do with conventional cameras.

In the final project, we extend our approach to synthesize a small portion of a lightfield from a few off-plane views, with an application to create cinematic effects with simulated, smooth camera motions that exhibit a sense of 3D parallax. We present a small-baseline multi-view stereo algorithm that computes dense view-dependent depthmaps for interpolating new views. We also describe the cinematic conventions of these effects by presenting a taxonomy of camera moves and other details that have been distilled from the observation of many hours of documentary film footage. We then present an automatic, content-aware approach to applying these cinematic conventions to an input of a few photographs. Finally, we evaluate and demonstrate the approach on a wide variety of scenes, and present a user study that compares the 3D cinematic effects to their 2D counterparts.

Chapter 3

BACKGROUND

The camera's only job is to get out of the way of making photographs.

—Ken Rockwell

Before describing our approach in detail over the next three chapters, we first discuss related work within the context of creating photorealistic visual media. This large body of research work is usually referred to as *image-based rendering*. Given that our approach is built upon the recent advances in image-based rendering, we first give an overview focusing mainly on the different representations and their corresponding rendering techniques. Since image-based rendering uses images as the primary substrate, which is also true for our approach, we next cover acquisition systems and methods that capture image data for image-based rendering. Finally, some 3D model reconstruction is discussed, serving as the basis for image based modeling. These three sections cover the parallax photography pipeline in Figure 2.2, which includes capturing, correspondence matching, and rendering.

3.1 IBR Techniques

Over the last decade, image-based rendering has attracted many researchers from various communities, including computer graphics, computer vision and signal processing. Many different algorithms have been developed, and a lot of progress has been made in terms of improving the rendering quality and increasing its generality. All these techniques attempt to solve the same basic problem: given a collection of images from known viewpoints, how do we generate images from unknown viewpoints?

We classify each rendering technique based on how image-centric or geometry-centric it is with an additional dimension on the number of input samples. As depicted in Figure 3.1, the x-axis of geometry separates the techniques into rendering with no geometry, rendering with implicit geometry,

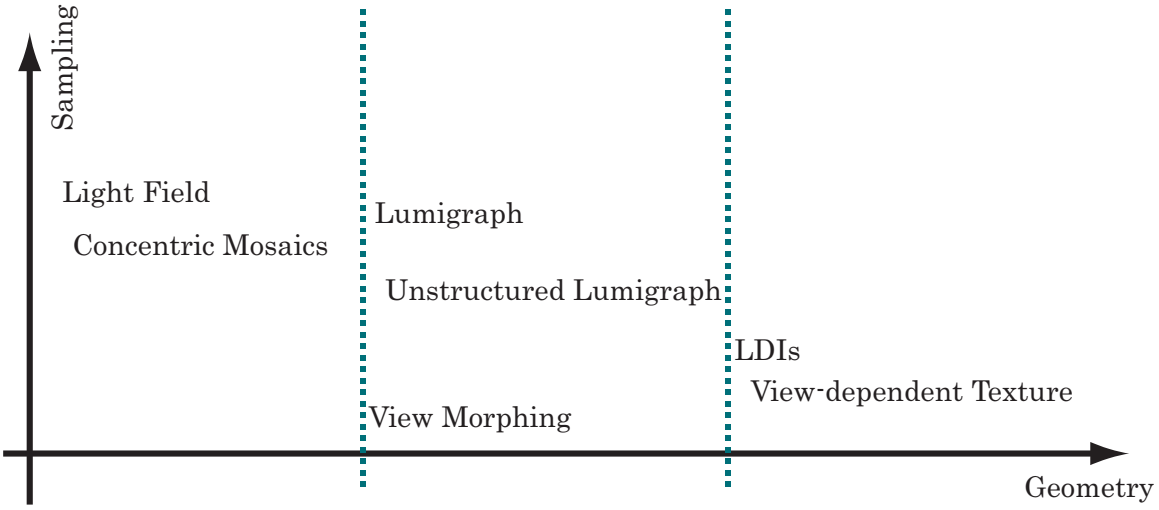


Figure 3.1: Categories used in this thesis, with representative approaches.

and rendering with explicit geometry; the y-axis of sampling indicates the number of input images for each technique ranging from sparse to dense.

Note that in this thesis, we sample the IBR space with a bias over techniques that could potentially be applied for our parallax photography problem. For more complete surveys on IBR techniques, readers are encouraged to read [48, 84, 104]. We report representative approaches with a focus on their representations and their rendering techniques.

3.1.1 Rendering with no geometry

We start with representative image-based rendering techniques with unknown scene geometry. Essentially, representations belonging to this category are all derived from the plenoptic functions introduced in Chapter 1.

Overview

Adelson and Bergen [2] defined the 7D plenoptic function as the intensity of light rays passing through the camera center at every 3D location (V_x, V_y, V_z), towards every possible direction (θ, φ), over any range of wavelengths (λ) and at any time(t), i.e., $P_7(V_x, V_y, V_z, \theta, \varphi, \lambda, t)$. The plenoptic

Dimension	View Space	Function Name
7D	free	Plenoptic function
5D	free	Plenoptic modeling
4D	bounding box	Light field/Lumigraph
3D	bounding circle	Concentric Mosaics
2D	fixed point	Image Mosaic

Table 3.1: A taxonomy of plenoptic functions categorized by their dimensionality and viewing space.

function provides a precise notion of the appearance of the world.

IBR, under the plenoptic function framework, can be defined as a set of techniques used to reconstruct a continuous representation of the plenoptic functions from observed discrete samples. The 7D plenoptic function is so general that it requires a huge amount of data to be fully sampled. Research on IBR is mostly about making reasonable assumptions to reduce the sample data size while keeping good rendering quality. By dropping the time axis with a static scene assumption and sampling wavelength, the 7D plenoptic function can be simplified to 5D. And by restricting the viewing space, the plenoptic function can be further reduced to lower dimension, from 5D to 2D. A taxonomy of plenoptic functions is summarized in Table 3.1.

Rendering techniques

For IBR techniques that do not require geometry information, they organize the reference images in ray space. The mapping and composition step of rendering is simply implemented as ray-space interpolation. Specifically, each ray that corresponds to target screen pixel, is mapped to the nearby sampled rays and is composite by ray-space interpolation . While the details in the rendering depend on the dimensionality and complexity of the ray space, these techniques all fundamentally target efficient organization and indexing of the reference rays for mapping onto screen space.

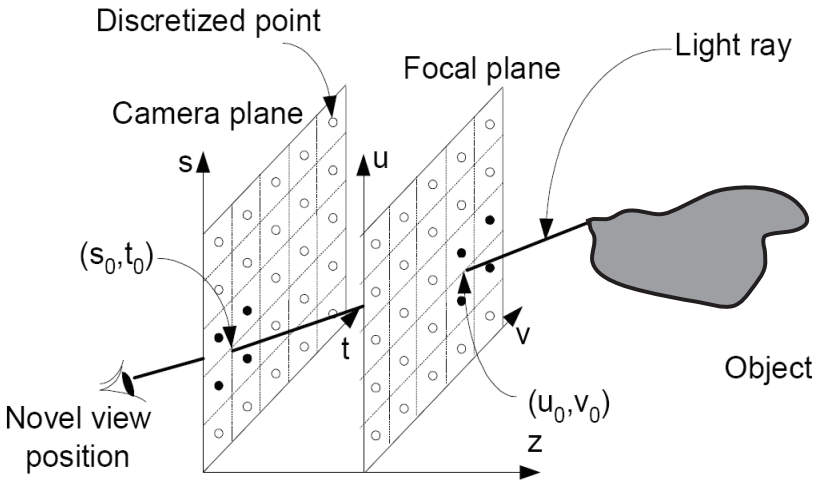


Figure 3.2: Two-plane representation of a light field.

5D Plenoptic modeling

McMillan and Bishop [63] introduced plenoptic modeling, which is a 5D function: $P_5(V_x, V_y, V_z, \theta, \varphi)$. They record a static scene by positioning cameras in the 3D viewing space, each on a tripod capable of continuous panning. At each position, a cylindrical projected image was composed from the captured images during the panning. This forms a 5D IBR representation: 3D for the camera position, 2D for the cylindrical image. To render a novel view from the 5D representation, the close-by cylindrical projected images are warped to the viewing position based on their epipolar relationship and image flow.

4D Light field and Lumigraph

Light field [58] and Lumigraph [36] both simplify the 7D plenoptic function into 4D by ignoring the wavelength and time, and assuming constant color along a ray direction.

As a result, we have this 4D plenoptic function: $P_4(u, v, s, t)$, where (u, v) and (s, t) are parameters of two planes of the bounding box, as shown in Figure 3.2.

Note we show the most widely used setup, where the two planes, namely the camera plane (u, v)

and the focal plane (s,t), are parallel. In general, they need not be parallel. The two planes are discretized so that a finite number of light rays are recorded. If we connect all the discretized points from the focal plane to one discretized point on the camera plane, we get an image (2D array of light rays). Therefore, the 4D representation is also a 2D image array.

The principles of light field and Lumigraph are the same, except that the Lumigraph leverages approximate geometry information for better rendering quality (reducing ghosting artifacts).

Capturing In the earlier light-field system, a capturing rig is designed to obtain uniformly sampled images. A number of new light-field capturing systems have been proposed in the last few years with some nice properties. We will discuss a few of them later in this chapter.

Lumigraph allows irregular sampling with a tracked hand-held camera. A hierarchical algorithm (rebinning) was proposed to resample the irregular samples onto the uniform grid on the camera and focal planes.

Rendering Under the two-plane parameterization, any ray passing through them can be indexed by the two intersection points and then be rendered using quadralinear interpolation of the neighboring 16 rays as illustrated in Figure 3.2. A novel view is decomposed into light rays, rendered separately, and reassembled together. All these operations can be done in realtime, independent of the scene complexity.

3D Concentric Mosaics

An interesting 3D parameterization of the plenoptic function, called Concentric Mosaics (CMs) [85], was proposed by Shum and He; here, the sampling camera motion is constrained to follow concentric circles on a plane. The light rays are then indexed by the camera position or the beam rotation angle α, and the pixel locations $s(u,v)$: $P_3(\alpha,u,v)$. This parameterization is equivalent to having many slit cameras rotating around a common center and taking images along the tangent direction. Each slit camera captures a manifold mosaic, inside which the pixels can be indexed by (α,u).

Capturing In concentric mosaics, the scene is captured by mounting a camera at the end of a level beam, and shooting images at regular intervals as the beam rotates. The entire capturing process can

be finished within 10 minutes, almost as easy as capturing a traditional panorama.

Rendering The rendering of concentric mosaics is slit-based. The novel view is split into vertical slits. For each slit, the neighboring slits in the captured images are located and used for interpolation. The rendered view is then reassembled using these interpolated slits.

3.1.2 Rendering with implicit geometry

In this section, we describe a class of techniques that relies on positional correspondences (typically across a small number of images) to render new views. This class has the term implicit to express the fact that geometry is not directly available.

Overview

Techniques under this category takes scene geometry information implicitly in the form of point feature correspondences, disparity maps, optical flow, etc. Novel views are computed based on direct manipulation of these positional correspondences.

Rendering techniques

Per pixel-based rendering is applied to representations that are created from 2D correspondences between reference images. Each pixel is usually rendered independently through forward mapping or backward mapping, depending on the representation. Typically, in per pixel-based rendering, the target view is always restricted to be close to the reference view. The pixel on the reference view is directly mapped to the target view with no composite.

View Interpolation

Chen and Williams' view interpolation method [21] is capable of reconstructing arbitrary viewpoints given two input images and dense optical flow between them. To generate an in-between view of the input image pair, the offset vectors in the optical flow are linearly interpolated and the pixels in the source images are moved by the interpolated vector to their destinations in the novel view. View

interpolation performs well when the two input images are close to each other, so that visibility ambiguity does not pose a serious problem.

View Morphing

From two input images, Seitz and Dyer's view morphing technique [79] reconstructs any viewpoint on the line linking two optical centers of the original cameras. View morphing guarantees that the rendered view is physically valid by introducing a prewarping stage and a postwarping stage. During the prewarping, the two reference images are rectified. After the rectification, the two images share the same image plane and their motion becomes perpendicular to their viewing axis. Linear interpolation is then used to get the intermediate view, followed by postwarping to undo the rectification.

As an extension to the view morphing technique, tri-view morphing [102] is a more recent system for creating the appearance of 3D by morphing between three views, making use of the trifocal tensor to generate the warping transforms among the views.

3.1.3 Rendering with explicit geometry

Representations that do not rely on geometry typically require many images for rendering, and representations that rely on implicit geometry require accurate image registration for high-quality view synthesis. In this section, we describe IBR representations that use explicit geometry. Such representations have direct 3D information encoded in them, either in the form of depth along known lines-of-sight, or 3D coordinates.

Overview

Building on a wave of new capturing devices and the rapidly improving state of the art in geometric reconstruction techniques, representations with explicit geometry are becoming more prevalent. Depending on the data size, resolution, rendering requirement and choice of algorithms, the geometry of an object or scene can be represented in numerous ways, including voxels, level-sets, polygon meshes, or depth maps. Many techniques represent geometry on a regularly sampled 3D grid (volume), either as a discrete occupancy function (e.g., voxels), or as a function encoding dis-

tance to the closest surface (e.g., level-sets). 3D grids are popular for their simplicity, uniformity, and ability to approximate any surface. Polygon meshes represent a surface as a set of connected planar facets. They are efficient to store and render and are therefore a popular output format for multi-view algorithms. Some methods represent the scene as a set of depth maps, one for each input view. This multi-depthmap representation avoids reconstructing the full 3D geometry, and the 2D representation is convenient particularly for smaller datasets.

Rendering techniques

The rendering algorithms of IBR representations with dense depth map are often similar to each other. The pixels of the reference view are first projected back to their 3D locations and then re-projected to the novel view. This process can be further accelerated by factorizing the warping process into a simple pre-warping stage followed by standard texture mapping. The pre-warp handles only the parallax effects resulting from the depth map and the direction of view. The subsequent texture-mapping operation handles the scaling, rotation, and remaining perspective transformation, which can be accelerated by standard graphics hardware. A similar factoring algorithm was performed for the LDI, where the depth map is first warped to the output image with visibility check, and colors are copied in afterwards.

Layered Depth Images

In order to deal with disocclusion artifacts in 3D warping, Shade *et al.*proposed Layered Depth Images, or LDIs [81], to store not only what is visible in the input image, but also what is behind the visible surface. The LDI is constructed either using stereo on a sequence of images with known camera motion or directly from synthetic environments with known geometries. In an LDI, each pixel in the input image contains a list of depth and color values where the ray from the pixel intersects with the environment. Though an LDI has the simplicity of warping a single image, it does not consider the issue of sampling density. Chang et al. [20] proposed LDI trees so that the sampling rates of the source images are preserved by adaptively selecting an LDI in the LDI tree for each pixel. While rendering the LDI tree, only the level of LDI tree that is the comparable to the sampling rate of the output image need to be traversed.

View-dependent texture mapping

To obtain visual effects of a reconstructed architectural environment such as highlights, reflections, and transparency, Debevec et al. in their Facade [23] work, used view-dependent texture mapping to render new views by warping and compositing several input images of an environment. This is the same as conventional texture mapping, except that multiple textures from different sampled viewpoints are warped to the same surface and averaged, with weights computed based on proximity of the current viewpoint to the sampled viewpoints. A three-step view-dependent texture mapping method was also proposed later by Debevec *et al.* [24] to further reduce the computational cost and to have smoother blending. This method employs visibility preprocessing, polygon-view maps, and projective texture mapping.

Along a very similar vein, Buehler *et al.*proposed the unstructured lumigraph rendering [19], which addressed a similar problem. They first proposed eight goals for IBR rendering: use of geometric proxies; unstructured input; epipole consistency; minimal angular deviation; continuity; resolution sensitivity; equivalent ray consistency and real-time. These goals served as the guidelines of their proposed unstructured lumigraph rendering approach. Weighted light ray interpolation was used to obtain light rays in the novel view. The weights are largely determined by how good the reference light ray is to the interpolated one according to the goals. A weighted blending field for the reference views is described to guarantee real-time rendering.

3.2 Capturing

We see a wide variety of approaches in image-based rendering. They vary in system setup and camera configuration, scene representation, rendering algorithm, and compression strategy. However, all these approaches strive towards the same goal of producing photorealistic depictions of the world.

There is a significant amount of work in developing capturing techniques, especially for IBR systems using no geometry information. Camera setups for light field acquisition is probably the most well-studied problem among all these techniques, yet it is still a very active research area. Some systems demonstrate the ability to handle dynamic scenes. Some provide dense configurations to sample at high angular and spatial resolution. And recently, hand-held light field capturing

systems have also been introduced. They are compact in size and easy to operate.

The earliest acquisition systems for light fields used a single moving camera. Levoy and Hanrahan used a camera on a mechanical gantry to capture the light fields of real objects [58]. They have since constructed a spherical gantry for capturing inward looking light fields. Gantries have the advantage of providing unlimited numbers of input images, but at a few seconds per image (letting the camera come to rest at each viewpoint, to avoid motion blur), it can take several hours to capture a full light field. Gantries also require very precise motion control, which is expensive. A simpler version has been proposed recently at a much lower cost using Lego Mindstorms ^{TM}. The Mindstorms motors have rotary encoders, so with enough gearing down and solid construction this can be made to be accurate and repeatable. However, for this line of systems, the biggest drawback, of course, is that they cannot capture light fields of dynamic scenes.

Using an array of digital cameras, which serves as a dense configuration for light field capture(e.g., [100, 103]), is currently the mainstream capturing approach for dynamic light field research. The Stanford Light Field Camera consists of 128 CMOS cameras, each has a resolution of 640 by 480 and is capable of capturing at 30 fps.

The research on designing cameras to simultaneously capture multiple views of a scene precedes the introduction of the light field concept. The earliest works of Lipmann [62] and Ives [45] among others, known as *integral photography*, used arrays of lenslets or pinholes placed directly in front of the film, creating multiple images on it just like an array of cameras. A related type of integral photography design places an array of positive lenses in front of a conventional camera to create an array of real images between the lenses and the camera. Then the camera takes a picture focused on those images (e.g., [70]). Such a design has also been extended to capture dynamic scenes by replacing the camera with an HDTV camera [65].

The more recent approaches of Adelson et al. [3] and Ng et al. [68], known as *plenoptic cameras*, effectively place a big lens in front of the array of lenslets (or cameras) considered in the first approach, forming an image on the array of lenslets. Each lenslet itself creates an image sampling the angular distribution of radiance at that point, which corresponds to one single direction observed from multiple points of view on the main lens aperture. This approach swaps the placement of spatial and angular samples on the image plane: instead of producing an array of ordinary images, as in integral photography, it creates what appears to be a single, recognizable image consisting of small

Camera design	Angular resolution	Spatial resolution	Output	System portability
Stanford camera array	high	high	video	bulky
NHK camera	medium	low	video	unknown
Ng camera	high	low	image	handheld
Dappled camera	high	low	image	handheld

Table 3.2: A summary of light field camera designs.

2D arrays of angular samples of a single point in the scene. Along the same line, Veeraraghavan *et al.* [97] describe a system for "dappled photography" for capturing radiance in the frequency domain. In this approach, the camera does not use microlenses, but replace the aperture with a cosine modulating mask. Again, this work suffers from the same problem as [68] in which only low spatial resolution lightfield is captured.

3.3 Multi-view stereo

The goal of multi-view stereo is to reconstruct a complete 3D object model from a collection of images taken from known camera viewpoints. Over the last few years, a number of high-quality algorithms have been developed, and the state of the art is improving rapidly. These approaches typically cast multi-view stereo as a variational problem, where the objective is to find the surface minimizing a global photo-consistency measure, regularized by explicit smoothness (geometric-consistency) constraints. Various optimization techniques are used, ranging from local methods such as gradient descent, level sets, or expectation maximization [107], to global ones such as graph cuts [53] and belief propagation. Seitz *et al.* [78] survey multi-view stereo algorithms and present a quantitative comparison of several multi-view stereo reconstruction algorithms together with some benchmark datasets.

As discussed earlier, the output of multi-view stereo is the geometry of a scene that can be represented in numerous ways: voxels, polygon meshes, and depth maps. We limit the scope to multi-view stereo algorithms that are tailored for image-based rendering applications. These methods all represent the scene as a set of depth maps, one for each input view [27, 47, 31, 53, 107].

We now summarize these methods with a focus on some fundamental properties: photo-consistency measure, visibility model, shape prior, and reconstruction algorithm.

3.3.1 Photo-consistency measure

In order to evaluate the visual compatibility of a reconstruction with a set of input images, researchers have proposed numerous photo-consistency measures which compare pixels in one image to pixels in other images to see how well they correlate. These measures can be classified into three categories: pixel based, window based and segment based. Pixel based measures work by taking a point, projecting it into the input images, and evaluating the amount of mutual agreement between those projections. A simple measure of agreement is the variance of the projected pixels in the input images. Pixel based measures can be extended to rectangle windows, and these window based metrics include sum of squared differences or normalized cross correlation [47]. Using the reasonable assumption that neighboring pixels with similar colors have similar or continuous depths, researchers have used image segments to simplify the stereo problem [107]. This has three important effects. First, it reduces the ambiguity associated with textureless regions. Second, by dealing with much larger segments, the computational complexity is reduced. Finally, noise tolerance is enhanced by aggregating over similarly colored pixels.

3.3.2 Visibility model

Given that scene visibility can change dramatically with viewpoint, all modern multi-view stereo algorithms employ visibility models to specify which views to consider when evaluating photo-consistency measures. These models vary from geometric reasoning to simple heuristics. In cases where scene points are visible more often than they are occluded, simple outlier rejection techniques can be used to select the good views. A heuristic often used in tandem with outlier rejection is to avoid comparing views that are far apart, thereby increasing the likely percentage of inliers [27, 31].

3.3.3 Shape prior

Approaches that represent the scene with depth maps typically optimize an image-based smoothness term that seeks to give neighboring pixels the same depth value. This kind of prior fits nicely into

a 2D Markov Random Field (MRF) framework [53], and can therefore take advantage of efficient MRF solvers. A disadvantage associating with such shape prior is that there is a bias toward frontal-parallel surfaces.

3.3.4 Reconstruction algorithm

Multi-view stereo algorithms follow a common paradigm where they first compute a cost function on a 3D volume, and then extract a surface from this volume. These algorithms differ in the definition of the cost function and the surface extraction method. Voxel coloring algorithm and its variants make a single sweep through the 3D volume and reconstructing voxels with costs by thresholding in one pass [80]. Other algorithms define a volumetric MRF and use max-flow or multi-way graph cut to extract an optimal surface [53]. While reconstructing a set of depth maps, these methods often enforce consistency constraints between depth maps to ensure a single consistent 3D scene interpretation [27, 47, 107].

Chapter 4

LAYERED DEPTH PANORAMAS

You don't take a photograph, you make it.

—Ansel Adams

4.1 Introduction

A single photograph of a scene is just a static snapshot with limited field of view captured from a single viewpoint. [1] Many techniques have been proposed to extend the ways in which a scene can be visualized by taking multiple photographs. These range from creating 2D panoramas from a few photographs (to extend the field of view) to creating 4D lightfields from a large number of images (to provide extensive freedom to explore a scene, with expensive capture and storage requirements).

In this chapter, we present a system that asks little more of the user than capturing a simple panorama from a sparse set of images with a hand-held camera. We provide a result that is only fractionally larger than a simple 2D panorama, yet affords the ability to view the result with both the wide field-of-view of panoramas and enough parallax between objects at different depths to create a more visceral sense of immersion in the scene.

The capture process is much like that for a traditional panorama in which a sparse set of images is taken about a single center of projection to avoid parallax. However, we instead require the user to merely hold the camera at arm's length to capture the parallax induced when moving the camera along an arc. We automatically recover a layered representation [9] in which multiple depths may exist for a single line of sight. Such a representation was called a layered depth image (LDI) [81]. Because our representation is a layered analogue of the panorama, we refer to it as layered depth panorama (LDP). The LDP removes the fundamental limitations of 2D mosaics by supporting viewpoint translation with reasonable extra cost in memory and computation. When viewed from any

[1] The work described in this chapter was originally presented as a paper [105] at CVPR 2007.

single point-of-view, the LDP appears like a normal 2D panorama; when the viewpoint moves off its center, the LDP exhibits motion parallax, thus providing a more immersive 3D experience.

I next review previous work in image-based scene modeling and stereo matching. In Section 3, we describe how to compute the LDP for this novel representation. We present some experiments on a few real world examples in Section 4. We conclude the chapter with a discussion of our results and a list of topics for future research.

4.2 Previous work

Our system builds on several algorithms previously developed for image-based rendering and stereo reconstruction. In this section, we review relevant work in these areas.

4.2.1 Image-based modeling and rendering

The techniques described below are specialized instances of *image-based rendering* [63, 58, 36, 19], where the goal is to create novel views from a collection of input images.

2D Panoramas. 2D panoramas are constructed by stitching together a collection of images taken from the same center of projection [93, 91]. They support viewing the scene from this point in any desired direction. Panoramas can be captured with or without a tripod, and can be automatically stitched [16]. However, they do not support view translation; this deprives users of motion parallax, which is an important cue in 3D scene perception.

Concentric mosaics. If we constrain the camera motion to planar concentric circles, we obtain a 3D plenoptic function called *concentric mosaics* [85]. Such mosaics can be formed by compositing slit images taken at different locations along each circle [71]. Like 2D panoramas, concentric mosaics do not require recovering geometric and photometric scene models. Moreover, they provide a much richer viewing experience by allowing users to move freely in a circular region and to observe significant parallax and lighting changes. However, there is a problem associated with not using appropriate geometry: the vertical scaling in the reconstructed views can appear incorrect. Also, concentric mosaics are much more data intensive than 2D panoramas, and require special hardware such as a motorized tripod with an extended arm.

Layered Depth Images. Layered depth images (LDIs) are images that have potentially more than

a single depth/color pair at each pixel [81]. These allow the scene to be rendered from multiple adjacent points of view without the introduction of either *cracks* (holes in the reconstruction) or spurious surfaces (that look like rubber sheets). When the pixels are organized into a small number of *layers*, the resulting representation can be efficiently rendered on a GPU [107].

Image-based editing. Image-based editing [86] bypasses the difficult modeling process by manually specifying geometry. 3D models are built by segmenting images into sprites that are mapped to separate planes. Image-based editing techniques take advantage of human knowledge of the scene, which allows them to maximize the 3D effect while minimizing the amount of depth data. However, manual geometry specification is slow and tedious, requiring more efficient user interfaces. In an earlier work [9], a semi-automated stereo matching approach was proposed to create such *layered scene descriptions*. Efficient image-based rendering techniques for such representations have also been developed [81, 107].

Dense depth map. Laser rangefinding technologies acquire dense, accurate depth maps that can be converted into high-quality models. Bahmutov *et al.* [8] model a real world scene using depth enhanced panoramas. Such panoramas with per-pixel depth are acquired using they call a *model camera*, which is a structured-light acquisition device. These methods produce good geometry but suffer from long acquisition times and high equipment cost. Passive stereo-based reconstruction techniques (described below) can capture the scene more quickly, since only a few images are required, but usually do not produce as high-quality a result.

Figure 4.1 shows the relative tradeoff between acquisition cost and rendering quality for various image-based modeling and rendering techniques. Our goal is to develop a solution that has almost as low acquisition cost as 2D panoramas, yet produces similar 3D effects as concentric mosaics. We leverage state-of-the-art stereo-based scene reconstruction techniques to achieve this goal.

4.2.2 Stereo-based scene reconstruction methods

The problem of reconstructing a scene from multiple cameras has received a lot of attention in the last few years [76, 78].

In voxel-based approaches, the scene is represented as a set of 3D voxels, and the task is to compute the visibility as well as the color of each voxels. One major limitation of voxel coloring [80]

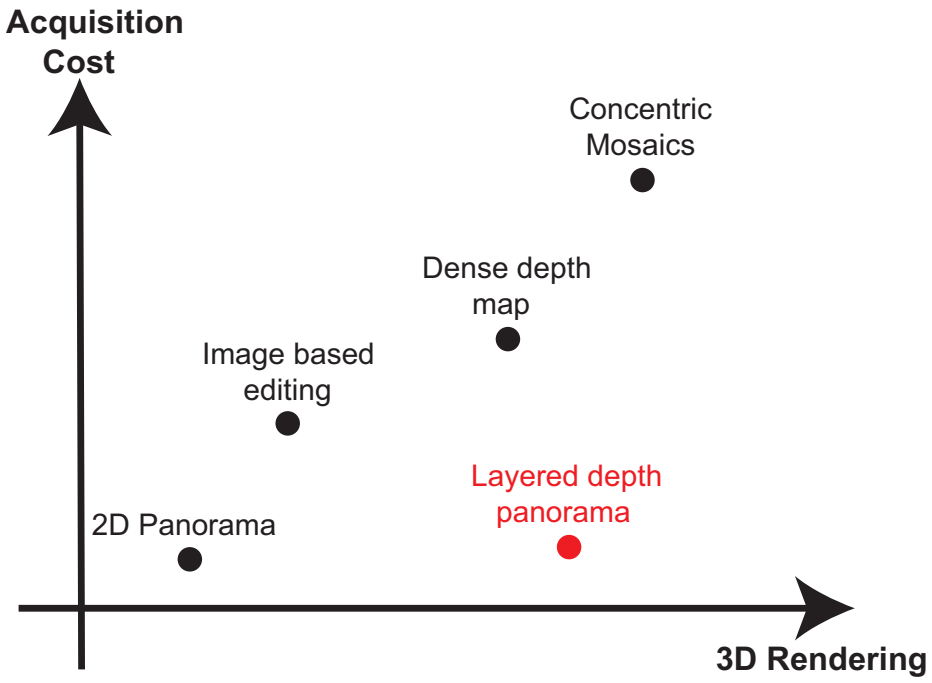


Figure 4.1: Cost/quality tradeoff for various modeling techniques.

is that "hard" decisions concerning voxel occupancy are made as the volume is traversed. Because the data is ambiguous, such decisions can easily be wrong, and there is no easy way to correct them.

Szeliski and Golland [92] applied an iterative framework to solve not only for depth, but for color and transparency as well. They iteratively aggregate visibility evidence to refine the reconstruction. Since this problem is grossly under-constrained and difficult to optimize, their results were not that encouraging. Baker *et al.* [9] proposed an alternative approach, where only a small number of layers is hypothesized and recovered (see also [14, 61]).

Kolmogorov and Zabih [53] took an approach that yielded excellent results for stereo [76], namely energy minimization via graph cuts, and generalized it to solve the scene reconstruction problem. They treat the input images symmetrically, handle visibility properly, and impose spatial smoothness while preserving discontinuities. Their work is one example of approaches that recover *multiple depth maps* simultaneously in the context of a global optimization framework [90, 47, 107, 31].

Stereo matching has also been applied to panoramic images, both in the case of panoramas taken from widely separated viewpoints [46], and for panoramas taken as concentric mosaics [59]. In both of these cases, a single dense depth map with a large field of view is directly computed from the multi-perspective panoramas. In this work, we create *layered depth panoramas*, which can represent multiple depths at each pixel in a panoramic coordinate system. Furthermore, we optimize the estimated colors using results from recent image stitching algorithms.

4.3 Approach

In this section, we describe the construction of the layered depth panorama (LDP) from multiple images, and how novel views are rendered.

4.3.1 The Layered Depth Panorama

A layered depth image (LDI) differs from a normal image in that each pixel stores one or more pixels along the line of sight represented by the pixel. The front element in the layered depth pixel samples the first surface seen along that line of sight; the next element samples the next surface seen along that line of sight, etc. Our LDP uses the same underlying concept adapted to a cylindrical parameterization to accommodate larger fields of view. A hand-held camera held at arm's length captures views to simulate an off-center camera rotation, from which we construct the LDP. As in an LDI, for the LDP we also select a 2D array of rays to form a layered representation. We would like these rays to be close to the captured rays, and we also wish to cover a wide field of view in order to maximize the range of viewpoints the user can virtually explore.

The original images are gathered roughly along a circular arc in space with a radius roughly equal to an arm's length. We first determine the arc that most closely follows the camera path. This establishes a cylindrical coordinate system (see Figure 4.2). The 2D array of rays is formed by the set of rays that pass through the arc and simultaneously lie in planes that pass through the center of the arc. In other words, these are the rays that would form the center vertical scanlines of cameras with optical centers lying on the arc and facing outward. This is a cylindrical version of pushbroom panoramas [77].

The 2D array of rays are parameterized by (θ, v) (see Figure 4.2). θ is the angle (or position)

along the arc through which all rays pass. v is the vertical position where the ray pierces a cylinder a *unit* distance from the arc (i.e., a cylinder with radius one greater than the radius of the arc). Discretizing θ and v defines the resolution of the LDP. Each discrete position, (θ, v), defines a 3D ray, also referred to as a *pixel* in each layer of the LDP.

Our goal is to recover the (possibly multiple) depths d associated with each pixel.

Depth, d, is discretized at increments proportional to $1/d$, to ensure that *disparity* increments are similar. Each discrete position (θ, v, d) represents a *voxel* in the cylindrical volume.

The LDP consists of a set of layers L_i, where i is the layer index. For each ray, (θ, v), we determine the depth d of objects intersected by the ray. We also determine the color, c, for the intersection points. Thus, $L_i(\theta, v) = (d, c)$ indicates the pixel with coordinate (θ, v) on the i-th layer has color c, and is at depth d. In other words, the voxel at (θ, v, d) is on a surface with color c; and it is the i-th colored voxel along the ray from the camera arc to (θ, v, d). The first layer is dense (i.e., there is a well defined d value for every (θ, v) pixel). Layers behind the first layer are kept as small as possible, just enough to fill holes seen through depth discontinuities when viewed from along the camera arc.

Our goal is to represent the scene with an LDP such that all the input images can be explained. We achieve this by sequentially solving each layer in the LDP from front to back with multi-view stereo. We begin with a cylindrical plane sweep algorithm to generate an initial disparity space image [43] (DSI). Later, the DSI is further refined based on visibility inferred from the LDP. We leverage state-of-the-art optimization techniques with improved matching costs and smoothness constraints to reconstruct each layer. The algorithm works in a iterative fashion, where we modulate the DSI based on reconstructed geometry from LDP and we update the LDP based on the improved DSI.

4.3.2 Cylindrical Plane Sweep Algorithm

Plane-sweep and space coloring/carving stereo algorithms support multi-image matching, enable reasoning about occlusion relationships, and are more efficient than traditional correlation-based formulations. Rather than searching for corresponding windows across images as in traditional stereo matching algorithms, plane sweep algorithms consider each candidate disparity as defining

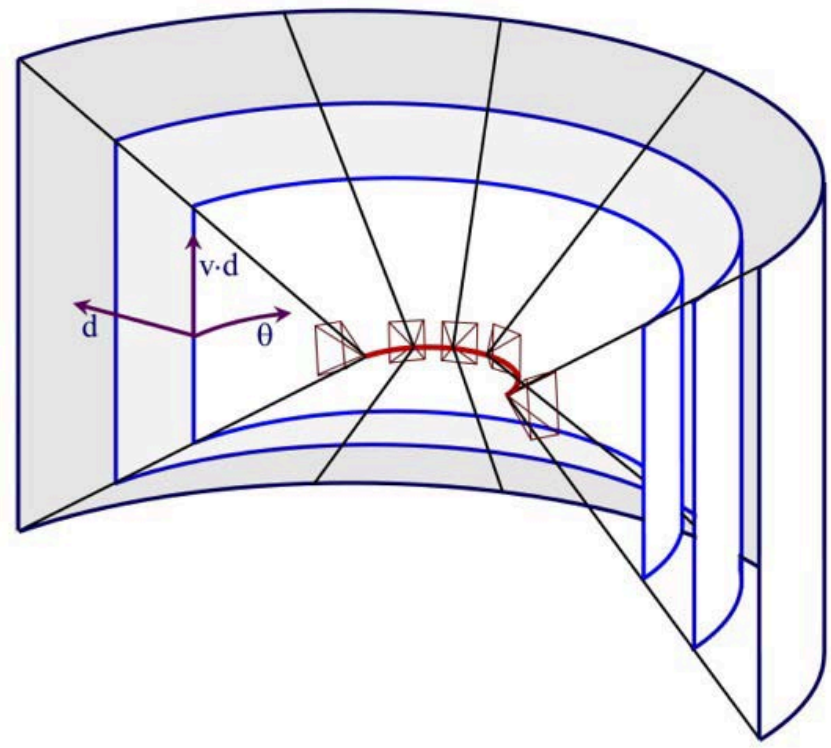


Figure 4.2: The original cameras lie along an arc shown in red. This arc defines the spine of the rays in the LDP and defines a 3D volume of concentric cylinders.

a plane in space and project all images to be matched onto that plane, using a planar perspective transform (homography).

We have generalized plane sweep stereo to perform a multi-image cylinder sweep stereo reconstruction. All images are projected onto cylinders at various depths d. A per-pixel robust variance of the collection of input pixels that map to an output pixel is first computed. These are then aggregated spatially using an efficient convolution algorithm (a moving average 5×5 box filter). Finally, we use aggregated shiftable windows, essentially seeking the lowest variance within ± 1 pixel and select the lowest value. This last step improves the performance of matching near depth discontinuities [47].

Thus, for every location (θ, v, d) in the cylindrical volume, we have $\mu(\theta, v, d)$ and $\phi(\theta, v, d)$, where μ is the median color and ϕ is the robust variance. This forms our raw disparity space image (DSI), the initial matching cost. Later, we will describe how we iteratively update the DSI based on

visibility information.

4.3.3 Optimization

For each layer L_i, we solve for its color and its depth separately.

Depth for the First Layer

Our first goal is to assign to each pixel $p = (\theta, v)$ on the first layer, L_1 a label corresponding to the depth (or disparity) of the first intersection along the ray (θ, v). Later, we will use almost the same formulation for the back layers. We formulate the problem of finding the disparity map for each layer as a global optimization. The objective is to find a disparity function d that minimizes a global energy given by

$$E(d) = E_{\text{data}}(d) + \lambda \cdot E_{\text{smooth}}(d). \tag{4.1}$$

The data term, $E_{\text{data}}(d)$, measures how well the disparity function d agrees with the input images. Using the disparity space formulation,

$$E_{\text{data}}(d) = \sum_{p \in (\theta, v)} \phi(p, d(p)), \tag{4.2}$$

where ϕ is the matching cost (robust variance) in the DSI. Recall that p represents the ray direction (θ, v) in our cylindrical coordinates.

The smoothness term $E_{\text{smooth}}(d)$ encodes the smoothness assumptions which in our case encourages a piece-wise smooth result:

$$E_{\text{smooth}}(d) = \sum_{(p,q) \in \mathscr{C}} \rho_d(d_p, d_q) \rho_I(\mu(p, d_p), \mu(q, d_q)), \tag{4.3}$$

where $\mathscr{C}$ is the set of 4-connected neighbors in (θ, v), d_X is the depth at X, ρ_d is a monotonically increasing function of disparity difference, and ρ_I is a monotonically decreasing function of intensity differences that lowers smoothness costs at high intensity gradients.

$$\rho_d(d_p, d_q) = min(|d_p - d_q|, c_1), \tag{4.4}$$

$$\rho_I(\mu(p, d_p), \mu(q, d_q)) = e^{c_2 |\mu(p, d_p) - \mu(q, d_q)|}, \tag{4.5}$$

We use $c_1 = 2.0$ and $c_2 = -0.01$ for all of our examples. Our smoothness term encourages disparity discontinuities to coincide with intensity/color edges, which accounts for some of the good performance of global optimization stereo approaches. Note because we do not have an image taken from the virtual camera, we approximate the intensity with the median from the DSI.

We balance the data term and the smoothness term using a constant $\lambda = 2.0$ for all examples. Once the global energy has been defined, we use the graph cut alpha-expansion algorithm of Boykov *et al.* [15] to solve for the label d.

Color

In addition to depth, we also recover the color for each pixel in each layer of the LDP. Since we are reconstructing a global texture for each layer, we will make the simplifying Lambertian assumption and determine a single color for each entry in the LDP.

We solve for the color of each entry in a layer in a similar fashion to the disparity determination, by leveraging a label optimization procedure. In this step, the labels identify the input image from which to pull the pixel color to assign to the layer. This avoids blurring caused by simply blending all input pixels that project to a particular voxel.

For each voxel $V = (\theta, v, d)$, we find all the input images that see it by reprojecting the voxel location back into those images accounting for visibility between the voxel and input cameras [92]. The indices of the visible input cameras form the set of candidate labels, and we find the best labeling once again using graph cuts.

As before, we define cost functions to express the desired properties of a labeling l:

$$E'(l) = E'_{\text{data}}(l) + \lambda \cdot E'_{\text{smooth}}(l). \tag{4.6}$$

The data term $E'_{\text{data}}(l)$ reflects the property that each pixel p in the scene should be imaged from a viewpoint l_p that is most aligned with the virtual camera. It is specified as

$$E'_{\text{data}}(l) = \sum_{p \in (\theta, v)} \cos^{-1}(p \cdot (V - C_{l_p})), \tag{4.7}$$

where V is the position of the voxel and C_{l_p} is the center of camera l_p. This forces using rays from the closest cameras to compute the color of the ray p.

$E'_{\text{smooth}}(l)$ measures how well the labeling agrees with our smoothness assumption. $E'_{\text{smooth}}(l)$ has the form

$$E'_{\text{smooth}}(l) = \sum_{(p,q)\in\mathscr{C}} \eta_l(l_p, l_q) \cdot \eta_d(d_p, d_q), \tag{4.8}$$

where $\mathscr{C}$ is the set of 4-connected neighbors in (θ, v), η_l is a monotonically increasing function of the distance between cameras, and η_d is a monotonically decreasing function of disparity differences that lowers smoothness costs at high depth discontinuities.

$$\eta_l(l_p, l_q) = min(|l_p - l_q|, c_3), \tag{4.9}$$

$$\eta_d(d_p, d_q) = e^{c_4|d_p - d_q|}, \tag{4.10}$$

We use $c_3 = 2.0$ and $c_4 = -0.1$ for all of our examples. Our smoothness term encourages the camera labeling to align with depth discontinuities. Again, we use the same graph cut alpha-expansion algorithm to compute the labeling l.

Depth and Color for Subsequent Layers

Computing the depth and color of layers beyond the first one proceeds almost exactly as for the first layer. One difference is that we first remove from consideration any voxels (θ, v, d) for which the depth d is less than or equal to the depth at the corresponding pixel in the first layer $L_1(\theta, v)$.

The robust variance of the intensity that projects to the remaining voxels is computed using median absolute variance (MAD). Note, however, that due to occlusion by the first layer, many voxels behind the first layer will no longer be visible to any input camera. In addition, voxels that are visible through the cracks induced by depth discontinuities will typically be visible in only one camera. The optimizations for depth and color then proceed as before for the second, and if desired, third layer.

Refinement of the LDP

Note that during determination of the first layer, we assumed all voxels were visible. However, even the first layer induces self occlusion; thus we can refine the process in an EM-like iteration [89]. Assuming the depths of the first layer are correct, we recompute the DSI taking into consideration

visibility information. From this we recompute the first layer's geometry, and proceed through the complete construction of the LDP one more time.

4.4 Results

In this section, we demonstrate our technique on three examples (two outdoor and one indoor) with varying amounts of scene complexity. Results are shown in Figures 4.3-4.5. Reconstructions at the end of each figure show the benefits of multiple depth layers for avoiding holes vs. reconstructions of a single layer as in [59]. Readers are encouraged to check out the supplementary material, which contains the inputs and results at the original resolutions, as well as videos of rendering results [2].

All scenes were captured by the user holding the camera arm length away from the body and capturing approximately 20 (800×600 pixel)images along an arc with approximately 75% overlap. The scenes exhibit large depth variation, resulting in significant parallax and thus large occluded areas. Two of the scenes contain numerous small surfaces, such as leaves and branches for the outdoor scene and individual fruits and signs for the indoor scene.

The camera positions were estimated using an off-the-shelf structure from motion (SFM) system [87] which recovers both the intrinsic and the extrinsic parameters of the camera. We fit a circular arc to the recovered camera positions using least-squares.

We constructed LDPs with two different parameterizations. The first simple representation with all rays converging on a central point at the center of the camera arc. This is a cylindrical analogue to a Layered Depth Image. The second, and more successful parameterization, is the one described in the chapter, which is a layered version of a cylindrical analogue to a "pushbroom" panorama. Figures 4.3 and 4.4 show how the cylindrical pushbroom results in a better depth estimate due to the rays being in better proximity to the original images. We used 16 depth labels and computed 2-3 layers depending on scene complexity. Each LDP computation took 1-2 hours on a 3.2GHz computer with 2GB memory.

The first and simplest example is shown in Figure 4.3. It depicts a couple sitting on a wall with more distant trees, buildings and a mountain in the background. This is typical of many "tourist" shots of a family member standing in front of a vista.

---

[2] `http://grail.cs.washington.edu/projects/ldp`

As expected, the depth map results show a clear separation between the foreground couple and the background. The first layer depicts the scene (with depth) similar to a view from the center of the arc. The second layer includes only those pixels hidden by the first layer that are revealed as one would move the viewpoint along the camera arc. The sparsity of this second layer shows the efficiency in the LDP representation while allowing a viewing experience depicting significant parallax (see the videos in supplementary materials).

The second and third scenes, a garden in front of a house (Figure 4.4), and a pile of melons in a market (Figure 4.5) are significantly more complex. The reconstruction once again faithfully finds depths for both the first and hidden layers. The second layers are not as sparse in these scenes due to the many depth discontinuities; however, they are still quite sparse compared to a full panorama. A third layer was also generated for the second scene (see images in supplementary materials). It is much sparser than the second layer, yet is helpful for filling small cracks in rendering.

The size of an *uncompressed* LDP is less than twice as large as a normal panoramic image. The data includes the first layer which is equivalent in size to a panorama. The two depth layers are 4 bits each (for 16 disparity layers) but are spatially coherent. The second layer texture and depth typically contain significantly fewer pixels although there is some overhead encoding the sparsity.

4.5 Discussion

Our system currently computes each layer sequentially. Such ordering dependency decreases the robustness of the system if errors get introduced at an early stage. Iterative methods could potentially alleviate such problem by solving all layers simultaneously, although this would result in a higher computational cost.

We can achieve smoother looking results if we allow voxels to be partially opaque at the boundaries of objects. Adding a matting component as a post-processing step for each layer as was done in [107] would definitely help.

The back layers in our LDP representation are usually quite sparse, containing many small yet non-rectangular shaped regions. Standard compression techniques support such type of data, yet with some overhead. We expect to be able to exploit compression methods such as in [107], with the added benefit that each layer should help predict voxels seen in further layers. Finally, we

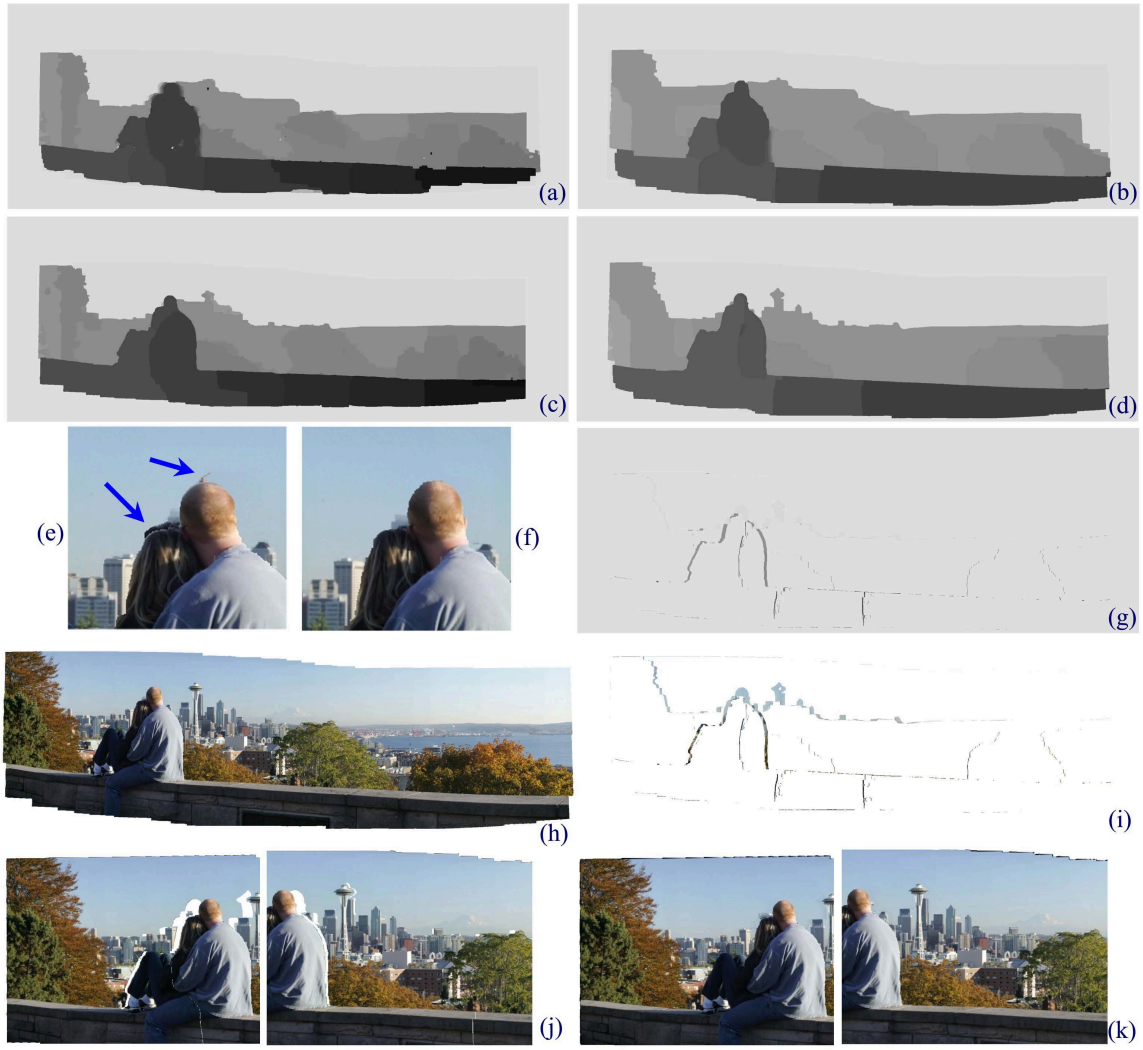


Figure 4.3: City scene and results: (a) and (b) show the first layer depths after one and two iterations generated from a global cylindrical center, (c) and (d) uses the arc-based pushbroom parameterization for depths, after one iteration and a second that accounts for visibility, (e) and (f) are details from the first layer textures comparison using the central and pushbroom parameterizations (note artifacts in the centered parameterization), (g) the second layer depth, (h) and (i) are the textures associated with the first and second layers, (j) two reconstructions from the first layer only showing obvious holes, and (k) the same reconstruction using two layers.

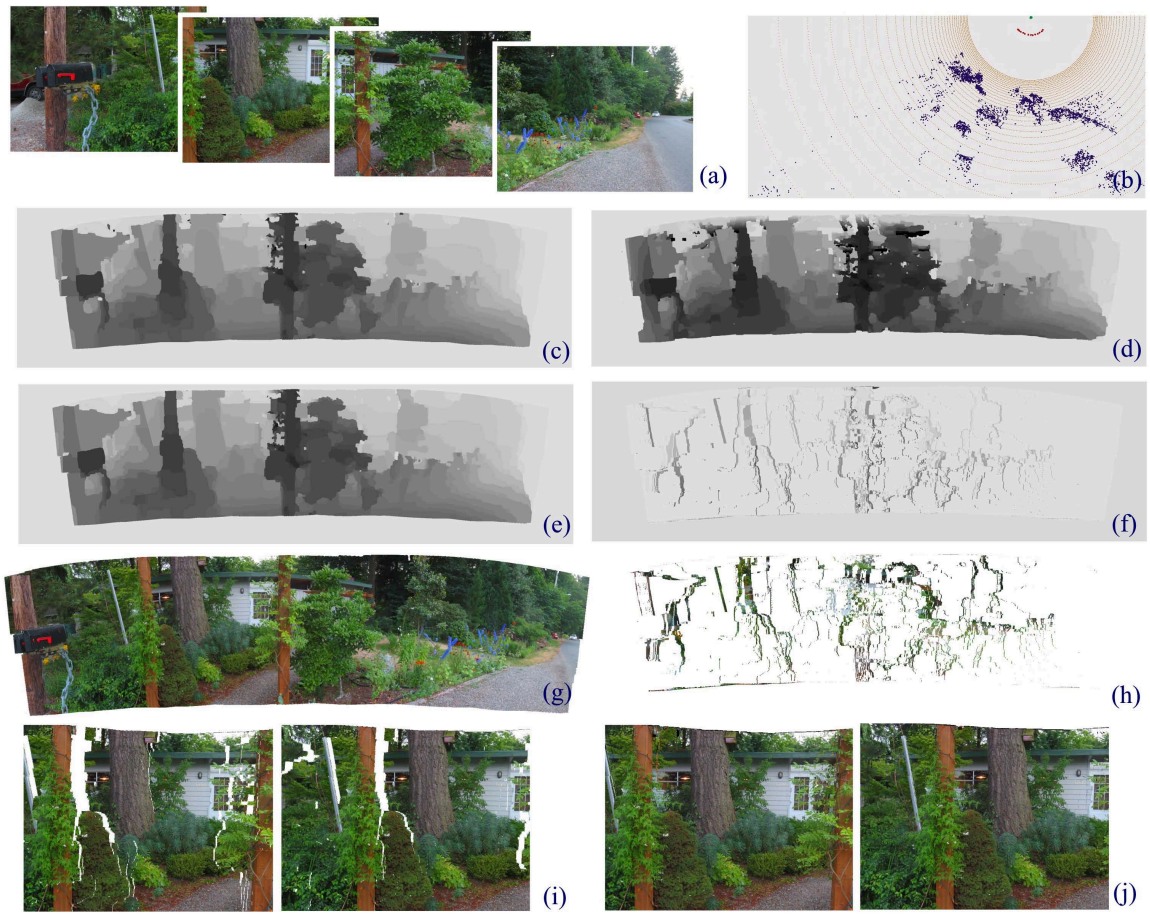


Figure 4.4: Garden scene and results: (a) are the input images (4 out of 14 shown here), (b) is a top-down view of camera positions and scene points recovered from SFM, (c) shows the front depth distribution after one iteration using our arc-based parameterization, (d) and (e) show the frontal depth distribution after multiple iterations and accounting for visibility, d) is generated from a global cylindrical center, and (e) uses our arc-based parameterization, (f) depths of second layer, (g) and (h) textures of first and second layer using our arc-based parameterizaton, (i) and (j) are two virtual views rendered with only the front layer (showing some holes), and all the layers.

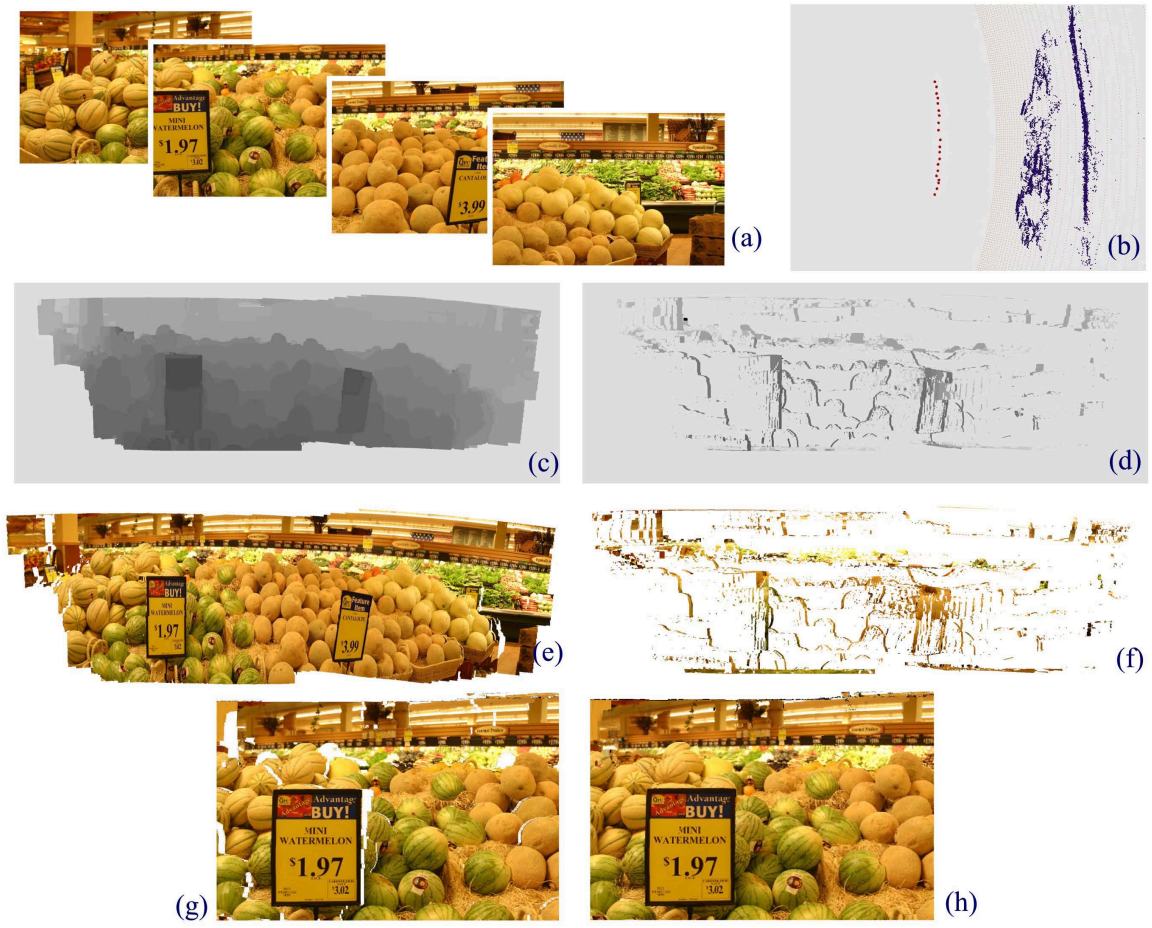


Figure 4.5: Market scene and results: (a) are the input images (4 out of 20 shown here), (b) camera positions and scene points recovered from SFM, (c) and (d) depth distributions of first and second layer, (e) and (f) texture of the first and second layer, (g) and (h) are virtual views rendered with only the front layer (showing some holes), and all the layers.

am exploring faster rendering methods to take advantage of current graphics hardware to make the viewing experience more interactive.

4.6 Conclusion and future work

In this chapter, we have developed a technique for creating a layered representation from a sparse set of images taken with a hand-held camera. This concise representation, which we call a layered depth panorama (LDP), allows the user to experience wide angle panoramas including the parallax associated with off-axis panning. The added 3D experience incurs a reasonable cost in terms of space efficiency (only about twice the size of the equivalent panorama). We formulate the problem of constructing the LDP as the recovery of color and geometry in a multi-perspective cylindrical disparity space. We introduce a new cylindrical pushbroom parameterization to closely follow the array of input images. Graph cut is leveraged to sequentially determine the disparity and color of each layer using multi-view stereo. As demonstrated both in the chapter and the supplementary videos from the project webpage, our approach is able to achieve high quality results on a variety of complex outdoor and indoor scenes.

Chapter 5

SPATIO-ANGULAR RESOLUTION TRADE-OFFS IN INTEGRAL PHOTOGRAPHY

Photography takes an instant out of time, altering life by holding it still.

—Dorothea Lange

5.1 Introduction

The *layered depth panorama* described in the previous chapter creates a global 3D presentation of a scene. While it is a compact representation, the LDP is prone to errors, especially at occlusion boundaries, leading to unsmooth viewing experiences. Moreover, the way the samples are captured limits the scenes to be static. In this chapter[1], we explore another dimension of parallax photography: design a high resolution handheld light field camera that captures multiple samples at once. Based on Georgeiv's new (integral) camera design with a bundle of lenses and prisms attached externally to the camera, we use an implicit geometry rendering technique to interpolate the light field with high quality, and achieve digital refocus and viewpoint shift.

The *light field* or *radiance density function* is a complete representation of light energy flowing along "all rays" in $3D$ space. This density is a field defined in the $4D$ domain of the optical phase space, the space of all lines in $3D$ with symplectic structure [37].

Conventional cameras, based on 2D image sensors, are simply integration devices. In a typical setting, they integrate over a $2D$ aperture to produce a $2D$ projection of the full $4D$ light field density. Integral Photography [62] was proposed almost a century ago to "undo" the integration and measure the complete $4D$ light field arriving at all points on a film plane or sensor.

As demonstrated by Levoy and Hanrahan [58] and Gortler *et al.* [36], capturing the additional two dimensions of radiance data allows us to re-sort the rays of light to synthesize new photographs,

[1]The work described in this chapter was originally presented as a paper [34] at EGSR 2006.

sometimes referred to as novel views. In the last decade, significant progress has been made in light field rendering to simulate a realistic camera with a finite aperture, producing depth of field effects. In this way, synthetic-aperture photography [58, 44] can compute photographs focused at different depths from a single light field, by simple numerical integration over the desired aperture.

Recently, Ng et al. [68] have shown that a full $4D$ light field can be captured even with a hand-held plenoptic camera. This approach makes light field photography practical, giving the photographer the freedom and the power to make adjustments of focus and aperture *after* the picture has been taken. In a way, it transfers the optics of the lens of the camera into the digital domain, greatly extending the types of postprocessing with software like Photoshop ^{TM}.

However, one drawback of the design of Ng et al. is that they require a large number of samples of the radiance: With their design, even with a 16-megapixel image sensor, the spatial resolution of the sampled light field is limited to 300×300 pixels.

This chapter surveys some of the previously proposed light field camera designs. Integral or light field photography is approached from the perspective of radiance analysis in geometrical optics. This provides a new way of looking at integral photography and the associated light field rendering. We then propose a new camera designs that produce higher spatial resolution than the camera of Ng et al., while trading-off the light field's angular sampling density. However, this lower angular resolution in the input is compensated for by inserting data synthesized by view interpolation of the measured light field.

We use three-view morphing to interpolate the missing angular samples of radiance. We demonstrate that such interpolated light fields generated from sparsely sampled radiance are generally good enough to produce synthetic aperture effects, new view synthesis and refocusing with minimal loss in quality. Moreover, with the same 16-megapixel sensor we are able to achieve a much higher spatial resolution of 700×700 pixels in the computed images.

We have implemented an integral camera that uses a system of lenses and prisms as an external optical attachment to a conventional camera. Using a computer vision based view interpolation algorithm, we demonstrate how our camera can be used to adjust the depth of field and synthesize novel views for scenes with high-speed action, which are impossible to do with conventional cameras.

5.2 Previous work on related cameras

Work done in integral / light field photography can be discussed in the framework of two types of design:

(1) The early works of Lipmann [62] and Ives [45] and others who use arrays of lenslets or pinholes placed directly in front of film creating multiple images on it, like an array of cameras. Optically similar to that is a physical array of digital cameras, which is the main approach used in current light field research. For one of the recent projects, see [100]. A related type of integral photography designs places an array of positive lenses in front of a conventional camera to create an array of real images between the lenses and the camera. Then the camera takes a picture focused on those images. See [70]. This approach is close to ours.

(2) We would like to consider the approach of Adelson et al. [3] and Ng et al. [68] as a second type. Effectively it is placing a big lens in front of the array of lenslets (or cameras) considered in the first approach, forming an image on the array of lenslets. Each lenslet itself creates an image sampling the angular distribution of radiance at that point, which corresponds to one single direction observed from multiple points of view on the main lens aperture. This is opposite to the first approach where each camera takes a real picture of the world. Spatial and angular dimensions are switched. A related technique is that of the Hartman-Shack sensor [95], which was also proposed a century ago to study wavefront shape in optics, with applications to Astronomy, medical studies of the eye and others.

The two types of previously proposed integral cameras can be viewed as sharing one goal – increasing angular resolution of the measured light field, which often comes at the cost of spatial resolution of the final $2D$ image generated by the system. In this chapter, we explore the trade-off between angular and spatial resolution and show that for typical scenes it is advantageous to use higher spatial resolution at the cost of angular resolution.

5.3 Optical designs

5.3.1 The plenoptic camera

In the plenoptic camera designs proposed first by Adelson et al. [3] and implemented and studied in detail recently by Ng et al. [68], the light field is captured by an array of 296^2 lenslets inside

a conventional camera. Each lenslet in this setting corresponds to a little camera producing an approximately 14×14 pixel image of the main lens aperture. Each pixel within that small image corresponds to one viewpoint on the aperture, while different lenslets correspond to different pixels in the final image. The result is an approximately 100-view light field with 90,000 pixels per view. (The number of effective views is 100 instead of 14^2 due to losses, which will be discussed later.)

Unfortunately, from the standpoint of professional photographers this system produces images with very low spatial resolution. An obvious way to improve upon this would be to use more lenslets (for example, 1,000,000), with fewer views/pixels under each lenslet (for example, 16).

The problem with such a design is that each small image of the main lens aperture created by a lenslet includes pixels at the aperture boundary that are either lost entirely, or noisy. Such a boundary pixel is only partially covered by the image of the aperture. In order to reconstruct the true irradiance corresponding to the illuminated part of that pixel we need to know exactly what percentage of it has been illuminated, and correct for that in software. In other words, we need very precise calibration of all pixels in the camera. Furthermore, captured pixel values are affected by small misalignments of less than a micrometer. A misalignment of a micrometer can change a boundary pixel value by more than 10%. This problem gets worse as the lenslets get smaller. In the limiting case of a 4×4 pixel image under each lenslet, all the pixels are boundary pixels (considering the Bayer array), providing no reliable information at all.

5.3.2 How do we capture 4D radiance with a 2D sensor?

A solution to this problem is to organize the $4D$ data needed for representing the light field as a field of $2D$ images, or a $2D$ array of $2D$ images. This can be done in two different ways, which will be discussed using the case of a 1D image detector, as shown in Figure 5.1.

If optical phase space ("light field space") were 2 dimensional, we would have one space dimension, x, and one angular dimension, θ. Radiance is measured at a point on a plane (line in our case) perpendicular to the optical axis t, where the coordinate x on the line specifies how far from the optical axis that point is. The ray through that point intersects the line at some angle, and the tangent of that angle will be called θ. See Figure 5.1a which shows radiance that smoothly changes in both directions, spatial (changing color) and angular (changing luminance). Because of the redundancy

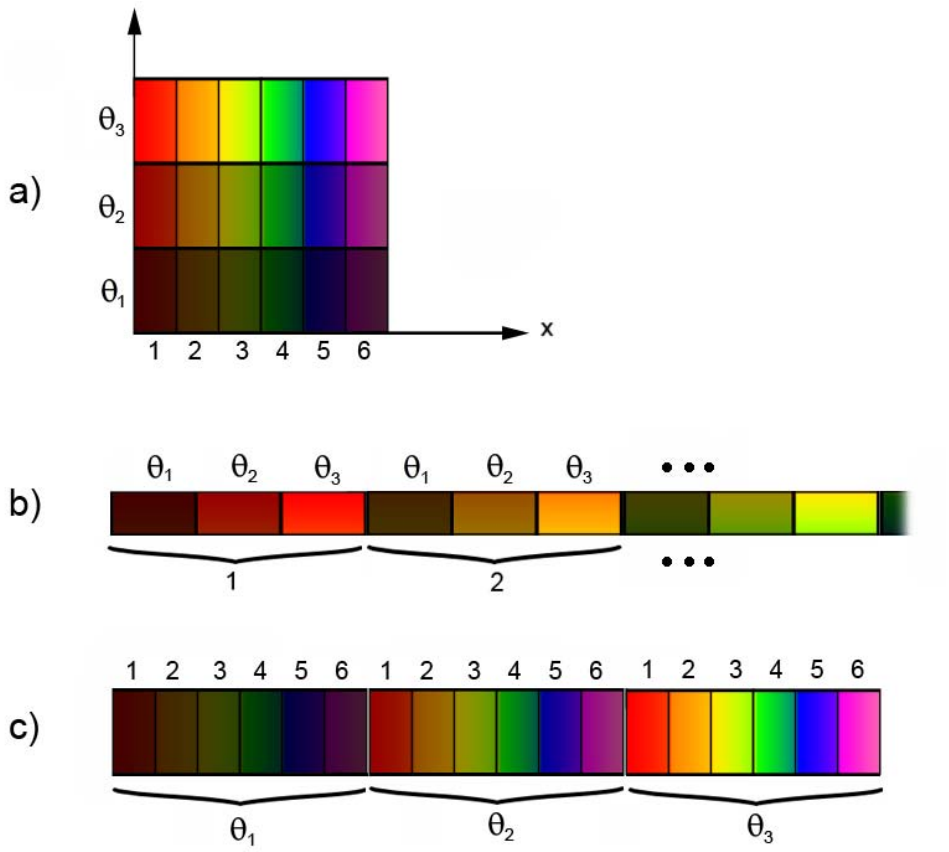


Figure 5.1: (a)Partitioning optical phase space (x, θ) into sampling regions for the light field camera. (b) The "Plenoptic" way of sampling. (c) The "Integral Photography" way of sampling.

of light fields noticed in early work (like [58] for example), we don't want to densely sample the dimension along which radiance is almost constant. In the plenoptic camera design this is the angular dimension. Adopting this space-saving approach we end up with the partitioning of the light space into rectangles or *radiance pixels*, long in the direction of θ, and short in the direction of x. (See Figure 5.1a) Radiance pixels are sparse (only 3 samples) along the angular direction, but a lot more samples (6) along the spatial direction. This is the type of partitioning of optical phase space practically chosen in most light field cameras.

In our example the sensor measuring the radiance is $1D$. In order to fit the 2D phase space of Figure 5.1a into $1D$, we need to rearrange light field data to fit in one single row, as a 1D array. The "plenoptic camera arrangement" Figure 5.1b puts all angular samples for pixel 1 (the first column)

in a row, then all angular samples for pixel 2 (the second column) next to them, and so on. Now we encounter the problem at the boundaries discussed above. Out of 3 angular samples only one is left intact. The left and right pixels in each sub-image, θ_1 and θ_3, are lost.

To avoid this problem we rearrange the optical data as in Figure 5.1c. All spatial samples at a given angle θ are grouped together. In this way we get a coherent image of lots of pixels representing θ_1-samples, then next to them we place all θ_2 samples, and so on. Again, boundary pixels are lost, but now they are much fewer as a percentage of all pixels in a sub-image.

Obviously, the method is more efficient. Also it is not new. Any array of conventional cameras samples the light field exactly in that way. What we gain here is a theoretical understanding why this approach is better and also, a chance to build an extensible framework of new camera designs (section 3.3).

Next we propose our theoretical model, which leads to improved camera designs, that greatly reduce the number of exposed boundary pixels between sub-images to virtually eliminate wasted pixels and make sparse radiance sampling practical.

5.3.3 Designs

In a traditional approach to the light field we would use an array of cameras to capture the above array of 2D images as in Figure 5.1c. For example, Figure 5.3a represents the arrangement of lenses in Integral Photography which produces that result. We propose a series of equivalent camera designs based on a formula from *affine optics*, which will be derived next. The proposed affine optics treatment of optical phase space can be used in other light field constructions.

Conventional Gaussian optics is linear in the following sense. All the equations are written relative to the *optical axis*, which plays the role of origin (or zero point) in optical phase space ("light field space"), treated as a vector space. In more detail, we consider a plane perpendicular to the optical axis and choose Cartesian coordinates x, y in the plane. A ray intersecting this plane at a given point is defined by the tangents of the two angles. We call those tangents θ and ϕ. (Note: θ and ϕ are not angles.) In light field terminology this is a version of the popular two plane parametrization. The light field or radiance density is a function in this $4D$ vector space, where the zero is defined as the point on the optical axis, with both $\theta = 0$ and $\phi = 0$. This is a typical treatment

in optics, see for example [35]. Now, a lens is defined by the linear transform:

$$\begin{pmatrix} x' \\ \theta' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \begin{pmatrix} x \\ \theta \end{pmatrix}, \tag{5.1}$$

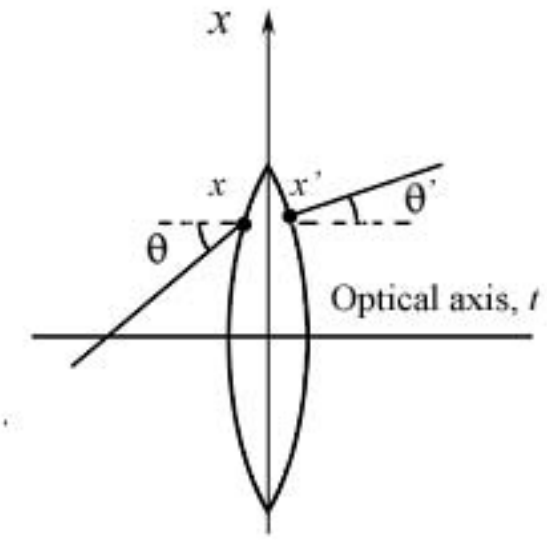


Figure 5.2: Light field transformation at the plane of a lens.

and a space translation of the light field from one plane to another separated by distance T is represented by the linear transform

$$\begin{pmatrix} x' \\ \theta' \end{pmatrix} = \begin{pmatrix} 1 & T \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ \theta \end{pmatrix}. \tag{5.2}$$

These and all other transforms used in Gaussian optics are *linear transforms* relative to the optical axis.

Unfortunately, in linear optics there is no representation for a lens shifted from the optical axis, as we would need in integral / light field photography. For example, in Figure 5.3a one would pick an arbitrary optical axis through one of the lenses, and then all the other lenses would be considered shifted relative to the optical axis, and not representable as linear transforms in this coordinate

system. In linear optics we have no way of writing a mathematical expression for the radiance valid at the same time everywhere in a light field camera.

To derive a rigorous description of this new situation we need a general mathematical framework that extends linear optics into what should be called *affine optics* (it adds translations to linear optics). A typical element representing an affine transform would be the prism. It tilts all rays by the same *fixed* angle α that depends only on the prism itself. Expressed in terms of the ray coordinates the prism transform is:

$$\begin{pmatrix} x' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ \theta \end{pmatrix} + \begin{pmatrix} 0 \\ \alpha \end{pmatrix}. \tag{5.3}$$

Now, a lens shifted a distance s from the optical axis would be treated as follows:

(1) Convert to new lens-centered coordinates by subtracting s.

$$\begin{pmatrix} x' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ \theta \end{pmatrix} - \begin{pmatrix} s \\ 0 \end{pmatrix} \tag{5.4}$$

(2) Apply the usual linear lens transform.

$$\begin{pmatrix} x'' \\ \theta'' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \begin{pmatrix} x-s \\ \theta \end{pmatrix} \tag{5.5}$$

(3) Convert to the original optical axis coordinates by adding back s.

$$\begin{pmatrix} q''' \\ \theta''' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \begin{pmatrix} x-s \\ \theta \end{pmatrix} + \begin{pmatrix} s \\ 0 \end{pmatrix} \tag{5.6}$$

We can re-write this equation as:

$$\begin{pmatrix} q''' \\ \theta''' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \begin{pmatrix} x \\ \theta \end{pmatrix} + \begin{pmatrix} 0 \\ \frac{s}{f} \end{pmatrix}. \tag{5.7}$$

Thus, we see that a shifted lens is equivalent to a lens with a prism. This result will be used to show that our proposed new designs are optically equivalent to arrays of cameras. This equivalence is exact.

Based on equation 7, Figure 5.3a is optically equivalent to Figure 5.3b. The array of shifted lenses has been replaced with one central lens and a set of prisms. Equation 7 represents the relations between focal lengths, shifts and prism angles that make the two systems equivalent.

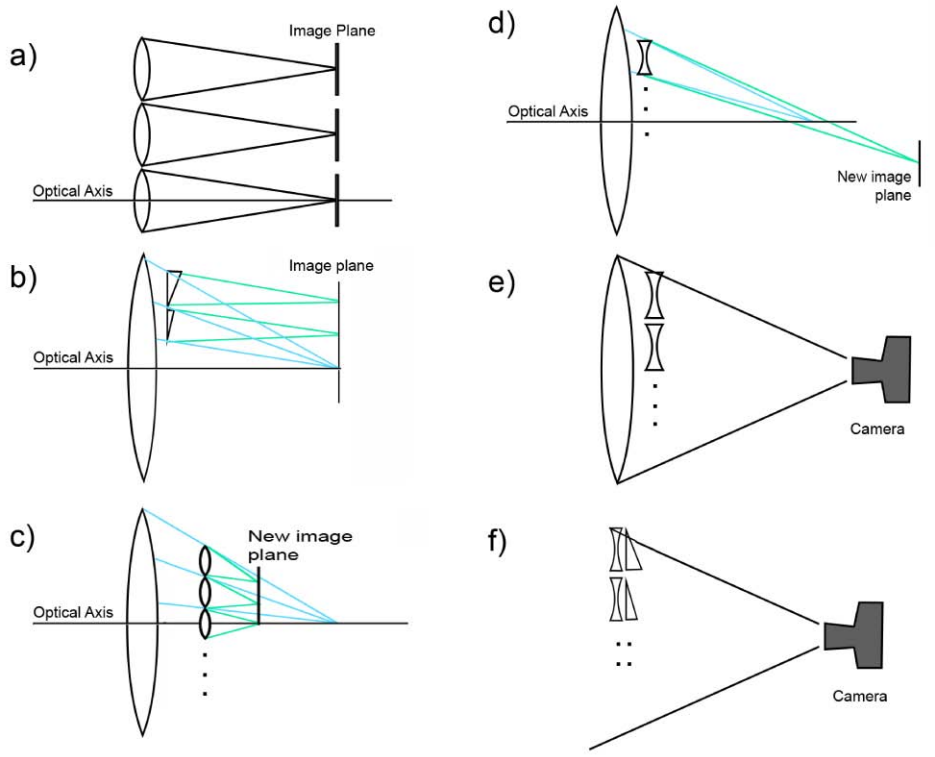


Figure 5.3: Six designs of light field cameras. (a) Integral photography. (b) One lens and multiple prisms. (c) Main lens and a lens array - used in integral photography. (d) Main lens and an array of negative lenses. (e) Same as 5, only implemented as *external* for the camera. (f) Example of external design of negative lenses and prisms that has no analog as internal.

From the point of view of simple intuition, different prisms tilt rays [that would converge to the same point] differently. Now they converge to different locations in the image plane forming different sub-images. Those different sub-images are of the type Figure 5.1c, which is the more efficient design. (Note that intuition is not sufficient to convince us that this approach is exact. Intuition only tells us that "this should work at least approximately".)

Figure 5.3c is also self-explanatory from the point of view of intuition. The additional small lenses focus light rays closer than the original focal plane of the main lens. Thus they form individual images instead of being integrated into one image as in traditional one-optical-axis cameras. Again, this is "at least approximately correct" as a design, and we need formula 7 to prove that it is exactly correct and to find the exact values of the parameters (in terms of equivalence with Figure 5.3b.)

In more detail, each of the shifted lenses in Figure 5.3c is equivalent to a big lens on the optical axis and a prism. The big lens can be combined in one with the main lens, and we get equivalence with Figure 5.3b.

Figure 5.3d is similar only with negative lenses. The construction should be obvious from the figure.

Figure 5.3e describes a design external to the camera. It is used in this chapter for the examples with 20 negative lenses. The whole optical device looks like a telephoto lens, which can be added as an attachment to the main camera lens. See Figure 5.6.

Figure 5.3f is our best design. We have implemented a version made up of 19 lenses and 18 achromatic prisms. See Figure 5.4. It is light weight and efficient compared to similar design with a big lens. In general, an array of prisms is cheaper than a big lens.

Figure 5.4: Our optical device consisting of lens-prism pairs.

As in the design of Figure 5.3e, the camera sees an array of virtual images created by the negative lenses, in front of the optical device and focuses upon them. The prisms shift these images appropriately, so the result is as if the scene is viewed by an array of parallel cameras. Again the idea is that a camera with a lens shifted from the optical axis is equivalent to a camera on the axis, a lens and a prism. We should also note that practically, the role of the negative lenses is to expand the field of view in each image, and that the prisms can be viewed as making up a Frensel lens focused

at the camera's center of projection. Other external designs are possible with an array of positive lenses creating real images between the array of lenses and the main camera lens.

We have built prototypes for two of the designs: Figure 5.3e with 20 lenses, cut into squares, and Figure 5.3d with 19 lenses and 18 prisms. Because of chromatic problems with our prisms currently we produce better results with the design on Figure 5.3e, which is used to obtain the results in this chapter. Also, our lenses and prisms for the design Figure 5.3d are not cut into squares, which leads to loss of pixels even with hexagonal packing, Figure 5.4. We are planning to build a version based on quality optical elements.

5.4 Synthetic aperture photography

Light fields can be used to simulate the defocus blur of a conventional lens, by re-projecting some or all of the images onto a (real or virtual) focal plane in the scene, and computing their average. Objects on this plane will appear sharp (in focus), while those not on this plane will appear blurred (out of focus) in the resulting image. This synthetic focus can be thought of as resulting from a large-aperture lens, the viewpoints of light field images being point samples on the lens surface. This method was proposed by Levoy and Hanrahan [58], first demonstrated by Isaksen et al. [44], and goes under the name of synthetic aperture photography in current work [96, 100]. It creates a strong sense of 3D; further, summing and averaging all the rays serves as a sharpen filter, hence the resulting image has superior signal-to-noise ratio(SNR) compared to the original inputs.

The projection and averaging approach to synthetic aperture requires a dense light field. However, we are working with relatively sparse samplings comprised of 20 images. Simply projecting and averaging such an image set results in pronounced ghosting artifacts, essentially the result of aliasing in the sampled light field. Stewart et al. [88] explore reconstruction filters to reduce the aliasing in undersampled light fields; however, even with 256 images some artifacts remain.

Instead, we address the aliasing problem by generating more camera views than those provided directly by the camera array through view morphing [79]. This is equivalent to generating a *synthetic light field* by carefully interpolating between the samples in our sparse camera data. Fundamentally, this is possible because of the well known "redundancy" of the light field [58], which in the Lambertian case is constant along angular dimensions at each point on the surface that is being observed.

In the following subsections, we describe our method for filling out the light field and for using it to generate synthetic aperture images.

5.4.1 Synthetic light field by tri-view morphing

Our sampling consists of viewpoints which lie on a grid. We tessellate it into a triangular mesh, as illustrated in Figure 5.5. Our goal is to be able to fill in arbitrary viewpoints within the grid. As described below, we do this by computing warps that allow view morphing between each pair of views connected by an edge. These warps are then combined to allow barycentric interpolation of views within each triangle of viewpoints.

View morphing with segmentation-based stereo

View morphing [79] is a method for interpolating two reference images to generate geometrically correct in-between views from any point on the line connecting the two initial centers of projection. To achieve this effect, a correspondence is needed between the pair of images.

Recently, color segmentation approaches have gained in popularity for dense correspondence computation. They use color discontinuities to delineate object boundaries and thus depth discontinuities. Also, they model mixed color pixels at boundaries with fractional contributions(matting) to reduce artifacts at depth discontinuities.

I build on the segment-based optical flow work of Zitnick et al. [106]. The idea behind their method is to model each pixel's color as the blend of two irregularly-shaped segments with fractional contributions α and then solve for a mutual segmentation between a pair of images which gives rise to segments with similar shapes and colors. We modify their flow algorithm in two ways. First, between each pair of images, we require the matched segments to lie along epipolar lines. Second, we simultaneously compute epipolar flow between an image and two neighbors defining a triangle, so that the segments in each image are consistent between neighbors needed for tri-view morphing, described in the next subsection.

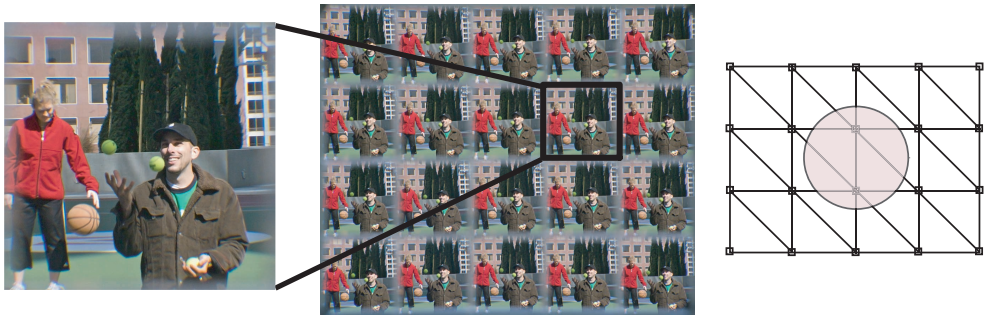


Figure 5.5: The set of 20 images (middle) is a sparse light field captured with our camera. A close-up of one of the images is shown on the left. The hazy edges are defocused images of the boundaries of the lenses; for the results in this paper, we discard these contaminated pixels. Each vertex on the right represents one camera view. We decompose the camera plane into triangles illustrated on the right. Any novel camera view inside these triangles can be synthesized using tri-view morhping. The circular region represents a possible virtual aperture we want to simulate.

Tri-view blending

Seitz et al. [79] demonstrated that any linear combination of two parallel views gives a valid interpolated projection of the scene. Multiple image morphing [33] or Polymorph [55] has been used to extend 2-view morphing to morphing among 3 or more views and into a complete geometrically correct image based 3D system [32]. Tri-view morphing [102] is a more recent system for multi-image morphing. It makes use of the trifocal tensor to generate the warping transforms among three views.

Here we summarize our method for tri-view morphing within triangles on the camera grid. Given three images I_1, I_2 and I_3, we morph to the target image I_s using barycentric coefficients λ_1, λ_2 and λ_3. Let W_{ij} be the warping vector field (or "flow") from image I_i to image I_j, according to the disparity map from I_i to I_j obtained using the segmentation-based stereo algorithm from Section 5.4.1. Ideally, it will convert image I_i into image identical to I_j. In general, warping any image I by a vector field W will produce a new image denoted as $I(W)$. We warp each of the input images to I_s using affine (barycentric) combination of the three vector fields, and then we blend them together based on the same barycentric coefficients.

$$I_{out} = \sum_{i=1}^{3}(\lambda_i I_i(\sum_{j=1}^{3}\lambda_j W_{ij}))$$

Note: we generally sample within the camera grid, so that the desired image is inside of a triangle defined by the three input images I_i, and then $\lambda_i \geq 0$ and $\Sigma_{i=1}^{3}\lambda_i = 1$. Extrapolation outside the grid is also feasible to some extent, in which case one or more barycentric coordinates will be negative.

5.4.2 Synthetic aperture rendering

To simulate the defocus of an ordinary camera lens, we first define an aperture location and size on the camera grid (see Figure 5.5). Then, we densely sample within this aperture using tri-view morphing. Finally, we determine an in-focus plane, project all images within the aperture onto this plane, and average.

5.5 Results

We implemented our camera design with a working prototype. We have been testing our prototype extensively on a wide range of regular photographic scenarios. Results shown here would highlight the capabilities of synthetic aperture photography.

5.5.1 Camera

The camera design we are using here is Figure 5.3e, with an array of 4×5 negative lenses cut into squares and attached to each-other with minimal loss of space. Before gluing together the lenses were placed with their flat side facing flat piece of glass, so we believe they are very well aligned on a plane and parallel to each-other. In this way, our optical design can be treated as a planar camera array with all views being parallel to each other. Since all lenses have the same focal length, -105 mm, their focal points are on one plane. This plane is perpendicular to the direction of view to the precision of lens manufacturing.

We calibrate the camera centers using an off-the-shelf structure from motion(SFM) system [17] which recovers both the intrinsic and the extrinsic parameters of the camera. For the purposes of synthetic aperture, one could also pursue the calibration method discussed by Vaish et al. [96], in which relative camera positions are recovered.

5.5.2 Renderings

With our camera prototype, 20 views are captured at a single exposure, each view has roughly 700 by 700 pixels. 24 triangles are formed to cover the entire viewing space. The relative locations of all the cameras are recovered by running SFM on the 20 images. Once the size, the location and the shape of the virtual lens is specified, we densely sample view points using our tri-view morphing algorithm at one reference depth. All examples shown here take around 250 views. Sweeping through planes of different depths corresponds to shifting all views accordingly. By shifting and summing all the sampled views, we compute synthetic aperture images at different depths.

In the seagull example of Figure 5.7, we demonstrate refocusing at three different depths from near to distant. The reader would gain more intuition on the quality of the refocusing from the supplementary videos on the project webpage[2].

For the juggling example (with input images in Figure 5.5) we present 3 sets of results, see Figure 5.8. On the first row we show three synthesized novel views inside a triangle of input images. Despite the slight motion blur of the tennis balls, the interpolated views look realistic with clean and smooth boundaries. The second row shows three synthetic aperture images focusing at three different depths. The third row shows results focused on the juggler with varying depth of field. The effect is created with varying aperture size. The left image and the middle image have the exact same virtual aperture. Notice that the number of samplings makes a huge difference as the left uses only 24 views, thus reveals strong aliasing in blurred regions; while the middle image uses over 200 views. The right image shows an even larger aperture that goes beyond the area of the input camera array, showing that view *extrapolation* also produces reasonable results.

The reader is encouraged to see the electronic version of this chapter for high resolution color images. The supplementary videos show sequences of synthetic aperture images as the focal plane sweeps through a family of planes that spans the depths of the scenes. The sharpness of objects on the focal plane together with the smooth blur indicates the accuracy of our technique. The size of the virtual aperture used in the juggling scene (Figure 5.8) and the seagull example (Figure 5.7) is about one quarter of the entire viewing region.

---

[2] `http://grail.cs.washington.edu/projects/lfcamera`

5.6 Conclusion and future work

In this chapter, we described several practical light field camera designs with the specific application to synthetic aperture photography. We compared the two ways of approaching light field capture, and argued that an important point in the camera space for integral photography is in a sparse sampling of the angular dimensions of the light field in order to achieve better spatial resolution. We explored the first (integral) camera design which produces results of higher resolution than the second ("plenoptic") design when both are based on sparse sampling with same resolution sensor.

We drew strength from the state-of-the-art computer vision technique as a post-processing tool to interpolate or "fill in" the sparse light field. We demonstrated the effectiveness of this framework with realistic refocusing and depth of field results. Averaging many intermediate views not only reduces sampling errors, but also makes errors caused by stereo matching much more tolerable, which was one of the insights of this approach.

Most of the computing cycles are spent on generating in-between views. An analysis on the sampling bounds would be helpful for better efficiency. How densely does one have to sample the viewing space in order to create non-aliased results? Furthermore, it would be interesting to explore the possibility of skipping the entire process of view interpolation and realizing refocusing directly from the disparity map.

We used 20 views in our camera implementation. For typical scenes we got good results, but for scenes with more complex $3D$ structure we began to observe artifacts. A detailed study of the relationship between optimal sampling rate and $3D$ scene complexity would be useful. It might be possible to dynamically adjust the number of captured views based on scene geometry, so that results with optimal resolution are achieved.

In the last few years, we have experienced the rapid rise of digital photography. Sensors are gaining in resolution. Capturing a light field with a single exposure becomes achievable in a realistic hand-held camera. This adds a whole new dimension to digital photography with the possibility of capturing a sparse light field with a compact camera design, and later post-processing based on computer vision. We hope that this work will inspire others to explore the possibilities in this rich domain.

Figure 5.6: Our sparse light field camera prototype, with 2 positive lenses and an array of 20 negative lenses in front of a conventional camera.

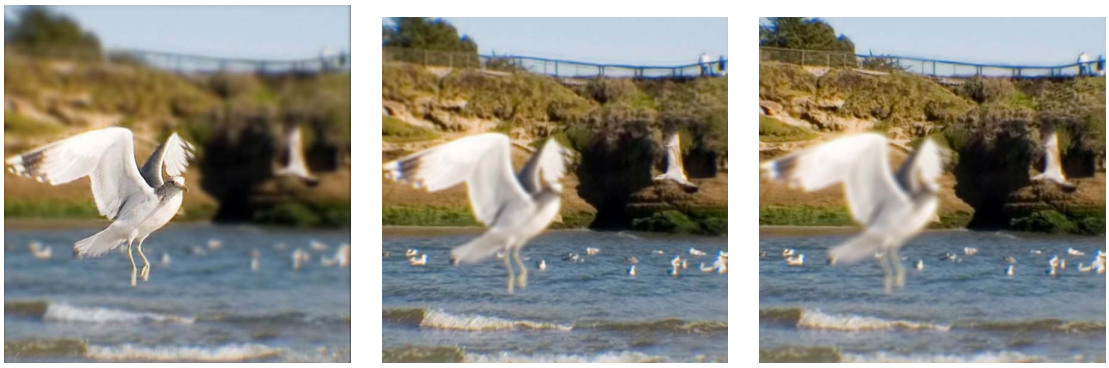
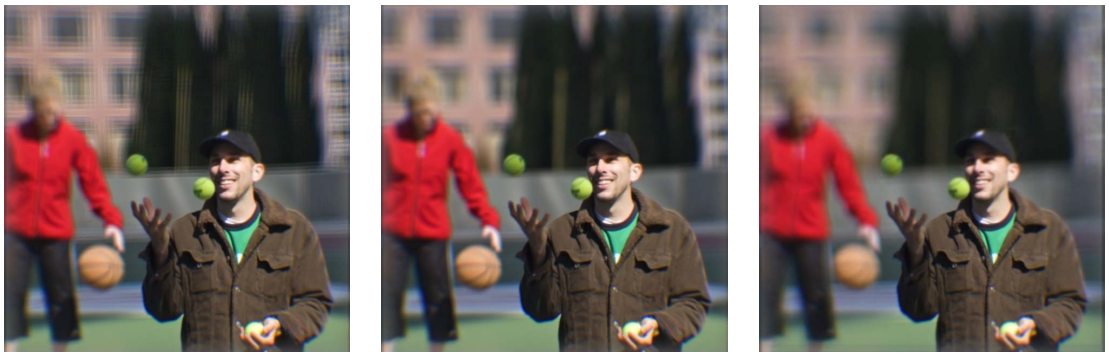


Figure 5.7: Synthetic aperture photography of flying birds. Refocusing to different depths.

(a) Three novel views generated using tri-view morphing.

(b) Synthetic aperture results with the focal plane moving from near to far.



(c) Synthetic aperture results with varying depth of field. (Left image demonstrates sparse sampling.)

Figure 5.8: Synthetic aperture photography of human motion focusing at different depths.

Chapter 6

PARALLAX PHOTOGRAPHY: CREATING 3D CINEMATIC EFFECTS FROM STILLS

I treat the photograph as a work of great complexity in which you can find drama. Add to that a careful composition of landscapes, live photography, the right music and interviews with people, and it becomes a style.

—Ken Burns

6.1 Introduction

In the previous chapter, we present a segmentation based tri-view morphing approach to interpolate the light field from an array of cameras on a plane. We extend this approach to off-plane views in this chapter. Moreover, rather than creating refocus and viewpoint shift through user interface, we present algorithms to automatically create smooth camera path exhibiting parallax while satisifying the cinematic conventions.

Documentary filmmakers commonly use photographs to tell a story. However, rather than placing photographs motionless on the screen, filmmakers have long used a cinematic technique called "pan & zoom," or "pan & scan," to move the camera across the images and give them more life.[1] The earliest such effects were done manually with photos pasted on animation stands, but they are now generally created digitally. This technique often goes by the name of the "Ken Burns effect", after the ocumentary filmmaker who popularized it.[2]

In the last few years, filmmakers have begun infusing photographs with an added sense of realism by injecting depth into them, thus adding *motion parallax* between near and far parts of the scene as the camera pans over a still scene. This cinematic effect, which we will call *3D pan & scan*, is now used extensively in documentary filmmaking. 3D pan & scan is now replacing traditional,

[1] http://blogs.adobe.com/bobddv/2006/09/the_ken_burns_effect.html

[2] http://en.wikipedia.org/wiki/Ken_Burns_Effect

2D camera motion in documentary films, as well as TV commercials and other media, because it provides a more compelling and lifelike experience.

However, creating such effects from still photos is painstakingly difficult. To create the effect from a still image, a photo must be manually separated into different layers, and each layer's motion animated separately. In addition, the background layers need to be painted in by hand so that no holes appear when a foreground layer is animated away from its original position.[1]

In this chapter we look at how 3D pan & scan effects can be created much more easily, albeit with a small amount of additional input. Indeed, our goal is to make creating such cinematic effects so easy that amateurs with no drawing skills or cinematic expertise can create them with little or no effort. To that end, we propose a solution to the following problem: given a small portion of a lightfield [58, 36], produce a 3D pan and scan effect automatically (or semi-automatically if the user wishes to influence its content). In our specific implementation, the input lightfield is captured with and constructed from a few photographs from a hand-held camera.

The 3D pan & scan effects are generated to satisfy three main design goals:

1. the results should conform to the cinematic conventions of pan & scan effects currently used in documentary films,

2. the produced effects should apply these conventions in a fashion that respects the content and limitations of the input data, and

3. the result should be viewable and editable in real-time, and temporally coherent.

Our approach takes as input a lightfield representation that contains enough information to infer depth for a small range of viewpoints. For static scenes, such lightfields can be captured with a standard hand held camera [73] by determining camera pose and scene depth with computer vision algorithms, namely structure-from-motion [39] and multi-view stereo [78]. Capturing and inferring this type of information from a single shot has also received significant attention in recent years, and many believe [57] that the commodity camera of the future will have this capability. There are now several camera designs for capturing lightfields [68, 34, 60], from which scene depth can be estimated [78]. Other specialized devices capture single viewpoints with depth. These include structured light scanners [11] — among them real-time time-of-flight imagers like the ZCam [1] and depth-from-defocus with controlled illumination [64] — and coded imaging systems [56, 97].

Lightfields with depth have the advantage that they can be relatively sparse and still lead to rea-

sonably high quality renderings [36, 60], at least for scenes that do not have strong view-dependent lighting effects (e.g., mirrored surfaces). However, such sparse inputs, taken over a small spatial range of viewpoints or even a single viewpoint, present limitations: novel viewpoints must stay near the small input set, and even then some portions of the scene are not observed and thus will appear as holes in new renderings. Our approach is designed to take these limitations into account when producing 3D pan & scan effects.

Our solution processes the input to produce 3D pan & scan effects automatically or semi-automatically to satisfy the three design goals listed above. To achieve the first goal, we describe a simple taxonomy of pan & scan effects distilled from observing 22 hours of documentary films that heavily employ them. This taxonomy is organized according to the number of "subjects of interest" (zero, one, or two). This taxonomy enables various communicative goals, such as "create an establishing shot of the entire scene" (e.g., if there are no special subjects of interest), or "transition from the first subject of interest to the second" (if two such subjects are identified). Second, we describe algorithms for analyzing the scene and automatically producing camera paths and effects according to our taxonomy. A face detector [98] automatically identifies subjects of interest, or the user can identify them interactively. Our solution then applies the appropriate effect by searching the range of viewpoints for a linear camera path that satisfies cinematic conventions while avoiding missing information and holes, and maximizing the apparent parallax in the 3D cinematic effect. Third, we introduce GPU-accelerated rendering algorithms proposed by Colburn with several novel features: (1) a method to interleave pixel colors and camera source IDs to multiplex rendering and guarantee optimal use of video memory; (2) the first GPU-accelerated version of the soft-z [74] technique, which minimizes temporal artifacts; and (3) a GPU-accelerated inverse soft-z approach to fill small holes and gaps in the rendered output.

In the rest of this chapter we describe the components of our approach, which include a taxonomy of the camera moves and other image effects found in documentary films (Section 6.3); techniques for automatically computing 3D pan & scan effects that follow this taxonomy (Section 6.4); a brief overview of our representation of a lightfield with depth and how we construct it from a few photographs (Section 6.5) using multi-view stereo(Section 6.6); and finally two rendering algorithms, both real-time (Section 6.7.1) and off-line (Section 6.7.2). Finally, we evaluate our approach in two ways. First, we numerically evaluate the success of our solution on over 200 input

datasets, and give a breakdown on the various sources of error (Section 6.8). Second, we support the motivation of this work with the results of a user study with 145 subjects that compares the effectiveness of 3D pan & scan effects versus their 2D counterparts (Section 6.9).

6.2 Related work

The process of creating a 3D pan & scan effect is challenging and time consuming. There are number of techniques that help in creating 3D fly-throughs from a single image, such as Tour Into the Picture [42] and the work of Oh *et al.* [69], though the task remains largely manual. Hoiem *et al.* [41] describe a completely automatic approach that hallucinates depths from a single image. While their results are impressive, substantially better results can be obtained with multiple photographs of a given scene.

To that end, image-based rendering (IBR) techniques use multiple captured images to support the rendering of novel viewpoints [49]. Our system builds a representation of a small portion of the 4D light field [58, 36] that can render a spatially restricted range of virtual viewpoints, as well as sample a virtual aperture to simulate depth of field. Rendering novel viewpoints of a scene by re-sampling a set of captured images is a well-studied problem [19]. IBR techniques vary in how much they rely on constructing a geometric proxy to allow a ray from one image to be projected into the new view. Since we are concerned primarily with a small region of the light field, we are able to construct a proxy by determining the depths for each of the input images using multi-view stereo [7], similar to Heigl *et al.* [40]. This approach provides us the benefits of a much denser light field from only a small number of input images. Our technique merges a set of images with depth in a spirit similar to the Layered Depth Image (LDI) [81]. However, we compute depths for segments, and also perform the final merge at render time. Zitnick *et al.* [107] also use multi-view stereo and real-time rendering in their system for multi-viewpoint video, though their constraint that cameras lie along a line allows some different choices. Most IBR systems are designed to operate across a much wider range of viewpoints than our system and typically use multiple capture devices and a more controlled environment [94, 57]. To date, the major application of capturing a small range of viewpoints, such as ours, has been digital re-focusing [64, 67].

A number of papers have used advanced graphics hardware to accelerate the rendering of im-

agery captured from a collection of viewpoints. The early work on light fields [58, 36] rendered new images by interpolating the colors seen along rays. The lightfield was first resampled from the input images. The GPU was used to quickly index into a lightfield data structure. In one of the early works leveraging per-pixel depth, Pulli *et al.* [74] created a textured triangle mesh from each depth image and rendered and blended these with constant weights. They also introduced the notion of a soft-z buffer to deal with slight inaccuracies in depth estimation. We take a similar approach but are able to deal with much more complex geometries, use a per-pixel weighting, and have encoded the first soft-z into the GPU acceleration. Buehler *et al.* [19] also rendered per-pixel weighted textured triangle meshes. We use a similar per-pixel weighting, but are also able to deal with much more complex and accurate geometries. We also use a ”reverse soft-z” buffer to fill holes caused by disocclusions during rendering.

Automatic cinematography that follows common film idioms has been explored in the context of virtual environments, e.g., by He *et al.* [99]; we focus on the idioms used in 3D pan & scan effects. Finally, “Photo Tourism” [87] supports the exploration of a large collection of photos of a scene; our result is meant to be captured and experienced much more like a single photograph, but with subtle amounts of parallax.

6.3 3D pan & scan effects

Our first design goal is to automatically create 3D pan & scan effects that follow the conventions in documentary films. To that end, we examined 22 hours of documentary footage in order to extract the most common types of camera moves and image effects. We examined both films that employ 2D pan & scan effects (18.5 hours, from the Ken Burns films *The Civil War, Jazz,* and *Baseball*) and the more recent 3D pan & scan technique (3.5 hours, *The Kid Stays in the Picture,* and *Riding Giants*). These films contained 97 minutes of 2D effects and 16 minutes of 3D effects. Of these 113 minutes, only 9 exhibited non-linear camera paths; we thus ignore these in our taxonomy (though, as described in Section 6.4.4, curved paths can be created using our interactive authoring tool). Of the remaining 104 minutes, 102 are covered by the taxonomy in Table 6.1 and described in detail below (including 13 minutes that use a concatenation of two of the effects in our taxonomy).

We organize the taxonomy according to the number of “subjects of interest” in a scene: zero,

Subjects of interest	Camera moves	Image effects
0	Establishing dolly, Dolly-out	
1	Dolly in/out, Dolly zoom	Change *dof*, Saturation/brightness
2	Dolly	Pull focus, Change *dof*

Table 6.1: A taxonomy of camera moves and image effects: *dof* refers to depth of field.

one, or two. For each number there are several possible camera moves. There are also several possible image effects, such as changes in saturation or brightness of the subjects of interest or background, or changes in depth of field. These effects are typically used to bring visual attention to or from a subject of interest. The complete set of 3D pan & scan effects in our taxonomy includes every combination of camera move and image effect in Table 6.1 for a specific number of subjects of interest (e.g., no image effect is possible for zero subjects of interest). The most typical subject of interest used in these effects is a human face.

For scenes with no specific subject of interest, we observed two basic types of "establishing shots." These shots depict the entire scene without focusing attention on any specific part. In one type of establishing shot, the camera simply dollies across the scene in order to emphasize visual parallax. We will call this an *establishing dolly*. In the other type of establishing shot, the camera starts in close and dollies out to reveal the entire scene. We will call this an *establishing dolly-out*.

For scenes with a single subject of interest, two types of camera moves are commonly used. The first uses a depth dolly to slowly move the camera in toward the subject, or, alternatively to pull away from it. We will call this type of move a *dolly-in* or *dolly-out*. A variant of this move involves also reducing the depth of field while focusing on the subject to draw the viewer's attention. Another variant, which can either be combined with a changing depth of field or used on its own, is an image effect in which either the subject of interest is slowly saturated or brightened, or its complement (the background) desaturated or dimmed. The other type of camera move sometimes used with a single subject of interest is a kind of special effect known as a *dolly zoom*. The camera is dollied back at the same time as the lens is zoomed in to give an intriguing, and somewhat unsettling, visual appearance. This particular camera move was made famous by Alfred Hitchock in the film, *Vertigo*, and is sometimes known as a "Hitchcock zoom" or "Vertigo effect." Like the other single-subject

camera moves, this move works equally well in either direction.

Finally, for scenes with two primary subjects of interest, the camera typically dollies from one subject to the other. We call this move, simply, a *dolly*. There are two variations of this move, both involving depth of field, when the objects are at substantially different depths. In the first, a low depth-of-field is used, and the focus is pulled from one subject to the other at the same time as the camera is dollied. In the other, the depth of field, itself, is changed, with the depth of field either increasing to encompass the entire scene by the time the camera is dollied from one subject to the other, or else decreasing to focus in on the second subject alone by the time the camera arrives there.

In general, any of the camera moves for scenes with n subjects of interest can also be applied to scenes with more than n. Thus, for example, scenes with two or more subjects are also amenable to any of the camera moves for scenes with just one.

6.4 Authoring

In this section, we describe how to generate 3D pan & scan effects, initially focusing on automatically generated effects that follow the taxonomy just described, and then concluding with an interactive key-framing system that uses our real-time renderer.

The input to this step is a lightfield with depth information. We assume that the input lightfield is sparse and therefore contains holes; while small holes can often be inpainted, large areas of unsampled rays are best avoided. Computing a 3D pan & scan effect automatically from this input requires solving three problems. First, an effect appropriate for the imaged scene must be chosen from the taxonomy in Table 6.1. Second, a linear camera path must be computed that follows the intent of the effect and respects the limited sampling of the input. Third, any associated image effects must be applied.

6.4.1 Choosing the effect

Choosing an effect requires identifying the number of subjects of interest. In general, it is difficult, sometimes impossible, to guess what the user (or director) intends to be the regions of interest in a scene. However, a natural guess for a scene with people is to select their faces. For our automatic system, we employ the face detector of Viola and Jones [98]. In particular, we run the face detector

on the centermost input view and count the number of faces. Then, one of the effects from the appropriate line in Table 6.1 is randomly chosen. The possible effects include image effects such as changing depth of field and focus pulls. Saturation and brightness changes, however, are left to the interactive authoring system, as they are less likely to be appropriate for a random scene.

6.4.2 Choosing a camera path

Each of the camera moves used in 3D pan & scan effects described in section 6.3 can be achieved by having the virtual camera follow a suitable path through camera parameter space. This parameter space includes the 3D camera location, the direction of the camera optical axis, and focal length. All of these parameters can vary over time. If we assume that all parameters are linearly interpolated between the two endpoints, the problem reduces to choosing the parameter values for the endpoints. The result is 6 degrees of freedom per endpoint – 3 for camera position, 2 for the optical axis (we ignore camera roll, uncommon in pan & scan effects) , and 1 for focal length – and thus 12 degrees of freedom overall (two endpoints). A candidate for these 12 parameters can be evaluated in three ways.

1. The camera path should follow the particular 3D pan & scan convention.

2. The camera path should respect the limitations of the input. That is, viewpoints that require significant numbers of rays not sampled in the input should be avoided (modulo the ability to successfully fill small holes).

3. The camera path should be chosen to clearly exhibit parallax in the scene (as we show in our user study in Section 6.9, users prefer effects that are clearly 3D).

Unfortunately, finding a global solution that best meets all 3 goals across 12 degrees of freedom is computationally intractable. The space is not necessarily differentiable and thus unlikely to yield readily to continuous optimization, and a complete sampling strategy would be costly, as validating each path during optimization would amount to rendering all the viewpoints along it.

We therefore make several simplifying assumptions. First, we assume that the second and third goals above can be evaluated by only examining the renderings of the two endpoints of the camera path. This simplification assumes that their measures are essentially smooth, e.g., a viewpoint along the line will not have significantly more holes than the two endpoints. While this assumption is

not strictly true, in our experience, samplings of the space of viewpoints suggest that it often is, as illustrated in Figure 6.1. Second, we assume that the camera focal length and optical axis are entirely defined by the specific pan & scan effect, the camera location, and the linear interpolation parameter. For example, a dolly effect starts by pointing at the first subject of interest, ends by pointing at the second subject of interest, and interpolates linearly along the way. The focal length is set to keep the subjects of interest a certain size. As a result of these assumptions, the problem of choosing a camera path reduces to finding two 3D camera locations, such that the renderings of both viewpoints contain few holes and exhibit significant parallax relative to each other.

Valid viewpoint sampling

We first discuss how to identify viewpoints from which we can successfully render the input photographs processed by our system (measure #2). The set of all viewpoints can be described by a hypervolume parameterized by the camera parameters described above. We constrain this hypervolume to the finite region of valid viewpoints, where a valid viewpoint is defined as a viewpoint from which the rendered scene is complete or contains holes that are likely to be inpainted easily. We take advantage of the interactive renderer described in Section 6.7 to quickly determine valid viewpoints. Given the scene rendered from viewpoint V, we evaluate the success of the rendering using the following metric H:

$$H(V) = \frac{\sum_{p \in V} [d(p)]^k}{w \times h}$$

where w and h are the width and height of the rendering, $d(p)$ is the minimum distance from pixel p to a valid pixel, i.e., the boundary of the hole ($d(p)$ is set to 0 if p is not inside a hole), and k is a constant. Larger values of k penalize larger holes (which are harder to inpaint) over many smaller holes; we found $k = 3$ to be a be good value. We use the fast distance transform from [29] to compute $d(p)$ quickly. We consider a viewpoint as valid if $H(V) < 2.0$.

The next step is to explore the hypervolume and define the 3D region of viewpoints that satisfy this viewpoint validity constraint. We assume that the coordinate system is aligned so that the input lightfield mostly samples rays from viewpoints spread roughly across the x and y axes and pointing down the z axis. We also assume that the centermost viewpoint is roughly at the origin of the

coordinate system. For some effects, such as dolly-out, the camera motion is constrained to travel down the z-axis of this coordinate system. Other effects have more freedom; for these, we search for two regions of valid viewpoints, one for the starting and one for the ending camera viewpoints (since these will have different constraints on pointing direction and focal length).

To exlore the range of valid viewpoints we uniformly sample along x and y at $z = 0$, and then search along z until $H(V)$ exceeds the threshold. The uniform sampling along x and y is done at a resolution of a 12×12 grid across a region that is 4 times the size of the bounding box of the viewpoints in the input lightfield (we have explored wider and denser samplings, and found these settings to be a good tradeoff between efficiency and quality). We search in each direction along z until $H(V)$ exceeds the threshold. We search using an adaptive stepsize, increasing or decreasing the stepsize by a factor of two depending on whether the $H(V)$ threshold is met or exceeded, respectively. Figure 6.1 demonstrates a real working example of such a grid (with a denser 100×100 sampling for visualization purposes), and the results of searching forwards in z.

Maximizing parallax

Next, we need an approach to measuring the parallax induced between two viewpoints. There are a number of possibilities for measuring parallax. We found that the most perceptually noticeable areas of parallax are those that are visible in one viewpoint and occluded in the other. We therefore project the starting viewpoint into the ending viewpoint and vice versa, and sum up the area of holes; this sum is the measure of parallax.

We assume that the starting and ending viewpoints will both be extremal in z; we thus have 144 candidates for both the starting and ending viewpoints (from sampling the 12×12 grid twice). We choose the highest scoring pair according to our measure of parallax. Since this measure requires projecting the starting viewpoint into the ending viewpoint and vice-versa, choosing the optimal pair would require $2 \times 144 \times 144$ projections, which is time-consuming. However, there is a strong correlation between the length of the camera path and the amount of parallax. We formed a training set of 12 examples and performed the full set of projections; we found that the best pair of viewpoints by our parallax metric was always among the top 12 pairs when sorted by the length of the camera path. We therefore increase the speed by only performing the projections on the 12 longest camera

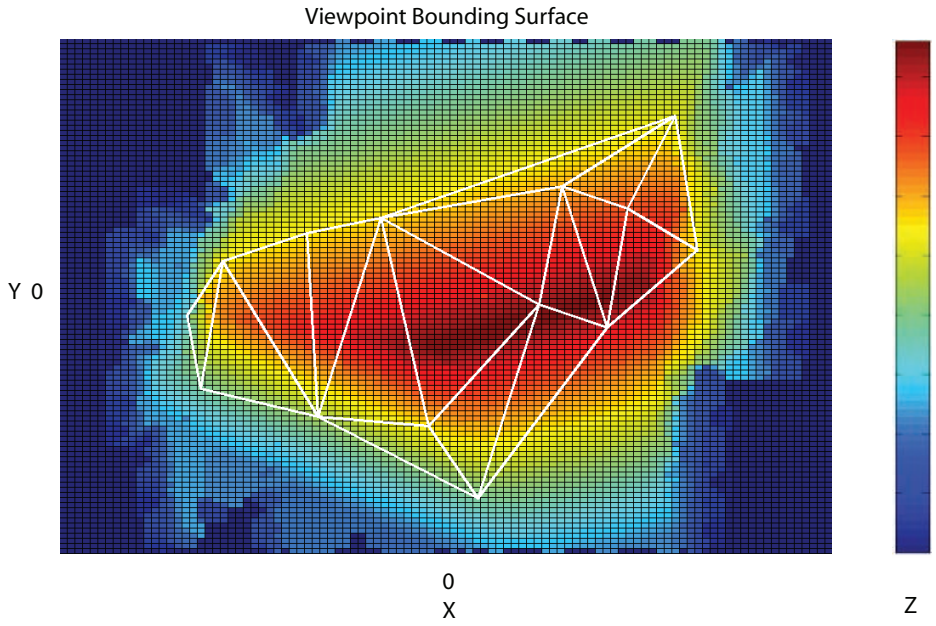


Figure 6.1: A 100×100 grid of valid viewpoints. A mesh connecting the original viewpoints sampled in the input lightfield is also overlaid in white on top of the grid. Note viewpoints in the central area are closer to the original sampled viewpoints; thus they can move a lot closer to the scene (larger z) than peripheral viewpoints.

paths, and choosing the one with the most parallax.

Constraints on the path

Each camera move in our taxonomy imposes specific constraints on the camera focal length, optical axis, and in some cases, camera motion. These constraints are designed to mimic the effect's typical appearance as we observed them in documentary films. The constraints make use of information that we assume is contained in the input lightfield, such as the 3D centroid of the scene (straightforward to compute given that the input lightfield contains depth formation), and the average focal length f of the capture device (in our case a standard camera). We now address each camera move, beginning with the case where there is no specific subject of interest.

Establishing dolly. The camera always points at the scene centroid, and the focal length is set to f for both ends of the camera path.

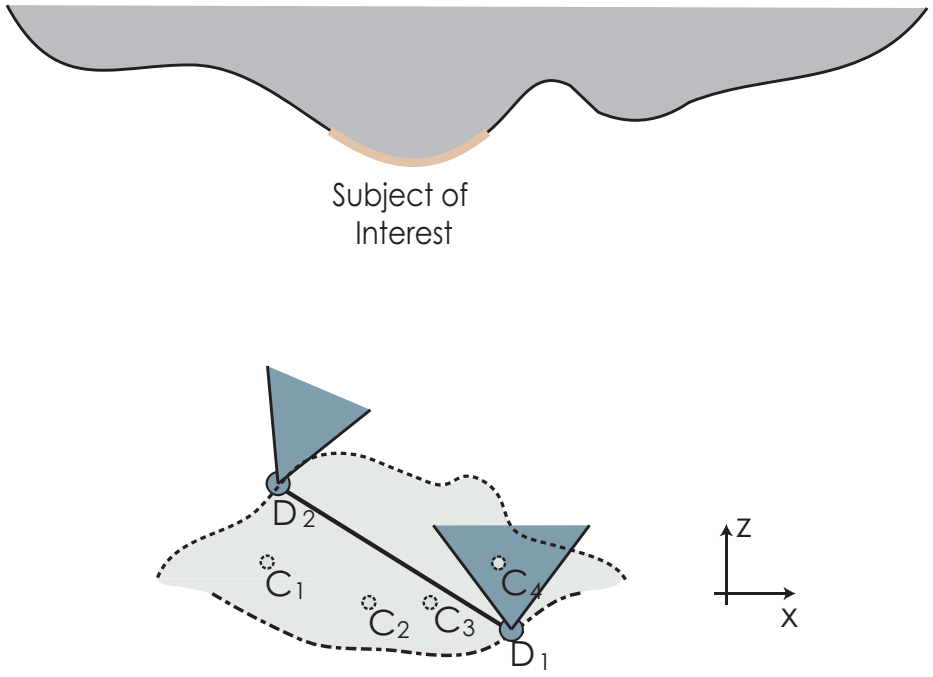


Figure 6.2: For one subject of interest, our algorithm locates the linear camera path (D_1D_2) for dolly-in/out within the valid hypervolume. D_1 and D_2 are on the front and back edges of the valid hypervolume with respect to the z axis. Here C_x denotes the position of a viewpoint sampled in the input. Note how D_2 adjusts its orientation and focal length to keep the subject of interest centered.

Establishing dolly-out. The camera moves along the z axis, always pointing at the scene centroid. The starting focal length is set to $1.5f$, and the ending focal length to f. In this case, no sampling grid across x, y is needed, (though search in the $+z$ direction and another in the $-z$ direction must be performed).

Next we consider the cases containing one or two subjects of interest. To test for the number of subjects of interest, we run the face detector, truncate to the two largest faces if more than two are found, and use the detected rectangles to construct a geometric proxy for the faces. We compute the median depth within each rectangle, and construct a 3D quadrilateral at that depth. Here we discuss 3D pan & scan effects for a single region of interest.

Dolly-in/out. In this case (Figure 6.2), the starting camera points at the lower half of the rectangle containing the subject of interest (so that the face is slightly above center), and the ending camera points at the scene centroid (or vice-versa). The focal length starts at $1.5f$ so that the face is

zoomed-in, and ends with f (or vice-versa).

Dolly zoom. Here the algorithm follows the same procedure as when executing an "establishing dolly-out" with one exception: the focal length is adjusted during the animation to force the region of interest to have the same size as seen in the starting viewpoint.

Finally, we consider two regions of interest.

Dolly. The camera starts by pointing at the first subject of interest, and ends by pointing at the second. The focal length starts at $1.5f$, and ends at the same focal length times the ratio of the size of the subjects of interest (so that the final subject of interest ends at the same size as the first).

6.4.3 Image effects

For one or more subjects of interest our solution may choose to add depth-of-field and/or focus pull effects. Depth-of-field adds another degree of freedom per camera endpoint, namely the aperture diameter. After the two camera endpoints are chosen, a maximum aperture diameter must be chosen so that it does not change the validity of a viewpoint; we therefore search for this maximum. We assume that an aperture diameter can be evaluated by the maximum $H(V)$ of the four corners of the bounding box of the aperture. Starting with an initial maximum aperture diameter of $1/3$ of the shortest dimension the bounding box of the viewpoints in the input lightfield, we search adaptively for bigger apertures until one of the four corner viewpoints is invalid.

To add a changing depth of field to a dolly or dolly in/out, the aperture is linearly interpolated from a pinhole to the maximum aperture. For a focus pull, the aperture is kept at the maximum, and the in-focus depth is simply transitioned from the depth of one subject to the other.

6.4.4 Interactive camera control

We also allow a user to design camera paths that are outside of the taxonomy we have described to better mimic all possible paths used in cinematography, or perhaps chain together a sequence of those that are within the taxonomy. The user is free to navigate, using the interactive renderer, to desired viewpoints and adjust camera settings, as well as color effects (like saturation) and enter them as keyframes along a cubic spline path. The speed of the camera can also be controlled by selecting the duration of the animation. Finally, the user can add depth-dependent brightness and

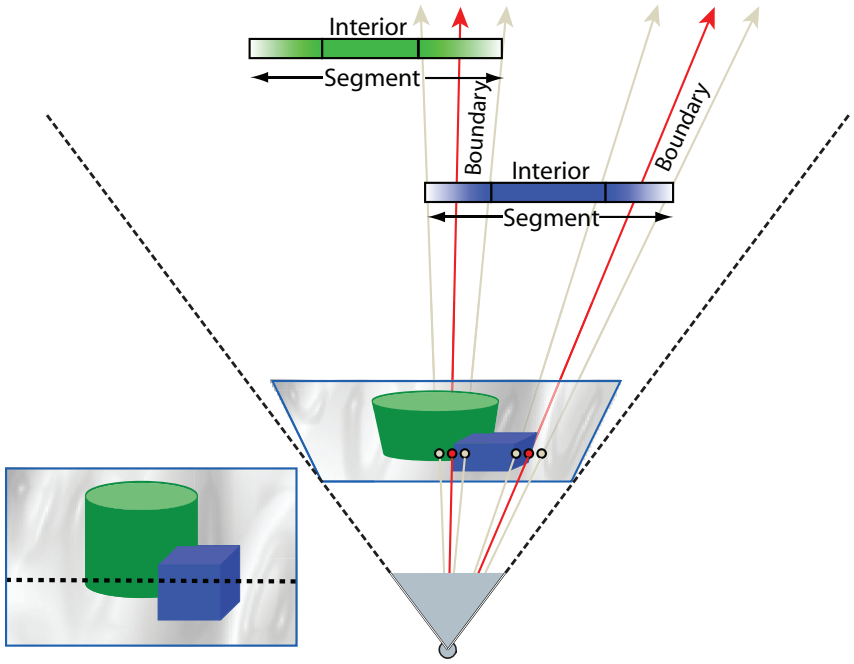


Figure 6.3: Segments are partitioned into two regions, opaque interior regions which do not overlap, and semi-transparent overlapping boundary regions which are decomposed into a two-color model representing color mixing between adjacent segments in the scene.

saturation effects tied to a manually specificed region of interest. In this case, the median depth of the region of interest is computed, and the effect is modulated with a Gaussian falloff from this depth and the bounding rectangle of the region of interest.

6.4.5 View morphing

For the special case where only two views are provided as input to the system, the method introduced in the previous chapter computes stereo correspondences using a soft epipolar constraint. The soft constraint allows for some scene motion. The automatic system simply interpolates linearly between correspondences, i.e., morphs between the views, regardless of scene content.

6.5 From input photographs to images with depth

The input lightfields used to create our results are constructed using existing computer vision techniques from a few photographs of a scene taken from slightly different viewpoints. First, the structure-from-motion system of Snavely *et al.* [87] is used to recover the relative location and orientation of each photograph, as well as the camera focal lengths.[3]

Next, we compute a depth map for each viewpoint using the multi-view stereo method described in Section 6.6. Rather than determining depth per pixel, many top-performing stereo algorithms determine a depth per segment after performing an over-segmentation of the input photographs. This includes the currently leading method [52] of the Middlebury stereo evaluation [76]. Per-segment depths generated by these algorithms are then used for image-based rendering [107]. We follow this approach to produce a set of planar RGBA-textured segments lying at some depth z for each viewpoint. Unfortunately, multi-view stereo algorithms are not perfect, and they will sometimes assign incorrect depths; these errors can sometimes be seen as errant segments that coast across the field of view (often near depth boundaries). To handle them, we allow users to click on these errant segments in our interactive viewer; this input is used to improve the depth estimates. The extent of this interaction is quantified in Section 6.8.

To account for pixels with mixed colors across segment and depth boundaries (Figure 6.3), we extend each segment to overlap adjacent segments as in [107]. The interiors of each segment are considered opaque (have an alpha of 1.0) and are textured with the color seen in the original viewpoint photographs. Pixels near the boundaries are decomposed into two pixels using a two-color model. The first pixel's color is based on the original pixel and the segment interior, and the second's on the original pixel and adjacent segment. These separated pixels are set with depths equal to the segment itself and the adjacent segment, respectively. The alpha values of the separated pixels sum to one, and when composited regenerate the original pixel. Since the second pixel is placed at the depth of the adjacent segment, it effectively extends the adjacent segment to create a region of overlap.

The result of this processing is, for each viewpoint, a set of textured segments, each placed

[3] Since our input contains only a small number of photographs from the same camera, we expect other structure-from-motion systems [39] should perform similarly.

at some some depth, with overlapping semi-transparent boundaries. Viewing the set of segments associated with a particular viewpoint from that viewpoint returns the original images. However, as we will see, when viewed from a slightly displaced location, the depth variations induce parallax, and the overlapping boundary regions help to fill the formation of holes along depth discontinuities.

6.6 Multi-view stereo

Our multi-view stereo algorithm can be viewed as an extension to the consistent-segmentation optical flow approach of Zitnick *et al.* [106]. Their intuition is that optical flow is simpler to compute between two images that are consistently segmented (by consistent, we mean that any two pixels in the same segment in image I_i are also in the same segment in image I_j), since the problem reduces to finding a mapping between segments. Conversely, consistently segmenting two images of the same scene is easier if optical flow is known, since neighboring pixels with the same motion are likely to be in the same segment. Their algorithm iterates between refining segmentation, with motion treated as constant, and refining motion, with segmentation treated as constant.

We apply this basic approach to multi-view stereo, and extend it in four significant ways: (1) we compute stereo rather than optical flow by enforcing a soft epipolar constraint; (2) we incorporate prior, sparse knowledge of the 3D scene computed by structure from motion; (3) we consistently segment a view with respect to its neighboring n views rather than a single other view; and (4) we add an extra stage that merges the disparities computed from the n neighboring views into a single set of segments and their associated depths, resulting in a final set of segments for each view.

We should note that the work of Georgiev *et al.* [34] also builds on the Zitnick *et al.* paper, imposing an epipolar constraint and computing disparities with respect to two neighbors. Their work, however, neither reconstructs nor uses any scene points derived through structure from motion, nor reconstructs depth maps — instead they perform view morphing on a specific camera manifold, customized for a special lens design.

Our multi-view stereo algorithm is applied to each reference view by comparing it to its n neighboring views. Let each view I_i contain k_i segments. Each pixel is treated as a mixture of two segments. Therefore, each pixel p of view I_i is assigned a primary and secondary segment, $s_1(p)$ and $s_2(p)$. A *coverage value* $\alpha(p)$ expresses the portion of the pixel that comes from the primary seg-

ment; thus, $0.5 < \alpha < 1$. Given the segmentation $(s_1(p), s_2(p))$, and the coverage values of each pixel, a mean color $\mu(k)$ for the k'th segment can be computed. Let C be the observed color of p in I, let $C_1 = \mu(s_1(p))$ be the mean color of the primary segment, and C_2 be the mean color of the secondary segment. Ultimately, we seek to compute a segmentation, such that the convex combination of the segment mean colors,

$$\alpha C_1 + (1-\alpha)C_2, \tag{6.1}$$

is as close to C as possible. Given a particular pair of mean colors C_1 and C_2, we can project the observed color C onto the line segment that connects them in color space to impute an α for that pixel, which amounts to computing

$$\alpha = \frac{(C-C_2)\cdot(C_1-C_2)}{\|C_1-C_2\|^2}. \tag{6.2}$$

where the numerator contains a dot product between two color difference vectors, and the result is clamped to α's valid range. In the end, the overlap between segments is usually fairly small; thus many pixels belong to exactly one segment. In such cases, we consider the pixel's primary and secondary segments to be the same, $s_1(p) = s_2(p)$, and set $\alpha = 1$.

To define a mapping between segments in two views, segment k in view I_i maps to a segment $\sigma_{ij}(k)$ in view I_j. Mappings are not required to be bijective or symmetric, which allows for occlusions and disocclusions. A mapping $\sigma_{ij}(k)$ implicitly defines a disparity for each pixel p that considers the k'th segment as primary, i.e., $s_1(p) = k$; the disparity $d_{ij}(k)$ is the displacement of the centroids of the segments. (Note that we use disparity and displacement interchangeably here, and that they correspond to 2D vectors.) In some cases, however, we are able to determine when a segment is partially occluded, making this disparity estimate invalid, and, as discussed in Section 6.6.3, we compute disparity by other means. For this reason, we separately keep track of a segment's disparity $d_{ij}(k)$, in addition to its mapping $\sigma_{ij}(k)$. Ultimately, we will combine the disparities $d_{ij}(k)$ to determine the depth of segment k.

The algorithm of Zitnick *et al.* iterates between updating the segmentation and disparities for two views. To handle n neighboring views when computing depths for a reference view I_i, our algorithm iterates between updating the segmentation of I_i and its n neighboring views I_j, and updating the mappings and disparities between I_i and each neighboring view (i.e., σ_{ij} and σ_{ji}). To compute segments and depths for all views, we loop over all the images, sequentially updating each image's

segmentation and then disparities. We repeat this process 20 times, after which depths are merged. Note that this entire process is linear in the number of original views.

6.6.1 Initialization

We initialize the segmentation for each image by subdividing it into a quadtree structure. A quadtree node is subdivided if the standard deviation of the pixel colors within the node is larger than a certain threshold. We set the threshold to 90 (color channel values are in the range of $[0..255]$), and we do not subdivide regions to be smaller than 8×8 pixels, to avoid over-segmentation. We initialize the mapping between segments using the sparse cloud of 3D scene points computed by structure from motion. Each segment is initialized with the median disparity of the scene points that project to it, or is interpolated from several of the nearest projected scene points if no points project within its boundaries. Each segment k in image I_i is then mapped to neighboring image I_j according to its initial disparity, and the mapping $\sigma_{ij}(k)$ is set to the segment in I_j that covers the largest portion of the displaced segment k.

6.6.2 Segmentation update

We first describe how Zitnick *et al.* update the segmentation of view I_i with respect to neighboring view I_j, given a current segmentation and mapping. We then describe how we extend this update to handle n views.

For each pixel p in view I_i, we consider the segments that overlap a 5×5 window around p. For each of these segments, treating it as primary, we then pair it with every other segment (including itself), compute α according to equation (6.2) (or set it to 1, for self-pairings), and compute a score for every pairing for which $\alpha > 0.5$. We then choose the highest scoring pair as the segments for this pixel. We commit this segmentation choice after visiting every image pixel in the same fashion.

The score of a segment pairing at pixel p is computed as follows. Given the primary and secondary candidate segments $s_1(p)$ and $s_2(p)$, as well as α and the observed color C, an *inferred* primary color C_1' can be calculated such that it satisfies the blending equation:

$$C = \alpha C_1' + (1 - \alpha) C_2. \tag{6.3}$$

Given the inferred primary color C_1' from a pair of candidate segments, we compute its score as follows:

$$N[C_1'; C_1, \Sigma(s_1(p))]\ v[p, s_1(p)]\ v[q, \sigma_{ij}(s_1(p))], \tag{6.4}$$

where $N[x; \mu, \Sigma]$ returns the probability of x given a normal distribution with mean μ and covariance matrix Σ, $v[p,k]$ measures the fraction of the 5×5 window centered at p covered by segment k, and where q is the pixel in I_j corresponding to p, i.e., $q = p + d_{ij}(s_1(p))$.

This scoring function encodes two objectives. The first is that the inferred primary color should have high probability, given the probability distribution of the colors of the primary segment. The second objective is that the primary segment should overlap significantly with a window around pixel p in I_i, and the corresponding segment should also overlap with a window around q in I_j.

Given this pairwise segmentation-update approach, the extension to n neighboring views is quite simple. When updating the segmentation for the reference view I_i, we multiply the product in equation (6.4) by the term $v[q, \sigma_{ij}(s_1(p))]$ for *each* neighboring view I_j, resulting in a product of $n+1$ overlap terms $v(\cdot)$.

6.6.3 Segmentation and disparity update

Given a segmentation for each view, in this step we update the mappings and disparities between segments in reference view I_i and each neighboring view I_j. We first describe the algorithm and objective function used by Zitnick *et al.* to choose this mapping, and then describe our extensions to incorporate epipolar constraints and a depth prior from the cloud of 3D scene points calculated by structure from motion.

For each segment k in I_i, we visit all segments within a large window around the centroid of the initial segment $\sigma_{ij}(k)$ in image I_j. we then score the compatibility of these candidate segments with k and set $\sigma_{ij}(k)$ to the candidate segment that yielded the highest score. (In our implementation, we repeat this process 20 times, starting with a 200×200 search window, steadily narrowing the search window with each iteration, down to 100×100.)

The scoring function for a segment k and a candidate mapping $\sigma_{ij}(k)$ is a product of three terms. The first term,

$$N[\mu(k); \mu(\sigma_{ij}(k)), \Sigma(\sigma_{ij}(k))]\ N[\mu(\sigma_{ij}(k)); \mu(k), \Sigma(k)], \tag{6.5}$$

measures how similar the colors are in the two corresponding segments, by measuring the probability of the mean color of segment k given the color distribution of segment $\sigma_{ij}(k)$, and vice versa. The second,

$$N[d_{ij}(k); \mu(d), \Sigma(d)], \tag{6.6}$$

is a regularization term that measures the probability of the implied disparity $d_{ij}(k)$ given a normal distribution of disparities whose mean and covariance $\mu(d), \Sigma(d)$ are computed using the disparities of each pixel in a 100×100 window centered at the centroid of segment k. The third,

$$s_{ij}[k, \sigma_{ij}(k)] \tag{6.7}$$

measures the similarity in shape between the two segments by comparing their sizes. The function s in this term is the ratio between the numbers of pixels in the smaller and larger segments. For these purposes, we extend the disparity update algorithm in several ways. For one, a candidate mapping segment $\sigma_{ij}(k)$ is only considered only if its centroid falls near the epipolar line l_k in I_j of the centroid of segment k in I_i. We cull from consideration segments whose centroids are more than 25 pixels from the epipolar line. We also contribute two additional terms to the product to be maximized when choosing mappings. The first term penalizes displacements that are not parallel to corresponding epipolar lines:

$$\exp(-\hat{e}(k) \cdot \hat{d}_{ij}(k)) \tag{6.8}$$

where $\exp(\cdot)$ is the exponential function, $\hat{e}(k)$ is the normalized direction of the epipolar line in image I_j associated with the centroid of segment k, and $\hat{d}_{ij}(k)$ is the normalized direction of displacement of that segment.

Finally, we (again) take advantage of the 3D scene points reconstructed in Section **??**. In this case, if one or more of these points project into a segment k in image I_i, we compute the median displacement $m_{ij}(k)$ of their re-projections into image I_j, and multiply one more term into the scoring function:

$$\exp(-||m_{ij}(k) - d_{ij}(k)||) \tag{6.9}$$

to encourage similarity in these displacements.

After iteratively optimizing all the disparities in the image according to the scoring function, a final pass is performed to account for segments that may have become partially occluded in moving

from image I_i to I_j. (Here, we return to the original algorithm of Zitnick *et al.*) In this pass, we visit each segment k in I_i and determine if the size of its corresponding segment is substantially different from the size of segment $\sigma_{ij}(k)$ in I_j. We also determine if the mapping for a segment k is not symmetric, i.e., if $k \neq \sigma_{ji}(\sigma_{ij}(k))$. If neither of these conditions is true, then we simply set the disparity $d_{ij}(k)$ to the difference of the centroids of k and $\sigma_{ij}(k)$. However, if either of these conditions is true, then we suspect a disocclusion. In that case, we attempt to "borrow" the disparity of k's neighbors. In particular, for each segment that overlaps k in image I_i (i.e., each distinct segment that is either primary or secondary to k atone or more pixels), we apply its disparity to k and compare the mapped segment against each segment it overlaps in I_j by computing the average square color difference within their overlap. After considering all possibilities, the disparity and segment mapping with minimum color difference are stored with segment k.

6.6.4 Depth merging

After the above two update steps are iterated to completion (20 iterations over all the images), the result is a segmentation and a set of n disparities for each segment, one disparity for each neighboring view. Since we need only one depth per segment, these disparity estimates must be combined. We do so in a weighted least squares fashion. To compute the depth of a segment k in image I_i, we consider the corresponding segment $\sigma_{ij}(k)$ in each neighboring view I_j. Each corresponding segment defines a 3D ray from the optical center of view I_j through the centroid of that segment on the view's image plane. Such a ray also exists for segment k in reference view I_i. we thus compute a 3D point that minimizes the Euclidean distance to these $n+1$ rays in a weighted least squares sense. Corresponding segments that we suspect are occluded in view I_i are given less weight. A mapping segment $\sigma_{ij}(k)$ is considered occluded if the mapping is not symmetric, i.e., $\sigma_{ji}(\sigma_{ij}(k)) \neq k$. We set the weights of these rays to 0.25, and the rest to 1.0.

6.7 Rendering algorithms

We have developed two rendering algorithms to display novel views from the textured segments with depth. The first rendering algorithm is implemented as a real-time renderer leveraging the GPU. This renderer allows a user to explore the scene and design camera paths that best depict the

scene. It is also used by the automatic camera path planner to evaluate whether novel viewpoints are ”valid” (i.e., wheter they can be rendered with minimal holes), and to estimate the amount of parallax to select paths that provide a strong 3D effect. The second algorithm is implemented as an off-line renderer and produces higher quality results; it is used to render the final result animation. Both renderers take the same basic approach; we therefore first describe the general rendering algorithm, and then we describe the specifics of the interactive renderer implementation including GPU acceleration, and finally we describe the differences in the off-line rendering algorithm.

To render the scene from a novel view, we project a ray from the origin of the novel view through each pixel in the image plane Figure 6.5b). If the ray intersects any segments, we combine their color and depth values and assign the combined color to the pixel. We calculate each segment's contribution in three steps. First we choose which segments should contribute to the the pixel value. Second, we compute a *blending weight* for each contributing segment color value. Finally we employ a *soft z-buffer* to resolve depth inconsistencies and normalize the weights to combine the remaining color values.

When choosing which segments should contribute to a pixel value, we only consider those segments belonging to three original viewpoints with rays most closely aligned with that corresponding to the pixel to be rendered. To select the viewpoints for each pixel of the novel view, we first construct a *view mesh* by projecting the viewpoint camera positions onto a 2D manifold in 3D space, in this case, a plane fit to the camera locations using least squares. The original vertices are then triangulated via a Delaunay triangulation [83]. We also extend the view mesh by duplicating the vertices on the mesh boundary (Figure 6.5a). These duplicate vertices are positioned radially outward from the mesh center at a distance four times the distance from the center to the vertices on the boundary. The original and duplicate vertices are then re-triangulated to form the final view mesh. The triangles on the view mesh boundary will contain two vertices with the same camera ID, while interior triangles will have three distinct camera IDs.

Given the novel view and the view mesh, we are now ready to determine which viewpoints will contribute to each pixel in the novel view and with what weights. Each pixel in the novel view defines a ray from the novel viewpoint through the pixel on the image plane. This ray is intersected with the view mesh (looking backwards if necessary). The viewpoints corresponding to vertices of the intersected triangle on the view mesh are *closest* for that pixel in the novel view, and thus the

ones whose segments will contribute to that novel view pixel. We assign a *blending weight* to each contributing segment equal to the barycentric coordinates for the corresponding viewpoint in the intersected triangle. This is similar to the weights given in [19].

The contributing segments also lie in some depth order from the novel view. Often, segments from different viewpoints represent the same piece of geometry in the real world and thus will lie at approximately the same depth. Slight differences in depth are due to noise and errors in the capture, viewpoint positioning and depth estimation. As the novel viewpoint changes, the exact ordering of these segments may change. Rendering only the closest segment may thus lead to popping artifacts as the z ordering flips. To avoid these temporal incoherencies, we implement a *soft z-buffer* [74]. A soft z-buffer allows us to consistently resolve conflicting depth information by combining all of the segments that may contribute to a pixel, and estimating the most likely RGBA and z values for the pixel. The soft z-buffer assigns a z-weight for each contributing segment beginning with a weight of 1.0 for the closest segment (at a distance z_0) dropping off smoothly to 0.0 as the distance increases beyond z_0. The z-weights are multiplied by the *blending weight,* and the results are normalized. The final pixel value is the normalized weighted sum of the textures from the contributing segments.

When the novel view diverges from the original viewpoints, the parallax at depth discontinuities may cause segments to separate enough so that a ray hits no segments. We are then left with a hole-filling problem. We address this later in the context of the interactive and offline renderers.

6.7.1 Interactive renderer

To render the scene from a novel view at interactive frame rates (at least 30 fps), we need to pose the rendering problem in terms of GPU operations. We now describe the rendering steps in terms of polygons, texture maps and GPU pixel shaders. We render the scene in four steps. First, we choose which segments should contribute to the pixel value and calculate the *blending weight* for each contributing segment color value. Second, we render all of the segments to three offscreen buffers. Third, we employ a *soft z-buffer* to resolve depth inconsistencies between the three offscreen buffers and combine their color values. Finally, we fill holes using a *reverse soft z-buffer* and local area sampling.

Rendering the extended view mesh To choose which segments should contribute to the pixel value and to calculate the *blending weights* we render the extended view mesh from the novel view to an offscreen buffer. Setting the three triangle vertex colors to red, green, and black encodes two of the barycentric coordinates in the Red and Green channels; the third coordinate is implicit. The Blue and Alpha channels are used to store an ordered 3-element list storing the ID's of the three viewpoints (we use 5 bits to encode a viewpoint ID, so 3 IDs can be stored in 16 bits, allowing for a total of 32 input viewpoints)[4].

When rendering the extended view mesh, there are two special cases that should be highlighted. First, if the novel view lies in front of the view mesh, the projection step requires a backwards projection (i.e., projecting geometry that is behind the viewer through the center of projection). Second, the projection of the view mesh nears a singularity as the novel view moves close to the view mesh itself. Therefore, if the novel view is determined to lie within some small distance from the view mesh, the view mesh is not rendered at all. Rather, the nearest point on the mesh is found. The blending weights and viewpoint IDs are then constant across all pixels in the novel view, set to the barycentric coordinates of the point on the view mesh and the vertex IDs of the triangle the point lies in.

Rendering segments Each segment is rendered as a texture mapped rectangle. The rectangle's vertex locations and texture coordinates are defined by the bounding box of the segment. A segment ID and associated viewpoint ID is recorded with each rectangle. Rather than create a separate texture map for each segment rectangle, we create two RGBA textures for each viewpoint, plus one *Segment ID* texture. As described in Section 6.5 pixels near the boundaries of segments are split into two overlapping layers to account for mixed pixels along the boundaries. Thus, the first RGBA texture contains the segment interiors and one of the two layers. The other RGBA texture contains the second layer along segment boundaries. Finally we create a third 2-channel 16-bit integer *Segment ID* texture map containing segment ID values indicating to which segment a pixel belongs.

For each segment, a pixel shader combines the texture maps and discards fragments within the segment bounding rectangles but laying outside the segment itself. To render a rectangle, we encode the segment ID as a vertex attribute (e.g. color or normal data). The shader uses this value in

[4]It is not strictly necessary to encode both viewpoint IDs and the barycentric weights into one off-screen buffer, but doing so saves a rendering pass and reduces texture memory usage.

conjunction with the *Segment ID* texture and the two RGBA textures to compose the segment color values on the fly.

```
void main(){
    if (SegmentID == SegIDMap[0] or
        SegmentID == SegIDMap[1])
        ExtractAndDraw();
    else
        discard;
}
```

Using this GPU based texture representation has two benefits, both of which increase rendering speed. First, it reduces the number of texture swaps from thousands per viewpoint to only three. Second, by removing the texture swap from the inner loop we are able to take advantage of vertex arrays or vertex buffer objects and utilize the GPU more efficiently.

We still need to choose which segments contribute to each pixel and by how much. A pixel is ultimately a sum of segments originating from at most three different viewpoints as encoded in the viewpointIDs in the offscreen rendered view mesh. We create three buffers to accumulate RGBA values, corresponding to the three viewpointIDs stored at each pixel in the rendered view mesh. When rendering a segment, we encode the segment's viewpoint ID as a vertex attribute. The pixel shader chooses to which of the three buffers a segment should contribute, if any, by matching the the segment's viewpoint ID with the ones encoded in the offscreen rendered view mesh at that pixel location. For example, if the segment's viewpoint ID matches the first of the view mesh's encoded viewpoint IDs (i.e., the one corresponding to the "red" barycentric coordinate), the pixel is accumulated in the first buffer using the first (red) barycentric coordinate as a weight. The same is done if there is a match with the second (green) or third of the view mesh's encoded viewpoint IDs, except the third barycentric weight is inferred from the other two (1 - red - green).

```
void main(){
  if (ViewID == ViewIDMap[0])
    Target = 0;
  else if (ViewID == ViewIDMap[1])
    Target = 1;
  else if (ViewID == ViewIDMap[2])
    Target = 2;
  else discard;
```

```
  if (SegmentID == SegIDMap[0] or
      SegmentID == SegIDMap[1])
        ExtractAndDraw(Target);
  else
     discard;
}
```

Before rendering any segments, the segments for each viewpoint are sorted to be processed in front-to-back order. The three rendering buffers are initialized to black background and zero alpha. To maintain proper color blending and z-buffering, the blending mode is set to GL_SRC_ALPHA_SATURATE, depth testing is enabled, and set to GL_ALWAYS. The pixel shader calculates the pre-multiplied pixel values and alphas to render.

Soft z-buffer The z-buffering is performed traditionally within a single viewpoint for each of the three buffers; however, we employ a *soft z-buffer* across the viewpoints to blend the three results. For each corresponding pixel in the three buffers, we compute a soft weight w_z by comparing each pixel's z-value with the z-value of pixel closest to the origin of the novel view. This distance Δz, where $\Delta z = z - z_{\text{closest}}$, is used to compute w_z in the following equation:

$$w_z(\Delta z) = \begin{cases} 1 & \text{if } \Delta z \leq \gamma \\ \frac{1}{2}(1+\cos\left(\frac{\pi(\Delta z-\gamma)}{\rho-2\gamma}\right)) & \text{if } \gamma < \Delta z \leq \rho-\gamma \\ 0 & \text{otherwise} \end{cases} \tag{6.10}$$

where ρ is the depth range (max$-$min) of the entire scene, and γ is set to $\rho/10$.

The set of w_z's are normalized to sum to one. The depth, z for that pixel is then given the sum of the z-values weighted by the w_zs. Each *blending weight* stored in the view mesh texture is multiplied by its corresponding w_z. These new blending weights are normalized. The final pixel value is computed by scaling each pixel by the normalized *blending weight*, and composited based on their alpha values.

Hole filling Holes occur when, due to parallax, a nearby segment separates from a more distant segment. A pixel with a z-value of 1 indicates a hole. we fill small holes of less than 6 pixels in diameter during the final *soft z-buffer* pass. We assume that any hole should be filled with information from neighboring pixels. Since holes occur due to disocclusion, given two neighbors, we prefer

to use the more distant one to fill the gap. To do so, we combine the pixel colors and z-values of the pixels in a 7×7 neighborhood. They are combined using the *soft z-buffer* calculation described above except in reverse. In other words, more distant z-values are given higher weights by inverting the ordering, by setting the z-values to $1 - z$.

In summary When iterating over the segments to be rendered, only three textures are (re-)loaded per viewpoint: the two RGBA texture maps; and a texture for an ID image to map pixels to segments. A fourth texture, the barycentric weight map from the view mesh, is computed once and used throughout.

As a result of this GPU approach, we can render scenes at 30-45 frames-per-second on an NVIDIA 8800 series graphics card, whereas an implementation using one texture per segment achieved only 7.5 frames-per-second and did not calculate the blending weights on a per pixel basis, use a *soft z-buffer*, or fill holes.

Depth of field and color effects Efficient, approximate depth-of-field rendering is accomplished using a variation on existing methods [26, 51, 38]. For each pixel, we calculate a circle of confusion based on a user defined aperture size, and blur the result of our rendered scene accordingly. The blurring operation is performed efficiently by loading the scene into a MIPMAP and indexing into it based on the blur kernel radius. To improve visual quality, we index into a higher resolution level of the MIPMAP than strictly needed, and then filter with a Gaussian filter of suitable size to achieve the correct amount of blur. Note that when focusing on the background in a scene, this approach will not result in blurred foreground pixels that partially cover background pixels as they should, i.e., the blurry foreground will have a sharp silhouette.

To avoid such sharp silhouettes, when processing a pixel at or behind the focus plane, the pixel shader blends the pixel with a blurred version of the image at that pixel. The blur kernel size is based on the average z-value of nearby foreground pixels. The blending weight given to this blurred version of the image is given by the fraction of the neighboring pixels determined to be foreground. The size of the neighborhood is determined by the circle of confusion computed from the user specified aperture and focal depth.

6.7.2 Off-line rendering

The higher quality off-line rendering algorithm differs from the interactive renderer in three main ways. First, we extend the *soft z-buffer* described above to increase the accuracy of our pixel value estimate. Second, the renderer uses a texture synthesis approach to fill any holes and cracks that might appear in a novel view due to sparse data generated from the input photographs. Finally, depth of field effects are rendered with increased quality by simulating a camera aperture.

Soft z-buffer The soft z-buffer calculation is very similar to the process described in the real-time renderer. However, rather than using a traditional hard z-buffering within each viewpoint followed by a soft z-buffer across viewpoints, all segments from all contributing viewpoints are combined in a uniform manner. We assemble a depth ordered list of elements at each pixel location as the segments are projected onto the scene. Each element contains the sampling viewpoint ID, the RGBA color value, z-value, the *blending weight*, and the soft weight w_z as computed above. The soft z-buffer weights, w_z are computed when the list is complete.

Hole filling To fill holes the offline renderer uses a more principled approach, in particular the in-painting algorithm of Criminisi *et al.* [22] — based on example-based texture synthesis — with two modifications. First, to accelerate computation, we search for matching (5×5) neighborhoods within a restricted window (100×100) around the target pixel, rather than over the entire image. The second, more significant, modification is based on the observation that nearly all large holes occur along depth discontinuities, because some region of background geometry was always occluded in the input photographs. In this case, the hole should be filled from background (far) regions rather than foreground (near) regions. We thus separate the depths of the pixels along the boundary into two clusters, and use these two clusters to classify pixels, as needed, as foreground or background. We then fill the hole with Criminisi's propagation order, using modified neighborhoods and neighborhood distance metrics. In particular, for a given target pixel to fill in, its neighborhood is formed only from pixels labeled background. If no such pixels exist in this neighborhood yet, then this pixel is placed at the bottom of the processing queue. Otherwise, the neighborhood is compared against other candidate source neighborhoods, measuring the L_2 differences between valid target pixels and all corresponding source pixels from a candidate neighborhood. For source pixels that are invalid

(foreground or unknown), we set their colors to 0, which penalizes their matching to generally non-zero, valid target neighborhood pixels. Whenever a pixel is filled in, it is automatically classified as background. Thus pixels with invalid neighborhoods (e.g., those centered on the foreground occluder) will eventually be processed as the hole is filled from the background side. When copying in pixel color, we also inpaint its z by weighted blending from known neighbouring pixels, again favoring the back layer. The inpainted z assists in region selection for color manipulation effects. The third row of Figure 6.6 shows the results of our inpainting algorithm for a novel viewpoint rendering of one of our datasets. Note that Moreno-Noguer *et al.* [64] also explored depth-sensitive inpainting, though their application has lower quality requirements since they use the inpainted regions for rendering defocused regions rather than novel viewpoints.

Depth of field Our rendering algorithm now provides a way to reconstruct any view within the viewing volume. In addition to changing viewpoint, we can synthetically focus the image to simulate camera depth of field. To do so, we apply an approach similar to what has been done in *synthetic-aperture photography* [58, 44]. We jitter viewpoints around the center of a synthetic aperture and reconstruct an image for each viewpoint. We then project all the images onto a given in-focus plane and average the result.

6.8 Results

We have tested our overall approach (including the construction of the lightfield from a few photographs) on 208 datasets capturing a variety of scenes. About half of the datasets (103 out of 208) produced successful results that were comparable to the results shown in this chapter. The other half were less successful. Here is a breakdown of what failed:

- Error due to data (35/208 or 17%): The data contained too much motion, resulting either in motion blur or poor correspondence.
- Error caused by structure-from-motion (SfM) (9/208 or 4%): too little parallax for SfM to recover the camera parameters.
- Error in stereo matching (56/208 or 27%): color shifts, too large a baseline, textureless regions, etc.

- Error on face detection (5/208 or 2%): false positives produced errant camera paths.

A subset of the successful results are included in this chapter. Only 2 examples were interactively authored. The rest were done entirely automatically through our pipeline. The best way to experience the cinematic effects produced by our system is in animated form, as shown in the supplemental video. However, a single rendered view of several of the results created using our system, as well as the number of input photographs used to construct them, can be seen in Figure 6.7. The number of input images ranges from 8 to 15, and are typically captured with just 3-4 inches of space between the viewpoints. Most of our datasets are captured at resolution 1200×800. On a 3GHz PC with 4GB memory and an NVIDIA 8800 class GPU, it usually takes about 3 hours to generate a 3D pan & scan effect automatically, including 10-15 minutes for structure-from-motion, 100-120 minutes for multi-view stereo, 2-5 minutes for computing the effect, and 25-35 minutes for the off-line rendering. As shown in Figure 6.7, four of the results required the user to click segments with errant depth (Section 6.5), which typically took 3-5 minutes of user time.

A number of our results depict people in seemingly dynamic poses; in these cases, we asked the subjects to hold still. We hope to experiment with hardware that can capture multiple viewpoint simultaneously in the near future. In addition, we include two results using two input views with small scene motion, for which our automatic system generates morphs from one view to the other.

Overall, our results offer a more compelling depiction of their scenes than a simple 2D pan & scan. However, occasional artifacts can be seen, often near depth boundaries. These artifacts come from depth mis-estimates; as multi-view stereo algorithms improve [78] these types of errors, as well as the need to occasionally click errant segments, should be reduced.

6.9 User study

In order to assess the perceptual quality of the 3D pan & scan effects, we conducted a user study study with 145 subjects, the majority of whom were in the IT industry but not in the graphics industry. Each subject was asked to watch 10 examples, with each example shown in two versions: a 2D pan & scan, and a 3D pan & scan. All effects were rendered at 480×320 pixels at 30 frames per second and were 3-5 seconds long. For each example, subjects had to answer whether they noticed any difference between the two versions after watching them as many times as they liked.

They also had to select which version they preferred, and provide reasons if possible.

The results show that there was a significant perceptual difference between 3D pan & scan and 2D pan & scan effects. Overall, the two versions of each example were judged to be different 94% of the time, and at least 80% of subjects noticed at least some difference between the two versions on any given example.

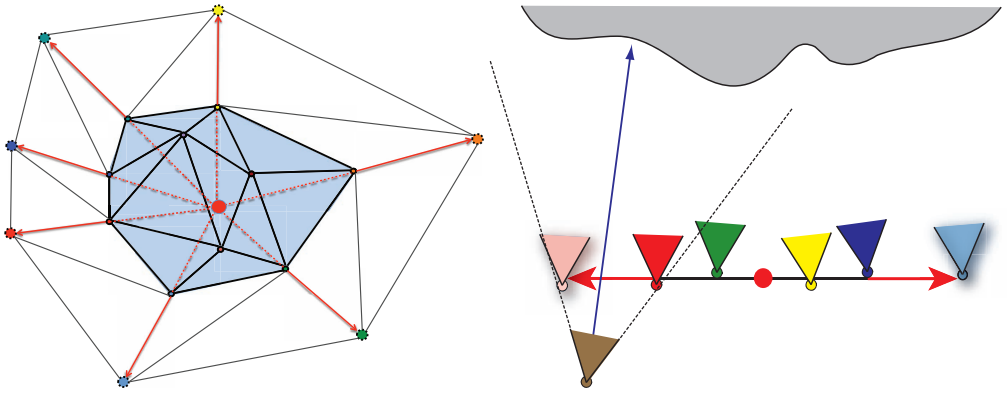
The results also show that majority of participants prefer the 3D versions to the 2D ones. Summing over all examples, 70% of subjects preferred 3D pan & scan, 16% preferred 2D pan & scan, and 14% had no preference. As shown in Figure 6.8, the preference rate exhibits correlations with the amount of parallax contained in each example. Note also that roughly half the people preferred the 2D to the 3D "dolly-zoom" example (#6 in Figure 6.8); in written comments they noted that the 3D effect was too dramatic and uncomfortable, which is the intended purpose of the effect. The users' written comments clearly show that apparent parallax was the dominant reason (69% of all reasons collected) behind the preference for the 3D results. Taken together, the results of the user study clearly indicate that the cinematic effects we created offer a more compelling depiction than a simple 2D pan & scan.

6.10 Conclusion

In this chapter we described a completely automatic approach to constructing cinematic effects from an input lightfield, which in our implementation we construct using a few snapshots of a static scene from a hand-held camera. Our system includes real-time rendering and interactive authoring components, as well as a taxonomy of common 3D pan & scan effects and a set of computational methods that can automatically achieve them. We have used our system to create a number of compelling results. Recent advances in computational photography have dramatically increased the amount of information that we can capture from a scene. Until now, techniques that capture depth along with an image have been used primarily for digital re-focusing, on the assumption that small parallax changes are uninteresting. On the contrary, we believe that subtle parallax can lead to a richer, more informative visual experience of a scene that feels fundamentally different than a still photograph or 2D pan & scan. As multi-view camera designs and computer vision techniques continue to improve, we see an opportunity for parallax photography to become a widely used

approach to capturing the moments of our lives.

Figure 6.4: Crab example: the top row shows one of the input images, the middel column is the visulization of its segmentation, and the bottom row is the corresponding depth map.



(a) (b)

Figure 6.5: **(a)** The view mesh is extended by creating copies of boundary vertices and placing them radially outward (red arrow) from the centroid (red dot) of all of the mesh vertices. These new vertices are then re-triangulated and added to the view mesh to form an extended view mesh. **(b)** The dotted blue arrow shows a ray projected through the image plane. The *blending weights* at this pixel correspond to the barycentric coordinates of the intersected triangle of viewpoints in the view mesh.

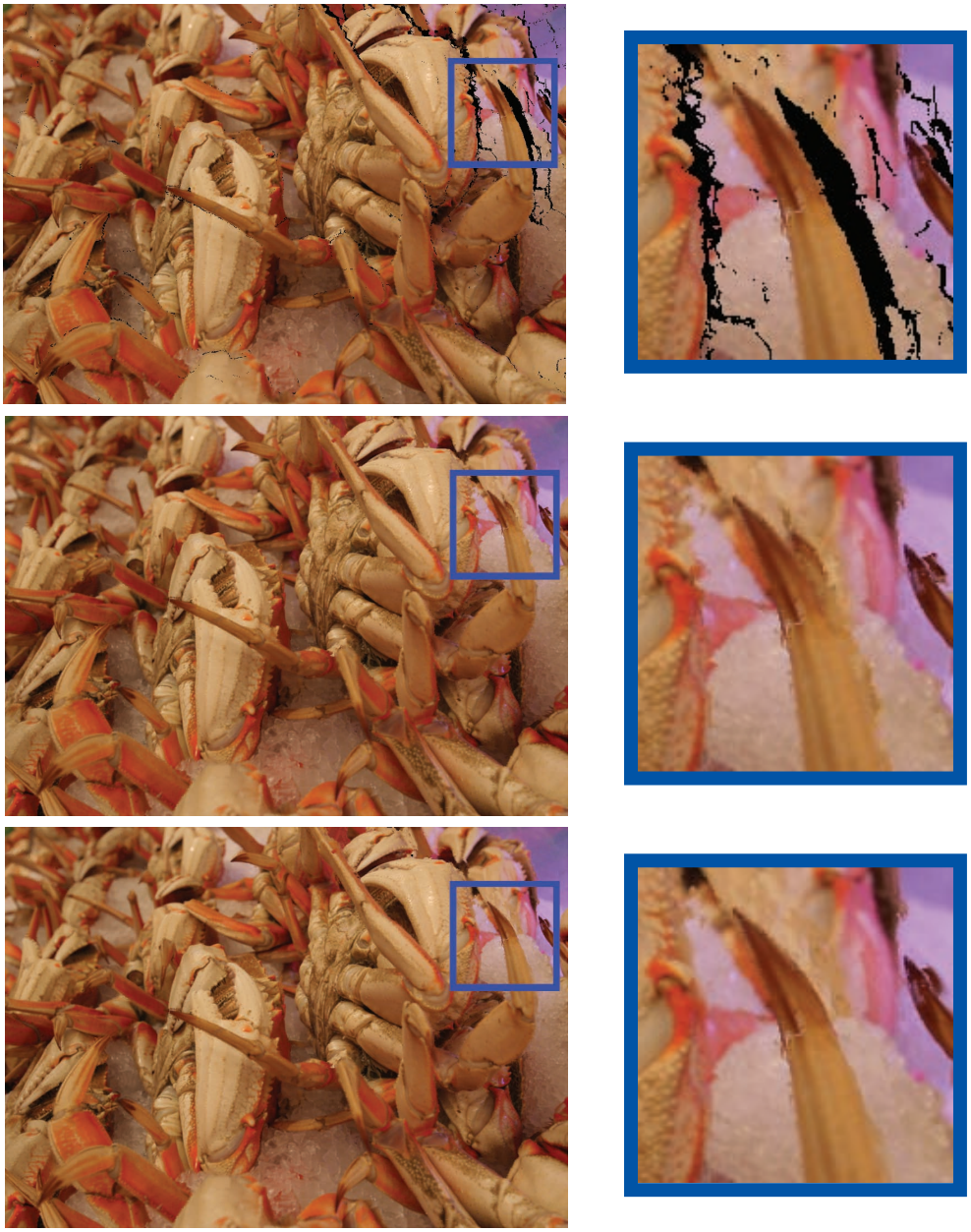


Figure 6.6: Three renderings of crabs at the market. The first row shows a novel viewpoint rendered from the segments of all the input photographs by the interactive renderer; many holes are visible. An inset, highlighted in blue, is shown on the right. The second row shows the result after inpainting without depth guidance; no holes remain but the result is incorrect. The final row shows the result after depth-guided inpainting in offline-rendering; no holes remain and the inferred background is correct.

Figure 6.7: A variety of results produced by our solution. (First row) One rendered view of each scene. (Second row) The number of input photographs followed by the number of clicks required to fix errant segments (in red). Note that only 4 datasets required any user intervention of this kind.

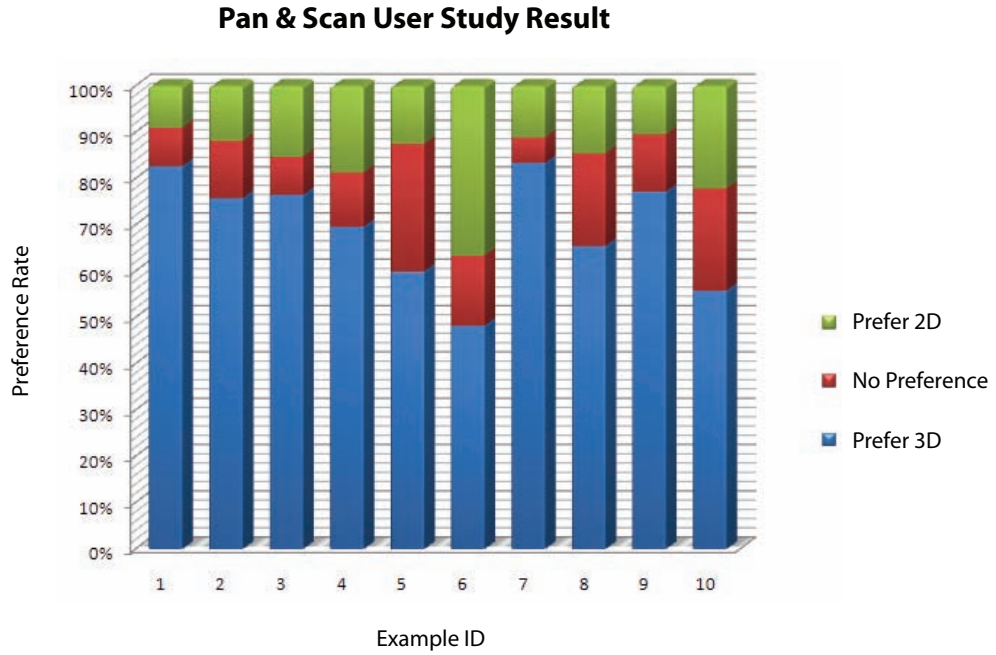


Figure 6.8: The preference rate of the 10 examples collected from the user study. Note that fewer than 65% of the subjects preferred the 3D version of examples 5, 8 and 10; these examples contained the most subtle parallax or motion, and thus support the notion that maximizing parallax is important.

Chapter 7

CONCLUSION

Visual ideas combined with technology combined with personal interpretation equals photography.

—Arnold Newman

7.1 Contributions

In this thesis we have designed a novel approach to combining multiple samples of the plenoptic function into new visual media focusing primarily on exhibiting parallax, and applied this approach to several different design tasks in which 3D perception gives rise to much more immersive visual experiences.

To apply this approach we begin by identifying the goals of the new visual media, and come up with representations satisfying the design needs. Solutions to different representations vary in terms of how samples of the plenoptic function get interpolated. Finally, the final viewing experience is optimized to best match viewers' visual memories.

We applied this approach in three chapters. In chapter 4, we introduced "layered depth panoramas", a concise global representation that allows the user to experience 3D by off-axis panning. In chapter 5, we exploit the spatial-angular tradeoff for lightfield cameras based on Georgiev's new (integral) camera design with a bundle of lenses and prisms attached externally to the camera. This optical design can be treated as a planar camera array with all views being parallel to each other. A sparse set of views can be captured at a single exposure, and be densified via tri-view morphing. The interpolated set of rays, or light field can be used to produce synthetic aperture effects, new view synthesis and refocusing. In chapter 6, we extended the approach from chapter 5 to to synthesize a small portion of a lightfield from a few off-plane views, with an application to create cinematic effects with simulated, smooth camera motions that exhibit a sense of 3D parallax. Along with this

overall approach to enhancing 3D perception with motion parallax by combining multiple samples, the thesis makes several contributions.

- **Concise representations for 3D panoramas.** We introduced "layered depth panoramas," a concise global representation that allows the user to experience 3D by off-axis panning. The system asks little more of the user than capturing a simple panorama from a sparse set of images with a hand-held camera. The final representation, which aims to explain every single pixel from all input images, is less than twice the size of a normal panorama in size, yet produces a smooth 3D viewing experience exhibiting strong parallax.

- **Spatial and angular analysis for light field cameras.** We surveyed some of the previously proposed light field camera designs. Integral or light field photography is approached from the perspective of radiance analysis in geometrical optics. This provides a new way of looking at integral photography and the associated light field rendering. We proposed new camera designs that produce higher spatial resolution than the camera of Ng et al., while trading-off the light field's angular sampling density. However, this lower angular resolution in the input is compensated for by inserting data synthesized by view interpolation of the measured light field.

- **Taxonomy of cinematic idioms used for 3D pan & scan and its corresponding authoring algorithm in camera space.** We describe the cinematic conventions of 3D pan & scan effects by presenting a taxonomy of camera moves and other details that were distilled from observation of many hours of documentary film footage. Following the taxonomy, which is organized by the number of subjects of interest in the scene, we present an automatic, content-aware approach to applying these cinematic conventions to an input of a few photographs.

7.2 Future work

At the end of chapters 4-6, we discussed limitations and potential areas of future work to improve on the techniques described in those chapters. To conclude this thesis, we now suggest more ambitious ideas for future research projects that exploits parallax by again combining information from

multiple samples of the plenoptic function. These ideas do not necessarily follow the approaches described above.

7.2.1 Finding light field data from internet photographs

Internet serves as a huge and ever-growing source of images and videos that could potentially be used for digital photography research. The "photo-tourism" system developed by Snavely *et al.* [87] is one of the first attempts at registering hundreds of photographs of popular tourist sites to create a sparse 5D time-varying light field. While photo-tourism is a great tool for browsing and organizing online photographs while delivering a sense of 3D, it is not designed to produce smooth transitions between nearby viewpoints as well as light field rendering systems can achieve.

The parallax photography approach we presented in the previous three chapters construct light field representations by combining photographs taken at approximately the same time from the same camera. This is to ensure that we sample rays under the same illumination, with the same imaging sensor, and of the same scene.

The challenge of constructing dense light field representations from internet photo collections include:

- register photographs into the unified 3D coordinate space so that a subset of photographs taken from similar viewpoints can be found.

- find photographs under similar lighting condition and further align them into the same color space.

- robustify the current segmentation based multi-view stereo algorithm to be less sensitive to color shifts and outliers (such as walking pedestrians).

In general, this project points to a general research direction of growing interest - reusing internet photo collections for new applications. And such approach has the potential of creating the largest light field people can ever capture both in geometric scale and sample density.

7.2.2 Fusing stills containing scene motion

Limited by the resolution of sensors, the state of the art handheld light field cameras suffer from the tradeoffs between spatial and angular resolutions. Using the approach proposed in chapter 7 increases the spatial resolution, however, this approach is limited to static scenes only. In previous chapter, we relax the epipolar constraint to handle subtle motions between a pair of photographs. With the rapid progress in both the multi-view stereo and optical flow algorithms, it is then possible to create 3D pan & scan effects from multiple images (more than 2) for moving scenes if the motion is not too large.

This problem poses a few challenges:

- identify the correct cause of motions between images and separate them to either scene motion or camera motion.

- solve geometry and 3D scene flow from limited samples. One can imagine an EM-like method that iterates between solving flow while fixing geometry and solving geometry while fixing flow.

- combine different views together to render new images. This might involves re-time the original samples.

- efficient user interaction interface for picking the timing and the viewpoint.

7.2.3 Combining video cameras

A frequent theme of this thesis is that the information provided by multiple samples of the plenoptic function is useful in creating better visual media. We believe that hand-held video cameras that simultaneously capture a number of synchronized videos from a number of different viewpoints would provide the information necessary to create better hand-held videos. This could be achieved either by attaching our lenslet array described in Chapter 5 onto a video camera or bundling a few small video cameras together to form a portable multi-camera array.

One of the biggest differences in quality between professional movies and home videos is the stability of the camera path; professionals use expensive gantries to carefully control the camera

motion, while regular users simply hold the camera in hand. Existing video stabilization [10] techniques are typically limited to translations or affine warps of each video frame. Multiple, synchronized video streams from different viewpoints, would allow the reconstruction of viewpoints within some range near the cameras. Buehler *et al.* [18] also posed video stabilization as an image-based rendering problem; however, their use of a monocular video stream limited their system to videos of static scenes.

Another challenge in video processing is re-timing, especially when a video must be slowed down. Doing so without introducing temporal jitter requires interpolating in-between frames. This task is typically accomplished by performing optical flow between video frames, and then flowing video colors along optical flow vectors to form in-betweens. Optical flow, however, often performs poorly at discontinuities. Multiple views can be used to identify depth discontinuities, and thus may be able to improve optical flow.

BIBLIOGRAPHY

[1] 3dv Systems. Z-cam. http://www.3dvsystems.com/home/index.html.

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

[3] T. Adelson and J. Wang. Single lens stereo with a plenoptic camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 99–106, 1992.

[4] Aseem Agarwala, Maneesh Agrawala, Michael Cohen, David H. Salesin, and Richard Szeliski. Photographing long scenes with multiv-viewpoint panoramas. *ACM Transactions on Graphics*, 25(3), 2006.

[5] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, and Michael Cohen. Interactive digital photomontage. *ACM Transactions on Graphics*, 23(3):294–302, 2004.

[6] Aseem Agarwala, Ke Colin Zheng, Chris Pal, Maneesh Agrawala, Michael Cohen, Brian Curless, David H. Salesin, and Richard Szeliski. Panoramic video textures. *ACM Transactions on Graphics*, 24(3):821–827, August 2005.

[7] Anonymous. A consistent segmentation approach to multi-view stereo. Technical Report 001, University of AnonymousVille, 2008.

[8] Gleb Bahmutov, Voicu Popescu, and Elisha Sacks. Depth enhanced panoramas. In *Proceedings of the conference on Visualization '04*, page 598.11, 2004.

[9] Simon Baker, Richard Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *Proc. of CVPR'98*, pages 434–441, 1998.

[10] S.B. Balakirsky and R. Chellappa. Performance characterization of image stabilization algorithms. *Real-time imaging*, 2(5):297–313, October 1996.

[11] J. Batlle, E. Mouaddib, and J. Salvi. Recent progress in coded structured light as a technique to solve the correspondence problem: a survey. *Pattern Recognition*, 31(7):963–982, July 1998.

[12] Eric P. Bennett, John L. Mason, and Leonard McMillan. Multispectral video fusion. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches*, page 123, New York, NY, USA, 2006. ACM.

[13] Eric P. Bennett and Leonard McMillan. Computational time-lapse video. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 102, New York, NY, USA, 2007. ACM.

[14] S. Birchfield and C. Tomasi. Multiway cut for stereo and motion with slanted surfaces. In *Proc. of ICCV'99*, pages 489–495, 1999.

[15] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. on PAMI*, 23(11), 2001.

[16] M. Brown and D. Lowe. Recognising panoramas. In *Proc. of ICCV'03*, volume 2, pages 1218–1225, 2003.

[17] M. Brown and D. G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *Proceedings of 5th International Conference on 3D Imaging and Modelling (3DIM)*, pages 21–30, 2005.

[18] Chris Buehler, Michael Bosse, and Leonard McMillan. Non-metric image-based rendering for video stabilization. In *Computer Vision and Pattern Recognition (CVPR 01)*, pages 609–614, 2001.

[19] Chris Buehler, Michael Bosse, Leonard McMillan, Steven Gortler, and Michael Cohen. Unstructured lumigraph rendering. *ACM Trans. Graph.*, pages 425–432, 2001.

[20] Chun-Fa Chang, Gary Bishop, and Anselmo Lastra. Ldi tree: A hierarchical representation for image-based rendering. *ACM Trans. Graph.*, pages 291–298, 1999.

[21] S. Chen and L. Williams. View interpolation for image synthesis. *ACM Trans. Graph.*, pages 279–C288, 1993.

[22] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *Proc. IEEE Computer Vision and Pattern Recognition (CVPR)*, Jun. 2003.

[23] P. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *ACM Trans. Graph.*, pages 11–20, 1996.

[24] P. Debevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *In Proc. 9th Eurographics Workshop on Rendering*, pages 105–116, 1998.

[25] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 369–378, August 1997.

[26] Joe Demers. Depth of field a survey of techniques. In Randima Fernando, editor, *GPU Gems 1*, chapter 23, pages 375–390. Addison Wesley, March 2004.

[27] Marc-Antoine Drouin, Martin Trudeau, and Sebastien Roy. Geo-consistency for wide multi-camera stereo. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 351–358, Washington, DC, USA, 2005. IEEE Computer Society.

[28] Raanan Fattal, Maneesh Agrawala, and Szymon Rusinkiewicz. Multiscale shape and detail enhancement from multi-light image collections. *ACM Trans. Graph.*, 26(3):51, 2007.

[29] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Distance transforms of sampled functions. Technical Report TR2004-1963, Cornell University, 2004.

[30] William T. Freeman and Hao Zhang. Shape-time photography. In *Conference on Computer Vision and Pattern Recognition (CVPR '03)*, pages 151–157, 2003.

[31] Pau Gargallo and Peter Sturm. Bayesian 3D modeling from images using multiple depth maps. In *Proc. of ICCV'05*, volume 2, pages 885–891, 2005.

[32] Todor Georgiev. 3d graphics based on images and morphing. *US Patent 6268846*, 1998.

[33] Todor Georgiev and Michael Wainer. Morphing between multiple images. *Tech. Rep.*, 1997.

[34] Todor Georgiev, Ke Colin Zheng, Brian Curless, David Salesin, Shree Nayar, and Chintan Intwala. Spatio-angular resolution tradeoffs in integral photography. In *Rendering Techniques 2006: 17th Eurographics Workshop on Rendering*, pages 263–272, June 2006.

[35] A. Gerrard and J. M. Burch. Introduction to matrix methods in optics. 1994.

[36] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael Cohen. The lumigraph. *ACM Trans. Graph.*, pages 43–54, 1996.

[37] V. Guillemin and S. Sternberg. Symplectic techniques in physics. 1985.

[38] Earl Hammon. Practical post-process depth of field. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 28. Addison Wesley, 2007.

[39] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[40] Benno Heigl, Reinhard Koch, Marc Pollefeys, Joachim Denzler, and Luc J. Van Gool. Plenoptic modeling and rendering from image sequences taken by hand-held camera. In *DAGM-Symposium*, pages 94–101, 1999.

[41] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Transactions on Graphics*, 24(3):577–584, August 2005.

[42] Youichi Horry, Ken ichi Anjyo, and Kiyoshi Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *Proceedings of SIGGRAPH 97*, pages 225–232, August 1997.

[43] Stephen S. Intille and Aaron F. Bobick. Disparity-space images and large occlusion stereo. In *Proc. of ECCV'94*, volume 1, pages 179–186, 1994.

[44] Aaron Isaksen, Leonard McMillan, and Steven J. Gortler. Dynamically reparameterized light fields. *ACM Trans. Graph.*, pages 297–306, 2000.

[45] H. Ives. Camera for making parallax panoramagrams. *J. Opt. Soc. Amer., 17*, pages 435–439, 1928.

[46] S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. *IJCV*, 25(2):167–183, November 1997.

[47] S. B. Kang and R. Szeliski. Extracting view-dependent depth maps from a collection of images. *IJCV*, 58(2):139–163, July 2004.

[48] Sing Bing Kang. A survey of image-based rendering techniques. *SPIE Inter. Symp. on Electronic Imaging: Science and Technology*, 3641:2–16, 1999.

[49] Sing Bing Kang and Heung-Yeung Shum. A review of image-based rendering techniques. In *IEEE/SPIE Visual Communications and Image Processing 2000*, pages 2–13, 2002.

[50] Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High dynamic range video. *ACM Transactions on Graphics*, 22(3):319–325, July 2003.

[51] Michael Kass, Aaron Lefohn, and John D. Owens. Interactive depth of field using simulated diffusion. Technical Report 06-01, Pixar Animation Studios, January 2006.

[52] Andreas Klaus, Mario Sormann, and Konrad F. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In *International Conference on Pattern Recognition (ICPR)*, pages 15–18, 2006.

[53] Vladimir Kolmogorov and Ramin Zabih. Multi-camera scene reconstruction via graph cuts. In *Proc. of ECCV'02*, pages 82–96, 2002.

[54] Oliver Landolt and Ania Mitros. Visual sensor with resolution enhancement by mechanical vibrations. *Auton. Robots*, 11(3):233–239, 2001.

[55] Seungyong Lee, George Wolberg, and Sung Shin. Polymorph: Morphing among multiple images. *IEEE Computer Graphics and Applications*, 1998.

[56] Anat Levin, Rob Fergus, Frédo Durand, and William T. Freeman. Image and depth from a conventional camera with a coded aperture. page 70, New York, NY, USA, 2007. ACM.

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

[58] Marc Levoy and Pat Hanrahan. Light field rendering. *ACM Trans. Graph.*, pages 31–42, 1996.

[59] Yin Li, Heung-Yeung Shum, Chi-Keung Tang, and Richard Szeliski. Stereo reconstruction from multiperspective panoramas. *IEEE Trans. on PAMI*, 26(1):45–62, 2004.

[60] Chia-Kai Liang, Tai-Hsu Lin, Bing-Yi Wong, Chi Liu, and Homer Chen. Programmable aperture photography: Multiplexed light field acquisition. *ACM Transactions on Graphics*, 27(3), 2008.

[61] M. Lin and C. Tomasi. Surfaces with occlusions from layered stereo. *IEEE Trans. on PAMI*, 26(8):710–717, 2004.

[62] Gabriel Lippmann. Epreuves reversible donnant la sensation du relief. *J. Phys. 7*, pages 821–825, 1908.

[63] L. McMillan and G. Bishop. Plenoptic modeling: an image-based rendering system. *ACM Trans. Graph.*, pages 39–46, 1995.

[64] Francesc Moreno-Noguer, Peter N. Belhumeur, and Shree K. Nayar. Active refocusing of images and videos. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 67, New York, NY, USA, 2007. ACM.

[65] T. Naemura, T. Yoshida, and H. Harashima. 3d computer graphics based on integral photography. *Optics Express*, 8, 2001.

[66] Shree K. Nayar and Tomoo Mitsunaga. High dynamic range imaging: Spatially varying pixel exposures. In *Conference on Computer Vision and Pattern Recognition (CVPR '00)*, pages 1472–1479, 2000.

[67] Ren Ng. Fourier slice photography. *ACM Transactions on Graphics*, 24(3):735–744, August 2005.

[68] Ren Ng, Marc Levoy, Mathieu Brdif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Tech. Rep.*, 2005.

[69] Byong Mok Oh, Max Chen, Julie Dorsey, and Frédo Durand. Image-based modeling and photo editing. In *Proceedings of ACM SIGGRAPH 2001*, pages 433–442, August 2001.

[70] F. Okano, H. Hoshino, J. Arai, and I Yuyama. Real-time pickup method for a three-dimensional image based on integral photography. *Applied Optics,*, pages 1598–1603, 1997.

[71] Shmuel Peleg, Moshe Ben-Ezra, and Yael Pritch. Omnistereo: Panoramic stereo imaging. *IEEE Trans. on PAMI*, 23(3):279–290, 2001.

[72] Georg Petschnigg, Richard Szeliski, Maneesh Agrawala, Michael Cohen, Hugues Hoppe, and Kentaro Toyama. Digital photography with flash and no-flash image pairs. *ACM Transactions on Graphics*, 23(3):664–672, August 2004.

[73] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, September 2004.

[74] Kari Pulli, Michael F. Cohen, Tom Duchamp, Hugues Hoppe, Linda Shapiro, and Werner Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In *Eurographics Rendering Workshop 1997*, pages 23–34, June 1997.

[75] Ramesh Raskar, Kar-Han Tan, Rogerio Feris, Jingyi Yu, and Matthew Turk. Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging. *ACM Transactions on Graphics*, 23(3):679–688, August 2004.

[76] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[77] S. Seitz and J. Kim. The space of all stereo images. *IJCV*, 48(1):21–38, June 2002.

[78] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *2006 Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 519–528, June 2006.

[79] S. M. Seitz and C. R. Dyer. View morphing. *ACM Trans. Graph.*, pages 21–30, 1996.

[80] Steven M. Seitz and Charles R. Dyer. Photorealistic scene reconstruction by voxel coloring. In *Proc. of CVPR'97*, pages 1067–1073, 1997.

[81] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. *ACM Trans. Graph.*, pages 231–242, 1998.

[82] E. Shechtman, Yaron Caspi, and Michal Irani. Increasing space-time resolution in video. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 753–768, London, UK, 2002. Springer-Verlag.

[83] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996.

[84] H.-Y. Shum, S.B. Kang, and S.-C. Chan. Survey of image-based representations and compression techniques. *IEEE Trans. On Circuits and Systems for Video Technology*, 13:1020–1037, 2003.

[85] Heung-Yeung Shum and Li-Wei He. Rendering with concentric mosaics. *ACM Trans. Graph.*, 33:299–306, 1999.

[86] Heung-Yeung Shum, Jian Sun, Shuntaro Yamazaki, Yin Li, and Chi-Keung Tang. Pop-up light field: An interactive image-based modeling and rendering system. *ACM Trans. Graph.*, pages 143–162, 2004.

[87] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Transactions on Graphics*, 25(3):835–846, July 2006.

[88] Jason Stewart, Jingyi Yu, Steven J. Gortler, and Leonard McMillan. A new reconstruction filter for undersampled light field. *Eurographics Symposium on Rendering*, pages 150–156, 2003.

[89] Christoph Strecha, Rik Fransens, and Luc Van Gool. Combined depth and outlier estimation in multi-view stereo. In *Proc. of CVPR'06*, volume 2, pages 2394–2401, 2006.

[90] R. Szeliski. A multi-view approach to motion and stereo. In *Proc. of CVPR'99*, volume 1, pages 157–163, 1999.

[91] R. Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends in Computer Graphics and Computer Vision*, 2(1), 2006.

[92] Richard Szeliski and Polina Golland. Stereo matching with transparency and matting. In *Proc. of ICCV'98*, pages 517–526, 1998.

[93] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic mosaics and environment maps. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 251–258, 1997.

[94] Dayton Taylor. Virtual camera movement: The way of the future? *American Cinematographer*, 77(9):93–100, September 1996.

[95] R. Tyson. Principles of adaptive optics. In *Academic Press*, 1991.

[96] Vaibhav Vaish, Bennett Wilburn, Neel Joshi, and Marc Levoy. Using plane + parallax to calibrate dense camera arrays. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[97] Ashok Veeraraghavan, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. Dappled photography: mask enhanced cameras for heterodyned light fields and coded aperture refocusing. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers*, page 69, New York, NY, USA, 2007. ACM.

[98] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[99] Li wei He, Michael F. Cohen, and David H. Salesin. The virtual cinematographer: A paradigm for automatic real-time camera control and directing. In *Proceedings of SIGGRAPH 96*, Computer Graphics Proceedings, Annual Conference Series, pages 217–224, August 1996.

[100] B. Wilburn, N. Joshi, V. Vaish, E. Talvala, E. Antunez, A. Barth, A. Adams, M. Levoy, and M. Horowitz. High performance imaging using large camera arrays. In *ACM Trans. Graph.*, 2005.

[101] Daniel N. Wood, Adam Finkelstein, John F. Hughes, Craig E. Thayer, and David H. Salesin. Multiperspective panoramas for cel animation. In *Proceedings of SIGGRAPH 97*, Computer Graphics Proceedings, Annual Conference Series, pages 243–250, August 1997.

[102] Jiangjian Xiao and Mubarak Shah. Tri-view morphing. *Computer Vision and Image Understanding*, 96(3):345–366, 2004.

[103] C. Zhang and T. Chen. A system for active image-based rendering. *ICME 2003*, 2003.

[104] Cha Zhang and T. Chen. A survey on image-based rendering - representation, sampling and compression. *EURASIP Signal Processing: Image Communication*, 19:1–28, 2004.

[105] Ke Colin Zheng, Sing Bing Kang, Michael F. Cohen, and Richard Szeliski. Layered depth panoramas. In *CVPR*. IEEE Computer Society, 2007.

[106] C. L. Zitnick, N. Jojic, and S.B. Kang. Consistent segmentation for optical flow estimation. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2005.

[107] C. Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics*, 23(3):600–608, August 2004.

VITA

Ke Colin Zheng received his Bachelor of Science in Computer Science and Engineering from Zhejiang University in 2001. He joined the Computer Science and Engineering department at the University of Washington right afterwards to pursue research with his advisor David Salesin and Brian Curless. He also spent four summers at Microsoft Research working with principal researcher Michael Cohen and Rick Szelisksi. In 2008, after seven years of study, he received the Doctor of Philosophy degree.