New Models and Methods for Matting and Compositing

Yung-Yu Chuang

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

Doctor of Philosophy

University of Washington

2004

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

Yung-Yu Chuang

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 Supervisory Committee:

_____

Brian Curless

_____

David H. Salesin

Reading Committee:

_____

Brian Curless

_____

David H. Salesin

_____

Richard Szeliski

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 Bell and Howell Information and Learning, 300 North Zeeb Road, Ann Arbor, MI 48106-1346, 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

New Models and Methods for Matting and Compositing

by Yung-Yu Chuang

Co-Chairs of Supervisory Committee:

Professor Brian Curless
Computer Science & Engineering

Professor David H. Salesin
Computer Science & Engineering

Matting and compositing are fundamental operations in graphics and visual effects. Despite having enjoyed wide usage for many years, traditional matting and compositing have limitations. Traditional matting methods either require special setups or cannot handle objects with complex silhouettes. Furthermore, the traditional compositing model is effective in modeling color blending effects but not reflection, refraction, and shadows. In this dissertation, we address these limitations and present a set of new compositing models and matting methods. To pull mattes of complex silhouettes from natural images, we introduce a principled statistical approach called Bayesian image matting. We also extend this algorithm to handle video sequences with the help of optical flow computation and background estimation. On the compositing side, previous work on environment matting has been shown to handle refraction and reflection, but the resulting mattes are not very accurate. We propose a more accurate environment matting model and method that requires using more images. For shadows, we develop a physically-motivated shadow compositing equation. Based on this equation, we introduce a shadow matting method for extracting shadow mattes from videos with natural backgrounds, and we demonstrate a novel process for acquiring the photometric and geometric properties of the background to enable creation of realistic shadow composites. Finally, we present a novel application of Bayesian image matting for animating still pictures.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my advisors, Brian Curless, David Salesin and Richard Szeliski, who have made this dissertation possible. I am indebted to them for their guidance, inspiration and encouragement throughout my graduate studies. They are of different research and advising styles but equally great. I have learned a lot from them about doing research, presenting results and advising students. I consider myself very fortunate to have them as my advisors.

This dissertation is built upon the work that appeared in several papers I coauthored with my advisors and collaborators, Doug Zongker, Dan Goldman, Aseem Agarwala and Joel Hindorff, who have contributed to this work in many different ways.

I wish to thank the truly amazing people of GRAIL for making the lab the best place to do research.

I am grateful to Lindsay Michimoto, Frankye Jones, Jennifer Wagner and Alicen Smith for shielding me from routine paper work; Stephen Spencer and CSE support group for their help with system and hardware.

Thank you to my parents and siblings for their love and support.

Most importantly, I want to thank my wife, Laura, for her love and confidence in me, and our son Jason for making our lives busy but joyful.

DEDICATION

To Laura and Jason

Chapter 1

INTRODUCTION

Matting and compositing are fundamental operations in graphics. In the matting process, a foreground element of arbitrary shape is extracted from an image. In the compositing process, the extracted foreground element is placed over a novel background image. Matting and compositing were originally developed for film production. Today, matting and compositing have become crucial and frequently used operations in visual effects production. They enable directors to insert new elements seamlessly into a scene or to transport an actor into a completely new location. Nearly all modern movies utilize digital matting and compositing in their production; notable examples include "Jurassic Park," "The Matrix," and "The Lord of the Rings." The potential importance of matting and compositing can perhaps be best indicated by observing how lucrative visual effects movies have become [80]. According to "The Internet Movie Database," as of April 2004, six of the ten best-selling movies ever released also won the "Best Visual Effects" Academy Awards.[1] We would argue that the success of these films is directly related to their stellar visual effects. Digital matting and compositing, therefore, have huge money-making potential. In addition to being put to use for visual effects, digital matting and compositing are used in much of today's media, including magazines, 2D arts and graphics, television, advertising, video post production, and multimedia title development.

Although they were developed several decades ago, matting and compositing are still somewhat of a black art. Not much scientific study has been done until recently. Hence, most existing matting and compositing techniques are still fairly *ad hoc* and often limited to solving a restricted version of the general problem. Motivated by some recent promising work on matting and compositing, this dissertation provides a novel set of new models and principled techniques so that matting and

[1] The other three of them, "Start Wars: Episode I" and two "Harry Potter" movies, are fantasy movies. It would be impossible to make them without special effects. The only exception is Pixar's animation, "Finding Nemo."

compositing can be used in less restricted environments, model more complex phenomena, and be applied to exciting new applications.

In the remainder of this chapter, we first discuss the evolution and terminology of matting and compositing. We then review the relevant scientific literature on matting and compositing. Next, we define the problems that this dissertation attempts to solve and state the general solution strategies. The chapter concludes with an overview of this dissertation.

1.1 History and terminology

1.1.1 The optical era

The origins of matting and compositing can be traced back to photography and cinematography. One early composite example is the massive print, "The Two Ways of Life," created by the photographer Oscar Rejlander in 1857 by selectively combining 32 different negatives [18]. The development of compositing, however, was mostly driven by the need for cinematography to convincingly merge images filmed at different times or locations onto a single strip of film. One of the early tricks for compositing is the *double exposure* technique, blocking out part of the film during the first shot in order to preserve an unexposed area for the addition of other elements in later shots [74].

For combining films, in the early 1900s, optical printers were built, and the art of *optical compositing* was born. In the optical compositing process, three pieces of film are required to create a composite [11]:

- *Foreground plate*: the film containing the foreground element to be composited onto a different background.
- *Matte*: a monochrome film that determines the opacity of the foreground plate. The matte outside the foreground element is black to hold out the unwanted background in the foreground plate. The matte inside the foreground element is clear.
- *Background plate*: the backdrop onto which the foreground will be composited. This background usually comes from a filmed background, a miniature model, or a *matte painting* (a photorealistic painting of a real or imaginary scene).

During the compositing process, first, the background plate is exposed through negative of the matte

onto a film strip to hold a copy of the background scene with an unexposed area of the same shape and size of the foreground element. This film strip is then rewound and re-exposed to the foreground plate, along with the matte [74]. Exposure occurs only where the matte is clear, effectively compositing the foreground element onto the background plate. The film strip now holds the composite. With optical printers, compositing became an easy task. However, generating appropriate mattes was still difficult and many techniques were invented. The process of producing mattes is referred to by a variety of terms including *matting*, *keying*, *matte extraction*, and *pulling a matte*. In this dissertation, we refer to this process as matting. When compositing a sequence of images, the matte has to move through the sequence to track the foreground element. Such a moving matte is often referred to as a *traveling matte*. In this dissertation, we refer to the process of generating a traveling matte as *video matting*.

Most optical matting techniques involve shooting the foreground element in front of a constant-colored backdrop in order to easily isolate the foreground element from the unwanted background. For example, in the Williams process [74], the foreground element was photographed against an evenly lit plain black backdrop. The matte was essentially generated by enhancing the contrast of the film of the foreground element. This was done by copying the film a number of times using high-contrast films. Another matting technique of this kind is the color difference method developed and patented by Petro Vlahos. In this method, the foreground element was often filmed in front of a bright blue background screen. Hence, this method is also known as *blue screen matting*. In a simpler form of the color difference method, the greater of red and green channels is subtracted from the blue channel and the result is inverted to map to the opacity value. This method was the most popular and effective optical matting method from the mid-1960s until the advent of digital alternatives in the late 1980s. With bluescreen photography, other color information can also be used; for example, *luma keying* uses luminance values to distinguish the foreground from the background while *chroma keying* uses chrominance values.

When filming with specialized backdrops was impossible or impractical, effects artists had to draw their mattes by hand. It was done on an apparatus called a *rotoscope*, invented by animator Max Fleischer in 1917. Hence, this process is generically known as *rotoscoping*. Although eliminating the need for a specialized backdrop, rotoscoping is an intricate and time-consuming process.

1.1.2 The digital era

Since the 1980s, with the advancement of digital technology, optical compositing equipment has been largely replaced with computers and specialized software. However, most techniques and concepts of optical compositing are directly applicable to the digital realm. Thus, a digital matte was created to composite two digital images in direct emulation of the optical compositing technique. Early digital matting and compositing programs simply mimicked the techniques developed for optical compositing.

The use of computers, however, enabled the development of improved matting and compositing processes. For example, it helps improve blue screen matting techniques by allowing additional controls to adjust the relationships between the channels and the thresholds of certain values until a visually acceptable matte is obtained [74]. Digital matting technology also makes the setup for blue screen matting easier. Unlike in optical processes, the exact shade and color of the backdrop is not critical when setting up and taking a shot. The operator simply identifies a few background pixels, and the matting software can then be calibrated to remove only those exact colors. An example is Mishima's algorithm [60] that we will describe thoroughly in Section 1.2.

For rotoscoping, artists can draw an editable (e.g., B-spline) curve around the foreground element at selected *keyframes*, often with the help of an image snapping tool that adheres to high-gradient areas. The curves can then be interpolated over time for the frames in between. However, such automation usually only works for a simple shape that moves in a relatively linear fashion. Hence, more often than not, every matte in a sequence still has to be hand-drawn. In addition, a post processing step is needed to convert the contour into opacity profiles. This step is usually done by *ad hoc* feathering, an operation that blurs edges.

Digital technology also enabled the development of new matting methods. For example, *difference matting* is another matting method that does not require filming the foreground element against a constant-color screen. Difference matting requires two versions of a scene to be shot, one with the foreground element, and one without. The image filmed without the foreground element is called a *clean plate*. Difference matting takes the difference between these two shots and maps that difference to an opacity value [72]. However, this process is error prone where there are similarities in color between foreground and background, requiring additional user interaction to correct the matte.

1.2 Related work

The previous section presents the matting and composting techniques mostly developed by the film industry. In this section, we discuss some recent scientific studies that are generally related to digital matting and compositing. Discussion of work related to more specific problem domains that we address in this dissertation is deferred to the "Related work" sections of individual chapters.

The digital analog of the optical composting process can be summarized by the *compositing equation* [69],

$$C = \alpha F + (1 - \alpha)B\,, \tag{1.1}$$

where C, F, and B are the composite, foreground, and background images, respectively, and α is the matte.[2] As in optical compositing, the matte represents each pixel's opacity component used to linearly blend between foreground and background colors.

Compositing, as described by equation (1.1), is a straightforward operation. There have been roughly three major developments in digital compositing [47]. In 1977, Alvy Ray Smith and Ed Catmull coined the term alpha channel for the matte because the matte plays a similar role as α in the classic linear interpolation formula, $\alpha a + (1-\alpha)b$. They also invented the concept of *integral alpha* to treat alpha as integral to an image [86]. In 1980, Bruce Wallace and Marc Levoy showed how to blend partially transparent images to produce images with alpha, making compositing operation associative [104]. In 1984, Thomas Porter and Tom Duff [69] introduced the compositing algebra of 12 compositing operators and showed how synthetic images with alpha could be useful in creating complex digital images. In their paper, the composting operation described by equation (1.1) is named the *over* operation (in particular, F over B). They also introduced the concept of pre-multiplication by alpha so that an identical operation can be applied to the pre-multiplied *rgb* channels as well as the alpha channel. As later proven by others [14, 87], pre-multiplication is important for many image processing operations such as filtering and subsampling.

Matting, on the other hand, is more difficult. The matting process is the inverse process of compositing, starting from a photograph (essentially composite images C) and attempting to solve for the foreground image F, background image B, and alpha matte α. Since F, B, and C have three

[2] We use the symbol α to represent both the opacity value at a pixel and the opacity image taken as a whole. At times, to be more specific, we use $\alpha(p)$ for the opacity value at a pixel p, the same for C, F and B.

color channels each, for each pixel, we have a problem with three equations and seven unknowns. Hence, matting is inherently under-constrained, and most matting techniques solve it by controlling the background, adding images, or adding *a priori* assumptions about the foreground, background, and alpha.

By shooting the foreground object against a known constant-color background, blue screen matting approaches reduce the number of unknows to four. This problem is, however, still under-constrained. Hence, some simple constraints and heuristics have been employed to make the problem tractable. Some of these constraints and heuristics are nicely summarized by Smith and Blinn [88]. One such example is the color difference method invented by Vlahos; by making an assumption about the relative proportions of red, green, and blue for the foreground colors, a matte is obtained with the equation,

$$\alpha = 1 - a_1(C_b - a_2 C_g)\,, \tag{1.2}$$

where C_b and C_g are the blue and green channels of the input image, respectively [88]. The user can control the tuning parameters, a_1 and a_2, until a satisfactory matte is obtained. While generally effective and easy to implement, these approaches are in fact clever tricks. They are not mathematically valid, require an expert to tune them, and can fail on fairly simple foregrounds.

A more recent improvement on blue screen matting was developed by Mishima [60] based on representative foreground and background color samples. In particular, the algorithm starts with two identical polyhedral (triangular mesh) approximations of a sphere in rgb space centered at the average value $\overline{B}$ of the background samples. The vertices of one of the polyhedra (the *background polyhedron*) are then repositioned by moving them along lines radiating from the center until the polyhedron is as small as possible while still containing all the background samples. The vertices of the other polyhedron (the *foreground polyhedron*) are similarly adjusted to give the *largest* possible polyhedron that contains *no* foreground pixels from the sample provided. Given a new composite color C, then, Mishima casts a ray from $\overline{B}$ through C and defines the intersections with the background and foreground polyhedra to be B and F, respectively. The fractional position of C along the line segment BF is α. Although it requires the user to identify background samples, Mishima's algorithm places less restrictions on the backdrop colors and lighting conditions.

While blue screen matting is generally effective, taking a photograph of the foreground object

in front of a bright blue screen entails both restrictions on the foreground colors and *blue spill* (reflection of blue light from the blue screen on the foreground object). To avoid these problems, an alternative is to use dual-film techniques. The dual-film methods require the use of two separate films that could be exposed to the same scene simultaneously. That is usually implemented using a beam-splitting prism that divides the light entering through the lens to send identical images of the scene toward two separate films. Different filters are often used for these two films. One recent example of dual-film matting is the infrared (IR) matting system designed by Debevec *et al.* [29]. In their system, the backdrop is made of a material reflecting infrared light but not visible light. Infrared light sources are used to illuminate the backdrop (and inevitably the foreground object). A monochrome camera with an IR-pass filter is used for obtaining the matte while the other CCD camera records the foreground colors. Because the foreground object reflects much less infrared light than the foreground object, the matte can easily be computed by thresholding the IR-filtered image. Ultraviolet light can also be used to produce mattes in a similar way. Another example of dual-film methods is the invisible-signal keying system developed by Ben-Ezra [6]. In this system, the backdrop is a silver screen preserving the polarization of light. A polarized light is used to illuminate the scene. The light reflected by the background keeps polarization, but the light reflected by the foreground object loses it. A polarization beam splitter then divides the light to two cameras to create an "in-phase" image with a silver background and an "out-of-phase" image with a dark background. The differences of these two images are then mapped to opacity values to form a matte.

Some dual-film methods use information other than light for matting. In *depth keying* [38], the foreground object is segmented by thresholding the depth values of the scene; and, in *thermo keying* [117], the temperature is used to isolate the actor from his surrounding. These techniques get rid of the need for backdrops and are more suitable for outdoor shooting. However, the methods using measurements other than light often lead to "jagged" mattes because those measurements are not directly related to the partial coverage that matting processes attempt to estimate. Smoothing such mattes by feathering can help with the jaggedness but does not generally compensate for gross errors. This makes these methods less suitable for pulling high-quality mattes for objects with intricate shapes such as hair strands and fur. A common weakness for all dual-film methods is their use of expensive specialized cameras.

To overcome the under-constrained problem, Smith and Blinn [88] provided a *triangulation* solution for matting that was independently (and much earlier) developed by Wallace [105]: take an image of the same object in front of multiple known backgrounds. This approach leads to an over-constrained system and can be solved using a least-squares framework. While this approach requires even more controlled studio conditions than the single solid background used in blue screen matting and is not immediately suitable for live-action capture, it does provide a means of estimating highly accurate foreground and alpha values for real objects.

When filming with specialized backdrops is impossible or impractical, it is necessary to pull a matte from a photograph of the foreground object taken with a natural background. In this case, we have to solve the full matting problem with seven unknowns. We call this problem *natural image matting*. Rotoscoping is a commonly used technique for solving this problem. However, as stated in the previous section, rotoscoping is often inefficient, and the conversion of mattes is *ad hoc*. Mitsunaga *et al.* [61] developed the AutoKey system to improve the rotoscoping process for video matting. In this system, the keyframe roto-curves are tracked over time so as to adhere to foreground contours. Tracked results tend to require less editing than interpolated results, but they often still require frame-by-frame hand adjustment in order to pull a high-quality matte. To avoid using *ad hoc* feathering in the conversion from contours to opacity profiles, they propose an adaptive feathering method. By taking the partial derivatives on both sides of the compositing equation (1.1), they observe that, under the assumption that F and B are roughly constant,

$$\nabla\alpha \approx \frac{1}{F-B}\nabla C\,, \tag{1.3}$$

where $\nabla = \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}\right]$ is the gradient operator. We call the above equation the *AutoKey constraint*. The AutoKey constraint suggests that the amount of blur in the matte depends on the image gradient. This approach, however, makes strong smoothness assumptions about the foreground and background and is designed for use with fairly hard edges in the transition from foreground to background; i.e., it is not well-suited for transparency and hair-like silhouettes. Recently, Agarwala *et al.* proposed a keyframe-based system to effectively reduce the amount of human effort for rotoscoping [1]. However, the difficulties of handling intricate silhouettes remain.

This dissertation is mainly inspired by two recent developments in matting and compositing: a probabilistic natural image matting algorithm by Ruzon and Tomasi [76] and environment mat-

ting and compositing by Zongker *et al.* [118]. The first development is related to pulling mattes from footage filmed in front of a natural background. Although such techniques exist, previous approaches fail to handle foreground objects with complex shapes. Ruzon and Tomasi proposed a statistical approach to address this deficiency [76]. With a modest amount of user intervention, their algorithm is capable of pulling a matte for a foreground object with intricate boundaries from a natural background by building statistics for foreground and background colors. Along this line, we develop a more principled matting approach for handling images and videos filmed with natural backgrounds. The second development is related to modeling phenomena that the traditional compositing model fails to capture. Although the traditional compositing model can model transport paths of scalar (non-color) attenuation and partial pixel coverage, it does not model how objects refract and reflect light. Zongker *et al.* introduced *environment matting and compositing* to generalize the traditional matting and compositing processes to incorporate refraction and reflection [118]. Following their work, we propose a novel compositing model for shadows and a practical process for shadow matting and compositing. Furthermore, we improve the environment matting algorithm to acquire more accurate environment mattes.

1.3 Problem statement

The compositing equation (1.1) defines how to perform the compositing operation, but not what compositing attempts to accomplish. To address this question, we seek ideas from practical compositing experts. The book, "The Art and Science of Digital Compositing," gives the following definition for digital compositing [18]:

> **Digital Compositing:** *The digitally manipulated combination of at least two source images to produce an integrated result.*

What does *integrated* mean? Another book on compositing, "Digital Compositing for Film and Video," offers an insight into this question [115]:

> *The ultimate artistic objective of a digital composite is to take images from a variety of different sources and combine them in such a way that they appear to have been shot at the same time under the same lighting conditions with the same camera.*

Hence, we can view compositing as an image-based rendering approach, because it uses images (samples of light) as the fundamental modeling primitives to synthesize a photorealistic composite image (a novel set of light samples).

An image C, either captured by an imaging system or synthesized by a rendering algorithm, records a discrete set of light samples over an image plane. For a given pixel p, the sample records an average color $C(p)$ of the light rays that strike that pixel. Each ray records the light transport from lights to the camera through a scene. Hence, the image C is a function of the camera characteristics, the lighting conditions, and the scene model,

$$C = Rendering(Model, Camera, Light)\,. \tag{1.4}$$

Previous image-based rendering approaches synthesize an image by manipulating a set of images acquired by either varying the camera views for a fixed scene and lighting (*view interpolation* [35, 49, 81, 114]) or varying the lighting conditions for a fixed scene and camera (*relighting* [28, 55]), or a combination of the two [58]. For example, for view interpolation, a set of images are taken from different viewpoints for a fixed scene and lighting condition,

$$C_i = Rendering(Model, Camera_{(i)}, Light), i = 1 \ldots n\,. \tag{1.5}$$

The image C' for a novel view $Camera'$ is then obtained by manipulating this set of images C_i.

In the context of compositing, we assume that the scene model can be decomposed into two parts, a foreground model $Model_F$ and a background model $Model_B$. Hence,

$$C = Rendering(\{Model_F, Model_B\}, Camera, Light)\,. \tag{1.6}$$

A compositing model describes how to approximate C from a foreground image F and a background image B through a matte, where

$$F = Rendering(\{Model_F\}, Camera, Light)\,, \tag{1.7}$$

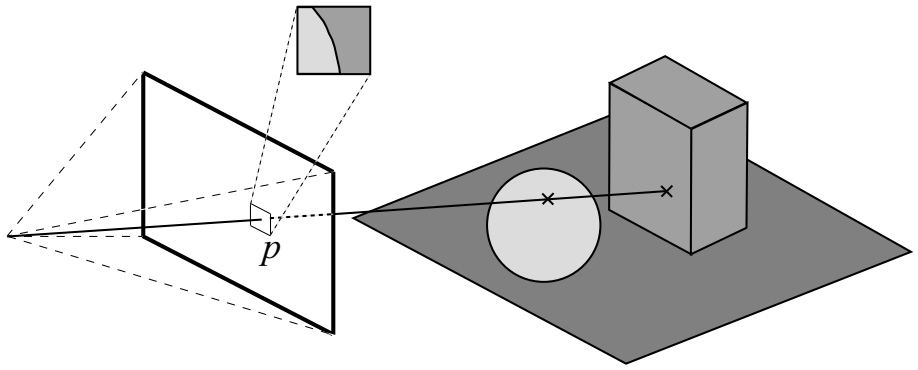
$$B = Rendering(\{Model_B\}, Camera, Light)\,. \tag{1.8}$$

To achieve this, the matte has to carry information about the light interaction between the foreground model and the other parts of the scene (the background model and lights). The traditional compositing model captures the inter-visibility between the foreground model and the background model.

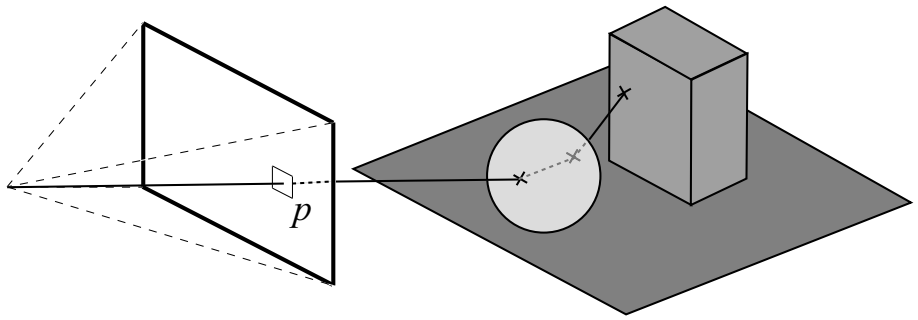
For example, in Figure 1.1(a), the foreground model only partially covers the pixel p. Hence, the matte is supposed to be an opacity map for the foreground object, correctly modeling partial coverage at the blending edges, as well as non-chromatic, non-refractive foreground transparency. The traditional compositing equation (1.1) describes how to blend the colors of F and B through the matte. This model, however, incorrectly represents some common lighting phenomena such as the occlusion of light in shadows (Figure 1.1(b)) and the change of the light path in refraction and reflection (Figure 1.1(c)).

The goal of a matting process is to extract the matte, essentially explaining how the image C is formed based on the compositing model. This process, however, is often ill-posed because there is more than one possible scene configuration that would result in the same image C. Common strategies for solving an ill-posed problem include reducing the number of unknowns, adding constraints, and adding statistical priors to the solution. For example, to reduce the number of unknowns, we can take a photograph of the foreground model in front of a "known" background model. We can even "control" the background model so that the matting problem becomes easier. An example of traditional matting algorithms using this strategy is blue screen matting. We call the methods requiring "known" backgrounds *active methods* and the ones without this restriction *passive methods*. To add constraints, we can take several photographs of the foreground model in front of different background models. An example is the triangulation algorithm. We call the methods requiring multiple images *multiple-frame methods* and the ones using a single image *single-frame methods*. To reduce the number of required images, a multiple-frame solution is often combined with an active method to control the background models and to explore the space of all possible backgrounds more efficiently. Hence, matting algorithms primarily differ in the number of images required and whether the background is "controlled" during the acquisition. For a passive single-frame matting problem, to make the problem tractable, statistical priors (or just "priors") are often added to arrive at plausible solutions. An example is the Ruzon-Tomasi algorithm for natural image matting. Algorithms in this category differ in terms of the choice of priors and the method of applying them to arrive at a solution.

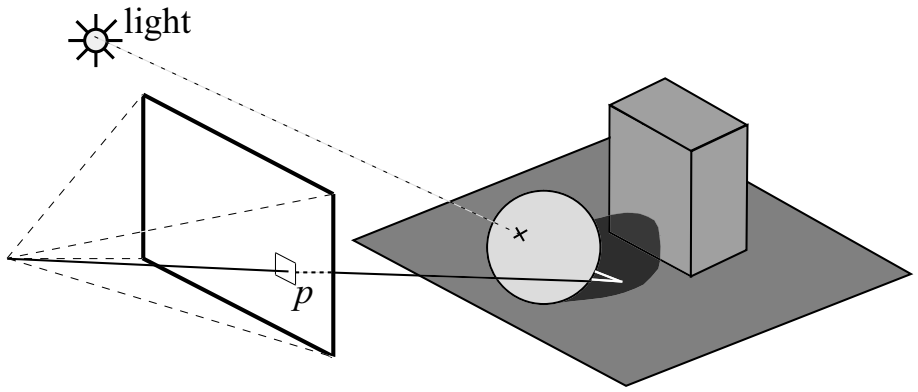
Table 1.1 gives a classification of previous matting algorithms before our work, based on whether the acquisition process is "controlled" and the number of images required. Generally, for matting, we prefer passive approaches over active ones and single-frame methods over multiple-frame tech-



(a) Partial coverage



(b) Refraction/reflection



(c) Shadows

Figure 1.1: Light transport and compositing. In the scenes shown in this figure, the spherical object is taken to be the foreground, while the rest of the scene constitutes the background. The traditional compositing equation effectively models partial coverage (a), assuming that the light goes straight from the background through the foreground to the camera. However, in the case of refraction and reflection (b), the light transport path is distorted by the foreground object and is not a straight line. Hence, the traditional compositing equation cannot handle this lighting effect. Moreover, this equation cannot handle shadows (c), which is the result of light occlusion.

Table 1.1: A summary of previous matting and compositing methods and the roadmap of this dissertation.[a] This table summarizes existing approaches before our work. Text in bold represents our work, to be discussed in later chapters.

			Traditional matting and compositing	Shadow matting and compositing	Environment matting and compositing (EM)
lighting phenomena			partial coverage transparency	shadows	refraction reflection
compositing			$C = \alpha F + (1-\alpha)B$ Porter & Duff [69]	**Chapter 4**	$C = F + \int W(\boldsymbol{x})B(\boldsymbol{x})d\boldsymbol{x}$ Zongker *et al.* [118][b]
matting	passive	single -frame	depth/thermo keying rotoscoping Ruzon-Tomasi [76] **Chapter 2**	**Chapter 4**	
		multiple -frame[c]	**Chapter 3**		
	active	single -frame	blue screen matting difference matting[d] dual-film [6, 29][e]		
		multiple -frame	$O(1)$: triangulation [88]		$O(\log k)$: Zongker [118] $O(k)$: **Chapter 5**

[a] This table is by no means complete; only representative work is listed. For matting algorithms, this table is more a categorization than a comparison. Because of the assumptions they make, these techniques actually solve related but somewhat different problems.

[b] The original environment matting equation proposed by Zongker *et al.* is slightly less general than the one listed here.

[c] The algorithms in this category take a sequence of images as input and output either a single matte or a sequence of mattes (e.g., video matting in Chapter 3). Because such algorithms usually take advantage of the temporal coherence within the input sequence, they do not work for a single image.

[d] While the background is not "controlled" in difference matting, the camera path often has to be controlled for the ease of obtaining clean plates. Hence, we count it as an active approach.

[e] This approach essentially requires two frames, but they are taken simultaneously.

niques. However, this does not mean that passive single-frame algorithms in Table 1.1 are superior to active multiple-frame techniques. All these techniques make different assumptions for the inherently under-constrained matting problem and turn out to solve related but somewhat different problems, each with their own pros and cons. For example, although blue screen matting methods require only a single frame, they make assumptions about and restrictions on the colors of the foreground object. On the other hand, the triangulation method requires multiple frames but places no such restrictions. As another example, rotoscoping does not involve expensive blue screens but requires intensive user intervention and only works for objects that have simple outlines. Hence, no single matting technique is clearly better than another and the choice of which one to use depends on the particular situation [18].

1.4 Overview of dissertation

Table 1.1 also provides a roadmap for this dissertation. The next two chapters explain our approach for matting from natural backgrounds. Chapter 2 describes our Bayesian framework for pulling mattes from images. Chapter 3 extends this framework to handle videos by interpolating user inputs using bidirectional optical flows. A novel technique for smoke matte extraction is also demonstrated. Chapters 4 and 5 deal with the phenomena that the traditional composting model fails to capture: shadows, reflection, and refraction. In Chapter 4, we develop physically-based shadow matting and compositing equations and use them to pull a traveling (moving) shadow matte from a source video filmed with a natural background. For shadow compositing, we propose an acquisition process for obtaining the photometric and geometric properties of the target background scene. In Chapter 5, we present our environment matting method for capturing a more accurate matte than the previous approach, at the expense of using more structured light backdrops. Chapter 6 describes an application of our Bayesian matting algorithm to creating a video texture from a single still image. Finally, Chapter 7 concludes this dissertation by summarizing our contributions and suggesting future research directions.

Chapter 2

BAYESIAN IMAGE MATTING

2.1 Introduction

While active matting approaches are usually more effective, they often require expensive setups, and the unnatural backdrops tend to induce worse performances by actors. Moreover, at times, it is impractical (e.g., outdoor shooting), inefficient (e.g., shooting for *background editing*[1]), or even impossible (e.g., pulling a matte from stock footage) to employ active approaches. In these cases, passive approaches are preferable. From Table 1.1, one option for pulling mattes without controlled backdrops is depth keying or thermo keying. While automatic, both methods require expensive, specialized cameras. In addition, these methods are not suitable for foreground objects with complex silhouettes, and they cannot be applied to stock footage. Another option is rotoscoping. Rotoscoping eliminates the requirement for expensive cameras and is applicable to stock footage, but at the expense of requiring user intervention. Still, like depth or thermo keying, it fails to pull satisfactory mattes for objects with intricate boundaries.

Recently, statistical approaches have been proposed to solve the natural image matting problem. These approaches attempt to pull mattes from natural (arbitrary) backgrounds by sampling colors from known foreground and background regions in order to obtain statistics for the foreground and background colors along the boundary. These statistics are then used to pull a matte in an unknown region in a statistically meaningful way [93]. As with rotoscoping, these methods require modest amounts of user interaction. However, unlike rotoscoping, they can handle difficult cases such as thin whisps of fur or hair.

In this chapter, we present a novel statistical method for solving the natural image matting problem.[2] We first survey the previous statistical approaches for natural image matting (Section 2.2), all

[1] In background editing, an object is inserted between the foreground element and the background plate

[2] This chapter describes joint work with Brian Curless, David H. Salesin and Richard Szeliski, first presented in IEEE CVPR 2001 [21].

of which are fairly *ad hoc*. We then introduce a new, more principled approach to natural image matting, based on a Bayesian framework (Section 2.3). While no algorithm can give perfect results (given that the problem is inherently under-constrained), our Bayesian approach appears to provide an improvement over existing approaches (Section 2.4). However, in regions of low contrast or high textures, the Bayesian algorithm may fail to give satisfactory results. We propose a regularization approach to improve the mattes for these cases (Section 2.5). Finally, we conclude this chapter by discussing extensions and applications of the Bayesian matting algorithm (Section 2.6).

2.2 Related work

As mentioned in Section 1.3, the natural image matting problem is under-constrained. Statistical approaches are effective for solving under-constrained problems because they provide a framework to encode prior knowledge about possible solutions. To build priors, most statistical matting approaches require the user to supply a hint image that partitions the input image into three regions: "foreground", "background," and "unknown," with the background and foreground regions having been delineated conservatively. We call such a hint image a *trimap*. In the unknown region, the matte can be estimated using the color statistics of the known foreground and background regions.

Most statistical matting techniques consist of two steps [93]. First, foreground and background color samples for each pixel in the unknown region are collected from the foreground and background regions. Based on the collected samples, statistical representations are calculated to summarize the foreground and background appearances. Second, the matte is estimated for each pixel in a particular order, given the foreground and background representations. In this section, we review three such techniques: Knockout,[3] developed by Ultimatte (and, to the best of our knowledge, described in patents by Berman *et al.* [7, 8]); the Ruzon-Tomasi algorithm [76]; and the technique of Hillman *et al.* [40]. The first two algorithms predate our Bayesian algorithm, while Hillman's algorithm is contemporary with ours.

For Knockout, after user segmentation, the next step is to extrapolate the known foreground and background colors into the unknown region. In particular, given a point in the unknown region, the

[3] Although Knockout is not strictly a statistical method, in this section, we generically classify the methods solving a problem using samples as statistical approaches. By this definition, Mishima's method is also a statistical method, but it is for blue screen matting, not natural image matting.

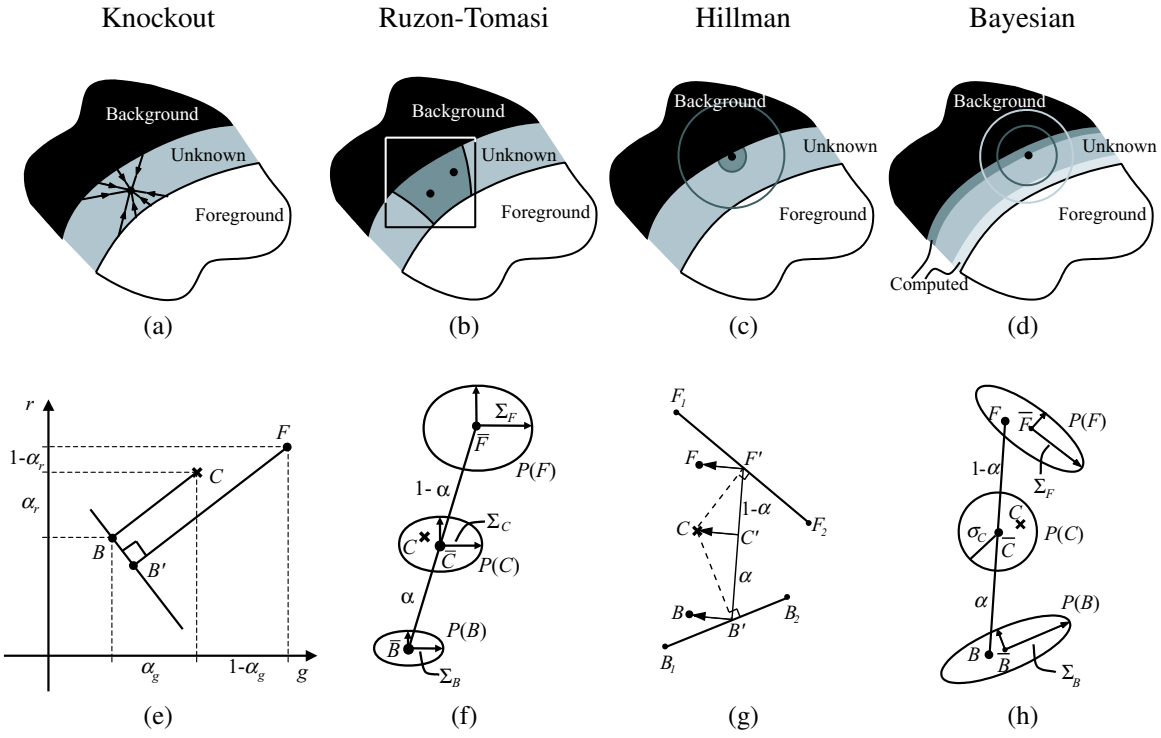


Figure 2.1: Summary of statistical natural image matting algorithms. Each of the algorithms shown in this figure requires a trimap to specify known background, known foreground, and unknown pixels. All algorithms estimate alpha for the unknown pixels using local color distributions of the nearby known foreground and background regions. The dark gray areas in (b) and (c) correspond to segments within the unknown region that will be evaluated using the same statistics derived from the square (b) or circular (c) region's overlap with the labeled foreground and background. The dark gray and light gray bands in (d) represent the computed unknown pixels while the marked pixel is being computed. The dark gray and light gray circles are the neighborhoods used for collecting the background and foreground statistics, respectively. Because the marked pixel is closer to the known background region, the neighborhood for the background is smaller than that for the foreground. Figures (e)-(h) show how matte parameters are computed using the Knockout, Ruzon-Tomasi, Hillman and our Bayesian approach, respectively.

foreground F is calculated as a weighted sum of the pixels on the perimeter of the known foreground region. The weight for the nearest known pixel is set to 1, and this weight tapers linearly with distance, reaching 0 for pixels that are twice as distant as the nearest pixel. The same procedure is used for initially estimating the background B' based on nearby known background pixels. Figure 2.1(a) shows a set of pixels that contribute to the calculation of F and B' of an unknown pixel.

The estimated background color B' is then refined to give B using one of several methods [8] that are all similar in character. One such method establishes a plane through the estimated background color with normal parallel to the line $B'F$. The pixel color in the unknown region is then projected along the direction of the normal onto the plane, and this projection becomes the refined guess for B. Figure 2.1(e) illustrates this procedure. Several simple variations on this plane computation and projection step are described in the patent [8].

Finally, Knockout estimates α according to the relation

$$\alpha = \frac{\varphi(C) - \varphi(B)}{\varphi(F) - \varphi(B)}\,, \tag{2.1}$$

where $\varphi(\cdot)$ projects a color onto one of several possible axes through rgb space (e.g., onto one of the r-, g-, or b- axes). Figure 2.1(e) illustrates alphas computed with respect to the r- and g- axes. In general, α is computed by projection onto all of the chosen axes, and the final α is taken as a weighted sum over all the projections, where the weights are proportional to the denominator in equation (2.1) for each axis.

Ruzon and Tomasi [76] take a probabilistic view that is somewhat closer to our own. First, they partition the unknown boundary region into sub-regions. For each sub-region, they construct a box that encompasses the sub-region and includes some of the nearby known foreground and background regions (see Figure 2.1(b)). The selected foreground and background pixels are then treated as samples from distributions $P(F)$ and $P(B)$, respectively, in color space. The foreground pixels are split into coherent clusters, and unoriented Gaussians (i.e., Gaussians that are axis-aligned in color space) are fit to each cluster, each with mean $\overline{F}$ and diagonal covariance matrix Σ_F. In the end, the foreground distribution is treated as a mixture (sum) of Gaussians. The same procedure is performed on the background pixels yielding Gaussians with mean $\overline{B}$ and covariance Σ_B. Every foreground cluster is then paired with every background cluster. Many of these pairings are rejected based on various "intersection" and "angle" criteria. Figure 2.1(f) shows a single pairing for a

Table 2.1: A summary of the five components in statistical natural image matting algorithms.

component	Knockout	Ruzon-Tomasi	Hillman	Bayesian
pixel ordering	raster order	region by region	raster order	onion peel
sample collection	Figure 2.1(a)	Figure 2.1(b)	Figure 2.1(c)	Figure 2.1(d)
color representation	extrapolated colors	axis-aligned Gaussians	oriented line segments	oriented Gaussians
matte estimation	Figure 2.1(e) (equation (2.1))	Figure 2.1(f) (ML[a])	Figure 2.1(g) ($C'B'/F'B'$)	Figure 2.1(h) (MAP[b])
color estimation	extrapolated colors	perturbation	perturbation	MAP[b]

[a] ML: maximum likelihood

[b] MAP: *maximum a posteriori*

foreground and background distribution.

After building this network of paired Gaussians, Ruzon and Tomasi treat the observed color C as coming from an intermediate distribution $P(C)$, somewhere between the foreground and background distributions. The intermediate distribution is also defined to be a sum of Gaussians, where each Gaussian is centered at a distinct mean value $\overline{C}$ located fractionally (according to a given alpha) along a line between the mean of each foreground and background cluster pair with fractionally interpolated covariance Σ_C, as depicted in Figure 2.1(f). The optimal alpha is the one that yields an intermediate distribution for which the observed color has maximum probability; i.e., the optimal α is chosen independently of F and B. As a post-process, the F and B are computed as weighted sums of the foreground and background cluster means using the individual pairwise distribution probabilities as weights. The F and B colors are then perturbed to force them to be endpoints of a line segment passing through the observed color and satisfying the compositing equation.

Instead of using axis-aligned Gaussians, Hillman *et al.* use principal component analysis (PCA) to represent color samples with oriented line segments [40]. The choice of oriented line segments is based on the observation that the color clusters tend to be prolate (cigar shaped) in *rgb* color space. This is probably because the pixels are either of the same basic color with varying degrees of illumination or else are part of a transition between two colors. The pixels in the unknown region are processed in turn, by scanning the image in raster order until reaching the next unprocessed pixel p.

All unprocessed pixels within a fixed radius of p share the same background and foreground models. For building the foreground model, pixels within a fixed radius of p in the known foreground region are collected (see Figure 2.1(c)). The same procedure is used to generate the background samples. PCA is used to find the major orientation of these color samples. Then, the collected color samples are projected onto the major axis found by PCA, essentially discarding the variances on the minor axes. Extreme points of the projected color samples are found to form the line segments F_1F_2 and B_1B_2, as depicted in Figure 2.1(g), for the foreground and background, respectively.

Figure 2.1(g) illustrates the matte and color estimation procedure. Given the line segments for foreground and background, to estimate the opacity value for an observed color C, the closest points, F' and B', to C on the line segments F_1F_2 and B_1B_2, respectively are found. Let C' be the projection of C onto the line segment $F'B'$. The opacity value α is then estimated as the ratio of the lengths of line segments $C'B'$ and $F'B'$. Finally, the foreground and background colors, F and B, are estimated by perturbing F' and B' in a similar way as in the Ruzon-Tomasi algorithm.

To sum up, these statistical natural image matting approaches primarily differ in (1) the pixel ordering for matte estimation, (2) the sample collection procedure, (3) the foreground and background color representations, (4) the matte estimation method, and (5) the foreground color estimation method. Table 2.1 summarizes these five components of the statistical natural matting approaches discussed in this chapter. In the next section, we present our own choices for these components.

2.3 Bayesian framework

The goal of a natural image matting algorithm is to solve for the foreground color F, the background color B, and the opacity α given the observed color C for each pixel within the unknown region of the image. Like other statistical matting algorithms, we solve the problem in part by building foreground and background probability distributions from a given neighborhood. Our method, however, uses a continuously sliding window for neighborhood definitions, marches inward from the foreground and background regions (such a filling order is called an *onion peel* order), and utilizes nearby *computed* F, B, and α values (in addition to these values from "known" regions) in constructing oriented Gaussian distributions, as illustrated in Figure 2.1(d). Further, our approach formulates the problem of computing matte parameters in a well-defined Bayesian framework and

solves it using the *maximum a posteriori* (MAP) technique. In this section, we describe our Bayesian framework in detail.

In MAP estimation, we try to find the most likely estimates for F, B, and α, given the observation C. We can express this as a maximization over a probability distribution P and then use Bayes's rule to express the result as the maximization over a sum of log likelihoods,

$$\begin{aligned} & \arg\max_{F,B,\alpha} P(F,B,\alpha \mid C) \\ &= \arg\max_{F,B,\alpha} P(C \mid F,B,\alpha)\, P(F)\, P(B)\, P(\alpha) \,/\, P(C) \\ &= \arg\max_{F,B,\alpha} L(C \mid F,B,\alpha) + L(F) + L(B) + L(\alpha)\,, \end{aligned} \tag{2.2}$$

where $L(\cdot)$ is the *log likelihood* $L(\cdot) = \log P(\cdot)$, and we drop the $P(C)$ term because it is a constant with respect to the optimization parameters. (Figure 2.1(h) illustrates the distributions over which we solve for the optimal F, B, and α parameters.)

The problem is now reduced to defining the log likelihoods $L(C \mid F,B,\alpha)$, $L(F)$, $L(B)$, and $L(\alpha)$. Using Gaussian noise to model the error in image acquisition, we can model the first term by measuring the difference between the observed color and the color predicted by the estimated F, B, and α,

$$L(C \mid F,B,\alpha) \;=\; -\|C - \alpha F - (1-\alpha)B\|^2 / \sigma_C^2\,. \tag{2.3}$$

This log-likelihood models error in the measurement of C and corresponds to a Gaussian probability distribution centered at $\overline{C} = \alpha F + (1-\alpha)B$ with standard deviation σ_C.

We use the spatial coherence of the image to estimate the foreground term $L(F)$. That is, we build the color probability distribution using the known and previously estimated foreground colors within each pixel's neighborhood N. The neighborhood N of a pixel p is initially set as a circular region of a user-defined radius centered at p and incrementally expanded until there are more than 15 samples within N. To model the foreground color distribution more robustly, we weight the contribution of each nearby pixel p in N according to two separate factors. First, we weight the pixel's contribution by $\alpha(p)^2$, which gives colors of more opaque pixels higher confidence. Second, we use a spatial Gaussian fall-off $g(p)$ with $\sigma = 8$ to stress the contribution of nearby pixels over those that are further away. The combined weight is then $w(p) = \alpha(p)^2 g(p)$.

Given a set of foreground colors and their corresponding weights, we first partition colors into several clusters using the method of Orchard and Bouman [64]. For each cluster, we calculate the weighted mean color $\overline{F}$ and the weighted covariance matrix Σ'_F,

$$\overline{F} = \frac{1}{W}\sum_{p\in N} w(p)\,F(p)\,, \tag{2.4}$$

$$\Sigma'_F = \frac{1}{W}\sum_{p\in N} w(p)\,(F(p)-\overline{F})\,(F(p)-\overline{F})^T\,, \tag{2.5}$$

where $W=\sum_{p\in N} w(p)$. Because foreground color samples are also observations from a camera, they are subject to the influence of imaging noise, as are the observations C. Hence, we should add the same amount of measurement variance σ_C^2 to the covariance matrices Σ'_F,

$$\Sigma_F = \Sigma'_F + \sigma_C^2\mathcal{I}. \tag{2.6}$$

By doing so, we also avoid most of the degenerate cases when all the color samples are similar.

The log likelihoods for the foreground $L(F)$ can then be modeled as being derived from an oriented elliptical Gaussian distribution, using the weighted covariance matrix as follows:

$$L(F) = -(F-\overline{F})^T\,\Sigma_F^{-1}\,(F-\overline{F})\,/\,2\,. \tag{2.7}$$

The definition of the log likelihood for the background $L(B)$ depends on which matting problem we are solving. For natural image matting, we use an analogous term to that of the foreground, setting $w(p)$ to $(1-\alpha(p))^2 g(p)$ and substituting B in place of F in every term of equations (2.4), (2.6), and (2.7). Note that the neighborhood for collecting background samples is determined similarly as the neighborhood for the foreground. These two neighborhoods are not necessarily the same. For constant-color matting, we calculate the mean and covariance for the set of all pixels that are labelled as background. For difference matting, i.e., when a clean plate is available, we have the background color at each pixel; we therefore use the known background color as the mean and a user-defined variance to model the noise of the background.

For now, we assume that the log likelihood for the opacity $L(\alpha)$ is constant (and thus omitted from the maximization in equation (2.2)). Later, in Section 2.5, we will describe a regularization approach to incorporate $L(\alpha)$.

Because of the multiplication of α with F and B in the log likelihood $L(C\,|\,F,B,\alpha)$, the function we are maximizing in (2.2) is not a quadratic equation in its unknowns. To solve the equation

efficiently, we break the problem into two quadratic sub-problems. In the first sub-problem, we assume that α is a constant. Under this assumption, taking the partial derivatives of (2.2) with respect to F and B and setting them equal to 0 gives

$$\begin{bmatrix} \Sigma_F^{-1} + \mathcal{I}\alpha^2/\sigma_C^2 & \mathcal{I}\alpha(1-\alpha)/\sigma_C^2 \\ \mathcal{I}\alpha(1-\alpha)/\sigma_C^2 & \Sigma_B^{-1} + \mathcal{I}(1-\alpha)^2/\sigma_C^2 \end{bmatrix} \begin{bmatrix} F \\ B \end{bmatrix} = \begin{bmatrix} \Sigma_F^{-1}\overline{F} + C\alpha/\sigma_C^2 \\ \Sigma_B^{-1}\overline{B} + C(1-\alpha)/\sigma_C^2 \end{bmatrix}, \quad (2.8)$$

where $\mathcal{I}$ is a 3×3 identity matrix. Therefore, for a constant α, we can find the best parameters F and B by solving the 6×6 linear system (2.8).

In the second sub-problem, we assume that F and B are constant, yielding a quadratic equation in α. We arrive at the solution to this equation by projecting the observed color C onto the line segment BF in color space,

$$\alpha = \frac{(C-B)\cdot(F-B)}{\|F-B\|^2}, \quad (2.9)$$

where the numerator contains a dot product between two color difference vectors. To optimize the overall equation (2.2) we alternate between assuming that α is fixed to solve for F and B using equation (2.8), and assuming that F and B are fixed to solve for α using equation (2.9). To start the optimization, we initialize α with the mean α over the neighborhood of nearby pixels and then solve the constant-α equation (2.8).

When there is more than one foreground or background cluster, we perform the above optimization procedure for each pair of foreground and background clusters and choose the pair with the maximum likelihood. Note that this model, in contrast to a mixture of Gaussians model, assumes that the observed color corresponds to exactly one pair of foreground and background distributions. In some cases, this model is likely to be the correct model, but we can certainly conceive of cases where mixtures of Gaussians would be desirable, say, when two foreground clusters can be near one another spatially and thus can mix in color space. Ideally, we would like to support a true Bayesian mixture model. In practice, even with our simple exclusive decision model, we have obtained better results than the existing approaches.

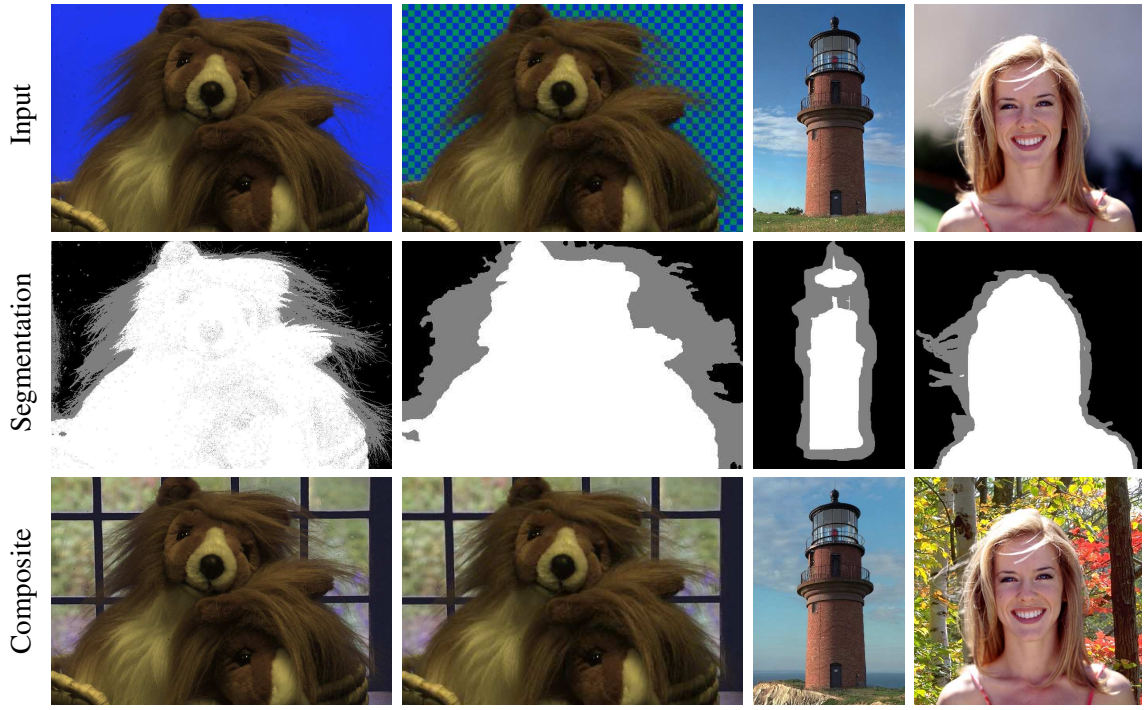


Figure 2.2: Summary of input images and results. Input images (top row): a blue screen matting example of a toy lion, a synthetic "natural image" of the same lion (for which the exact solution is known), and two real natural images, (a lighthouse and a woman). Input trimap segmentation (middle row): conservative foreground (white), conservative background (black), and "unknown" (grey). The leftmost trimap was computed automatically (see text), while the rightmost three were specified by hand. Compositing results (bottom row): the results of compositing the foreground images and mattes extracted through our Bayesian matting algorithm over new background scenes. (Lighthouse image and the background images in composite courtesy Philip Greenspun, http://philip.greenspun.com. Woman image was obtained from Corel Knockout's tutorial, Copyright © 2001 Corel. All rights reserved.)

2.4 Results and comparisons

We tested our Bayesian approach on a variety of different input images, both for blue screen and for natural image matting. Figure 2.2 shows four such examples. In the rest of this section, we discuss each of these examples and provide comparisons between the results of our algorithm and those of previous approaches.[4]

[4]For natural image matting, we compare our Bayesian algorithm with Knockout and the Ruzon-Tomasi algorithms but not the Hillman algorithm.

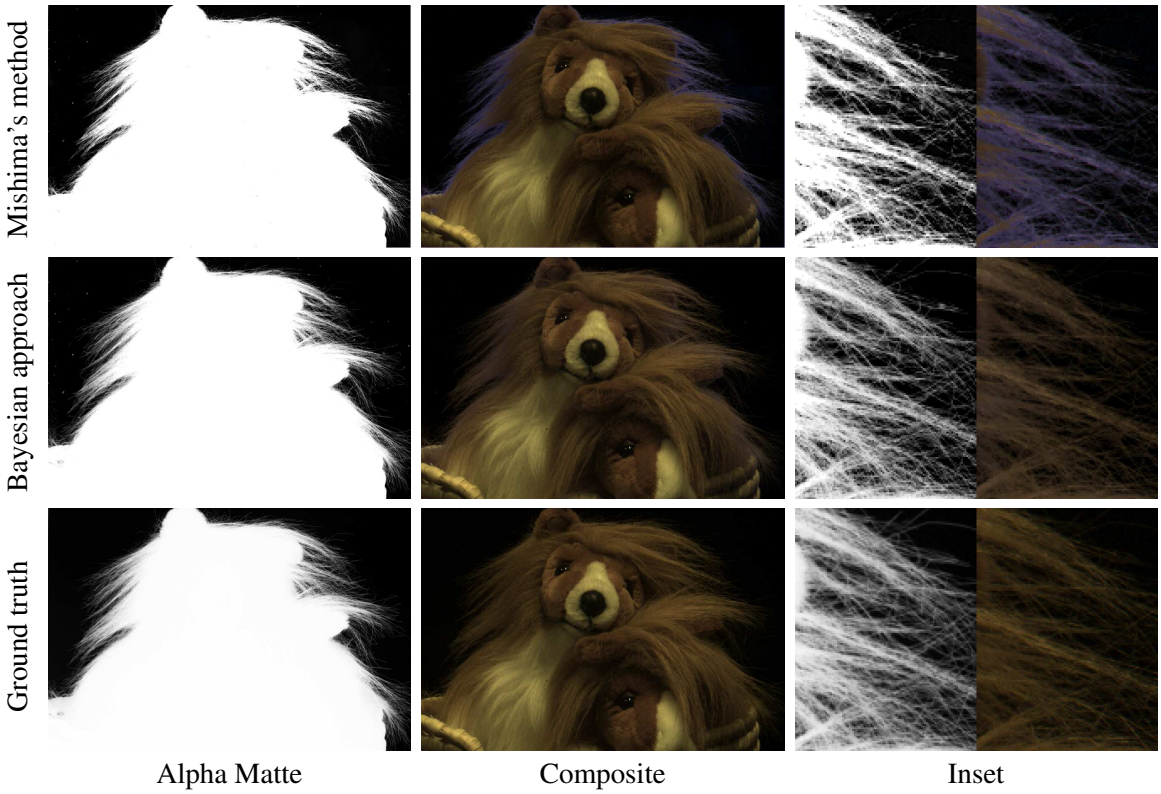


Figure 2.3: Blue screen matting of lion (taken from leftmost column of Figure 2.2). Mishima's results in the top row suffer from "blue spill." The middle and bottom rows show the Bayesian result and ground truth, respectively.

2.4.1 Blue screen matting

We filmed our target object, a stuffed lion, in front of a computer monitor displaying a constant blue field. In order to obtain a ground-truth solution, we also took radiance-corrected, high dynamic range [30] pictures of the object in front of five additional constant-color backgrounds. The ground-truth solution was derived from these latter five pictures by solving the overdetermined linear system of compositing equations (1.1) using singular value decomposition.

We compare our Bayesian approach to one of the best blue screen matting algorithms, Mishima's algorithm. Both Mishima's algorithm and our Bayesian approach require an estimate of the background color distribution as input. For blue screen matting, a preliminary trimap segmentation can be performed more-or-less automatically using the Vlahos equation (1.2). Setting a_1 to be a large

number generally gives regions of pure background, while setting a_1 to a small number gives regions of pure foreground. The leftmost image in the middle row of Figure 2.2 shows the preliminary segmentation produced in this way, which was used as input for both Mishima's algorithm and our Bayesian approach to build the foreground and background statistics.

In Figure 2.3, we compare our results with Mishima's algorithm and with the ground-truth solution. Mishima's algorithm exhibits obvious "blue spill" artifacts around the boundary, whereas our Bayesian approach gives results that appear to be much closer to the ground truth.

2.4.2 Natural image matting

For natural image matting, we have implemented an integrated system for trimap editing and Bayesian matting. For trimap editing, the user draws unknown regions with painting tools or intelligent scissors [62] and selects which partition or partitions are to be flood-filled as foreground, with the remainder flood-filled as background. After performing Bayesian matting, if there are errors in the resulting matte, the user can directly edit the matte with alphas of 0 and 1. The edited matte is then converted to a trimap for Bayesian matting until a satisfactory result is obtained.

Figure 2.4 provides an artificial example of "natural image matting," one for which we have a ground-truth solution. When taking the ground-truth solution for the previous blue screen matting example, in addition to the constant-color backgrounds, we also filmed the stuffed lion in front of a known checkerboard background. We then attempted to use four different approaches for pulling the matte: a simple difference matting approach (which takes the difference of the image from the known background, thresholds it, and then blurs the result to soften it); Knockout; the Ruzon and Tomasi algorithm, and our Bayesian approach. The ground-truth result is repeated here for easier visual comparison. Note the checkerboard artifacts that are visible in Knockout's solution. The Bayesian approach gives mattes that are somewhat softer, and closer to the ground truth, than those of Ruzon and Tomasi.

Figure 2.5 repeats this comparison for two (real) natural images (for which no difference matting or ground-truth solution is possible). Note the missing strands of hair in the close-up for Knockout's results. The Ruzon and Tomasi result has a discontinuous hair strand on the left side of the image, as well as a color discontinuity near the center of the inset. In the lighthouse example, both

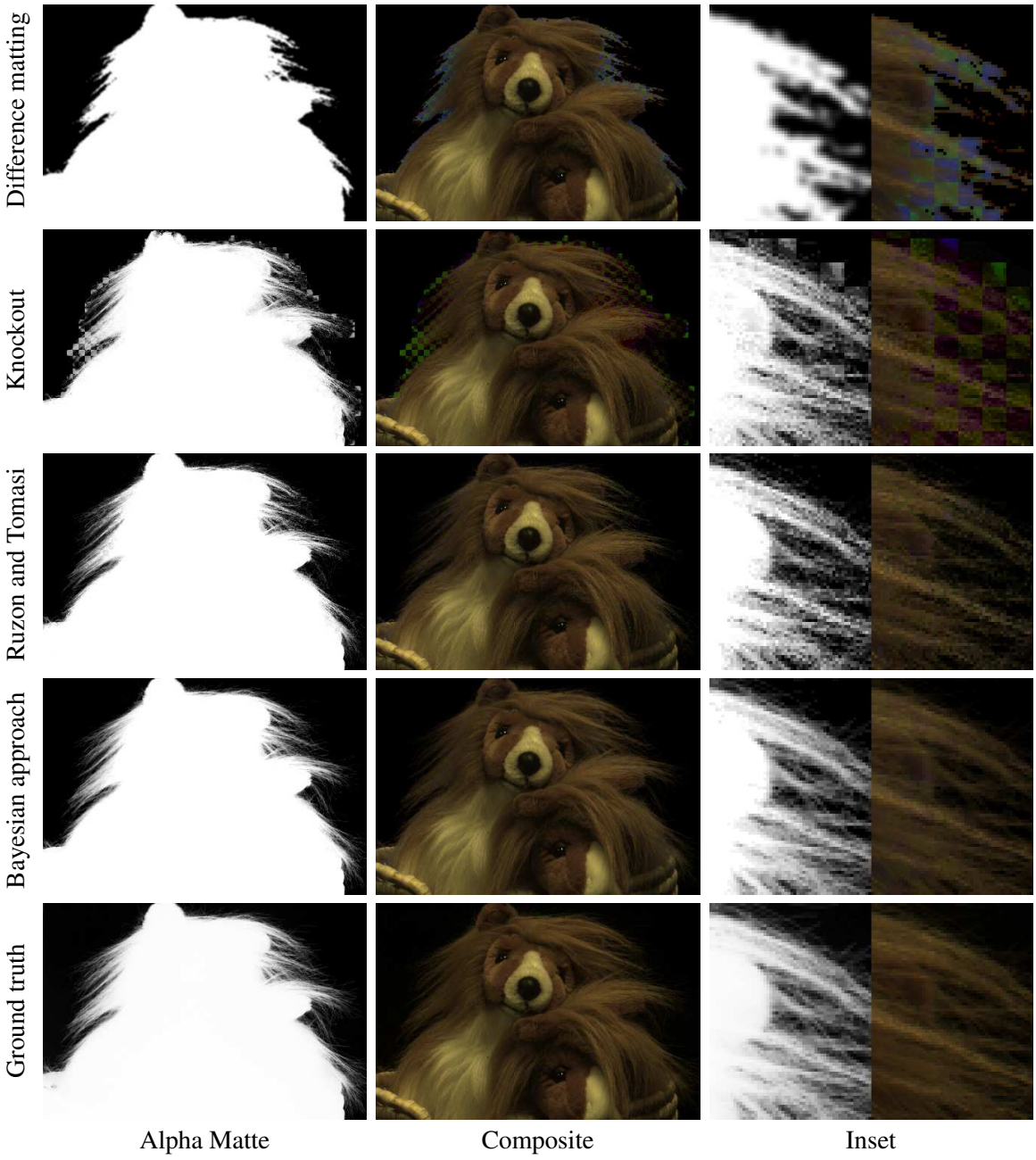


Figure 2.4: "Synthetic" natural image matting. The top row shows the results of difference image matting and blurring on the synthetic composite image of the lion against a checkerboard (column second from left in Figure 2.2). Clearly, difference matting does not cope well with fine strands. The second row shows the result of applying Knockout; in this case, the interpolation algorithm poorly estimates background colors that should be drawn from a bimodal distribution. The Ruzon-Tomasi result in the next row is clearly better, but exhibits a significant graininess not present in the Bayesian matting result on the next row or the ground-truth result on the bottom row.

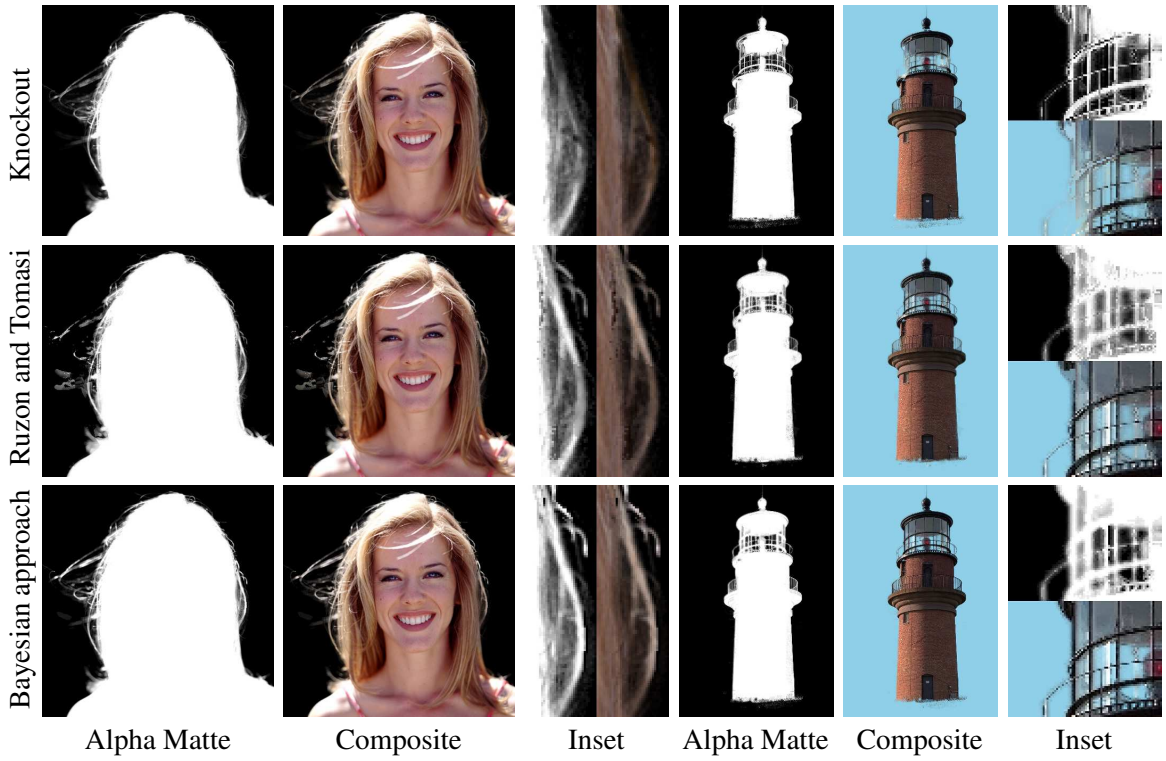


Figure 2.5: Natural image matting. These two sets of photographs correspond to the rightmost two columns of Figure 2.2, and the insets show both a close-up of the alpha matte and the composite image. For the woman's hair, Knockout loses strands in the inset, whereas Ruzon-Tomasi exhibits broken strands on the left and a diagonal color discontinuity on the right, which is enlarged in the inset. Both Knockout and Ruzon-Tomasi suffer from background spill as seen in the lighthouse inset, with Knockout practically losing the railing.

Knockout and Ruzon-Tomasi suffer from background spill. For example, Ruzon-Tomasi allows the background to blend through the roof at the top center of the composite inset, while Knockout loses the railing around the lighthouse almost completely. The Bayesian results exhibit none of these artifacts.

2.5 Regularization

In Section 2.3, the Bayesian matting algorithm estimates α, F, and B for each pixel independently without the likelihood $L(\alpha)$. Hence, when the foreground and background color statistics are similar, this algorithm could lead to a poor matte with jagged edges. This is because, in these regions, foreground and background estimates are very close in color space, and equation (2.9) becomes ill-

conditioned. To alleviate this problem, one common solution is to impose smoothness constraints to make the problem well-posed.

We use a Markov Random Field (MRF) approach [50] to solve the matting problem as a whole and incorporate constraints into the solution. To simplify analysis, we assume that α is the only hidden variable. Hence, we can write the joint probability of the MRF as

$$P(\alpha, C) = \frac{1}{Z} \prod_{p \in \Omega} \Phi(\alpha(p); p, C) \prod_{(p,q) \cap \Omega \neq \phi} \Psi(\alpha(p), \alpha(q)) \,, \tag{2.10}$$

where (p, q) represents a pair of neighboring pixels, Ω is the unknown region, $(p, q) \cap \Omega \neq \phi$ means at least one of p and q is in the unknown region, and Z is the normalization constant. Note that, here, α and C represent the matte and the observation image as a whole instead of the opacity value and the color at each pixel in equation (2.2). The evidence function $\Phi(\alpha'; p, C)$ describes the likelihood of a particular alpha value α' being assigned to $\alpha(p)$ given the observation C. The compatibility function Ψ encodes the prior for α. Such a prior could be hypothesized or learned. The optimal alpha matte can then be found by maximizing the above probability for the given observation C. Taking the $\log$ of equation (2.10), we can see that finding the MAP estimate is equivalent to minimizing a function of the following form,

$$\begin{aligned} L(\alpha, C) &= \sum_{p \in \Omega} -\log \Phi(\alpha(p); p, C) + \sum_{(p,q) \cap \Omega \neq \phi} -\log \Psi(\alpha(p), \alpha(q)) \\ &= \sum_{p \in \Omega} E_\Phi(\alpha(p); p, C) + \sum_{(p,q) \cap \Omega \neq \phi} E_\Psi(\alpha(p), \alpha(q)) \,. \end{aligned} \tag{2.11}$$

To complete our MRF formulation for the matting problem, we have to define these two energy functions $E_\Phi(\cdot)$ and $E_\Psi(\cdot)$.

For a pixel p, the evidence energy $E_\Phi(\alpha'; p, C)$ evaluates how likely $\alpha(p)$ is to take the value α' given the observation C. To calculate $E_\Phi(\alpha'; p, C)$, for the hypothesis $\alpha(p) = \alpha'$, we follow the same color sampling procedure described in Section 2.3 and calculate foreground and background color statistics $\{\overline{F}, \Sigma_F, \overline{B}, \Sigma_B\}$ for the pixel p using equation (2.4) and (2.6). We then solve the linear equation (2.8) to obtain the optimal foreground and background colors, $\hat{F}$ and $\hat{B}$, for the given α'. Finally, we substitute $\hat{F}$, $\hat{B}$ and α' into equation (2.2) to get the associated energy (a sum of negative likelihoods), and assign it to $E_\Phi(\alpha'; p, C)$.

For the compatibility function $E_\Psi(\cdot)$, a naïve choice is the quadratic function, $(\alpha(p)-\alpha(q))^2$, that ensures smoothness of the resulting matte. Note that, with this quadratic energy, computing the second term of equation (2.11) is equivalent to computing the sum of gradients of the alpha matte within the unknown region, $\sum_{p\in\Omega}|\nabla\alpha|^2$. This energy, however, makes α smooth everywhere, leading to poor results at boundaries. To preserve discontinuities, we modulate the smoothness term by a function of the image gradient, $\varphi(|\nabla C|)$. We set $\varphi(a)\propto\frac{1}{\max(a,\epsilon)}$, where ϵ is a small positive number to prevent $\varphi(a)$ from becoming infinite, to enforce smoothness on the matte in low-contrast regions while preserving discontinuities in high-contrast regions. Finally, a global constant λ is used to control the compromise between the evidence and the compatibility energies. Plugging in the quadratic compatibility energy, equation (2.11) becomes

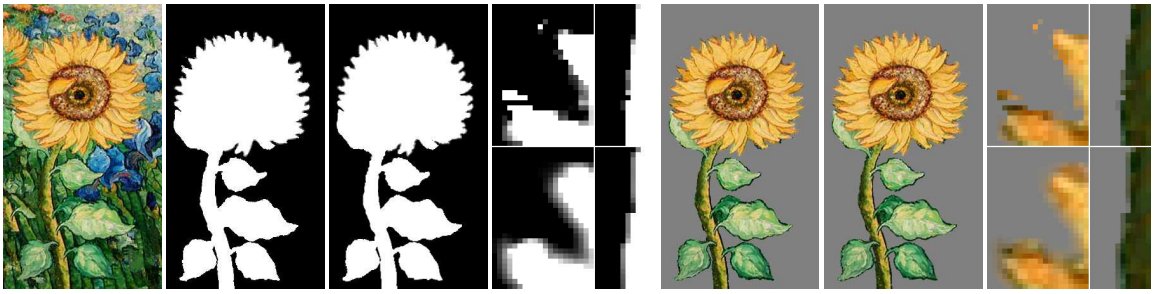
$$L(\alpha,C)=\sum_{p\in\Omega}E_\Phi(\alpha(p);p,C)+\lambda\sum_{p\in\Omega}\varphi(|\nabla C|)|\nabla\alpha|^2\,. \tag{2.12}$$

We call this method *regularization* because solving the MAP setting with a prior energy of a quadratic form is equivalent to the regularization of an ill-posed problem [50].

Now that the MRF has been formulated, to find the optimal α, we could use Belief Propagation [34] or Graph Cut algorithms [17]. However, to apply these algorithms, for a 256-level matte, we have to solve equation (2.8) 256 times for every pixel p. This is computationally intensive. Because the evidence energy function $E_\Phi(\cdot)$ is usually very smooth, to further simplify the computation, we instead approximate $E_\Phi(\alpha(p);p,C)$ with a quadratic function, $\left(\frac{\alpha(p)-\mu_p}{\sigma_p^2}\right)^2$, with center μ_p and width σ_p. Therefore, equation (2.12) becomes

$$L(\alpha,C)=\sum_{p\in\Omega}\left(\frac{\alpha(p)-\mu_p}{\sigma_p^2}\right)^2+\lambda\sum_{p\in\Omega}\varphi(|\nabla C|)|\nabla\alpha|^2\,. \tag{2.13}$$

We estimate μ_p and σ_p by sampling $E_\Phi(\cdot)$ uniformly at 16 different alpha's, and fitting a quadratic function to the samples. Intuitively, μ_p represents the optimal alpha solved by the Bayesian matting algorithm, and σ_p is the confidence level to the solution μ_p. When σ_p is large, the evidence energy function is flat and the optimal $\alpha(p)$ could be pulled further from μ_p by p's neighbors. When σ_p is small, the energy function is sharp and the optimal $\alpha(p)$ can not be pulled too far away from μ_p. Now, because the likelihood equation (2.13) becomes quadratic, we can find the optimum by solving a linear system, $A\alpha=b$. Matrix A is very sparse with only only 13 non-zero elements per row on



(a) (b) (c) (d) (e) (f) (g)

Figure 2.6: Results of regularization. Given the input image (a), we applied the Bayesian matting algorithm to pull out an alpha matte (b) for a flower and composite the flower over a gray color (e). Because the background is highly textured, there are some artifacts in the matte and composite such as mis-classified pixels and over-hardened edges. Such artifacts are more noticeable in the top portions of insets (d) and (g) showing two close-ups of the alpha matte and composite, respectively, without regularization. With regularization, we obtained a better matte (c) and composite (f). In the bottom portions of the insets (d) and (g), most of the artifacts are eliminated with regularization.

average. Iterative methods such as the conjugate gradient method are very efficient for solving this kind of system. Overall, this approach is much faster than solving a Belief Propagation or Graph Cut problem. After having estimated α by optimizing equation (2.13), we estimate the foreground and background colors by solving equation (2.8) using the estimated α.

Figure 2.6 shows the results of regularization and compares them to those of Bayesian matting. Generally, regularization gives smoother mattes than Bayesian matting. However, regularization can sometimes over-smooth the mattes sometimes. Hence, it is preferable for simpler, coherent structures but could be harmful for the mattes containing complex structures such as hair.

2.6 Extensions and applications

In this section, we discuss the extensions and applications reported by others after our Bayesian matting algorithm was published.

We have provided Industrial Light & Magic (ILM) with a copy of our Bayesian matting software to evaluate it in an production setting. While the Bayesian matting algorithm is designed for natural image matting, ILM has primarily been using it to estimate the foreground colors and refine the mattes extracted from blue screen images. There are two reasons for this. First, blue screen matting is still more widely used than natural image matting. Bayesian matting tends to outperform

blue screen matting algorithms, because it does a better job of handling spatial color variations. Second, the Bayesian matting algorithm does not work particularly well for the regions where the foregrounds or backgrounds are highly textured.

Layer-based representations have many potential applications in computer graphics, and the recent developments of natural image matting algorithms have inspired many of them. For example, Reche *et al.* [73] use a natural image matting algorithm to extract alpha mattes from a small number of calibrated photographs of a tree to allow interactive rendering of realistic 3D trees. Yamazaki *et al.* modify natural image matting algorithms to build billboards for image-based rendering [116]. In addition, in Chapter 6, we present an application of Bayesian matting to animating pictures.

Chen *et al.* specialize the Bayesian algorithm for solving the grayscale matting problem [19]. They also add the likelihood for α,

$$L(\alpha) = -\frac{(\alpha - \overline{\alpha})^2}{\sigma_\alpha^2 |\nabla C|}\,, \tag{2.14}$$

where $\overline{\alpha}$ is the average alpha of neighboring pixels and σ_α is a user-specified constant modeling the noise of α. The magnitude of color gradient $|\nabla C|$ is used here to avoid over-smoothing alpha. Finally, they combine grayscale image matting with color transferring techniques to perform region-based colorization. Shum *et al.* exploit similar smoothness constraint for $L(\alpha)$ and call it *coherence matting* [84]. They use this technique to construct *coherent layers* for an image-based rendering system called *pop-up light field*.

The quality of the input trimap is critical to the success of statistical image matting algorithms. However, specifying trimaps could be a painstaking process. Blake *et al.* propose an easier way to specify trimaps based on an adaptive "Gaussian Mixture Markov Random Field" model [13]. More recently, several methods for efficient foreground extraction have been proposed [52, 75].

One weakness of most statistical natural image matting algorithms is that they are slow. For thousands of unknown pixels, it usually takes a couple of minutes to estimate the matte. Tan presents a closed-form solution for a fast probabilistic matting algorithm [99]. This algorithm is reported to obtain mattes of comparable quality to those of existing algorithms in a couple of seconds.

The most important development for natural image matting after Bayesian matting is probably the Poisson matting algorithm proposed by Sun *et al.* [93]. As observed by Mitsunaga *et al.*, the

gradient of the matte is proportional to the gradient of the input image (equation (1.3)). Hence, instead of using color statistics, Poisson matting directly reconstructs a matte from a continuous matte gradient field by Poisson integration using boundary conditions specified by the input trimap. That is, it solves for the matte α^* that minimizes the discrepancy between its own gradient and the image gradient,

$$\alpha^* = \arg\min_{\alpha} \int_{p\in\Omega} |\nabla\alpha(p) - \frac{1}{F(p) - B(p)}\nabla C(p)|^2 dp, \tag{2.15}$$

in which $F(p)$ and $B(p)$ are estimated from the neighboring known foreground and background colors assuming foreground and background colors are fairly smooth. Furthermore, a rich set of filtering tools are provided to allow the user to improve the resulting matte by interactively manipulating the matte gradient field. Easier control is the biggest strength of Poisson matting. Because of the difficulty in obtaining perfect mattes, it is common practice in industry to allow the user to adjust parameters until the results are satisfactory. However, for most natural image matting approaches, the only way to adjust a matte is to edit the trimap. This appraoch is less intuitive and the results are usually less predictable. By formulating the matting problem in terms of Poisson equations from image gradient fields, Poisson matting allows the user to operate directly on the gradient fields to adjust a matte in easier, interactive, and more intuitive ways.

Summary

In this chapter, we have developed a Bayesian approach to solving several image matting problems: constant-color matting, difference matting, and natural image matting. Though sharing a similar probabilistic view with Ruzon and Tomasi's algorithm, our approach differs from theirs in a number of key aspects. It uses (1) MAP estimation in a Bayesian framework to optimize α, F and B simultaneously, (2) oriented Gaussian covariances to better model the color distributions, (3) a sliding window to construct neighborhood color distributions that include previously computed values, and (4) a scanning order that marches inward from the known foreground and background regions. We have also proposed a regularization approach to overcome the "jagged-matte" problem when Bayesian matting becomes ill-posed. To sum up, our approach has an intuitive probabilistic motivation, is relatively easy to implement, and compares favorably with previous approaches for matte extraction.

Chapter 3

VIDEO MATTING

3.1 Introduction

The Bayesian matting algorithm presented in the previous chapter is capable of pulling a matte from a single image with an unknown background. For special effects, however, it is more common to extract a traveling matte from a video filmed against a natural background, instead of a single matte from an image. This scenario often occurs when new elements are inserted into complex live-action footage. For example, in "Jurassic Park," computer-generated dinosaurs were inserted into real jungle environments [74]. The focus of this chapter is to move beyond the single-frame realm and extend natural image matting algorithms to handle a sequence of images. A naïve approach would be to create a trimap by hand for each frame. In addition to being inefficient, this approach can result in unusable mattes. Even if each individual matte looks reasonable, when displaying the whole sequence at the proper speed, matte edges may crawl, elements may flicker, and motions may be jerky [18] because manual frame-by-frame matting does not enforce temporal consistency.

In this chapter, we present a framework to extend our Bayesian matting algorithm to video matting in a more automatic and temporally consistent way, enabling the extraction of high-quality traveling mattes around complex foreground silhouettes filmed against natural backgrounds.[1] While video requires matting many frames and introduces a need for temporal coherence, it also has an inherent data redundancy that we can utilize to facilitate the matting process. Our approach employs computer vision algorithms to incorporate as much information from other frames as possible. In particular, we use bi-directional optical flow to interpolate user-supplied keyframe trimaps, thus reducing user involvement. Competing flow estimates are combined based on information about where flow is likely to be accurate. Further, when the background can be estimated with mosaicking

[1]The work described in this chapter was first presented in the ACM SIGGRAPH 2002 paper co-authored with Aseem Agarwala, Brian Curless, David H. Salesin and Richard Szeliski [20].

techniques, we improve both the trimap and the matte by borrowing the background colors from the nearest frame in which those colors are known to be uncontaminated by the foreground.

The remainder of the chapter is organized as follows. First, we discuss the related work on layer extraction and the existing techniques we build upon (Section 3.2). We then present our video matting algorithm (Section 3.3). Next, we describe a novel technique for estimating mattes for smoke blowing across a known background (Section 3.4). Then, we show results of applying our matting algorithm to video footage both with and without background estimation (Section 3.5). Finally, we describe some extensions developed by others and summarize this chapter (Section 3.6).

3.2 Related work

Video matting is related to the layer extraction problem in the computer vision [103, 106] and multimedia [37, 46] communities. Layer extraction algorithms segment images into several coherent layers based on information such as colors and motions. Wang and Adelson pioneered representing moving images with layers [106]. They used motion analysis to segment a sequence of images into a set of layers. Later, Weiss and Adelson proposed an expectation maximization (EM) algorithm to perform such segmentation automatically [111]. Most layer extraction algorithms, however, can only generate binary masks for classifying pixels as foreground or background, instead of mattes depicting opacity values.

More recently, Jojic and Frey use a generative-model-based approach to solve the layer extraction problem [43]. They use a *flexible sprite model* to represent each layer as a combination of a sprite appearance map and a sprite mask, both modeled with pixel-by-pixel Gaussians. Using a *generalized EM* and a *variational inference* algorithm, their method efficiently learns the best model parameters from the sequence and automatically decomposes the sequence into layers. For videos that can be explained by this generative model, their method outputs a reasonably good soft segmentation, although it is typically still not good enough for high quality compositing. Furthermore, this model is considerably restrictive and works best for sprites with translational motions.

Hillman *et al.* have extended their natural image matting approach to handle videos by inferring a trimap for the current frame from the previous frame [40]. They match a low-resolution version of the current frame to the previous frame and classify each pixel as foreground or background if

the corresponding pixels are mostly of the same foreground/background class. Undecided pixels are classified by searching for corresponding edges in the previous frame and using their foreground and background color statistics. The resulting trimap tends to be noisy, and a morphology operation is used in a post-processing step to clean up the trimap. Unfortunately, it is hard to gauge the quality of their approach since only three static frames from a simple sequence are presented.

Our video matting algorithm flows keyframe trimaps using optical flow. Over the years, researchers have developed a number of techniques for estimating optical flow, many of which are compared and summarized by Barron *et al.* [5]. Two assumptions common to many of these techniques are that the color of a source and destination pixel should be similar, and that the flow field should exhibit some amount of spatial coherence. Thus, the problem becomes one of optimizing a data term (color similarity) plus a regularization term (flow smoothness). One of the better-performing optical flow techniques is due to Black and Anandan [12]. In addition to estimating regularized flow, their technique employs robust statistics to avoid large errors caused by outliers and to allow for discontinuities in the flow field. Their method handles large motions using a multi-scale approach to flow estimation. In our video matting process, we have incorporated Black's optical flow algorithm, for which an implementation is available on the author's webpage.

Our method uses background estimation to reconstruct clean plates to improve matting whenever possible. Background estimation is straightforward when the camera is locked down or when it is attached to a motion control rig that permits reproducing camera motion both with and without the actors. In some cases, however, a partial clean plate can be assembled from nearby video frames if the background motion can be reliably estimated from frame to frame. Such *image mosaicking* techniques have been used in MPEG-4 video coding to compactly transmit a static portion of the scene viewed from a panning camera [46]. Once a conservative foreground mask (sometimes called a *garbage matte*) has been specified, through either manual [46] or automated [106] means, the remaining background fragments can be assembled using perspective image mappings to form a composite mosaic [97], which can then be reprojected into each original frame to form a dynamic clean plate. While these previous approaches have been successfully applied in image coding and surveillance, they have not been used to obtain pixel-accurate alpha mattes.

3.3 Video matting

Our approach to video matting combines these earlier techniques with a modest amount of user interaction, which consists of garbage matte specification if background estimation is needed, and hand-drawn keyframe trimaps. Figure 3.1 illustrates the flow of user interaction, data, and computation, which we summarize here.

Where possible, a background plate B can appreciably improve the quality of the mattes obtained. To aid this process, the user draws a set of garbage mattes G that conservatively eliminate the foreground and enable background estimation. Next, the user draws trimaps at selected keyframes using the system described in Section 2.4.2. These keyframe trimaps K can be fairly crude, as shown in Figure 3.3, and thus can be drawn quickly. The choice of keyframes is something the user adapts to with experience; for example, keyframes are helpful in areas where the topology of the foreground layer changes.

Once the initial set of trimaps is specified, the labelings are passed through the volume using optical flow, resulting in trimaps T at every frame. The flow of information considers where optical flow is likely to succeed and where it might fail. In order to narrow the bands of uncertainty in the trimaps as they flow through the volume, they are converted to alpha mattes α by the Bayesian matting process at each step of the process and then converted back into trimaps. If background information is available, the flow process can be improved using better alpha mattes (and thus trimaps) and by using a form of difference matting that improves the trimaps in regions where flow fails.

Finally, if the matte is not satisfactory, the user can select a frame and edit the trimap with the trimap editing tool. In practice, we provide an image of the alpha matte to edit, as this tends to expose the structure of the image more than the trimap, but we permit the user to paint alphas of only 0 or 1. The edited alpha matte is then converted to a trimap and becomes a new keyframe. We then re-run the trimap interpolation method to make maximum use of the new information.

The output of the system is the estimated foreground F, estimated background $\hat{B}$, and alpha α for every frame. Note that even when the background is available, the estimated background color may be slightly different in order to satisfy the maximum-likelihood criterion in Bayesian matting.

In the remainder of this section, we discuss in greater detail the fundamental computational blocks: background estimation, trimap interpolation, and alpha estimation.

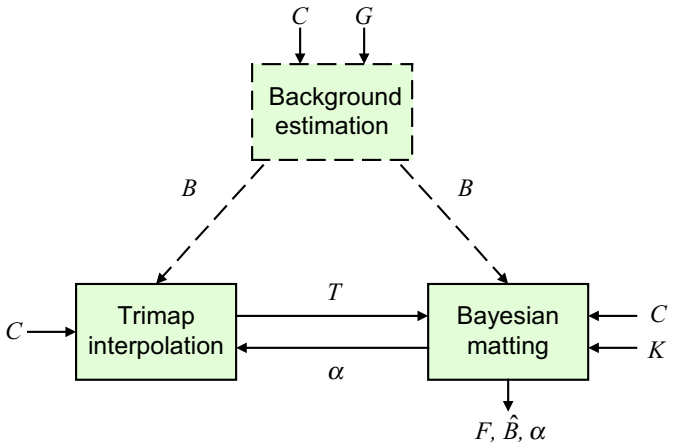


Figure 3.1: Video matting flow chart. The primary computational blocks of our process are the background estimation, trimap interpolation, and Bayesian matting components. Each block receives the original image sequence C as input. For background estimation, the user also provides a garbage matte G to remove the foreground; then, a background B is estimated and used in trimap flow and matting. To obtain a video matte, the user provides keyframe trimaps K, which are then converted to alpha mattes α and passed to the trimap interpolator. The interpolated trimaps T are refined through alpha matting at each step. Once the trimaps are complete, the Bayesian matting algorithm generates an estimated foreground F, background $\hat{B}$, and alpha matte α for all frames.

3.3.1 Background estimation

Before beginning background estimation, a garbage matte is necessary to mask out, in a conservative manner, all pixels that could possibly contain foreground contributions. While automatic methods have been published for estimating a garbage matte, and indeed we have tried using the keyframe trimaps to assist in this process, they do not always work. Making errors in the background plate can seriously degrade the quality of the alpha matte. Thus, we follow the approach commonly used in the film industry: we require the user to provide the garbage mattes. For our system, the user must draw a rectangle at a handful of keyframes. These rectangles are automatically interpolated over all the frames and can then be adjusted immediately by the user to obtain conservative garbage mattes. For a sequence of 100 frames, it took about 5 minutes to specify garbage mattes.

Given the garbage mattes, we use the mosaicking method of Szeliski and Shum [97] to compute frame-to-frame registration. This method works under the assumption that the background undergoes only planar-perspective transformation. This assumption is equivalent to requiring that

the background be planar or, less restrictively, that the camera's optical center translate a negligible amount relative to the distance to the background between frames being registered. Instead of constructing a single clean plate, we fill in the missing parts of each frame with the color from the temporally nearest frame that contains a background color for that pixel. In practice, we find that copying pixels from nearby keyframes has the advantage of reducing errors due to small amounts of parallax or gradual temporal variations that may arise with illumination changes, motion blur, and defocus.

3.3.2 Trimap interpolation

To pull a complete video matte, we require a trimap for each frame. Constructing the trimaps manually is tedious and time-consuming; on the other hand, a fully automatic approach is unlikely to succeed in giving high-fidelity mattes. Thus, we have developed a semi-automatic system that calculates trimaps for every frame using hand-drawn trimaps for selected keyframes.

To take advantage of spatiotemporal coherence within the video volume, we employ optical flow. Optical flow provides an estimate of inter-frame motion at each pixel in a video sequence. Given two neighboring frames, C_t and C_{t+1}, we can think of each pixel p in C_{t+1} as coming from a shifted location $p+u(p)$ in C_t,

$$C_{t+1}(p) = C_t(p + u(p)) \,, \tag{3.1}$$

where $u(p)$ describes the velocity of the pixel and is itself a spatially varying function over the image. We refer to u as the *flow field* between two frames. The flow field acts as a guide for passing trimap labelings through the volume between keyframes. In principle, we could simply start from the first keyframe and flow its trimap forward in time. However, consider *disocclusions* in an image sequence, i.e., when a feature not present in one frame appears in the next frame. Optical flow breaks down here because it cannot find source pixels that explain the new feature. On the other hand, if we view this same event when running through the frames in the reverse direction, the event becomes an *occlusion*, and optical flow does not have a problem.[2] This tells us that disocclusions will cause errors in flow that can frequently be resolved by viewing flow in the opposite direction.

[2]This insight has also been noted and used in the image coding literature [95]

The solution is clear: run flow in both directions — forward from one keyframe and backward from the next — and combine the observations according to a measure of per-pixel accuracy for each prospective flow.

In the following sections we describe how we measure the accuracy of optical flow and then present our algorithm for running and combining bi-directional flow.

Accuracy of optical flow

In this section, we devise a method for determining the accuracy of optical flow. This accuracy test is built on precomputed error maps, i.e., the per-pixel prediction errors from one frame to another. To create a forward error map ε_t^f for a flow from frame $t-1$ to t, we first compute the image predicted by flow, $C_t(p - u(p))$, bilinearly resampled onto the pixel grid. At each pixel of this warped image, we can compute the difference between the predicted color and the observed color as the L_2 distance between the pixels in *rgb* color space, $\varepsilon_t^f = \|C_t(p) - C_t(p - u(p))\|$. A similarly computed backward error map ε_t^b measures the accuracy of the flow from frame $t+1$ to t.

After computing error maps for each frame in both directions of optical flow, we combine these to create two sets of *accumulated error maps*. Since we are propagating trimap values from the nearest keyframes, it is not enough to know the error for single frames of optical flow. Instead, we need the accumulated error in optical flow in both directions from the nearest keyframes. If frame t is a keyframe, the forward accumulated error map ξ_{t+1}^f is simply set to ε_{t+1}^f. To compute ξ_{t+2}^f, we first warp the values of ξ_{t+1}^f forward in time using the calculated flow. Then, we set ξ_{t+2}^f equal to this warped accumulated error map plus ε_{t+2}^f. This set of calculations is performed from each keyframe forward until the following keyframe is reached. The backward accumulated error maps ξ_t^b are computed similarly. Thus, at a frame t', the accumulated error maps $\xi_{t'}^f$ and $\xi_{t'}^b$ give us a measure at each pixel of the accuracy of flow estimation from the previous and following keyframes.

Combining forward and backward flow

Once optical flow and the error maps have been calculated, we flow the trimaps forward in time from the hand-drawn keyframes. That is, a trimap is formed at frame $t+1$ by warping the keyframe trimap t using the calculated forward flow vectors. We add an additional *validity bit* for each pixel

p of the flowed trimaps that indicates whether the calculated trimap value is trusted; this validity bit $\upsilon_t(p)$ is set to 0 if $\varepsilon_t^f(p)$ is greater than a certain threshold (experimentally set to 30). When calculating trimap $t+2$ from trimap $t+1$, the validity bits are also warped forward and combined conjunctively. To perform forward warping on the validity bits, they are first converted into real numbers. If the real-value forward warped version of the validity map υ_{t+1} to frame $t+2$ is υ'_{t+1}, the validity bit at p in trimap $t+2$ is conservatively set to $(\upsilon'_{t+1}(p) = 1.0) \wedge (\varepsilon_{t+2}^f(p) < 30)$.

After calculating each warped trimap, we improve it by performing Bayesian alpha estimation. Untrusted pixels whose validity bits are 0 are labelled as "unknown": we are not confident of the labelling of these pixels and do not want them corrupting the color distributions. The result of the alpha estimation is thresholded (within 10 gray levels of "definitely foreground" or "definitely background") and used to improve the trimap. If alpha estimation confirms the labelling of an untrusted pixel, the validity bit is set to 1. If a pixel labelled as "unknown" is identified as "foreground" or "background", the label is changed accordingly.

In the second pass, we start from each keyframe and flow backwards in time. Now, however, we must combine the trimap prediction in the forward direction with the trimap prediction in the backward direction. This is made simple by the accumulated error maps that we calculated earlier. First, the backward trimap prediction for frame t is calculated using a technique symmetric to the forward trimap calculation. Then, at each pixel, we use the lesser of ξ_t^f and ξ_t^b to select a trimap value. In practice, we add an additional penalty term to this comparison; if the forward trimap label is "unknown" we add a penalty of 50 to ξ_t^f, and likewise for the backward trimap. We add this additional penalty for "unknown" pixels because they are near depth discontinuities and should be trusted less; in particular, optical flow tends to perform poorly at the boundaries between foreground and background layers that have distinct motions, yet are blended together. Although Black's algorithm does allow for discontinuities in the flow field, we have found that this allowance does not always work, particularly for complex silhouettes.

If a background plate is available, it is useful to incorporate this information when flow is invalid from both directions. In the event that the forward and backward validity bits are both 0, we can resort to a simple form of difference matting. In particular, we compute the *rgb* L_2 distance between the observed color and the background color and apply user defined thresholds to map the distance to a trimap value.

Finally, during the second pass each computed trimap is passed through alpha estimation, and the results are used to improve the trimap as described earlier.

3.3.3 Bayesian matting with background

When the background is not available, we run the matting algorithm on the video frames and trimaps exactly as described in Chapter 2. When available, however, the estimated background offers three distinct advantages. First, the distribution of the background is tighter and more accurate. (However, due to sensor noise, the background color is still not a point in color space, but is modeled by a Gaussian distribution with a small standard deviation centered at the estimated color.) Figure 3.3 shows that the extracted alpha matte is much improved with the help of the clean plate. Second, we no longer need to compute background statistics by sampling neighborhoods and computing means and covariances, thus speeding up the matting process. Finally, the neighborhood windows no longer have to be large enough to span the unknown region and include pixels in the known background region. These last two factors yield a factor of 10 speedup in the matting process.

3.4 Smoke matting

We have also developed a simple extension for extracting mattes of flowing, participating media, such as smoke, given a known background. The example in Figure 3.4 illustrates the "smoke matting" process for a smoking actor. First, applying the video matting technique described in the previous section to just the actor, we pull his matte and remove him from the scene (Figure 3.4(b)). We then compute the difference between the matted-out image and the background image (Figure 3.4(c)), and all pixels whose differences are among the largest 5% are selected for estimating foreground statistics.

The challenge of applying Bayesian matting to extracting smoke is that there is no opaque region of smoke from which we can build a prior probability for foreground color. By treating the color of the smoke as a constant value that is simply composited with a varying alpha over the background, we need only discover that foreground color in order to estimate the matte. For each selected pixel p_i, if it has been mixed with smoke, we expect its color C_i to lie somewhere along a line ℓ_i in *rgb* color space between the pixel's known background color B_i and the foreground smoke color $\overline{F}$. By

taking all of the selected pixels, we can construct a set of these lines that, barring degenerate configurations, will roughly intersect at the foreground smoke color (Figure 3.4(d)). Thus, we compute an estimate of the foreground color $\overline{F}$ as the least-square nearest intersection of all the lines.

To find the least-square intersection, we first express each line ℓ_i in a parametric form, B_i+tU_i, where $U_i=(C_i-B_i)/\|C_i-B_i\|$ is the normalized direction vector of the line ℓ_i. Then, we seek the point $\overline{F}$ that minimizes the cost function $\Lambda(F;\ell_i)$, the sum of the squares of the distance of a point F to the lines ℓ_i; that is,

$$\overline{F} = \arg\min_F \Lambda(F;\ell_i)\,, \tag{3.2}$$

$$\Lambda(F;\ell_i) = \sum_i \|F-B_i\|^2 - [(F-B_i)\cdot U_i]^2\ . \tag{3.3}$$

At the optimum, the derivative of equation (3.3) with respect to F is equal to zero, which leads to the solution

$$\overline{F} = \left[\sum_i (U_i U_i^T - \mathcal{I})\right]^{-1} \sum_i \left[(U_i U_i^T - \mathcal{I})B_i\right]\ . \tag{3.4}$$

After finding the estimate of the foreground color $\overline{F}$, we project $\overline{F}$ onto each of the original lines ℓ_i to obtain the projections $F_i = B_i + [(\overline{F}-B_i)\cdot U_i]U_i$. It can be proved that $\overline{F}$ equals the average of these projections F_i. Thus, a covariance matrix Σ_F can be calculated by

$$\Sigma_F = \sum_i (F_i-\overline{F})(F_i-\overline{F})^T\,. \tag{3.5}$$

To be more robust, the contribution of each line ℓ_i can be weighted by $\|C_i - B_i\|$ in the above calculations for $\overline{F}$ and Σ_F, but we did not find this necessary in practice. The calculated variance Σ_F along with $\overline{F}$ form a Gaussian function used by Bayesian matting as the foreground distribution. The smoke matte is then calculated at every pixel in the image without the actor, and then the actor and smoke mattes are combined (Figure 3.4(e)).

Although the result looks realistic, a more principled approach would be to formulate the whole problem as a Bayesian framework that includes variances for C_i and B_i, and optimizes for all the variables simultaneously:

$$\begin{aligned} & \arg\max_{F,\alpha_i,C_i,B_i} P(F,\alpha_i,C_i,B_i) \\ = \ & \arg\max_{F,\alpha_i,C_i,B_i} L(\alpha_i|F,C_i,B_i)L(F|C_i,B_i)L(C_i)L(B_i), \end{aligned} \tag{3.6}$$

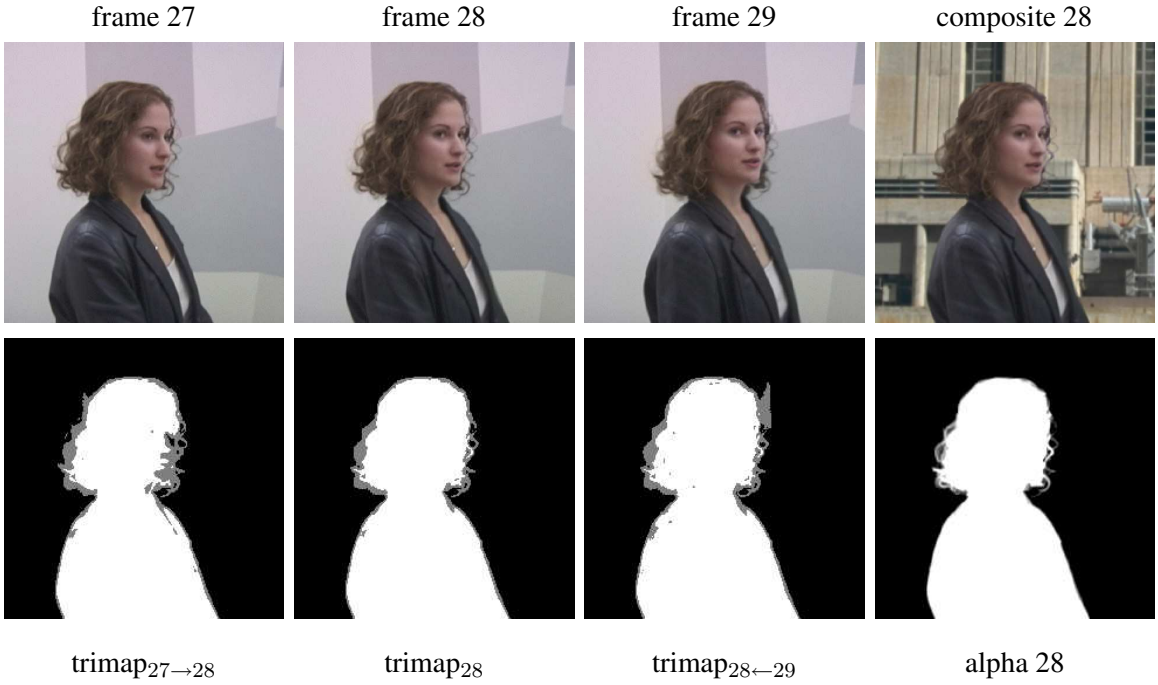


Figure 3.2: Combination of bi-directional flow. For frames 27, 28, and 29 (shown above) and the current trimaps for frames 27 and 29 (neither shown), we estimate the trimap for frame 28. Flowing the trimap in the forward direction ($27 \rightarrow 28$) yields a trimap with extra uncertainty due to disocclusion as the actor's face turns to the right. The trimap predicted by flowing backward ($28 \leftarrow 29$) has less uncertainty in those regions, but suffers from disocclusions where the background is newly exposed. By combining the information in both these trimaps, we can compute trimap 28 automatically, which is better than either one alone. The right column shows a composite into a new scene (top) using the pulled matte (bottom) based on the combined trimap.

where $L(C_i)$ and $L(B_i)$ model errors in the measurements of C_i and B_i respectively, $L(F|C_i, B_i)$ is the sum of squared distances defined in equation (3.3), and $L(\alpha_i|F, C_i, B_i)$ can be similarly defined as equation (2.3). However, this approach would involve many more unknowns and the optimization would be much more complex. Instead, we have taken a simpler approach and found that it works well in practice.

3.5 Results

We have applied our video matting algorithm to a number of video sequences. In this section, we present stills for several instructive examples.

(a) (b) (c) (d)

Figure 3.3: Background estimation in Bayesian matting. On the far left are the keyframe trimap and estimated background for a single frame. Using the original image (a) and the trimap without the background, Bayesian matting pulls a matte with errors, as shown in a composite over blue (b). Including the background but using instead the method of Ruzon and Tomasi gives an improved result (c), but flaws in the matte are still visible. Applying the Bayesian matting with background results in a higher-quality matte and composite (d).

Figure 3.2 demonstrates the importance of using bi-directional optical flow for computing interpolated trimaps. For this example, no background estimation is performed. In flowing from keyframe 27 to 28, a disocclusion of the foreground occurs, causing errors in the optical flow, which lead to "unknown" labels in these regions. Flowing from 29 to 28 solves this problem, but introduces some disocclusion in the background. Combining the two flows using the forward/backward sweep described in Section 3.3.2 gives a more accurate trimap and a better matte and composite.

Figure 3.3 illustrates the utility of background estimation. After the user provides the garbage matte, a background sequence is constructed. The keyframe trimap, combined with the input image, can be used to create a matte using the Bayesian method even without the background, but the matte contains a number of errors, as shown in the composite over blue. When including the background, but using the method of Ruzon and Tomasi, the matte also exhibits artifacts. In fact, when playing this result back as a video, temporal flashing artifacts arise, which are likely due to the static neighborhoods assembled independently for each frame. Finally, using the Bayesian method combined with the background yields a matte that, while not perfect, is free of many of the artifacts of the other two methods, and composites well over novel backgrounds.

Figure 3.4 shows a result of smoke matting and illustrates a composite over an edited background. In this case, we acquired the original background by locking down the camera and filming in the absence of the actor. Note that while the resulting matte is not perfect around the silhouette

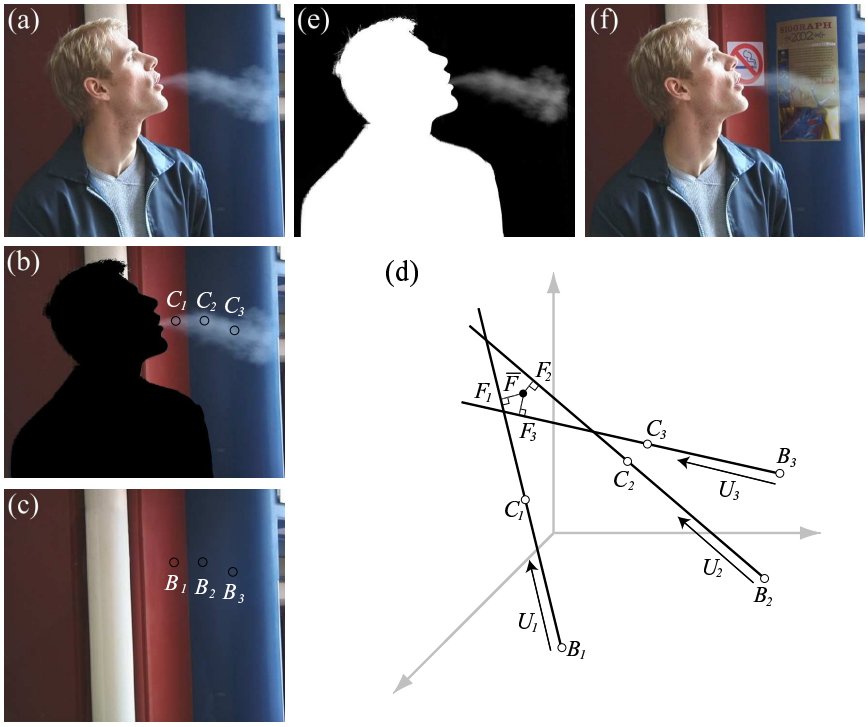


Figure 3.4: Smoke matting. The input image (a) is part of a sequence for which keyframe trimaps and trimap flow have been computed, and for which a background plate (c) is available. Using the alpha matte as a garbage matte, the foreground actor is removed (b). After applying the participating-media matting algorithm described in Section 3.4, we obtain a matte for the smoke, which is combined with the actor's matte to yield a complete matte (e). We can then composite the foreground over an edited version of the background as shown here (f).

of the actor, for the purposes of background editing in a particular region, the matte is good enough. If more edits were required, the user could focus them only in the regions that would be composited over the background edits.

Figure 3.5 and 3.6 give more examples for background editing. For the example in Figure 3.5(a), we first used the method described in Section 3.3.1 to estimate the part of the background (Figure 3.5(b)) that is occluded by the foreground actor. Using Bayesian image matting, the dinosaur on the left side of the background plate is extracted, and a reflected copy is inserted to the right (Figure 3.5(c)). The extracted foreground actor is then composited back onto the edited background to finish the composite (Figure 3.5(d)). We also show an application of matting for non-photorealistic rendering (NPR). In Figure 3.5(e), a non-photorealistic filter is applied to the whole image; note

(a) Input frame (b) Estimated background

(c) Edited background (d) Composite

(e) NPR rendering without matting (f) NPR rendering with matting

Figure 3.5: Background editing. For an input frame (a) from a sequence, the clean plate (b) is first estimated using the method described in Section 3.3.1. The dinosaur in the left side of the clean plate is extracted using Bayesian matting and a mirrored copy is added to the right to obtain the edited background (c). We then composite the foreground back into the edited background as shown here (d). When applying NPR filters on the whole image, the result (e) does not respect foreground boundaries and strokes are spread across the boundaries. With the help of matting, there are no such artifacts in the composite (f). (Universal Studios and Amblin Entertainment hold the copyright on the image from "Jurassic Park" shown here, and have granted non-exclusive permission for its use.)

(a) Input frame (b) Composite

Figure 3.6: Background editing. For the input image (a) from a given sequence, we obtain the composite (b) by adding a sign on the background and compositing the actor onto the edited background. (Universal Studios and Amblin Entertainment hold the copyright on the image from "For Love of the Game" shown here, and have granted non-exclusive permission for its use.)

that the strokes are spread across the boundary of the foreground element. Instead, we can extract the foreground element using our video matting algorithm, apply non-photorealistic filters to the foreground element and the background plate separately, and combine the filtered versions together. There are no spreading artifacts in the composite (Figure 3.5(f)) with the help of matting. Figure 3.6 shows another background editing example in which we add a sign to the background.

Both the execution time and the amount of user interaction depend heavily on the nature and resolution of the sequence. Overall, our system can be divided into an unsupervised pre-processing phase and an online phase. For the 640x480 Kim sequence (Figure 3.3), pre-processing took about 80 seconds per frame; the calculation of optical flow was by far the largest component of this time. For the online phase, drawing a keyframe trimap for this sequence took about 2 minutes per trimap. Interpolating these trimaps took 12 seconds per frame, with alpha estimation as the bottleneck. The interpolation process is unsupervised, and the user can draw more keyframes in parallel. As mentioned in Section 3.3, the user can then hand-edit any errors to form additional keyframes; this is quick and requires only a couple minutes for the entire sequence.

All of our algorithms scale linearly with the number of pixels (resolution), but other factors such as the area of unknown regions in the trimaps also affect running time. Choosing the frequency of keyframes also depends on the sequence. Complicated geometry such as wispy hair generally requires a keyframe every 10 frames. Sequences of simpler geometry require keyframes every 20–

Table 3.1: Details for the five test sequences for video matting.

sequence	frames	initial keys	edited keys	frames/sec
Amira (Fig 3.2)	91	10	2	15
Kim (Fig 3.3)	101	11	4	15
Smoke (Fig 3.4)	176	10	1	15
Jurassic (Fig 3.5)	96	11	1	30
Baseball (Fig 3.6)	161	12	3	30

30 frames. Care should be taken to add keyframes when objects enter or leave the field of view. The number of keyframes required for each sequence, along with the number of additional hand-edited keyframes after the first pass of the algorithm, are given in Table 3.1.

3.6 Extensions

In this section, we discuss two extensions that solve the video matting problem with less user input by adding more observations and priors on alpha mattes. Wexler *et al.* assume that motions of both foreground and background layers can be modeled by planar-projective transformations [112]. The basic idea is to register all frames to a reference view by stabilizing the foreground element, essentially having the background sweep past behind the fixed foreground. They also assume that the background for each frame is known by estimating the background in a similar way to the method described in Section 3.3.1. Therefore, for each pixel p, we have four unknowns for α and F, with $3n$ equations for a sequence of n images. Even though the problem appears to be over-constrained, it can still become ill-posed if the foreground pixel moves over a uniform background section. Therefore, three kinds of constraints are used to regularize the solution: bounded constraints ($0 \leq \alpha \leq 1$ and $0 \leq F \leq 1$), spatial consistency (essentially the smoothness prior in Section 2.5) and probability distribution of alpha values. For the last constraint, they observe the typical distribution of alpha values and encourage the solved alpha matte to exhibit a similar distribution. Although this algorithm requires very little user input, the assumptions about rigid and planar foreground motions are quite restrictive.

Apostoloff and Fitzgibbon solve the video matting problem with a Bayesian framework [3].

They also assume that the background is known. Hence, they essentially solve for

$$\arg\max_{F,\alpha} L(C, B \mid F, \alpha) + L(F) + L(\alpha)\,. \tag{3.7}$$

Their definitions for $L(C, B \mid F, \alpha)$ and $L(F)$ are similar to ours (Section 2.3). For $L(\alpha)$, similar to Wexler *et al.*, they encode three constraints: spatiotemporal consistency (a smoothness constraint for a space-time volume of the video), a probability distribution over alpha values (similar to Wexler *et al.*), and learned spatiotemporal consistency. For the last constraint, they use a library of ground-truth sequences to build a joint distribution $P(\nabla C, \nabla\alpha)$ that encodes the tight correlation between ∇C and $\nabla\alpha$. Inspired by the statistics of derivatives of natural images, they fit the collected histogram of $P(\nabla C, \nabla\alpha)$ with a mixture of a t-distribution and a Gaussian. In addition, they use difference matting to automatically generate coarse trimaps. Although this approach solves the video matting problem in an automatic way, because the difference matting does not always work well, there could be holes in the resulting mattes even though spatiotemporal consistency helps fill some of them. In addition, the extracted mattes tend to be too smooth, making this approach less preferable for pulling mattes of complex silhouettes.

Summary

In this chapter, we extended the Bayesian matting algorithm introduced in the previous chapter to handle videos. Our framework pulls together pieces of existing research and combines their strengths while working around their weaknesses. The result is a new kind of rotoscoping approach. With a modest amount of user interaction, our framework enables detailed matte extraction for actors with complex silhouettes filmed against natural backgrounds. In addition, we have introduced a simple procedure for extracting mattes of participating media filmed against a known background.

Chapter 4

SHADOW MATTING AND COMPOSITING

4.1 Introduction

The matting algorithms in the previous two chapters are based on the traditional compositing equation. Although traditional compositing is effective in modeling partial coverage and transparency, as explained in Section 1.3, it fails to correctly capture some lighting phenomena, such as shadows. However, shadows provide important visual cues for depth, shape, contact, movement, and lighting in our perception of the world [68], and thus are often essential in the construction of convincing composites. Shadow elements for compositing are typically created either by hand or by extracting them from blue screen plates.

The manual approach is commonly called *faux shadow* in the film industry [115]. For this technique, artists use the foreground object's own alpha matte to create its shadow. By warping or displacement-mapping the shadow, it can be made to drape over simple objects in the background plate. However, this approach has several limitations. First, an alpha matte is a flat projection of the object from the point of view of the camera that filmed it. If the view from the light is too far from the camera's point of view, the silhouette of the alpha matte may be noticeably different from the silhouette of the correct shadow, and the resulting synthetic shadow will be unconvincing. Second, the shadow color characteristics are manually adjusted by the compositor and do not necessarily match the shadow characteristics of the real scene. Most importantly, this approach becomes unwieldy for casting shadows on backgrounds with highly complex geometry. Figure 4.1(c) shows such artifacts.

The second main approach is to extract shadows from the foreground plates using luma keying or blue screen matting. These techniques provide a better approximation to the correct shadow characteristics. However, depending on the compositing model used, it may be difficult to obtain photometrically realistic results. For example, in Figure 4.1(d), note that the blue screen composite gives a noisy shadow with the wrong density, and it creates a double shadow where the ground plane

was already in shadow.

Regardless of the shadow extraction method, target background scenes with complex geometry present special compositing challenges. In many cases, a rough model must be built so the actors cast shadows onto the model as they would onto the target scene. This model may be a physical blue screen model onto which the actor casts his real shadow for extraction, or a computer-generated virtual model onto which faux shadows are cast using a renderer. In either case, it is often difficult to construct a model that matches the target object exactly, so additional manual warping and rotoscoping is required to align the transferred shadows to the receiving geometry.

In this chapter, we introduce a new process for shadow matting and compositing that captures all of these effects realistically.[1] We develop a physically-motivated shadow compositing equation, and design a matting and compositing process based on this equation. For shadow matting, we extract a shadow density map to describe the degree to which each pixel is in shadow. In contrast to previous approaches, our matting method works for natural backgrounds. For shadow compositing, we use an active illumination approach to extract an *illumination map* and a *displacement map* for the destination scene. These maps describe the shadow appearance and distortions over the novel target background. We recover the displacement map without requiring the calibration of the camera or the position of the light source, using an arbitrarily textured planar reference region. Using these acquired maps, we can realistically transfer shadows from one scene to another. Our method imposes certain restrictions on the lighting, camera placement, and at least some of the geometry in the source and target scenes, which we discuss when evaluating the merits and limitations of our method.

Figure 4.1 shows an overview of our approach and compares our results to faux shadow, to blue screen matting and compositing, and to ground truth. Our method correctly occludes both diffuse illumination and specular highlights, retains soft shadow edges, warps shadows convincingly across arbitrary background geometry, and seamlessly blends newly introduced shadows with those already present in the background plate.

In the following sections, we first discuss related work (Section 4.2). We then develop our shadow matting equation and shadow matting algorithm for scenes with identical source and desti-

[1]This chapter describes joint work with Dan B Goldman, Brian Curless, David H. Salesin and Richard Szeliski, first presented in ACM SIGGRAPH 2003 [22].

(a) Foreground scene (b) Background scene (c) Faux shadow

(d) Blue screen composite (e) Our method (f) Reference photograph

Figure 4.1: Sample result from our matting and compositing algorithm for shadows. Given a foreground element photographed against a natural background (a), we seek to matte the element and its shadow and then composite it over another background (b). The result of faux shadow (c) exhibits noticeable artifacts because the view from the light is too far from the camera's point of view. In addition, the shadow color characteristics are assigned in an *ad hoc* way and the shadow does not conform to the geometry of the background for compositing. Using a blue screen (not shown) to extract the shadow, followed by conventional matting and compositing, we obtain a result (d) with double darkening of the existing shadow and without proper warping of the cast shadow. The results of our new shadow matting and compositing method (e) compare favorably with an actual photograph (f). Note the correct dimming of the specular highlight, the convincing geometric deformation, and the seamless matte edges where the foreground and background shadows meet.

nation background geometry (Section 4.3). We describe shadow compositing onto simple (planar) destination backgrounds (Section 4.4) and shadow warping for more geometrically complex backgrounds (Section 4.5). Then, we present results using our technique (Section 4.6). Finally, we discuss the limitations, working range, and pros and cons of our approach (Section 4.7).

4.2 Related work

Much research has been done on shadows in both the vision and graphics communities. For example, Finlayson *et al.* [32] have attempted to remove shadows from a single image, and Pellacini *et al.* [66] implemented a user interface for adjusting shadow placement using direct manipulation. However, none of this research deals directly with shadow matting and compositing.

The intrinsic image approach [110] attempts to decompose a video of an object under different illumination conditions into a reflectance map and a sequence of light maps. These light maps could be used as shadow mattes, though Weiss does not attempt to transfer them to other scenes. Matsushita *et al.* [57] attempt to morph shadows for different lighting conditions using Weiss's approach. They capture the lightfield of a static scene with several light sources. With multiple cameras, they recover a view-dependent model using multiview stereo. A shadow mask is obtained by simply thresholding the illumination maps. Finally, the shadow for a novel light source is obtained by warping the shadow masks of neighboring samples and the estimated geometry.

Petrovic *et al.* [68] also estimate 3D geometry for casting shadows, but with different goals and methods. Their method is intended to create shadow mattes for cel animation, so their estimates of 3D geometry can be somewhat more approximate. They create a 3D model for the scene by inflating the character and estimating simple geometry for the background from user gestures.

Researchers have developed a number of shape-from-shadow techniques. For example, Savarese and collaborators [78] observe the self-shadowing of an object under different lighting conditions and carve a model to generate a plausible solution to match those observations. Our method builds on the shadow scanning approach of Bouguet and Perona [16]. However, we avoid explicitly reconstructing a 3D geometric model, which requires calibration of the camera and light sources. Instead, we estimate the displacement map in the image space directly.

Our compositing method is similar to Debevec's differential rendering approach for composit-

ing synthetic objects into real scenes [27]. Debevec records the differences between the rendered radiances with and without the synthetic objects in the scene. The differences are then added to the real destination background to make the composite. To avoid explicitly modeling the geometry and BRDF of the destination scene, we take a different approach.

4.3 Shadow matting

In this section, we develop our shadow compositing equation and describe our algorithm to estimate shadow mattes.

As mentioned in Section 1.3, the compositing equation (1.1) mainly describes how colors should be blended between foreground and background due to partial coverage, transparency, or motion blur. Some of the previous approaches for shadow matting assume this model and try to determine α and F for the shadow [115]. This approach effectively represents the shadow as a translucent dark layer. However, the standard compositing model does not hold for shadows because shadows are caused by the occlusion of light sources rather than color blending.

4.3.1 The shadow compositing equation

To determine an appropriate model for shadow compositing, we treat the problem within a simplified lighting framework. In particular, we assume that a single, primary point light source (sometimes called a *key light*) is responsible for dominant cast shadows within a scene. The remaining, secondary lighting is either dim or of such wide area (as in sky illumination) that its cast shadows are comparatively negligible. If we further assume no interreflections, we can model the observed color C at a pixel as

$$C = S + \beta I \,, \tag{4.1}$$

where S is the shadowed color, I is the reflected contribution of the primary light source, and β is the visibility to that light source. Let L be the color of a pixel when not in shadow. Substituting $I = L - S$ into equation (4.1) and rearranging, we obtain the *shadow compositing equation*,

$$C \;\; = \;\; \beta L + (1 - \beta) S \,. \tag{4.2}$$

Equation (4.2) can be thought of in terms of images or layers. In this case, S is the *shadow image*, L is the *lit image*, and β is the *shadow density matte* (or just *shadow matte*) representing the per-pixel visibility of the light source casting the shadow. Note that β may be fractional. Such values represent the same phenomena as fractional α values, including motion blur and partial coverage. Fractional β also allows us to simulate penumbral effects. (However, for non-point light sources, our model is only an approximation. This limitation is discussed in more depth in Section 4.7.)

The lit and shadow images L and S depend on the lighting conditions and albedos of the source scene. Hence, they are not directly transferable from scene to scene, unlike the shadow matte β, which is what we estimate during the matting process. For compositing, we therefore require two new images: the lit and shadow images, L' and S', of the new (destination) scene (Section 4.4).

4.3.2 Estimating the shadow matte

During the matting process, given the observed color C, we need to recover the shadow matte β. We therefore first need to estimate the shadow image S and the lit image L. We assume the image sequence is taken from a static camera with a static background. We can estimate the lit image and the shadow image using max and min compositing [96], i.e., finding the darkest and brightest value at each pixel. As with smoke matting (Section 3.4), we first use the video matting algorithm in Section 3.3 to extract the mattes of the foreground objects and exclude them from the max/min compositing. For each pixel, we then compute

$$S = \min_f C_f \quad \text{and} \quad L = \max_f C_f \,, \tag{4.3}$$

where the min and max are computed across all frames f independently at each pixel. Given the color images C, L, and S, which can be thought of as 3-vectors at each pixel, we estimate the shadow matte β using

$$\beta = \frac{(C - S) \cdot (L - S)}{\|L - S\|^2} \,. \tag{4.4}$$

This equation simply projects the observed color C onto the color line between L and S and computes the parametric distance of the projection along that line. (It is also the least squares estimate of β given a noisy color sample C.)

The method works quite well where we have good estimates for both L and S. However, the β estimates are noisy where L and S are similar. This happens wherever some part of the background is never covered by the shadow or always lies in shadow. Where $L = S$ we cannot recover β at all, but this signifies that the pixel is completely unaffected by the presence of shadow, so we mark these pixels as unknown β. Small areas with unknown β can be filled in using inpainting [9] or other hole-filling approaches, although such hole-filling was not necessary for the examples in this dissertation. Figure 4.3 illustrates the min and max composite and recovered β for a sample input. Note that our shadow compositing equation is derived in radiance space. In practice, we used the gamma-corrected pixel values directly but did not observe any visible artifacts.

4.4 Shadow compositing

To perform shadow compositing, we require the lit and shadow images L' and S' corresponding to the novel background scene. We assume that the novel background is captured with a static camera whose relation to the primary light source is the same as it was in the source scene. As explained in Section 1.3, such an assumption is commonly used for compositing to avoid difficult *view interpolation* and *relighting* operations. Equation (4.2) can then be used to calculate the composite color due to shadowing as a function of these two images as

$$C' = \beta L' + (1 - \beta)S' \,. \tag{4.5}$$

For a synthetic scene, it is easy to render both the lit and the shadowed versions of the scene. For a natural scene, we perform *photometric shadow scanning* by moving an object between the light and scene such that every part of the scene that we wish to composite shadows into is in shadow at some time. As in the shadow matting process previously described, we then use max/min compositing operations to recover the shadow and lit images corresponding to the scene.

With the recovered photometric parameters and source shadow matte β, we use the shadow compositing equation to make the composite as shown in Figure 4.4(a–c). The most noticeable flaw in this image is that the cast shadow does not conform to the geometry of the destination background scene as it should. We address this shortcoming in the following section.

4.5 Estimating shadow deformations

In this section, we show how to transfer a shadow cast on a source planar surface onto a target background with arbitrary geometry, assuming that some region of the target background has a planar region matching the source planar surface. To accomplish this, we construct a *displacement* or *warping map* w that places each pixel p in the target image in correspondence with some point $w[p]$ in the source image. Such a displacement map is sufficient because shadows cast from point light sources can be described using a 2D map and projection matrix, and the projected image of this map onto any geometry is some distortion of the map.

We use the same shadow compositing equation (4.5) but with a warped $\beta' = \beta[w[p]]$. Since the values of the displacement map $w[p]$ are not constrained to integer image coordinates, we use bilinear interpolation when resampling the shadow map β.

Our method for estimating the shadow displacement map is based on Bouguet's shadow scanning approach [16]. Shadow scanning is an active illumination method, and as such its reconstructions contain gaps wherever surfaces are occluded from the view of either the light source or camera. Fortunately, in the context of shadow compositing, these gaps coincide exactly with regions of the image unaffected by shadows and with regions not visible to the camera. Therefore, shadow scanning is ideally suited for our application. Furthermore, we can avoid Bouguet's calibration of the camera and light source by not doing full 3D reconstruction. Instead, we perform multiple passes with different scan orientations in order to compute a 2D shadow warping function. We call this process *geometric shadow scanning* to distinguish it from the photometric shadow scanning described in the previous section, which recovers no geometric (deformation) properties of the scene.

As in Bouguet's work, we require that our target background include some planar region, which is specified by the user via a hand-drawn mask (Figure 4.4(d)). We refer to this region as the *reference plane region* and denote the 3D plane defined by the points in this region as π (Figure 4.2(a)). Consider a pixel p through which we can see a 3D point P in the target background. For such a point P, there is a projection Q on the reference plane that lies at the intersection of π and the shadow ray V that passes through P. Now consider a stick or rod that casts a shadow on P. By observing the shadow line on the reference plane, we know that Q must lie along this line. If we re-orient the stick such that the shadow still covers point P, we obtain a second shadow line on the reference plane

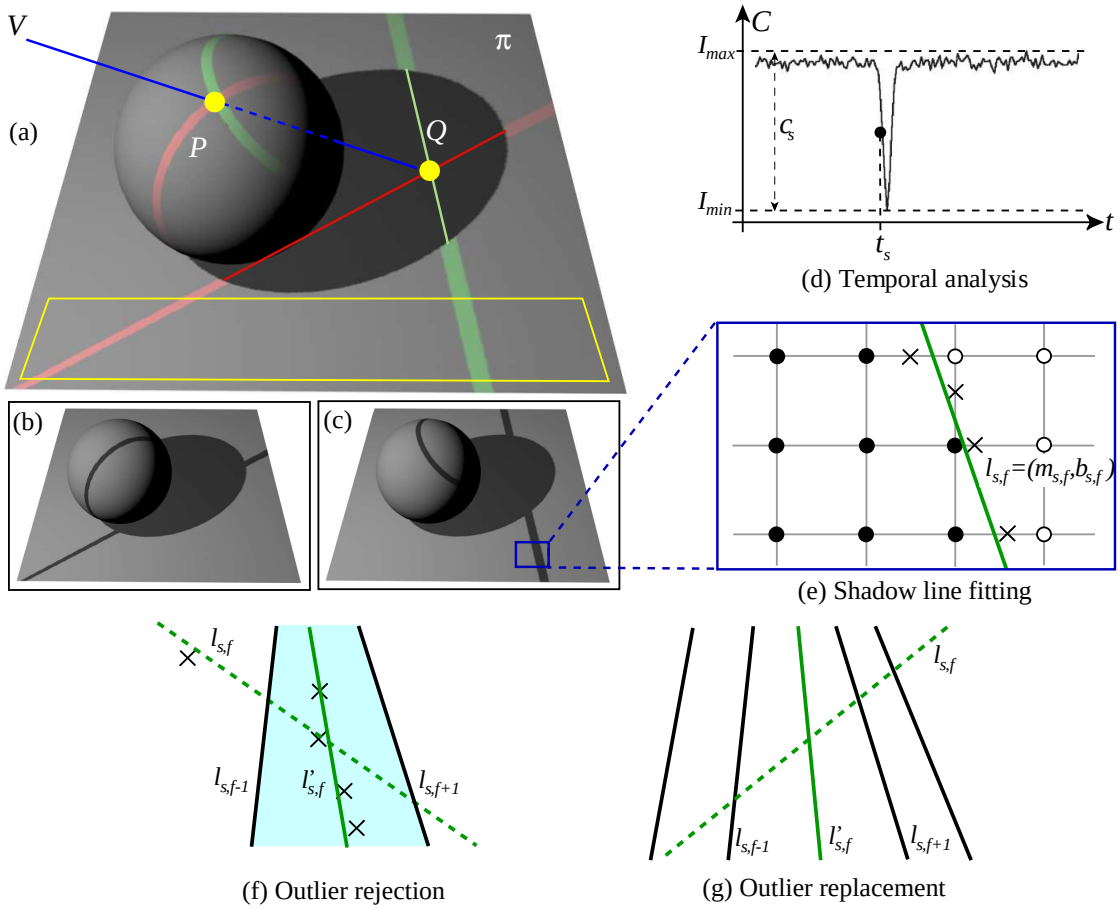


Figure 4.2: Illustration of the principle and details of geometric shadow scanning. Point Q is the point on the reference plane π that lies behind point P along ray V (a). It is found at the intersection of two shadow lines whose orientation is estimated from the reference planar region, outlined in light yellow. For reference, the pair of images below (b,c) shows 3D renderings of the two individual shadow lines. We estimate all shadow edge crossings using **temporal analysis** (d) to identify for each pixel p the shadow time $t_s[p]$, computed as the first time the pixel color goes below a threshold halfway between its min and max values, I_{min} and I_{max}. In the reference plane region, we can then determine the **shadow line** (e) for frame f of scan s, $l_{s,f} = (m_{s,f}, b_{s,f})$, by linearly interpolating between neighboring pixels with shadow times less than and greater than f (shown as black and white dots, respectively) and then fitting a line through the interpolated points. In some cases, line fits are poor due to spurious shadow time samples used in fitting. In the **outlier rejection** step (f), we identify lines with high error (the green dotted line $l_{s,f}$), discard samples outside the region defined by the nearest valid shadow lines on both sides (the cyan region between $l_{s,f-1}$ and $l_{s,f+1}$), and then re-fit the line (the green solid line $l'_{s,f}$). Inconsistent shadow lines may still occur when fitting to too few or closely spaced samples. In the **outlier replacement** step (g), we identify these lines (e.g., the green dotted line $l_{s,f}$) by detecting rapid changes in line slopes between frames and then replace them by interpolating neighboring line coefficients. The solid green line $l'_{s,f}$ is the replacement computed by interpolating between $l_{s,f-1}$ and $l_{s,f+1}$.

that must also intersect Q. We can compute Q as simply the intersection of the two shadow lines on the reference plane. In our implementation, we solve directly for q, the image-space projection of Q, which is in fact the warping map $w[p]$, by performing all computations in image coordinates, as described in the rest of this section. (Note that the displacement between $w[p]$ and p is actually a measure of projective depth (parallax) in a plane-plus-parallax formulation of 3D reconstruction [45].)

Acquisition. In practice, we use a series of directional scans, so that we can interpolate shadow lines for pixels that are not covered by a shadow edge at an integer frame time. Since the shadows may vary in width and the objects receiving the shadow may be very thin, we identify the leading temporal edge of the shadow lines, rather than trying to identify spatial shadow edges. Spatial edge detection is also less reliable in the presence of reflectance variations; the temporal analysis avoids this problem by focusing on a single pixel at a time.

To simplify the scanning process, a person moves the scanning stick by hand, potentially casting his or her own shadow onto the scene. Our algorithm is designed to ignore any additional moving shadows in the image frames, as long as the stick's shadow always appears to one side of the extra shadow. In principle, two passes suffice to find intersections, but we use three to five passes in order to improve the quality of the results and to cover more of the target background object. Figure 4.4(d) shows a frame from a geometric shadow scanning sequence. The user-specified reference plane region is shown with hatched blue lines.

Algorithm. Here is an outline of our algorithm, with details about each boldfaced step in the paragraphs that follow.

1. For each directional scan s:

   (a) Label each pixel location p with the continuous time $t_s[p]$ at which it is first crossed by the shadow edge using **temporal analysis**.

   (b) For each frame f, fit a **shadow line** equation $x = m_{s,f}y + b_{s,f}$ to the points r in the reference plane region labeled with time $t_s[r] = f$.

   (c) Perform **outlier rejection and replacement** on the line equation parameters $(m_{s,f}, b_{s,f})$ to reduce noise.

2. For each pixel location p:

   (a) For each scan s, interpolate the line equation parameters $(m_{s,\lfloor t_s[p] \rfloor}, b_{s,\lfloor t_s[p] \rfloor})$ and $(m_{s,\lceil t_s[p] \rceil}, b_{s,\lceil t_s[p] \rceil})$ from the nearest integer frames of the scan sequences to obtain the line equation parameters $(m_{s,t_s[p]}, b_{s,t_s[p]})$ for this pixel.

   (b) Compute the **intersection** q of lines for all scans s.

   (c) Store q as the value of the warp function for pixel p: $w[p] \leftarrow q$.

3. Optionally, perform **anisotropic diffusion** to smooth the displacement map $w[p]$ while maintaining its structure.

Details. We perform **temporal analysis** [16] to identify the shadow time $t_s[p]$ (Figure 4.2(d)). The person holding the scanning stick stays behind the trailing edge of its shadow, so we find the first frame when the pixel color goes below a threshold halfway between its min and max values, I_{min} and I_{max}. We linearly interpolate from the previous frame time to compute the shadow edge time $t_s[p]$. Also, we define the *shadow contrast* $c_s[p]$ as $I_{max} - I_{min}$; this value is later used as a confidence measure.

For **shadow line** fitting, Bouguet [16] used spatial analysis to find points on a planar region crossing the shadow edge and then fit those points to a line. However, Bouguet's method assumes that the reference plane is uniform in color and material. Since our goal is to handle natural scenes with reflectance variations on the reference plane, we instead use temporal analysis to determine the shadow lines more accurately (Figure 4.2(e)). For each frame f, we check each pixel r within the reference planar region. For every neighboring pair of pixels for which the signs of $t_s[r] - f$ are different, we estimate the zero-crossing point and add it to a list of candidate points on the shadow edge. After finding all such zero-crossings, we fit them to a line using linear regression. This line is the shadow line for frame f of scan s, parameterized by the tuple $(m_{s,f}, b_{s,f})$. (This is similar to a 2D version of the marching cubes method [54] for tracing isocurves, except we only collect the points, and need not connect them into a curve.)

Because of noise and bad shadow time estimates in the reference plane, some shadow lines may be poorly estimated. Accordingly, we perform **outlier rejection and replacement** to fix poorly fitted lines. For the lines whose fitting errors (as measured by the residual) are larger than some

threshold, we reject all the points outside the image region defined by the nearest valid shadow lines on both sides and refit the line without those outliers (Figure 4.2(f)). Even after this refitting, some lines may still be poorly estimated, either because we have too few points to fit or because the points are too close together in the image. Since the orientation of the scanning stick varies smoothly across the sequence, we identify and reject poorly fit lines by noting rapid changes in the slopes m between frames. We then replace the rejected shadow line by interpolating the line coefficients of the neighboring frames (Figure 4.2(g)). (We have not found discontinuities in the intercept b values to be of use for outlier rejection, since the scanning may progress at widely varying rates.)

After interpolating the shadow lines from integer frames for each scan, we use weighted least squares to calculate the **intersection** point q of all the shadow lines corresponding to shadow edges crossing point p. We determine the weight for each line based on two factors. First, if the shadow contrast for the scan is low, the estimated shadow time $t_s[p]$ for that pixel will be less accurate. Second, if the shadow line was poorly fit, it might have a significant impact on the location of intersection. In practice, we weight each line by $w_c w_f$, where the first term w_c is defined as the square of the shadow contrast $c_s[p]^2$, and the second term, w_f, is defined as $\exp(-(E_{s,t_s})^2)$, where E_{s,t_s} is the interpolated line fitting error for scan s at time t_s.

In a final step, we apply an **anisotropic diffusion** algorithm to smooth the displacement map while maintaining its structure [67]. Errors in the displacement map are most severe for pixels with low shadow contrast. Our diffusion process therefore flows data from high confidence areas to low confidence areas. Regions with very low contrast, e.g., areas that were already in shadow or were never shadowed, will be filled with nearby displacements that could be quite different from the true displacements. However, these areas will essentially be unaffected by shadows we want to transfer, so the values of the displacement map in those regions are, in principle, unimportant (see Section 4.7 for further discussion). In the end, the displacement map effectively places all pixels on the reference plane from which we extracted the source shadow matte (Figure 4.4(e)) to match the geometry of the target background (Figure 4.4(f)). Figure 4.4(g) shows the warped version of the shadow matte (Figure 4.3(d)) distorted by the recovered displacement map.

(a) Source frame (b) Source lit image

(c) Source shadow image (d) Recovered β

Figure 4.3: Shadow matting. Starting from a source image sequence (one frame shown in (a)), we first remove the foreground character using video matting. Our shadow matting algorithm recovers lit and shadow images (b,c) using max/min compositing. It then estimates β by projecting observed pixel colors onto the color lines between them (d).

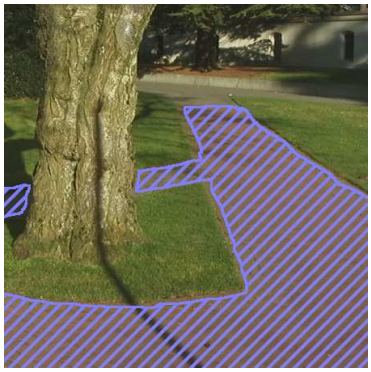
4.6 Results

To illustrate our method, we filmed some outdoor scenes in uncontrolled settings (Figure 4.3) and transferred the shadows to different target backgrounds (Figure 4.4(h,i)). Note that the source scenes were captured with complex textured backgrounds. Video matting (Chapter 3) was employed to extract the matte of the foreground actor. Our outdoor destination backgrounds were scanned with three to five passes of a $96'' \times 3'' \times 1''$ wooden board (Figure 4.4(d)). We took care to sweep the scanning stick so as to cover all the regions where the replacement shadow might be cast, while simultaneously covering a reasonable part of the reference plane. In all our examples, we used a

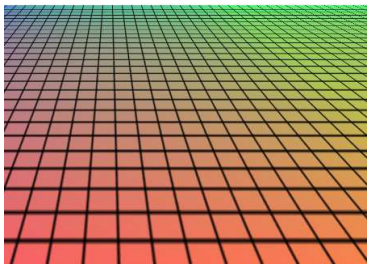
(a) Target lit image

(b) Target shadow image

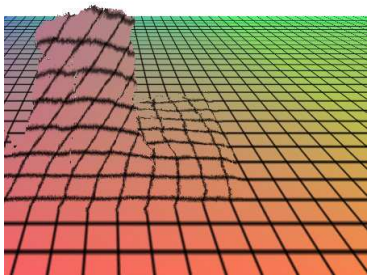
(c) Composite without displacement



(d) Geometric shadow scanning



(e) Regular grid on π



(f) Displacement map visualization

(g) Warped β

(h) Composite with displacement

(i) Using another background

Figure 4.4: Shadow compositing. Lit and shadow images are recovered for target geometry as well (a,b). Our composite remains unconvincing because the shadow does not conform to the background geometry (c). We acquire multiple, oriented scans of a straight line shadow (d), and compute line equations in the user-specified reference plane region, shown here with hatched blue lines. We then recover a displacement map (f) from the target scene to the source reference plane (e). This map distorts the shadow matte into the correct shape (g). In the composite (h), the shadow conforms to the background geometry faithfully. The results using a second background are shown in (i).

(a) Source frame (b) Target frame (c) Composite

Figure 4.5: An example of more complicated background geometry. The foreground element and shadow are extracted from a source frame (a) and transferred to a target frame (b) containing ivy to obtain a composite (c).

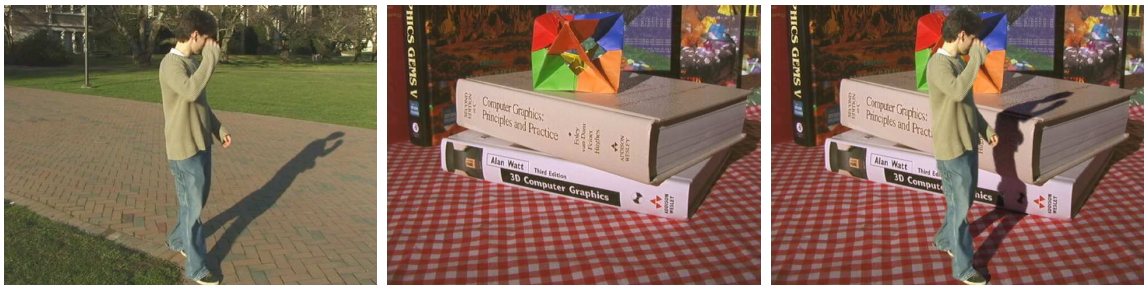
section of the ground plane as our reference plane, and our outdoor scenes were filmed at hours of the day when the sun lay low on the horizon, so that an 8-foot stick sufficed to scan an adequate region. In addition, we matched the camera pose and lighting by simply looking at a source image with a shadow on the ground plane, and then "eye-balling" how the camera should be placed in the target sequence by looking at another cast shadow.

Figure 4.5 demonstrates the advantages of our method over previous methods on a target scene with complex geometry. The surface of the ivy is bumpy and challenging to model manually. There is also severe self-shadowing in this scene (Figure 4.5(b)). Using previous methods, the composite would have double-shadowing in these regions. With our method, the shadows deform naturally to match the background geometry. Furthermore, there is no double-shadowing (Figure 4.5(c)). In Figure 4.6, we transferred an outdoor foreground to an indoor background scene scanned with wooden dowels.

4.7 Discussion

Our present matting and compositing method has a number of important restrictions. First, our shadow compositing equation (4.2) is strictly only valid for scenes with one dominant, pointlike light source, and it does not model potentially complex effects arising from interreflections.[2] Second, to estimate the lit and shadow images, we require a static camera. Third, in order to construct the

[2]Appendix A contains a more detailed analysis on the conditions when the shadow compositing equation holds.



(a) Source frame (b) Target frame (c) Composite

Figure 4.6: *Honey, I Shrunk the Grad Student!* A source (a) and target (b) for a miniature composite (c).

shadow displacement map, we require the source background to be planar and the target background to contain a planar reference region. Finally, we require that the relationship of the dominant light source, reference plane, and camera be matched in the source and target scenes.

Despite these restrictions, we believe that in many settings our approach provides a less restrictive capture mechanism compared to previous shadow extraction and compositing methods. As explained in Section 1.3, existing techniques typically share our requirement for a single matched key light source and matched cameras in order to avoid the difficult "view interpolation" and "relighting" problems. However, they also require the construction and calibration of matching geometry (physical or virtual sets) and the use of painstakingly lit bluescreens for source capture. Our technique requires neither matching geometry nor blue screens.

Furthermore, some of the theoretical restrictions on our technique can be relaxed in practice. For instance, the camera and light directions need not match precisely. As shown in the previous section, approximate matches are good enough to create convincing composites. In addition, the dominant light sources in our scenes are not perfect point lights, but no objectionable artifacts are evident. Appendix A contains a more detailed analysis that explains why there are no noticeable artifacts. For less point-like sources, we can transfer approximate penumbrae as long as the source and target backgrounds are at a similar distance to the casting object. We could potentially even blur or sharpen the shadow matte to create a *faux shadow* with a softer or harder penumbra.

Finally, we would like to extend the operating range of the shadow matting and compositing equations, and experiments suggest that at least some extensions are possible. For instance, assum-

ing the source and target backgrounds are Lambertian and geometrically similar, we have been able to matte and composite plausible shadows cast by multiple light sources without taking separate images for each light source, as shown in Figure 4.7(a). A sketch of the proof can be found in Appendix A. In addition, our shadow density mattes are currently single-channel mattes bounded between 0 and 1. Using unbounded, multichannel β values, our method can also be modified to transfer approximate color-filtered shadows (Figure 4.7(b)) and, for similar source and target background geometries, caustics (Figure 4.7(c)). (Note that, while these transparent, refractive shadow results are encouraging, the glass foreground objects were directly copied from the input frames using a rotoscoped matte.)

Summary

We have introduced a physically-based shadow matting and compositing method. Our approach has many advantages over previous approaches to shadow extraction and compositing. First, it can extract both the photometric and geometric information required for shadow compositing from natural (planar) scenes. Second, it can cast shadows onto scenes that already contain shadows, self-shadowing, and specularities. Third, it can cast shadows onto complex geometry without manually modeling the scene (at the price of having to do a shadow scan). Because we use the same camera to capture our warping function as we do to capture the images of the scene itself, we avoid the difficulties of having to accurately register an independently reconstructed model to our image.

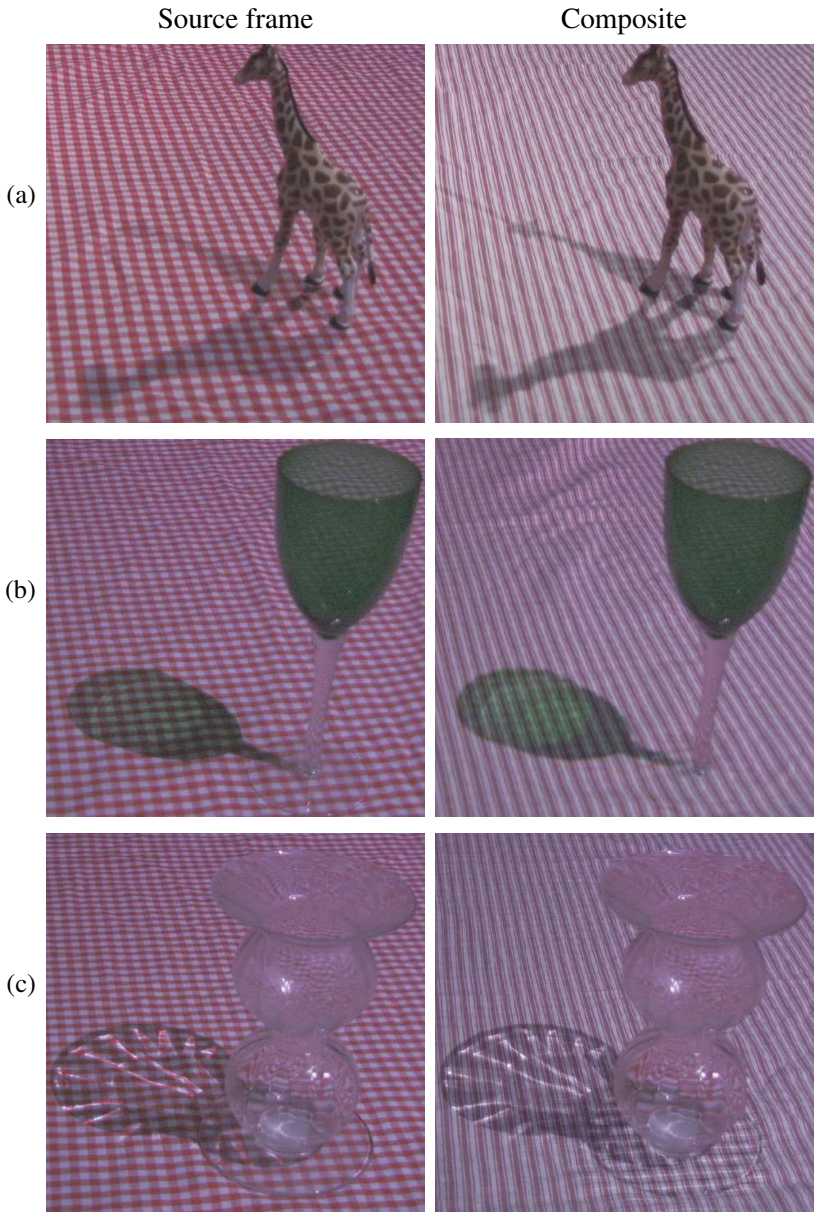


Figure 4.7: Our method can be extended to transfer shadows of multiple light sources (a), color-filtered shadows (b), and caustics (c). For multiple light sources, the shadow image is taken with all the lights turned off. Using multi-channel β's permits color-filtered shadows. We obtain caustics by allowing β to exceed unity.

Chapter 5

HIGH ACCURACY ENVIRONMENT MATTING AND COMPOSITING

5.1 Introduction

Like shadows, refraction and reflection are lighting phenomena that the traditional compositing model fails to capture. Zongker *et al.* introduced environment matting and compositing to capture these effects [118]. After making a set of approximations, and by photographing an object in front of a sequence of structured light backdrops, Zongker *et al.* demonstrate the ability to capture and render the effects of reflection, refraction, and colored filtering of light from a background. The backdrops consist of a hierarchy of finer and finer horizontal and vertical square-wave stripes from which the matte can be extracted with $O(\log k)$ images for a $k \times k$ pixel grid. This choice of backdrops is inspired by a related technique developed for 3D range scanning [10]. In practice, however, this approach has a number of shortcomings.

First, Zongker *et al.* assume that each image pixel collects light from a single region of the background, augmented with an alpha component for straight-through partial coverage. This assumption fails when we consider the effects of simultaneous reflection and refraction at a dielectric. Second, the method is tuned to capturing highly specular interactions, but breaks down in the presence of surfaces that are even moderately rough. Third, the mappings that are captured assume axis-aligned filtering of background pixels. At pixels where this assumption does not hold, this axis alignment results in excessive blurring and degrades the quality of the final composite. Fourth, the original method accounts for colored filtering of light, but does not account for the effects of dispersion, which necessitate different mappings per color channel and give rise to prismatic rainbowing effects.

In this chapter, we propose a more accurate model and matting method to address these limitations at the expense of using additional structured light backdrops (requiring $O(k)$ images).[1] Again

[1]This chapter describes joint work with Douglas E. Zongker, Joel Hindorff, Brian Curless, David H. Salesin and

taking inspiration from the 3D range-scanning literature, our approach uses a background consisting of a single stripe swept over time in the vertical, horizontal, and two diagonal directions. In each case, the stripe orientation is perpendicular to the sweeping direction. By combining sweeps with stripes of varying widths and intensity profiles, we demonstrate an efficient method for extracting higher quality environment mattes than the previous method.

We first discuss related work in Section 5.2. We then describe the general environment matting model in Section 5.3. In Section 5.4, we present our model for high accuracy environment mattes and a method to estimate the matte parameters. Then, we describe the experimental procedure used and show results in Section 5.5. We conclude in Section 5.6 with a discussion of further extensions and applications by others.

5.2 Related work

As with the original environment matting paper, our technique is inspired by range-scanning *illumination patterns*. The structured-light range-scanning literature suggests many possible ways to capture spatially varying properties of an object [10]. We should note that the end goals are substantially simpler in the range-scanning case. Range scanners attempt to recover just a handful of parameters per pixel: primarily, depth and reflectance. Environment-matting procedures, on the other hand, generally need to recover a continuous, spatially varying wavelength-dependent mapping from a background to the image plane. Even with the approximations described in Section 5.4, we must estimate at least 21 parameters per pixel, and sometimes many more.

The most "brute force" range-scanning method is to sweep a beam of light over an object in a raster pattern. Such an approach, while $O(k^2)$ in time since each range-image pixel is acquired sequentially, is actually practical for triangulation and imaging radar systems [10], since the reflected light seen by the sensor is known to have followed a straight line from the object. As a result, objects can be imaged with fast 1D (triangulation) or 0D (imaging radar) sensors. These faster sensors make the acquisition speeds comparable to, and in some cases better than, the $O(k)$ swept-stripe techniques described below. By contrast, typical objects used in environment matting will cause light from the background to bend through or reflect off of the object in unpredictable ways, thus

requiring a full 2D sensor array to capture the light. In this case, the $O(k^2)$ penalty is prohibitive.

Using a swept plane of light, $O(k)$ images can provide shape information through optical triangulation [10]. Our technique uses such a pattern, though multiple oriented sweeps are required to capture all the parameters. Note that this particular environment matting technique bears some resemblance to the spacetime analysis described by Kanade *et al.* [44] and Curless and Levoy [26], in which the authors study the time evolution of reflected light and triangulate over space and time.

By projecting a hierarchy of progressively finer stripe patterns, the required number of images for optical triangulation can be reduced to $O(\log k)$ [77]. Zongker *et al.* [118] use such a stripe hierarchy with some accompanying compromises over the swept-stripe technique.

Our work also has some connection to BRDF acquisition. Though we do not explicitly solve for the BRDF, one could certainly imagine using environment matting to capture reflection functions over a uniformly coated surface of known geometry, such as a sphere. A more direct connection lies in the BRDF fitting work of Ward [108]. Using an elliptical Gaussian model for rough specular reflection, he achieves excellent matches to goniometric samples. This model is the motivation for our choice of oriented, elliptical, Gaussian weighting functions described in Section 5.4.

5.3 The environment compositing equation

We begin by developing a general expression for the environment compositing equation and showing how it reduces to the traditional compositing equation, as well as the equation developed by Zongker *et al.* [118].

An imaging system, such as a CCD camera, records a discrete set of samples over an image plane. Let's assume for the moment that we have a camera that measures the irradiance at each wavelength separately. Then, for a given pixel p,[2] the camera records a value C for each wavelength. Following the environment mapping work of Blinn and Newell [15], we can express this color in terms of an infinitely distant environment illumination $\boldsymbol{E}(\boldsymbol{\omega})$,

$$C \;=\; \int \boldsymbol{W}(\boldsymbol{\omega})\, \boldsymbol{E}(\boldsymbol{\omega})\, d\boldsymbol{\omega}\,. \tag{5.1}$$

[2]Throughout this section, the derivations of equations are for a pixel p of the foreground plate. For example, C is for $C(p)$ and $\boldsymbol{W}(\boldsymbol{x})$ is for $\boldsymbol{W}(\boldsymbol{x}; p)$. Exceptions are $\boldsymbol{E}(\boldsymbol{\omega})$ and $\boldsymbol{T}(\boldsymbol{x})$, which represent the environment and are hence independent of p.

The weighting function $\boldsymbol{W}$ comprises all means of transport of environment lighting from all directions $\boldsymbol{\omega}$ through a foreground object to the camera, including any blurring due to the camera optics and area integration at a sensor cell. This equation holds under the assumption that none of the materials that are scattering light from the environment exhibit any wavelength coupling (e.g., fluorescence).

Next, we rewrite this equation as a spatial integral over a bounding surface (e.g., an environment map). Further, we augment the equation to include an additive foreground color F. This foreground color is typically due to some additional lighting that is separate from the environment map, though it could encompass object emissivity as well. Under these assumptions, our equation becomes

$$C \;=\; F \;+\; \int \boldsymbol{W}(\boldsymbol{x})\, \boldsymbol{T}(\boldsymbol{x})\, d\boldsymbol{x}\,, \tag{5.2}$$

where $\boldsymbol{T}(\boldsymbol{x})$ is the environment map. From this equation, we can develop a series of approximations that allow us to embed a foreground object in a new environment with varying degrees of quality.

To arrive at the traditional compositing equation (1.1), we assume that the straight-through background pixel is the only environment sample that affects the camera pixel. Let $\sqcap_p$ be the rectangular-area support of the pixel p on the background. We describe the (in this case, monochromatic) weighting function as

$$\boldsymbol{W}(\boldsymbol{x}) \;=\; (1-\alpha)\, \boldsymbol{\Pi}(\boldsymbol{x}; \sqcap_p)\,, \tag{5.3}$$

where α represents the foreground's transparency or partial pixel coverage, and $\boldsymbol{\Pi}(\boldsymbol{x}; A)$ is the box function of unit volume supported over an arbitrary axis-aligned area A. Next we define $\mathcal{M}(\boldsymbol{T},\, A)$ as the "texture-mapping operator" that performs the area integral and returns the average value of the texture $\boldsymbol{T}$ over region A,

$$\mathcal{M}(\boldsymbol{T},\, A) \;\equiv\; \int \boldsymbol{\Pi}(\boldsymbol{x}; A)\, \boldsymbol{T}(\boldsymbol{x})\, d\boldsymbol{x}\,. \tag{5.4}$$

Finally, defining the filtered background B to be the integral over the pixel's support,

$$B \;\equiv\; \mathcal{M}(\boldsymbol{T},\, \sqcap_p)\,, \tag{5.5}$$

and substituting the previous three equations into equation (5.2), gives the traditional compositing equation,

$$C \;=\; F \;+\; (1-\alpha)\, B\,. \tag{5.6}$$

Note that α typically does not have any wavelength dependence and thus cannot model color-filtered transparency. In addition, F is a measured quantity that is added directly to the attenuated background—in effect, it is pre-multiplied by α.

Zongker *et al.* model more complex lighting effects by approximating the environment as a set of m texture maps $\boldsymbol{T}_i(\boldsymbol{x})$ (the six sides of a bounding cube for instance), and by using more general light transport paths. Their weighting function is

$$\boldsymbol{W}(\boldsymbol{x}) \;=\; (1-\alpha)\,\boldsymbol{\Pi}(\boldsymbol{x};\sqcap_p) \;+\; \sum_{i=1}^{m} R_i\,\boldsymbol{\Pi}(\boldsymbol{x};A_i)\,. \tag{5.7}$$

In their formulation, the A_i represent various axis-aligned regions, each lying on a different texture map (corresponding, typically, to a different face of the environment cube). The R_i are reflectance coefficients describing the amount of light from the designated area of texture map i that is reflected or transmitted by the object at a given wavelength. In this formulation, R_i captures color-filtered transparency, and α represents only partial pixel coverage of the object. Substituting this weighting function into equation (5.2) gives the environment compositing equation[3] used by Zongker *et al.*,

$$C \;=\; F \;+\; (1-\alpha)\,B \;+\; \sum_{i=1}^{m} R_i\,\mathcal{M}(\boldsymbol{T}_i,\,A_i)\,. \tag{5.8}$$

Note that this formulation not only permits colored filtering of light, but also enables effects such as reflection and refraction since the light contributing to a pixel can be scattered from parts of the environment other than just the pixel directly behind the object. This approach, however, does have several distinct limitations. First, the components of the weighting function are assumed to be separable products of wavelength functions R_i and spatial functions $\boldsymbol{\Pi}(\boldsymbol{x};A_i)$. Thus, phenomena such as dispersion are not handled, since these require the weighting functions to shift spatially with wavelength. Second, the axis-aligned rectangle weighting functions do not simulate the effects of, for example, smooth BRDF's, which when mapped onto a background have a smooth, oriented footprint. Finally, other than the straight-through α-component, the approach models only a single mapping from a texture face to the camera. In reality, multiple mappings to the same face can and do happen and must be modeled, for example, when reflection and refraction at an interface cause view rays to split into distinct groups that collect light from the same texture map.

---

[3] In their paper, Zongker *et al.* called it the "environment matting equation."

Our objective, then, is to choose a different model for the weighting function that is more physically motivated and whose parameters are still easy to acquire using a simple apparatus.

5.4 High accuracy environment matting

To address the limitations of the weighting function described in Zongker *et al.*, we generalize it to a sum of Gaussians,

$$\boldsymbol{W}(\boldsymbol{x}) \;=\; \sum_{i=1}^{n} R_i \, \boldsymbol{G}_i(\boldsymbol{x}) \,. \tag{5.9}$$

In our formulation, we allow any number of contributions from a single texture map. Here, R_i is an attenuation factor, and each $\boldsymbol{G}_i$ is a normalized, elliptical, oriented $2D$ Gaussian,

$$\boldsymbol{G}_i(\boldsymbol{x}) \;\equiv\; \boldsymbol{G}_{\text{2D}}(\boldsymbol{x};\; \boldsymbol{c}_i,\; \boldsymbol{\sigma}_i,\; \theta_i) \,, \tag{5.10}$$

where $\boldsymbol{G}_{\text{2D}}$ is defined as

$$\boldsymbol{G}_{\text{2D}}(\boldsymbol{x}; \boldsymbol{c}, \boldsymbol{\sigma}, \theta) \;\equiv\; \frac{1}{2\pi\sigma_u\sigma_v} \exp\left[-\frac{u^2}{2{\sigma_u}^2} - \frac{v^2}{2{\sigma_v}^2}\right] \tag{5.11}$$

with

$$u \;=\; (x - c_x)\cos\theta - (y - c_y)\sin\theta \,, \tag{5.12}$$

$$v \;=\; (x - c_x)\sin\theta + (y - c_y)\cos\theta \,. \tag{5.13}$$

Here, $\boldsymbol{x} = (x, y)$ are the pixel coordinates, $\boldsymbol{c} = (c_x, c_y)$ is the center of each Gaussian, $\boldsymbol{\sigma} = (\sigma_u, \sigma_v)$ are the "unrotated" widths (a.k.a. standard deviations) in a local uv-coordinate system, and θ is the orientation. Figure 5.1 illustrates these parameters. Thus, our weighting function is some n-modal Gaussian with each term contributing a reflective or refractive effect from the object. Substituting into equation (5.2), we arrive at a new form of the compositing equation,

$$C \;=\; F \;+\; \sum_{i=1}^{n} R_i \int \boldsymbol{G}_{\text{2D}}(\boldsymbol{x};\; \boldsymbol{c}_i,\; \boldsymbol{\sigma}_i,\; \theta_i)\, \boldsymbol{T}(\boldsymbol{x})\, d\boldsymbol{x} \,. \tag{5.14}$$

(In this equation, we use $\boldsymbol{T}(\boldsymbol{x})$ to represent the set of all texture maps. The n modes of the weighting function are distributed over m textures, where n may be larger than m in general. The choice of the particular texture map used in computing a given Gaussian contribution i should be assumed to be implicitly controlled by the position $\boldsymbol{c}_i$ of the Gaussian weighting function.)

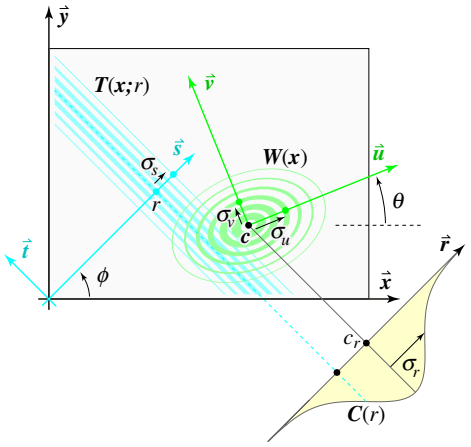


Figure 5.1: Illustration of the variables used in recovering an unknown elliptical, oriented Gaussian by sweeping out convolutions with known Gaussian stripes. As a tilted stripe $\boldsymbol{T}(\boldsymbol{x}; r)$ of width σ_s and position r sweeps across the background in direction s, it passes "under" the elliptical Gaussian weighting function $\boldsymbol{W}(\boldsymbol{x})$ associated with a single camera pixel. The camera records the integral of the product of the stripe and the weighting function, which describes a new, observed function $C(r)$ as the stripe sweeps. The center c_r and width σ_r of this observed function are related to the center $\boldsymbol{c}$ and width $\boldsymbol{\sigma} = (\sigma_u, \sigma_v)$ of the weighting function and the width of the stripe through equations (5.22) and (5.23).

The key advantages of this weighting function over the one used by Zongker *et al.* are that: (1) the spatial variation can be coupled with wavelength to permit modeling of dispersion; (2) it supports multiple mappings to a single texture; and (3) it approximates the behavior of BRDF's more closely (using oriented Gaussian weighting functions rather than axis-aligned box functions).

In practice, each of the values $C, F, R_i, \boldsymbol{c}_i, \boldsymbol{\sigma}_i, \theta_i$ and $\boldsymbol{T}$ in equation (5.14) is implemented as an *rgb* vector. So, this equation actually represents three independent equations, one for each of the color components. Our unknowns are $F, R_i, \boldsymbol{c}_i, \boldsymbol{\sigma}_i, \theta_i$, which means that each pixel encodes $3+18n$ parameters.

5.4.1 Swept Gaussians for environment matting

Recovering the environment matte requires taking a set of images of an object in front of a sequence of backdrops. Our method consists of three steps: (1) identifying pixels outside the object silhouette, (2) recovering the foreground color, and (3) applying a set of novel background *stimulus functions* to estimate the remaining parameters in the matte.

In the first step of our high-accuracy matting method, we identify pixels that are outside the silhouette of the object. This step is desirable for two reasons: it saves us the computational effort of estimating the matte parameters at these pixels, and it prevents us from making potentially noisy estimates of how straight-through background pixels map to the image, which would result in shimmering artifacts when rendering. To identify these pixels, we use the method of Zongker *et al.* In particular, we display a coarse-to-fine sequence of horizontal and vertical square-wave background patterns with and without the object. If we measure the same color (within a user-specified tolerance) at a pixel both with and without the object for each background, then we consider the pixel to map straight through. The overhead of taking these additional images is small compared to the total acquisition time.

To recover the foreground color, we photograph the object against two solid backgrounds. Replacing $\boldsymbol{T}(\boldsymbol{x})$ in equation (5.14) with a single backdrop of constant color T' and integrating, we get

$$C \;=\; F \;+\; RT' \,, \tag{5.15}$$

where $R \equiv \sum_{i=1}^{n} R_i$. Given the two images, we have two equations in two unknowns for each color channel, i.e., the foreground color F and the aggregate attenuation factor R. Solving the system of equations yields the foreground color.

Once we have the silhouette mask and the foreground color, we can solve for the remaining parameters of equation (5.14) using a large set of controlled backdrops (i.e., stimulus functions). Zongker *et al.* use a hierarchical set of square-wave stripe patterns in both the vertical and horizontal directions. They encounter difficulties with this method for two reasons: (1) the square waves are not good stimuli for recovering smooth functions, and (2) there is no obvious way to recover multiple mappings to the backdrop using these stimuli. To combat the first problem, we choose a smooth set of stimulus functions. To address the second, we constrain the stimuli to be narrow in one dimension, sweeping over time to reveal multiple mappings to the same background. Our choice of stimulus function, then, is a set of swept Gaussian stripes.

Let's see how we can use sweeping stripes to recover some of the parameters of our weighting functions. To begin, let us assume that the weighting function is unimodal and axis-aligned ($n = 1$ and $\theta = 0$). Under these assumptions, we can omit the summation and the subscript i in

equations (5.9) and (5.10) and then decompose the 2D Gaussian weighting function into two 1D components,

$$\boldsymbol{W}(\boldsymbol{x}) \;=\; R\,\boldsymbol{G}_{\text{1D}}(x; c_x, \sigma_u)\,\boldsymbol{G}_{\text{1D}}(y; c_y, \sigma_v)\,, \tag{5.16}$$

where

$$\boldsymbol{G}_{\text{1D}}(x; c, \sigma) \;=\; \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(x-c)^2}{2\sigma^2}\right]. \tag{5.17}$$

Our first stimulus function will be a vertical stripe that is constant in y and has a 1D Gaussian intensity profile in x with width σ_s,

$$\boldsymbol{T}(\boldsymbol{x}) \;=\; \boldsymbol{G}_{\text{1D}}(x; 0, \sigma_s)\,. \tag{5.18}$$

Now consider sweeping the stripe horizontally, displacing it at each step by some offset r,

$$\boldsymbol{T}(\boldsymbol{x}; r) \;=\; \boldsymbol{G}_{\text{1D}}(x - r; 0, \sigma_s) \;=\; \boldsymbol{G}_{\text{1D}}(r - x; 0, \sigma_s)\,. \tag{5.19}$$

The camera observation at a pixel is then given by

$$\begin{aligned}
C(r) \;&=\; \int \boldsymbol{W}(\boldsymbol{x})\,\boldsymbol{T}(\boldsymbol{x}; r)\,d\boldsymbol{x} \\
&=\; \iint R\,\boldsymbol{G}_{\text{1D}}(x; c_x, \sigma_u)\,\boldsymbol{G}_{\text{1D}}(y; c_y, \sigma_v)\,\boldsymbol{G}_{\text{1D}}(r-x, 0, \sigma_s)\,dx\,dy \\
&=\; \int R\,\boldsymbol{G}_{\text{1D}}(x; c_x, \sigma_u)\,\boldsymbol{G}_{\text{1D}}(r-x; 0, \sigma_s)\,dx \cdot \int \boldsymbol{G}_{\text{1D}}(y; c_y, \sigma_v)\,dy \\
&=\; R\,\boldsymbol{G}_{\text{1D}}(r; c_x, \sigma_u) * \boldsymbol{G}_{\text{1D}}(r; 0, \sigma_s) \cdot 1 \\
&=\; R\,\boldsymbol{G}_{\text{1D}}(r; c_x, \sqrt{{\sigma_u}^2 + \sigma_s^2})\,.
\end{aligned} \tag{5.20}$$

Thus, at each pixel, we expect to record a Gaussian evolving over time. Given an illumination stripe of known width, we can now estimate the *rgb* parameters c_x and σ_u using the procedure described below in Section 5.4.2. By symmetry, we can recover the vertical center coordinate and width by sweeping a horizontal Gaussian stripe in the vertical direction behind the foreground object. Thus, for the case of a single, unoriented Gaussian weighting function, a horizontal and a vertical swept Gaussian stripe are enough to estimate all the remaining parameters of the environment matte.

Figure 5.1 illustrates the more general case of a sweeping stripe that is constant in the t-direction and has Gaussian profile in the s-direction. This stripe is oriented at an angle ϕ with respect to the

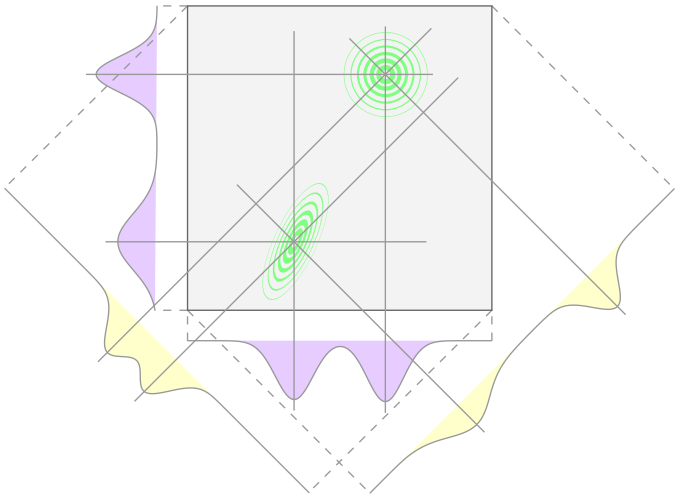


Figure 5.2: The green concentric rings depict a possible bimodal weighting function. The curves around the image indicate the convolved projections resulting from sweeping horizontal, vertical, and left and right diagonal Gaussian stripes across the screen independently. The horizontal and vertical sweeps alone (purple) are insufficient to determine the mode positions, but by adding additional diagonal sweeps (yellow) the correct modes can be determined.

xy-coordinate system and travels in the s-direction. Under these circumstances, it is straightforward to show that the observation at a pixel will be

$$C(r) \;=\; R\,\boldsymbol{G}_{\text{1D}}(r; c_r, \sigma_r)\,, \tag{5.21}$$

where

$$c_r \;=\; c_x \cos\phi + c_y \sin\phi\,, \tag{5.22}$$

$$\sigma_r \;=\; \sqrt{{\sigma_u}^2 \cos^2(\phi-\theta) + {\sigma_v}^2 \sin^2(\phi-\theta) + \sigma_s^2}\,. \tag{5.23}$$

Here, c_r is the center of the weighting function projected onto the r-axis, and σ_r is the projected, convolved standard deviation of the observed Gaussian.

Horizontally and vertically swept stripes alone ($\phi = 0^\circ$ and 90°, respectively) are not enough to determine the weighting function, so we introduce two diagonal passes at $\phi = 45^\circ$ and -45°. The additional oriented stripes serve another purpose: disambiguating multiple mappings to the backdrop. As shown in Figure 5.2, a bimodal weighting function results in two Gaussian images over time at a pixel as the stripe sweeps across. If we use just the horizontal and vertical stripes, the

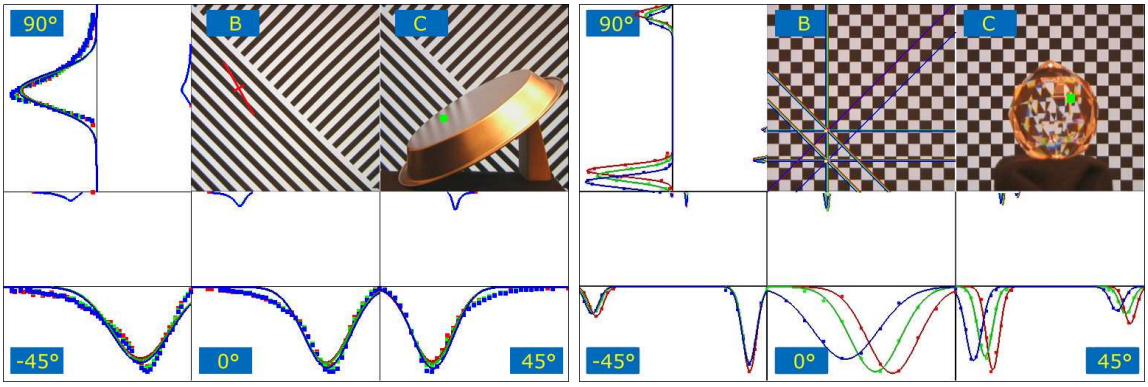
two modes recorded in each sweep yield multiple indistinguishable interpretations of the bimodal weighting function. The oriented stripes can be used to disambiguate these choices, as described below.

5.4.2 Estimating the matte parameters

In practice, our acquisition process entails stepping each Gaussian stripe across a computer screen and recording a set of samples for each sweep and for each color channel. Given this data, we seek the best set of parameters that explain the measurements. We estimate these parameters (separately for each color channel) in four steps: (1) identifying the number of Gaussians modes in the response, (2) solving for the projected centers and widths associated with each Gaussian mode, (3) intersecting the centers to localize the Gaussian modes, and (4) computing the parameters for each Gaussian mode.

To identify the number of Gaussians for the response to a given stripe sweep, we search for a series of peaks above the noise floor of the sensor. To make this process more robust, we first filter the 1D response function, and then identify the peaks. The locations of the peaks are the starting points for the projected centers of the projected modes. If the projected modes are clearly separated, we also estimate the projected widths by examining the extent of the signal that is above the noise floor. For two overlapping modes, we compute the distances from the left mode to the left extent and the right mode to the right extent and then estimate widths accordingly. For more overlapping modes, we compute the total width and divide by the number of modes. In any case, these center and width estimates are simply starting points for a Levenberg-Marquardt optimization procedure [70] that takes the original data, the number of Gaussians, and the initial center and width estimates in order to find the best centers and widths that explain the data.

Next, we use the sets of projected centers to choose the most likely locations of the Gaussian modes $\boldsymbol{G}_i$ (from equation (5.10)). The centers computed in the previous step should each correspond to the center of a Gaussian mode as projected onto the axis defined by the stripe. We then construct a line passing through each projected center point running parallel to the stripe's t direction. We consider all 4-tuples of horizontal, vertical, and two diagonal lines, and hypothesize their intersections by computing the point closest to each set of four lines. We measure the distance of



(a) Pie tin (b) Crystal

Figure 5.3: Screen shots of an interactive environment matte explorer. The explorer allows a user to interactively select a pixel on the composited image in the upper right pane (highlighted here in green). For any given pixel, the explorer shows the estimated Gaussian weighting functions for each color channel in the vertical, horizontal, and two diagonal directions. Each function is drawn twice: once on the scale of the pixel grid, and once on an expanded scale that magnifies the area around the peak. For the pie tin (a), the upper-middle panel shows the estimated weighting function for the selected pixel. (Here, only the weighting function for the red channel is shown.) The estimated oriented weighting function clearly shows the need to model the orientation for the weighting function. For the crystal (b), the upper-middle panel has superimposed stripes showing the estimated centers of the red, green, and blue responses in each direction. The crystal object here clearly demonstrates the effects of chromatic dispersion by the offset stripes in the different color channels. In addition, the upper-middle panel shows the resolution of the bimodal weighting function.

that point to each of the lines, and apply a user-specified tolerance to reject or accept the purported intersection.

Finally, given the set of Gaussian modes selected by the intersection process, we determine the parameters of each Gaussian mode. For each identified mode, we have estimates of the convolved, projected parameters according to equations (5.22) and (5.23). We compute R as the average of individually computed R's. The center $\boldsymbol{c}_i$ of $\boldsymbol{G}_i$ is simply the closest point to the lines as described in the previous step. Finally, we have four equations that relate the width and orientation of each Gaussian mode to the four measured widths. We solve this over-constrained, non-linear system of equations by finely sampling the space of possible orientations, solving for the linear-least-squares-best width parameters, and then choosing the orientation and widths that yield the lowest overall error.

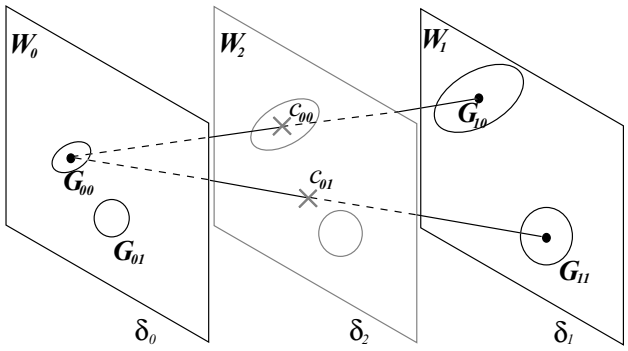


Figure 5.4: The resolution of the correspondence between Gaussians. Assuming that there are two Gaussians in both mattes, to decide the corresponding Gaussian for $\boldsymbol{G}_{00}$ in the matte $\boldsymbol{W}_1(\boldsymbol{x})$, we calculate the two interpolated centers c_{00} and c_{01} assuming $\boldsymbol{G}_{00}$ corresponds to $\boldsymbol{G}_{10}$ and $\boldsymbol{G}_{11}$ respectively. The interpolated center c_{00} coincides with a Gaussian in the matte $\boldsymbol{W}_2(\boldsymbol{x})$ but c_{01} does not. Hence, we associate $\boldsymbol{G}_{00}$ with $\boldsymbol{G}_{10}$.

The result of this sequence of steps is a reasonable estimate for the number of Gaussian modes and their parameters. As a final step, we apply a full Levenberg-Marquardt optimization to find the best $\boldsymbol{c}_i$, $\boldsymbol{\sigma}_i$, and θ_i that explain all of the measurements. Figure 5.3 illustrates the response at a given pixel for each sweep and the estimated weighting function.

5.4.3 Depth correction

Thus far, our method allows us to acquire environment mattes for objects at a fixed distance from the backdrop. Hence, the composites are effectively performed in front of backdrops at the same depth as the mattes were acquired. To composite the object at a novel depth, we need to acquire a matte at the same depth.

To allow composting at arbitrary depths, as suggested by Zongker *et al.* in their paper [118], we extract two different environment mattes $\boldsymbol{W}_0(\boldsymbol{x})$ and $\boldsymbol{W}_1(\boldsymbol{x})$ for the foreground object using backdrops placed at two different depths, δ_0 and δ_1. Given a novel depth δ, to approximate the environment matte $\boldsymbol{W}(\boldsymbol{x})$ at that depth, we linearly interpolate the parameters of $\boldsymbol{W}_0(\boldsymbol{x})$ and $\boldsymbol{W}_1(\boldsymbol{x})$. However, to perform this interpolation, we have to find the associations between Gaussians $\boldsymbol{G}_{0i}$'s and $\boldsymbol{G}_{1i}$'s in the mattes $\boldsymbol{W}_0(\boldsymbol{x})$ and $\boldsymbol{W}_1(\boldsymbol{x})$. We assume that both mattes have the same number of Gaussians. In a rare case, if the numbers of Gaussians do not match, we drop the Gaussians

Figure 5.5: A photograph of our experimental setup for acquiring high accuracy environment mattes.

with smaller attenuation factors R_i's in the matte with more Gaussians. To find the correspondence between Gaussians, we acquire a third environment matte $\boldsymbol{W}_2(\boldsymbol{x})$ using a backdrop located at a depth δ_2 in between the two original depths. If two Gaussians correspond, their interpolated center at depth δ_2 must coincide with the center of some Gaussian in $\boldsymbol{W}_2(\boldsymbol{x})$. Hence, we can use this fact to resolve the correspondence between Gaussians. Figure 5.4 gives an example of resolving such ambiguity. After finding the correspondences between Gaussians, the parameters for $\boldsymbol{W}(\boldsymbol{x})$ can be interpolated using $\boldsymbol{W}_0(\boldsymbol{x})$ and $\boldsymbol{W}_1(\boldsymbol{x})$ accordingly. Note that $\boldsymbol{W}_2(\boldsymbol{x})$ is captured only for finding the correspondences between $\boldsymbol{G}_{0i}$'s and $\boldsymbol{G}_{1i}$'s, and is not necessary to be stored for compositing.[4]

5.5 Results

Figure 5.5 shows our experimental setup. A Sony DCR-TRV900 digital video camera records images of an object as one of three monitors presents a sequence of stimulus functions. We correct for non-linearities in the video camera using Debevec and Malik's method [30]. To calibrate each monitor's brightness settings, we display a sequence of solid gray images and record them with

[4]In principle, $\boldsymbol{W}_2(\boldsymbol{x})$ can also be used to determine if the Gaussians shrink to a zero width and expand again from $\boldsymbol{W}_0(\boldsymbol{x})$ to $\boldsymbol{W}_1(\boldsymbol{x})$, but we did not implement it.

the radiometrically corrected camera. After averaging the gray values within each image, we have a mapping between gray values on the computer and displayed radiance. Each stripe image is adjusted so that the profile is Gaussian in radiance space.

After calibration, we begin imaging the object by extracting the foreground color and silhouette mask as described in the previous section. Next, we display the sequence of background patterns. We translate each Gaussian stripe across the screen in steps of $\sigma_s/2$ to ensure enough samples for the estimation procedure. We typically use $\sigma_s = 2$ or 4 (measured in camera pixels) requiring about 300 or 150 stripe positions, respectively, per horizontal, vertical, or diagonal sweep. Due to the lack of synchronization between the monitor refresh and the camera, we are unable to capture at video rates; instead, a typical capture plus digital video transfer requires roughly 30 minutes. (With a synchronized system and real-time transfer to PC memory, we expect acquisition could take less than a minute.) Processing time for an environment matte is typically about 20 minutes on a 400 MHz Pentium II PC with 128 MB of RAM.

We demonstrate the accuracy of our new environment matting algorithm on three objects. For each example, we render the matte by explicitly integrating the oriented Gaussian filters over the background. The results are shown in Figures 5.6 and 5.7.

The first object is a crystal in the shape of a regularly triangulated sphere, shown in Figure 5.6(a). The planar facets give rise to prismatic rainbowing effects due to dispersion. This effect is captured by our new matting algorithm because we estimate a different Gaussian weighting function (with a different center) for each color channel. This effect is not modeled by the old matting algorithm, which breaks down even further due to the multiple mappings at pixels that straddle crystal facets.

The multiple mapping problem is more clearly demonstrated by our next object, a beer glass laid on its side (Figure 5.6(b)). Due to the grazing angle, simultaneous reflection and refraction at the top of the glass results in bimodal mappings to the background. The old method simply cannot handle this phenomenon, whereas the new method captures the effect realistically.

Finally, we captured an environment matte for a pie tin with a rough surface, oriented to cause tilted reflections from the backdrop. Figure 5.6(c) demonstrates the failure of the old method to capture the large, smooth weighting function indicative of surface roughness, in contrast to the new method's success.

Figure 5.7 demonstrates the importance of capturing the orientation of the weighting function.

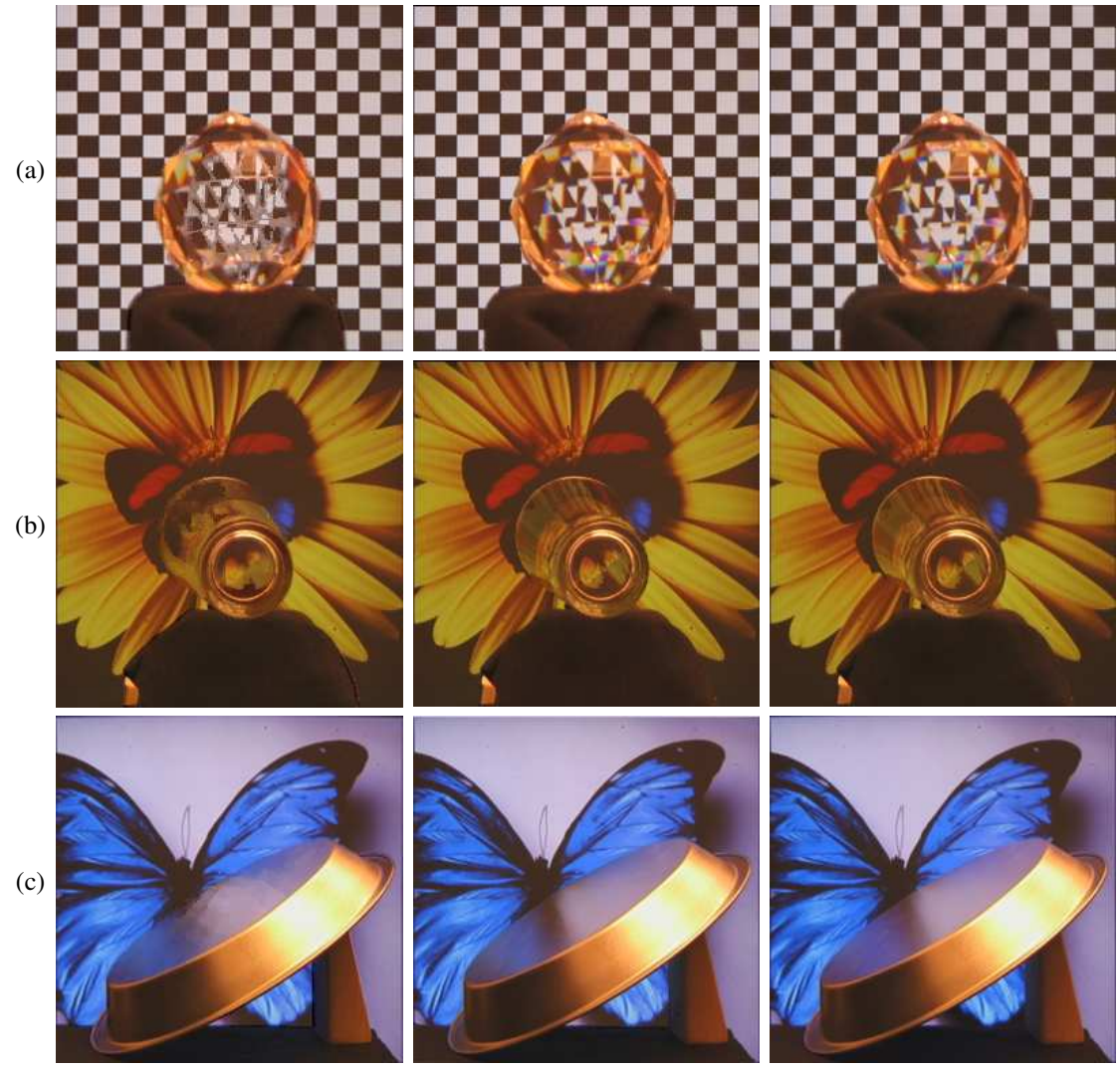
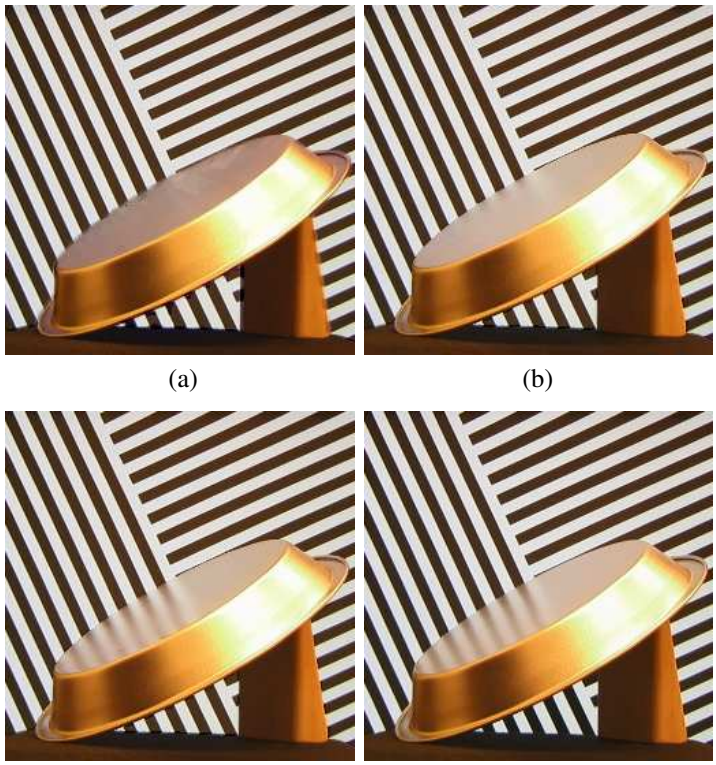


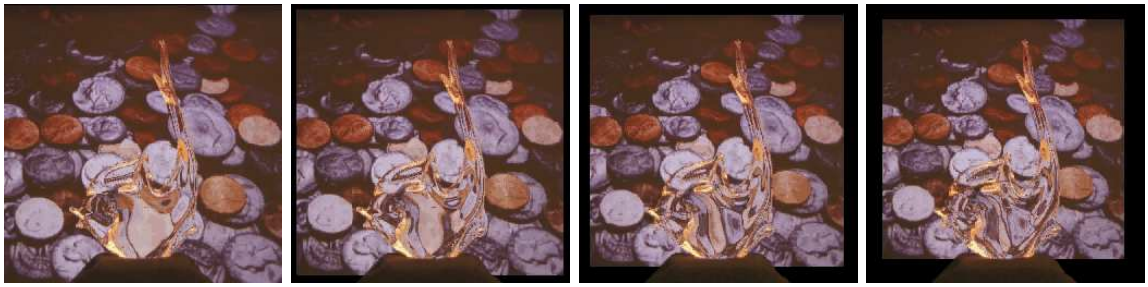
Figure 5.6: Comparisons between the composite results of the previously published algorithm, the high accuracy environment matting technique described here, and reference photographs of the matted objects in front of background images. Lighting in the room contributed a yellowish foreground color F that appears, e.g., around the rim of the pie tin in the bottom row. (a) A faceted crystal ball causes rainbowing due to prismatic dispersion, an effect successfully captured by the high accuracy technique since shifted Gaussian weighting functions are determined for each color channel. (b) Light both reflects off and refracts through the sides of a glass. This bimodal contribution from the background causes catastrophic failure with the previous unimodal method, but is faithfully captured with the new multi-modal method. (c) The weighting functions due to reflections from a roughly-textured pie tin are smooth and fairly broad. The new technique with Gaussian illumination and weighting functions handles such smooth mappings successfully, while the previous technique based on square-wave illumination patterns and rectangular weighting functions yields blocky artifacts.



(a) (b)

(c) (d)

Figure 5.7: Oriented weighting functions reflected from a pie tin. (a) As in Figure 5.6, the previous method yields blocky artifacts for smooth weighting functions. (b) Using the high accuracy method with unoriented Gaussians ($\theta = 0$) produces a smoother result. (c) Results improve significantly when we orient the Gaussians and solve for θ. In this case, $\theta \approx 25^\circ$ over most of the bottom surface (facing up) of the pie tin. (d) Reference photograph.



(a) (b) (c) (d)

Figure 5.8: An environment matte rendered at novel depths. Parts (a)-(d) are composites of a glass fish rendered at four different depths. The composites (a) and (d) correspond to the two depths at which we extracted mattes, whereas the composites (b) and (c) are composited using the interpolated environment mattes at two novel depths.

When we apply the new method without estimating orientation (i.e., by simply using the widths determined by the horizontal and vertical sweeps), the texture lines running at 25° off of vertical are significantly blurred. After estimating the orientation, we obtain a matte that faithfully preserves these details.

Figure 5.8 shows results for depth correction, captured for a glass fish. We captured environment mattes of the glass fish at two different depths corresponding to Figure 5.8(a) and (d). Figure 5.8(b) and (c) show the composites using the interpolated environment mattes at two novel depths.

5.6 Extensions and applications

As in traditional matting, environment matting involves an inherent tradeoff: the amount of input data required and the use of an active or passive illumination versus the restriction and quality of the resulting matte (See Table 1.1). Our method and the previous method of Zongker *et al.* represent two points in the environment matting design space. In this section, we discuss three other points in this design space, one of them (real-time environment matting) explored contemporaneously with the method described in this chapter, and two that followed.

Although relaxing some restrictions on the weighting functions placed by the previous method of Zongker *et al.*, our approach still restricts the weighting function to be a sum of a limited number of oriented elliptical Gaussians. Peers *et al.* propose an environment matting technique, called wavelet environment matting (wavelet EM), that makes no assumption on the form of the weighting function at the expense of using even more backdrops [65]. Theoretically, their method would require $O(k^2)$ backdrops. To be more efficient, they use a set of wavelet illumination patterns B_i to exploit their hierarchical nature for reducing the number of backdrops. During acquisition, an error-tree is employed to decide the subsequent patterns to use based on previously recorded photographs. Hence, the displayed patterns are determined on fly, adapting to the observed properties of the environment matte. The process repeats until reaching a stop criterion. In practice, they set a time limit of 12 hours as the stopping criterion and found it sufficient for capturing most types of objects. On average, thousands of photographs are captured. Instead of being represented by a set of parameters, the weighting function is directly represented by a set of photographs C_i acquired in front of the wavelet backdrops B_i. During compositing, the novel background B for compositing is

projected onto the space spanned by the wavelet backdrops B_i to obtain coefficients a_i so that $B \approx \sum_i a_i B_i$. The composite is then computed as $\sum_i a_i C_i$. This avoids the optimization procedure for finding parameters of the weighting functions, but requires a large amount of storage for the images C_i. The authors report that an average of 2.5GB is required for a single object. One strength of this approach is that it allows the capture of diffuse reflections, which proved difficult to capture using our swept stripe patterns, because the illumination due to any single stripe was too weak. However, by acquiring high-dynamic-range radiance images, we suspect the weak illumination problem is solvable. In any case, it would be instructive to compare our Gaussian weighting function to their wavelet model for handling diffuse objects.

In contrast to the approaches that attempt to improve the quality of the mattes, real-time environment matting (real-time EM) requires only one backdrop and allows the capture of environment mattes for transparent objects in motion, such as water pouring into a glass, at the expense of the quality of the resulting matte [23]. Real-time EM utilizes a calibrated color ramp B as the backdrop. This approach essentially represents the weighting function by a warping function. For a pixel p, real-time EM projects the observation color $C(p)$ from black onto the calibrated background manifold in *rgb* space. The position of the projected point gives the position q on the background contributing to p. This procedure, however, leads to significant random distortions, because observation noise directly and significantly affects the estimated mappings to background pixels. To overcome the effects of noise, a non-linear discontinuity-preserving filter [67] is applied to the matte. While allowing capture of moving objects, this technique is only designed to work with perfect specular refractive/reflective colorless objects.

The methods we have discussed so far are all active approaches, and hence require a calibrated laboratory setting. Wexler *et al.* [113] propose an approach, called image-based environment matting (image-based EM), to pull an environment matte from a sequence of images C_i in which a static foreground object is filmed in front of a moving background or by a panning camera in front of a natural background. For their method to work, clean plates B_i are required. These could be obtained by mosaicking the moving-background in a manner similar to the one described in Section 3.3.1. For each pixel p in the environment matte, a probability map P_i is built for a background B_i: $P_i(q) \propto \exp(-\|C_i(p) - B_i(q)\|^2)$, where q is a pixel on the background. This probability map potentially has many peaks where the color $B_i(q)$ on the background is similar to $C_i(p)$. Since

we have multiple backgrounds, we can integrate these probability maps P_i's by multiplying them together to get rid of the false peaks, essentially identifying the location of the pixel contributing to p on the background. Although, in principle, their weighting function (essentially the integrated probability map) allows an arbitrary weighting function, the way of constructing the probability map restricts the refractive and reflective properties of the object. As with real-time EM, their method is only effective for capturing colorless specular objects. Furthermore, image-based EM is a multiple-frame approach, requiring multiple frames for extracting a single environment matte. Hence, the object must be static during the acquisition, and the success of this method relies on having enough background samples or sufficiently rich backdrop images.

Thus far, all methods discussed capture environment mattes for a fixed viewpoint. Matusik *et al.* [59] combine environment matting and reflectance fields to acquire an image-based representation of transparent and refractive objects including their 3D shapes. They employ the high accuracy environment matting method presented in this chapter to acquire the environment mattes from many views. The object can then be viewed from arbitrary novel viewpoints by interpolating the environment mattes of nearby viewpoints. For interpolation, as described in Section 5.4.3, they have to resolve the ambiguity of matching Gaussians. To resolve matching ambiguities, they assume that there are at most two Gaussians, one for reflection and one for refraction, and only match Gaussians of the same type. Since they also capture the geometry, they can compute the angle between the surface normal and the vector from the surface point to the center of the Gaussian. A Gaussian is classified as reflective if this angle is smaller than 90°; otherwise, it is transmissive.

Summary

This chapter presents a method for capturing a higher quality environment matte in which each pixel can see one or more different Gaussian regions of the environment on a per-channel basis. This method allows for the capture of complex and subtle interactions of light with objects with multi-modal refraction and reflection, or with prismatic color dispersion, thus, advancing the state of the art in environment matting.

Chapter 6

APPLICATION: ANIMATING PICTURES

6.1 Introduction

Matting and composting have many applications. This chapter presents a novel application of natural image matting for taking a still picture and making it move in convincing ways.

When we view a photograph or painting, we perceive much more than the static picture before us. We supplement that image with our life experiences: given a picture of a tree, we imagine it swaying; given a picture of a pond, we imagine it rippling. In effect, we bring to bear a strong set of "priors" (to use the technical jargon from statistics), and these priors enrich our perceptions.

In this chapter, we explore how a set of explicitly encoded priors might be used to animate pictures on a computer. The *fully automatic* animation of *arbitrary* scenes is, of course, a monumental challenge. In order to make progress, we make the problem easier in two ways. First, we use a semi-automatic, user-assisted, approach. In particular, we have a user segment the scene into a set of animatable layers and assign certain parameters to each one. Second, we limit our scope to scenes containing passive elements that respond to natural forces in some oscillatory fashion. The types of passive elements we explore include plants and trees, water, floating objects like boats, and clouds. The motion of these objects is driven by the same natural force, namely, wind. While these may seem like a limited set of objects and motions, they occur in a large variety of pictures and paintings, as shown in Figure 6.2.

It turns out that all of these elements can be animated in a similar way. First, with the help of our Bayesian matting algorithm (Chapter 2), we can segment the picture into a set of user-specified layers. As each layer is removed from the picture, inpainting is used to fill in the resulting hole. Next, for each layer, we synthesize a *stochastic motion texture* using spectral methods [90]. Spectral methods work by generating a noise spectrum in the frequency domain, applying a (physically based) spectrum filter to that noise, which is specific to the type of natural force and to the type

and parameters of the passive object being affected, and computing an inverse Fourier transform to create the stochastic motion texture. This motion texture is a time-varying 2D displacement map, which is applied to the pixels in the layer. Finally, the warped layers are recomposited to form the animated picture for each frame.

The resulting moving picture can be thought of as a kind of video texture [79]—although, in this case, a video texture created from a single static image rather than from a video source. Thus, these results have potential application wherever video textures do, i.e., in place of still images on Web sites, as screen savers or desktop "wallpapers," or in presentations and vacation slide shows. In addition, creating video textures from a static image rather than from a video source has certain advantages.

First, because they are created synthetically, animated pictures allow greater creative control in their appearance. For example, the wind direction and amplitude can be tuned for a particular desired effect. Second, consumer-grade digital still cameras generally provide much higher image quality and greater resolution than their video camera counterparts. These advantages may allow video textures to be used in entirely new situations that were not previously practical. For example, controllable, high-resolution video textures might be usable for animated matte paintings in special effects.

We begin with a discussion of related work (Section 6.2) followed by an overview that describes the basic flow of our system (Section 6.3). We then address our most important subproblem, namely synthesizing stochastic motion texture (Section 6.4). Finally, we discuss our results (Section 6.5) and conclude with a summary.

6.2 Related work

Our goal is to synthesize a stochastic video from a single image. Hence, our work is directly related to the work on video textures and dynamic textures [79, 89, 98, 107, 109]. Like our work, video textures focus on "quasi-periodic" scenes. However, the inputs to video texture algorithms are short videos that can be analyzed to mimic the appearance and dynamics of the scene. In contrast, the input to our work is only a single image.

Our work is, in spirit, similar to the "Tour Into the Picture" system developed by Horry *et al.* [41].

Their system allows users to map a 2D image onto a simple 3D box scene based on some interactively selected perspective viewing parameters such as vanishing points. This allows users to interactively navigate into a picture. Criminisi *et al.* [25] propose an automated technique that can produce similar effects in a geometrically correct way. More recently, Oh *et al.* [63] developed an image-based depth editing system capable of augmenting a photograph with a more complicated depth field to synthesize more realistic effects. In our work, instead of synthesizing a depth field to change the viewpoint, we add motion fields to make the scene change over time.

For certain classes of motions, our system requires the users to specify a skeleton for a layer. It then performs a physically-based simulation on the skeleton to synthesize a motion field; it is therefore similar to *skeleton-based animation* approaches. Litwinowicz and Williams [53] use keyframe line drawings to deform images to create 2D animations. Their system is quite useful for traditional 2D animation. However, their technique is not suitable for modeling the natural phenomena we target because such motions are difficult to keyframe. In addition, they use a smooth scattered data interpolation to synthesize a motion field without any physical dynamics model.

Our work also has similar components to the *object-based image editing* system proposed by Barrett and Cheney [4], namely, *object selection*, *matte extraction*, and *hole filling*. Indeed, Barrett and Cheney have also demonstrated how to generate a video from a single image by editing and interpolating keyframes. Like Litwinowicz's system, the focus is on key-framed rather than stochastic (temporal texture-like) motions.

An earlier attempt to create the illusion of motion from an image was the "Motion without movement" paper by Freeman *et al.* [33]. They apply quadrature pairs of oriented filters to vary the local phase in an image to give the illusion of motion. While the motion is quite compelling, the band-pass filtered images do not look photorealistic.

Even earlier, at the turn of the (20th) century, people painted outdoor scenes on pieces of masked vellum paper and used series of sequentially timed lights to create the illusion of descending waterfalls [39]. People still make this kind of device, which is often called a *kinetic waterfall*. Another example of a simple animated picture is the popular Java program *Lake applet*, which takes a single image and perturbs the image with a set of simple ripples [36]. Though visually pleasing, these results often do not look realistic because of their lack of physical properties.

Working on an inverse problem to ours, Sun *et al.* [94] propose a *video-input driven animation*

(VIDA) system to extract physical parameters, like wind speed, from real video footage. They then use these parameters to drive the physical simulations of synthetic objects to integrate them consistently with the source imagery. They estimate physical parameters from observed displacements; we synthesize displacements using a physical simulation based on user-specified parameters. They target a similar set of natural phenomena to those we study, such as plants, waves, and boats, which can all be explained as *harmonic oscillations*.

To simulate our dynamics, we use physically-based simulation techniques previously developed in computer graphics for modeling natural phenomena. For waves, we use the Fourier wave model to synthesize a time-varying height field. Mastin *et al.* [56] were the first to introduce statistical frequency-domain wave models from oceanography into computer graphics. In a similar way, we synthesize stochastic wind fields [82, 92] by applying a different spectrum filter. When applying the wind field to trees, since the force is oscillatory in nature, the corresponding motions are also periodic and can be solved more robustly and efficiently in the frequency domain [83, 91].

Aoki *et al.* [2] coupled physically-based animations of plants with image morphing techniques as an efficient alternative to the expensive physically-based plant simulation and synthesis, but only demonstrated their concept on synthetic images. In our work, we target real pictures and use our approach as a way to synthesize video textures for stochastic scenes.

Our system requires users to segment an image into *layers*. We use our Bayesian image matting algorithm to extract alpha mattes from the input image (Chapter 2). To fill in holes left behind after removing each layer, we use an *inpainting* algorithm [9, 24, 31, 42].

6.3 System overview

Given a single image I, how can we generate a continuously moving animation? The approach we follow is to break the image up into several layers and to then synthesize a *motion texture*[1] and apply it to each layer individually.

A motion texture is essentially a *time-varying displacement map* defined by a motion type, a set of motion parameters, and optionally a motion skeleton. A displacement map D is a set of

[1]We use the terms *motion texture* and *stochastic motion texture* interchangeably in the paper. The term *motion texture* was also used by others [51, 71] to refer to linear dynamic system learned from motion capture data.

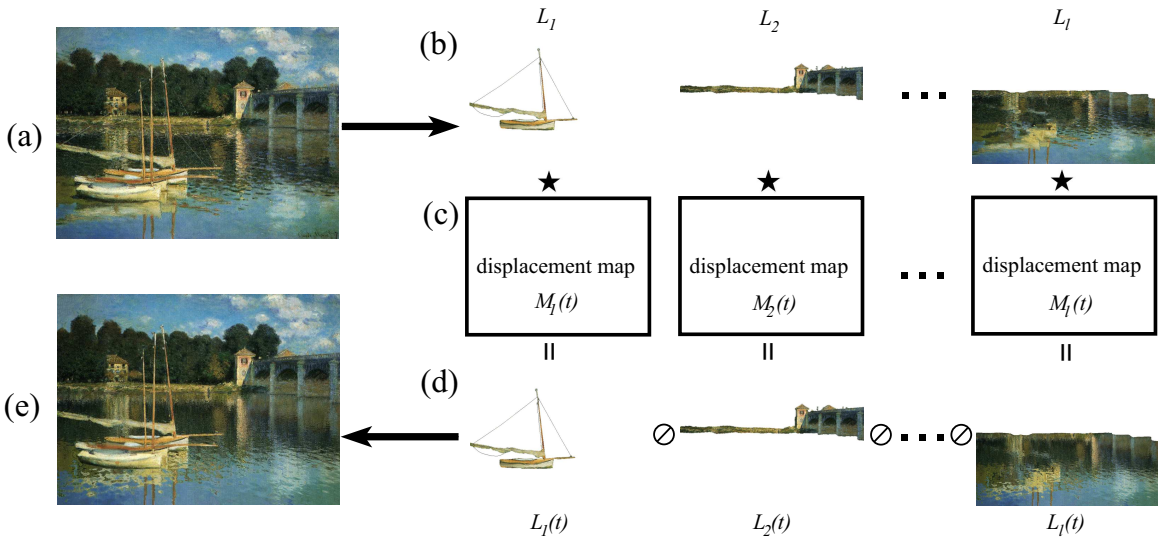


Figure 6.1: Overview of our system. The input still image (a) is manually segmented into several layers (b). Each layer L_i is then animated with a different stochastic motion texture $M(t)$ (c). Finally, the animated layers $L_i(t)$ (d) are composited back together to produce the final animation $I(t)$ (e). The input painting is Claude Monet's *The Bridge at Argenteuil*.

displacement vectors,

$$D(p) = (d_x(p), d_y(p)) \,, \tag{6.1}$$

for pixels $p = (x, y)$. A motion texture $M(t)$ is a mapping from a time t to a displacement map D.

Applying a displacement field D directly to an image I results in a forward warped image I' such that

$$I'(x + d_x(p), y + d_y(p)) = I(x, y) \,. \tag{6.2}$$

However, since forward mapping is fraught with problems such as aliasing and holes, we actually use inverse warping, $I' = D' \star I$, where

$$I'(x, y) = I(x + d'_x(p), y + d'_y(p)) \,. \tag{6.3}$$

We could compute the inverse displacement map D' from D using the two-pass method suggested by Shade *et al.* [81]. Instead, since our motion fields are all very smooth, we simply dilate them by the extent of the largest possible motion and reverse their sign.

With this notation in place, we can now describe the basic workflow of our system (Figure 6.1), which consists of three steps: *layering* and *matting*, *motion specification* and *editing*, and finally *rendering*.

Layering and matting. The first step, *layering*, is to segment the input image I into layers so that, within each layer, the same motion texture can be applied. For example, for the painting in Figure 6.1(a), we have the following layers: water, sky, bridge and shore, three boats, and the eleven trees in the background (Figure 6.1(b)). To accomplish this, we use the interactive Bayesian matting system described in Section 2.4.2.

Because some layers will be moving, occluded parts of the background might become visible. Hence, after extracting a layer, we use an inpainting algorithm to fill the hole in the background behind the foreground layer. We use the example-based inpainting algorithm of Criminisi *et al.* [24] because of its simplicity and its capacity to handle both linear structures and textured regions. Note that the inpainting algorithm does not have to be perfect, since only pixels near the boundary of the hole are likely to become visible. However, sometimes, we do have to perform manual inpainting to better maintain the layers' structures, for example, for the boats in Figure 6.1. After the background image has been inpainted, we work on this image to extract the next layer. We repeat this process from the closest layer to the furthest layer to generate the desired number of layers. Each layer L_i contains a color image F_i, a matte α_i, and a depth δ_i (an index to indicate compositing order). The depth could be automatically assigned according to the order in which the layers are extracted.

Motion specification and editing. The second component of our system lets the user *specify* and *edit* the motion texture for each layer. Currently, we provide the following motion types: *trees* (swaying), *water* (rippling), *boats* (bobbing), *clouds* (translation), and *stone* (no motion). For each motion type, the user can tune the motion parameters and specify a motion skeleton, where applicable. We describe the motion parameters and skeletons in greater detail for each motion type in Section 6.4.

Since all of the motions we currently support are driven by wind, the user controls a single wind speed and direction, which is shared by all the layers. This allows all the layers to respond to the wind consistently. Our motion synthesis algorithm is fast enough to animate a single layer in

real-time. Hence, the system can provide instant visual feedback to changes in motion parameters, which makes motion editing easier. Each layer L_i has its own motion texture, M_i, as shown in Figure 6.1(c).

Rendering. During the *rendering* process, for each time instance t and layer L_i, a displacement map $M_i(t)$ is synthesized. This displacement map is then applied to F_i and α_i to obtain $L_i(t) = M_i(t) \star L_i(0)$ (Figure 6.1(d)). Notice that the displacement is evaluated as an absolute displacement of the input image $I(0)$ rather than a relative displacement of the previous image $I(t-1)$. In this way, repeated resampling is avoided.

Finally, all the warped layers are composited together from back to front to synthesize the frame at time t, $I(t) = L_1(t) \oslash L_2(t) \oslash \ldots \oslash L_l(t)$, where $\delta_1 \geq \delta_2 \cdots \geq \delta_l$ and $\oslash$ is the *over* operation (Figure 6.1(e)). The user can also specify a time-varying wind field to create a more realistic animation.

6.4 Stochastic motion textures

In this section, we describe our approach to synthesizing the stochastic motion textures that drive the animated image. In Section 6.4.1, we describe the basic principles (spectral methods). We then describe the details of each motion type, i.e., trees (Section 6.4.2), water (Section 6.4.3), bobbing boats (Section 6.4.4), and clouds (Section 6.4.5).

6.4.1 Stochastic modeling of natural phenomena

Many natural motions can be seen as harmonic oscillations [94], and, indeed, hand-crafted superpositions of handfuls of sinusoids have often been used to approximate many natural phenomena for computer graphics. However, this simple approach has some limitations. First of all, it is tedious to tune the parameters to produce the desired effects. Second, it is harder to hook all the motions together in a consistent way, since they lack a physical basis. Lastly, the resulting motions do not look natural since they are strictly periodic — irregularity actually plays a central role in modeling natural phenomena.

One way to add randomness is to introduce a noise field. Introducing this noise directly into the

temporal or spatial domain often leads to erratic and unrealistic simulations of natural phenomena. Instead, we simulate the noise in the frequency domain, and then sculpt the spectral characteristics to match the behaviors of real systems that have intrinsic periodicities and frequency responses. Specific spectrum filters need to be applied to model specific phenomena, leading to so-called *spectral methods*.

The spectral method for synthesizing a stochastic field in general has three steps: (1) generate a complex Gaussian random field in the frequency domain, (2) apply a domain-specific spectrum filter, (3) compute the inverse Fourier transform to synthesize a stochastic field in the time or frequency domain. A nice property of this method is that the synthesized stochastic field can be tiled seamlessly. Hence, we only need to synthesize a patch of reasonable size and tile it to produce a much larger stochastic signal. This tiling approach works reasonably well if the size of the patch is large enough to avoid objectionable repetition.

To realistically model natural phenomena, the filter should be learned from real-world data. For the phenomena we simulate, plants and waves, such experimental data and statistics are available from other fields, e.g., structural engineering and oceanography, and have already been used by the graphics community to create synthetic imagery [56, 82, 92]. We use these methods to synthesize our stochastic motion textures in the following sections.

6.4.2 Plants and trees

The branches and trunks of trees and plants can be modeled as physical systems with mass, damping, and stiffness properties. The driving function that causes branches to sway is typically wind. Our goal is to model the spectral filtering due to the dynamics of the branches applied to the spectrum of the driving wind force.

To model the physics of branches, we take a simplified approach introduced by Sun *et al.* [94]. In particular, each branch is represented as a 2D line segment parameterized by l, which ranges from 0 to 1. This line segment is drawn by the user for each layer. Displacements of the tip of the branch $d_{\mathrm{tip}}(t)$ are taken to be perpendicular to the line segment. Modal analysis indicates that the displacement perpendicular to the line for other points along the branch can be simplified to the

form

$$d(l,t) = \left[\frac{1}{3}l^4 - \frac{4}{3}l^3 + 2l^2\right] d_{\text{tip}}(t)\,. \tag{6.4}$$

We approximate the (scalar) displacement of the tip in the direction of the projected wind force as a dampled harmonic oscillator,

$$\ddot{d_{\text{tip}}}(t) + \gamma \ddot{d_{\text{tip}}}(t) + 4\pi^2 f_o^2 d_{\text{tip}}(t) = r(t)/m\,, \tag{6.5}$$

where m is the mass of the branch, $f_o = k_s/m$ is the natural frequency of the system, $\gamma = k_d/m$ is the velocity damping term [94]. These parameters have a more intuitive meaning than the damping (k_d) and stiffness (k_s) terms found in more traditional formulations. The driving force $r(t)$ is derived from the wind force incident on the branch, as detailed below.

Taking the temporal Fourier transform of this equation gives us

$$D_{\text{tip}}(f) = \frac{R(f)\exp^{-i2\pi\theta}}{2\pi m\left[(f^2 - f_o^2)^2 + \gamma^2 f^2\right]^{-1/2}}\,, \tag{6.6}$$

where $R(f)$ is the Fourier transform of the driving force and f is the temporal frequency. The phase shift θ is given by,

$$\tan\theta = \frac{\gamma f}{f^2 - f_o^2}\,. \tag{6.7}$$

We can see from equation (6.6) that the dynamical system is acting as a non-zero phase spectral filter on the forcing spectrum $R(f)$.

Next, we model the forcing spectrum for wind. Experimental evidence [85, page 55] indicates that the temporal velocity spectrum of wind at a point takes the following form,

$$V(f) \sim \frac{v_{\text{mean}}}{(1 + \kappa f/v_{\text{mean}})^{5/3}}\,, \tag{6.8}$$

where v_{mean} is the mean wind speed and κ is generally a function of altitude which we take to be a constant. We therefore modulate a random Gaussian noise field $G(f)$ with the velocity spectrum to compute the spectrum of a particular (random) wind velocity field,

$$\tilde{V}(f) = V(f)G(f)\,. \tag{6.9}$$

The force due to the wind is generally modeled as a drag force proportional to $\tilde{v}^2(t)$. However, in our experiments, we have found that making the wind force directly proportional to wind velocity produces more pleasing results.

Finally, we assemble equations (6.6)-(6.9) to construct the spectrum of the tip displacement $D_{\text{tip}}(f)$, take the inverse Fourier transform to generate the tip displacement, $d_{\text{tip}}(t)$, and distribute the displacement over the branch according to equation (6.4). The displacement of points in the layer away from the skeleton is obtained by projecting all pixels orthogonally onto the original skeleton and using the corresponding displacement.

The user can control the resulting motion appearance by independently changing the mean wind speed v_{mean} and the natural (oscillatory) frequency f_o, mass m, and velocity damping term γ of each branch.

6.4.3 Water

Water surfaces belong to another class of natural phenomena that exhibit oscillatory responses to natural forces like wind. In this section we describe how one can specify a water plane in a photograph and then define the mapping of water height out of that plane to displacements in image space. We then describe how to synthesize water height variations, again using a spectral method.

The motion skeleton for water is simply a plane; we assume that the image plane is the *x-y plane* and the water surface is parallel to the *x-z plane*. To correctly model the perspective effect, the user roughly specifies where the plane is. This perspective transformation T can be fully specified by the focal length and the tilt of the camera, which can be visualized by drawing the horizon [25].

After specifying the water plane, the water is animated using a time-varying height field $h(q,t)$, where $q = (x_q, y_0, z_q)$ is a point on the water plane. To convert the height field h to the displacement map $M_t(p)$, for each pixel p we first find its corresponding point, $q = (x_q, y_0, z_q) = Tp$, on the water plane. We then add the synthesized height $h(q,t)$ as a vertical displacement, which gives us a point, $q' = (x_q, y_0 + h(q,t), z_q)$. We then project q' back to the image plane to get $p' = T^{-1}q'$. The displacement vector for $M_t(p) = p' - p$ is therefore

$$M_t(p) = T^{-1}[Tp + (0, h(Tp,t), 0)] - p\,. \tag{6.10}$$

The above model is technically correct if we want to displace objects on the surface of the water.

In reality, the shimmer in the water is caused by local changes in surface normals. Therefore, a more physically realistic approach would be to use *normal* mapping, i.e., to convert the surface normals computed from the spatial gradients of $h(q,t)$ into two-dimensional displacements. We have found that our current approach produces pleasing, realistic-looking results but plan to study the more physically-motivated normal reflection model in the future.

To synthesize a time-varying height field for the water, we use the time-varying wind velocity discussed in the previous section to synthesize a height field matching the statistics of real waves, as described by Mastin *et al.* [56].

The spectrum filter we use for waves is the *Phillips spectrum* [102], which is a power spectrum describing the expected square amplitude of waves across all spatial frequencies, $\mathbf{s}$,

$$P_h(\mathbf{s}) \sim \frac{\exp\left[-1/\left(s\eta\right)^2\right]}{s^4} |\hat{\mathbf{s}} \cdot \hat{\mathbf{v}}_{\mathrm{mean}}|^2 , \tag{6.11}$$

where $s = \|\mathbf{s}\|$, $\eta = v_{\mathrm{mean}}^2/g$, g is the gravitational constant and $\hat{\mathbf{s}}$ and $\hat{\mathbf{v}}_{\mathrm{mean}}$ are normalized spatial frequency and wind direction vectors.

The square root of the power spectrum describes the amplitude of wave heights, which we can use to filter a random Gaussian noise field,

$$H_0(\mathbf{s}) = a\sqrt{P_h(\mathbf{s})}G(\mathbf{s}) , \tag{6.12}$$

where a is a constant of proportionality and H_0 is an instance of the height field which we can now animate by introducing time-varying phase. However, waves of different spatial frequencies move at different speeds. The relationship between the spatial frequency and the phase velocity is described by the well-known dispersion relation,

$$w(s) = \sqrt{gs}. \tag{6.13}$$

The time varying height spectrum can thus be expressed as

$$H(\mathbf{s},t) = H_0(\mathbf{s}) \exp\left[iw(s)t\right] + H_0^*(-\mathbf{s}) \exp\{-iw(s)t\} , \tag{6.14}$$

where the H_0^* is the complex conjugate of H_0. We can now compute the height field at time, $h(q,t)$ as the two-dimensional inverse Fourier transform of $H(\mathbf{s},t)$ with respect to spatial frequencies $\mathbf{s}$.

We take the generated height field and tile the water surface using a scale parameter, τ, to control the spatial frequency.

There are thus several motion parameters related to water: wind speed, wind direction, the size of the tile N, the amplitude scale a, and the spatial frequency scale τ. The wind speed and direction are controlled globally for the whole animation. We find that a tile of size $N = 256$ usually produces nice looking results. Users can change a to scale the height of the waves/ripples. Finally, scaling the frequencies by τ changes the scale at which the wave simulation is being done. Simulating at a larger frequency scale gives a rougher look, while a smaller scale gives a smoother look. Hence, we call τ the *roughness* in our user interface.

6.4.4 Boats

We approximate the motion of a bobbing boat by a 2D rigid transformation composed of a translation for heaving and a rotation for rolling. A boat moving on the surface of open water is almost always in oscillatory motion [94]. Hence, the simplest model is to assign a sinusoidal translation and a sinusoidal rotation. However, this often looks fake. In principle, we could build a simple model for the boat, convert the height field of water into a force interacting with the hull and solve the dynamics equation for the boat to estimate its displacement. However, since our goal is only to synthesize an approximate solution, we directly use the height field of the wave to move the boat, as follows.

We let the user select a line close to the bottom of the boat in the image. Then, we sample several pixels along the line. For each pixel p_i, we look up its corresponding displaced pixel $p_i' = T^{-1}[Tp + (0, h(Tp, t), 0)]$ as described in the previous section. Finally, we use linear regression to fit a line through these p_i'. The position and orientation of the fitted line then determine the heaving and rolling of the boat.

6.4.5 Clouds

Another common element for scenic pictures is clouds. In principle, clouds could also be modeled as a stochastic process. However, we need the stochastic process to match the clouds in the image at some point, which is harder. Since clouds often move very slowly and their motion does not attract

(a) Japanese temple (b) Harbor (c) The Boat Studio (d) Sunflowers

Figure 6.2: Sample input images we animated using our technique. The first two pictures are photographs of a Japanese Temple (a) and a harbor (b). The rightmost two paintings are Claude Monet's painting *Le bateau atelier (The Boat Studio)* (c) and Van Gogh's *Sunflowers* (d).

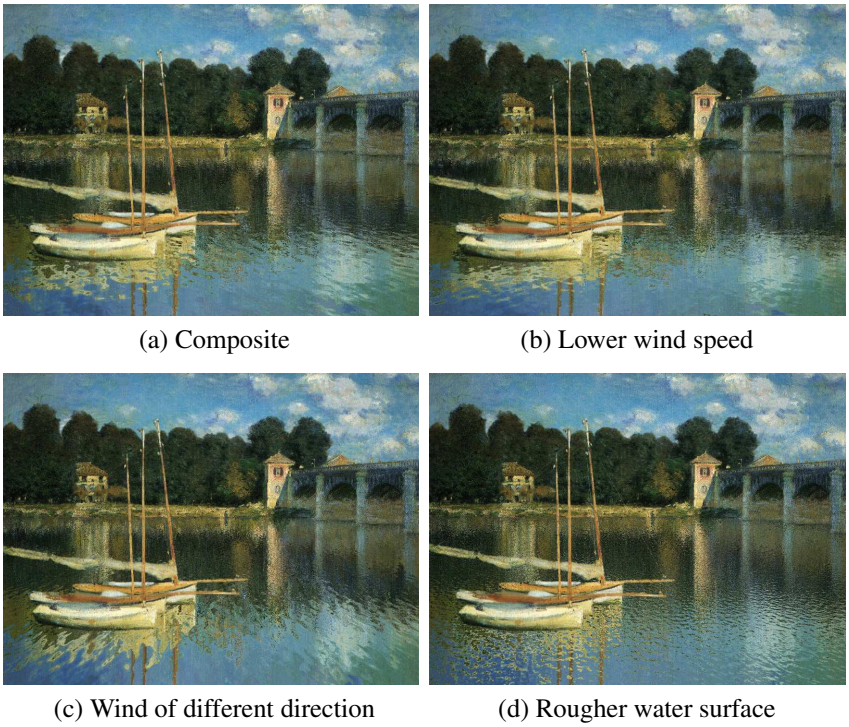
too much attention, we simply assign a translational motion field to them. As with water, we have to extend the clouds outside the image frame in some way, since their motion in one direction will create holes that we have to fill.

6.5 Results

We have applied our system to several photographs and famous paintings (Figure 6.1(a) and 6.2). Here, we summarize some of the results and discuss some details in creating them.

It takes from several minutes to several hours to animate a picture depending on the complexity of the input picture. Matting and inpainting is the most time-consuming part in our system and usually consumes more than 95 percent of the overall time for animating a picture. For example, for the picture in Figure 6.2(a), because of the complicated structure, it is difficult to specify the trimap for the branches on the left. Since the background is smooth in color, we end up using a garbage matte to take out the tree and use inpainting first to estimate the background. Taking advantage of the known background, the trimap can be painted fairly roughly and matting can be done much faster as explained in Section 3.3.3. We model a total of 10 branches on the left and the right. We use a small wave amplitude and high roughness to give the ripples a fine-grained look. For the harbor picture in Figure 6.2(b), we animate the water and have nine boats swaying with the water. The cloud and sky are animated using a translational motion field.

Our technique actually works even better with paintings, perhaps because in this situation we



(a) Composite (b) Lower wind speed

(c) Wind of different direction (d) Rougher water surface

Figure 6.3: An example of controlling the appearance by tuning physical parameters. We can control the appearance of the water surface by adjusting some physical parameters such as wind speed. We show one of the composites (a) as the reference, in which the wind blow at 5 m/s in z direction. We decrease the wind speed to 3 m/s (b) and change the wind direction to be along y axis (c). Finally, we change the scale of the simulation to render water with finer ripples (d).

are less sensitive to anything that does not look perfectly realistic. Figure 6.1(a) and 6.2(c,d) show three paintings we have animated. For Claude Monet's painting in Figure 6.2(c), we animate the water with lower amplitude roughness to keep the strokes intact. We also let the boat sway with the water. The other Monet painting shown in Figure 6.1(a) is a more complex example, with more than 20 layers. We use this example to demonstrate that we can change the appearance of the water by controlling the physical parameters. In Figure 6.3, we show look of the water under different wind speeds, directions, and simulation scales.

For Van Gogh's sunflower painting (Figure 6.2(d)), we use our stochastic wind model to animate 40 plant layers. With a simple sinusoidal model, the viewer usually can quickly figure out that the plants swing in synchrony and the motion looses a lot of its interest. With the stochastic wind model,

the flowers' motions de-correlate in phase and the resulting animation is more appealing.

Summary

In this chapter, we have applied Bayesian matting algorithm to an intriguing problem, animating still pictures for a limited class of scenes—in particular, scenes that contain passive elements responding to natural forces in an oscillatory fashion. In conclusion, we are pleased by the ease with which it is possible to breathe life into pictures, based on our Bayesian image matting algorithm, inpainting, and stochastic modeling algorithms.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this dissertation, we have introduced a novel set of composting models and matting methods which expand the working range of matting and composting in two directions: eliminating the need for special setups for acquiring foreground plates in traditional matting, and moving beyond the limitations of the traditional compositing model. We also present a novel application of matting and compositing to animate still pictures. In particular, this dissertation makes the following contributions:

- **A probabilistic framework for natural image matting.** The Bayesian framework introduced in Chapter 2 provides a more principled solution and better results than previous methods. Solving the natural image matting problem within a probabilistic framework also leads us to a more general problem setting, Markov Random Fields, where we have proposed a regularization approach to incorporate smoothness priors into solutions.

- **A system for video matting.** We have identified and adapted a set of existing algorithms and devised an overall framework for applying them together to solve the video matting problem with natural backgrounds. This framework not only reduces the user's burden of specifying trimaps but also encourages temporal consistency in the resulting mattes.

- **A novel matting algorithm for smoke.** With the assumptions of a known background and a constant foreground color, we have developed a simple but effective method to extract mattes for flowing, participating media such as smoke.

- **A framework for shadow matting and compositing.** Based on a simplified lighting framework, we have derived a shadow compositing equation for a single key light. We have also introduced a set of matting methods to extract shadow mattes and acquire photometric and geometric properties of the background for making realistic shadow composites.

- **A more accurate model and method for environment matting.** We have improved the quality of environment matting by proposing a parametric model that is simple and compact and approximates real weighting functions expected for many materials. We have also designed a practical matting algorithm for estimating the parameters of this model.

- **An application of matting and compositing to animating pictures.** We have applied our Bayesian matting algorithm to a novel application, animating pictures. With the help of the matting algorithm, we are able to synthesize realistic video textures from still photographs and paintings by augmenting layers with time-varying motion fields.

To sum up, we complete Table 1.1 by filling in our work and follow-on work by others. The completed Table 7.1 shows a comprehensive categorization of existing compositing models and matting algorithms. As mentioned in Section 1.3, there is no clear winner in this table. Matting algorithms often involve making decisions on assumptions about the properties of the foreground elements, the quality of mattes, the illumination methods, and the number of images required. These decisions span a rich and interesting design space. We feel we have only just begun to explore this space. Thus, there are a number of research opportunities for the future.

An improved Bayesian matting algorithm. Although our Bayesian matting algorithm offers improvements over previous methods, it has several drawbacks. First, it is not fast enough to provide instant feedback to the user's interaction. This makes it difficult to edit trimaps to obtain satisfactory mattes. The current computation bottleneck is sample gathering. A possible way to speed up this process is to segment the input image into many regions of similar colors and to pre-compute the color statistics for each region. Second, the onion peel order may not be the best filling order because it does not take the image content into account. For a similar problem, image inpainting, Criminisi *et al.*show that the filling order is crucial [24]. By appropriately arranging the filling order, results can be dramatically improved. The filling order could be biased toward the pixels that are on the continuation of edges and whose estimates potentially have higher confidence. For matting, we could give higher priority to those pixels with higher image gradients, more computed pixels in their neighborhoods and neighbors that are classified as pure foreground or background.

Table 7.1: A summary of matting and composting techniques.[a] We use colors to distinguish our work from others; dark red for our own work; dark blue for the work before ours; and dark green for the work after ours.

			Traditional matting and compositing	Shadow matting and compositing	Environment matting and compositing
lighting phenomena			partial coverage transparency	shadows	refraction reflection
compositing			$C=\alpha F+(1-\alpha)B$ Porter & Duff [69]	$C'=\beta L'+(1-\beta)S'$ [Section 4.3]	$C=F+\int W(\boldsymbol{x})B(\boldsymbol{x})d\boldsymbol{x}$ Zongker *et al.* [118][b]
matting	passive	single -frame	depth/thermo keying rotoscoping Ruzon-Tomasi [76] Chapter 2 Poisson matting [93]	faux shadow	
		multiple -frame[c]	Chapter 3 Wexler [112]	shadow matting [Section 4.3]	image-Based EM [113]
	active	single -frame	blue screen matting difference matting[d] dual-film [6, 29][e]	blue screen matting	real-time EM [23]
		multiple -frame	$O(1)$: triangulation [88]	$O(k)$: shadow scanning [Section 4.5]	$O(\log k)$: Zongker [118] $O(k)$: Chapter 5 $O(k^2)$: wavelet EM [65] [f]
matte	foreground		F, α	β	$F, \alpha, \boldsymbol{W}$ [g]
	background		B	L, S, w	B

[a] This table is by no means complete; only representative work is listed. For matting algorithms, this table is more a categorization than a comparison. Because of the assumptions they make, these techniques actually solve related but somewhat different problems.

[b] The original environment matting equation proposed by Zongker *et al.* is slightly less general than the one listed here.

[c] The algorithms of this category take a sequence of images as input and output either a single matte (image-based EM) or a sequence of mattes (e.g., video matting in Chapter 3). Because such algorithms usually take advantage of the temporal coherence within the input sequence, they do not work for a single image.

[d] While the background is not "controlled" in difference matting, the camera path often has to be controlled for the ease of obtaining clean plates. Hence, we count it as an active approach.

[e] This approach essentially requires two frames, but they are taken simultaneously.

[f] Wavelet EM algorithm is adaptive. The actual number of photographs depends on the reflective/refractive properties of the objects. However, theoretically, it requires $O(k^2)$ images for the worst case when the object is extremely specular.

[g] The actual information stored in the environment matte $\boldsymbol{W}$ depends on the particular form of the weighting function.

A more principled solution to natural image matting. We laid out a MRF framework for the matting problem in Section 2.5. However, to make the problem more tractable, we only solved a simplified version. There are several possible improvements. First, instead of assuming a quadratic function, we can allow a more general evidence energy function in equation (2.13) and find the optimum by using Belief Propagation or Graph Cut algorithms. Second, in Section 2.5, we use a smoothness prior as the compatibility measurement. It is possible to build the priors by studying the statistics of ground truth alpha mattes [3]. We can even combine Poisson matting and Bayesian matting by encoding the AutoKey constraint (equation (1.3)) into the prior. Thus, for every pixel, we use the discrepancy between the matte gradient and the image gradient as the compatibility measurement, so that equation (2.12) becomes

$$L(\alpha, C) = \sum_{p \in \Omega} E_{\Phi}(\alpha(p); p, C) + \lambda \sum_{p \in \Omega} \left(\nabla \alpha(p) - \frac{1}{\overline{F}(p) - \overline{B}(p)} \nabla C(p) \right)^2 . \tag{7.1}$$

Note that, because F and B in equation (1.3) are not known yet, we use the gathered $\overline{F}$ and $\overline{B}$ as their approximations. Moreover, F, B and C are vectors and we can use their grayscale versions for this calculation. Finally, our MRF formulation (equation (2.10)) only solves for the alpha matte. A better solution should solve for F, B and α simultaneously and exploit their individual priors.

A more efficient interactive segmentation method. The quality of the input trimap is critical to the success of the Bayesian matting algorithm.[1] A considerable amount of user interaction is required to specify a good trimap and thus usually takes several minutes to accomplish. Hence, efficient methods for specifying trimaps would make the matting process much easier. Recently, several efficient foreground extraction methods have been proposed [52, 75, 100, 101]. With only a few strokes, these methods are capable of isolating a foreground element from its background. These methods, however, often give a hard segmentation and synthesize a trimap by dilating the object boundary. This makes it difficult to apply these methods to objects with wide unknown areas. In addition, these methods actually perform foreground segmentation with a statistical flavor similar to Bayesian matting. Hence, it should be possible to solve the matting problem with several strokes in a principled way by combining the strengths of both Bayesian matting and these sketch-based foreground extraction methods.

[1] This problem could potentially be alleviated by carefully designing the filling order as mentioned earlier.

An improved video matting method. The framework we have developed for video matting relies on accurate optical flow estimation. However, existing optical flow algorithms do not consider the color blending effects between layers. An optical flow algorithm that incorporates the notion of blended, complex foreground and background layers could improve flow estimates and allow us to accumulate foreground and background color distributions *temporally* as well as spatially.

Our current framework is designed for extracting a single foreground layer. Without background estimation, this framework can be directly applied to extract multiple layers from the closest to the furthest. However, we do not expect such a naïve approach to work well because the motions of multiple dynamic layers will degrade the performance of optical flow algorithms. To improve the results, we have to estimate the background. If the motions are somewhat rigid, parametric motion models such as affine transformations could be used to robustly predict the underlying background when the foreground element crosses other moving elements.

A less restrictive shadow matting and compositing method. Our present shadow matting and composting method imposes a set of restrictions. Nevertheless, there are several possible ways to extend the operating range of the shadow matting and compositing method. For example, the planarity constraints for the source background could be relaxed. For a source scene that is not fully planar, but has at least a planar segment, we could simply shadow scan the source scene to get its displacement map relative to its own ground plane, and use this to produce an unwarped (planar) shadow matte. It would also be useful to have interactive editing tools for adjusting shadow direction and shape. Such tools could extend the operating range of our technique to cases in which the lights or reference planes are somewhat misaligned between the source and target scene.

More data points in the environment matting design space. From Table 7.1, an obvious missing data point for environment matting is a single-frame method for natural backgrounds. Another general area of future work is to develop more sophisticated tools for extracting environment mattes of similar quality but requiring fewer images or better quality with the same number of images. For example, for real-time capture, we could use a more principled Bayesian approach to fitting matte parameters given noisy image streams. For higher accuracy methods, we could use other stimulus functions such as Gabor functions to achieve a better matte-quality-to-number-of-image

ratio. Another promising direction is to employ the Fourier projection slice theorem [48] to capture an arbitrary weighting function with $O(k \log k)$ images.

In this dissertation, we explore creative possibilities in the rich space of matting and compositing. We believe that our work has made matting and compositing more powerful tools and are glad to see research on matting and compositing starts to flourish.

BIBLIOGRAPHY

[1] Aseem Agarwala, Aaron Hertzmann, Steve Seitz, and David H. Salesin. Keyframe-based tracking for rotoscoping and animation. *ACM Transactions on Graphics*, 23(3), 2004. Special issue: Proceedings of ACM SIGGRAPH 2004. To appear.

[2] Masakastsu Aoki, Mikio Shinya, Ken Tsutsuguchi, and Naoya Kotani. Dynamic texture: Physically-based 2D animation. In *SIGGRAPH 1999 Conference Sketches and Applications*, page 239, 1999.

[3] Nicholas Apostoloff and Andrew W. Fitzgibbon. Bayesian video matting using learnt image priors. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2004)*, 2004. To appear.

[4] William A. Barrett and Alan S. Cheney. Object-based image editing. *ACM Transactions on Graphics*, 21(3):777–784, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[5] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[6] Moshe Ben-Ezra. Segmentation with invisible keying signal. In *Proceedings of IEEE —Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, pages 32–37, 2000.

[7] Arie Berman, Arpag Dadourian, and Paul Vlahos. Method for removing from an image the background surrounding a selected object. U.S. Patent 6,134,346, 2000.

[8] Arie Berman, Paul Vlahos, and Arpag Dadourian. Comprehensive method for removing from an image the background surrounding a selected object. U.S. Patent 6,134,345, 2000.

[9] Marcelo Bertalmio, Guillermo Sapiro, Vicent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of ACM SIGGRAPH 2000*, pages 417–424, 2000.

[10] Paul Besl. Active optical range imaging sensors. In Jorge L.C. Sanz, editor, *Advances in Machine Vision*, chapter 1, pages 1–63. Springer-Verlag, 1989.

[11] David Biedny, Bert Monroy, and Nathan Moody. *Photoshop Channel Chops*. New Riders Publishing, 1998.

[12] Michael J. Black and P. Anandan. The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[13] Andrew Blake, Carsten Rother, Matthew Brown, Patrick Perez, and Philip Torr. Interactive image segmentation using an adaptive GMMRF model. In *Proceedings of European Conference on Computer Vision (ECCV 2004)*, pages 418–441, 2004.

[14] James F. Blinn. Compositing, part 1: Theory. *IEEE Computer Graphics & Applications*, 14(5):83–87, November 1994.

[15] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–546, 1976.

[16] Jean-Yves Bouguet and Pietro Perona. 3D photography on your desk. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 1998)*, pages 43–50, 1998.

[17] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, November 2001.

[18] Ron Brinkman. *The Art and Science of Digital Compositing*. Morgan Kaufman, 1999.

[19] Tongbo Chen, Yan Wang, Volker Schillings, and Christoph Meinel. Grayscale image matting and colorization. In *Proceedings of Asian Conference on Computer Vision (ACCV 2004)*, pages 1164–1169, 2004.

[20] Yung-Yu Chuang, Aseem Agarwala, Brian Curless, David H. Salesin, and Richard Szeliski. Video matting of complex scenes. *ACM Transactions on Graphics*, 21(3):243–248, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[21] Yung-Yu Chuang, Brian Curless, David H. Salesin, and Richard Szeliski. A Bayesian approach to digital matting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume II, pages 264–271, 2001.

[22] Yung-Yu Chuang, Dan B Goldman, Brian Curless, David H. Salesin, and Richard Szeliski. Shadow matting and compositing. *ACM Transactions on Graphics*, 22(3):494–500, 2003. Special issue: Proceedings of ACM SIGGRAPH 2003.

[23] Yung-Yu Chuang, Douglas E. Zongker, Joel Hindorff, Brian Curless, David H. Salesin, and Richard Szeliski. Environment matting extensions: Towards higher accuracy and real-time capture. In *Proceedings of ACM SIGGRAPH 2000*, pages 121–130, 2000.

[24] Antonio Criminisi, Patrick Perez, and Kentaro Toyama. Object removal by exemplar-based inpainting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume II, pages 721–728, 2003.

[25] Antonio Criminisi, Ian D. Reid, and Andrew Zisserman. Single view metrology. *International Journal of Computer Vision*, 40(2):123–148, 2000.

[26] Brian Curless and Marc Levoy. Better optical triangulation through spacetime analysis. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 1995)*, pages 987–994, 1995.

[27] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of ACM SIGGRAPH 1998*, pages 189–198, 1998.

[28] Paul Debevec, Tim Hawkins, Chris Tchou, Haarm-Pieter Duiker, Westley Sarokin, and Mark Sagar. Acquiring the reflectance field of a human face. In *Proceedings of ACM SIGGRAPH 2000*, pages 145–156, 2000.

[29] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. A lighting reproduction approach to live-action compositing. *ACM Transactions on Graphics*, 21(3):547–556, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[30] Paul E. Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of ACM SIGGRAPH 1997*, pages 369–378, 1997.

[31] Iddo Drori, Daniel Cohen-Or, and Hezy Yeshurun. Fragment-based image completion. *ACM Transactions on Graphics*, 22(3):303–312, 2003. Special issue: Proceedings of ACM SIGGRAPH 2003.

[32] Graham D. Finlayson, Steven D. Hordley, and Mark S. Drew. Removing shadows from images. In *Proceedings of European Conference on Computer Vision (ECCV 2002)*, pages 823–836, 2002.

[33] William T. Freeman, Edward H. Adelson, and David J. Heeger. Motion without movement. In *Computer Graphics (Proceedings of ACM SIGGRAPH 1990)*, pages 27–30, 1991.

[34] William T. Freeman, Egon C. Pasztor, and Owen T. Carmichael. Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47, 2000.

[35] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. In *Proceedings of ACM SIGGRAPH 1996*, pages 43–54, 1996.

[36] David Griffiths. Lake java applet, 1997.

[37] Chuang Gu and Ming-Chieh Lee. Semiautomatic segmentation and tracking of semantic video objects. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(5):572–584, 1998.

[38] Ronen Gvili, Amir Kaplan, Eyal Ofek, and Giora Yahav. Depth keying. In *Proceedings of SPIE Electronic Imaging 2003 Conference*, 2003.

[39] Terry Hathaway, Dave Bowers, Don Pease, and Siegfried Wendel. `http://www.mechanicalmusicpress.com/history/pianella/p40.htm`.

[40] Peter Hillman, John Hannah, and David Renshaw. Alpha channel estimation in high resolution images and image sequences. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume I, pages 1063–1068, 2001.

[41] 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 ACM SIGGRAPH 1997*, pages 225–232, 1997.

[42] Jiaya Jia and Chi-Keung Tang. Image repairing: Robust image synthesis by adaptive nd tensor voting. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, volume I, pages 643–650, 2003.

[43] Nebojsa Jojic and Brendan J. Frey. Learning flexible sprites in video layers. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, volume II, pages 199–206, 2001.

[44] Takeo Kanade, Andrew Gruss, and L. Richard Carley. A very fast VLSI rangefinder. In *IEEE International Conference on Robotics and Automation*, volume 39, pages 1322–1329, April 1991.

[45] Rakesh Kumar, P. Anandan, and Keith Hanna. Direct recovery of shape from multiple views: A parallax based approach. In *Proceedings of International Conference on Pattern Recognition (ICPR 1994)*, pages 685–688, 1994.

[46] Ming-Chieh Lee, Wei ge Chen, Chih-lung Bruce Lin, Chuang Gu, Tomislav Markoc, Steven I. Zabinsky, and Richard Szeliski. A layered video object coding system using sprite

and affine motion model. *IEEE Transactions on Circuits and Systems for Video Technology*, 7(1):130–145, 1997.

[47] Marc Levoy. Merging and transformation of raster images for cartoon animation. `http://graphics.stanford.edu/papers/merging-sig81/`.

[48] Marc Levoy. Volume rendering using the fourier projection-slice theorem. In *Proceedings of the conference on Graphics Interface '92*, pages 61–69, 1992.

[49] Marc Levoy and Pat Hanrahan. Light field rendering. In *Proceedings of ACM SIGGRAPH 1996*, pages 31–42, 1996.

[50] Stan Z. Li. *Markov Random Field Modeling in Image Analysis*. Springer-Verlag, 2001.

[51] Yan Li, Tianshu Wang, and Heung-Yeung Shum. Motion texture: a two-level statistical model for character motion synthesis. *ACM Transactions on Graphics*, 21(3):465–472, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[52] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. *ACM Transactions on Graphics*, 23(3), 2004. Special issue: Proceedings of ACM SIGGRAPH 2004. To appear.

[53] Peter Litwinowicz and Lance Williams. Animating images with drawings. In *Proceedings of ACM SIGGRAPH 1994*, pages 409–412, 1994.

[54] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Computer Graphics (Proceedings of ACM SIGGRAPH 1987)*, pages 163–169, 1987.

[55] Vincent Masselus, Pieter Peers, Philip Dutré, and Yves D. Willems. Relighting with 4D incident light fields. *ACM Transactions on Graphics*, 22(3):413–620, 2003. Special issue: Proceedings of ACM SIGGRAPH 2003.

[56] Gary A. Mastin, Peter A. Watterberg, and John F. Mareda. Fourier synthesis of ocean scenes. *IEEE Computer Graphics & Applications*, 7(3):16–23, March 1987.

[57] Yasuyuki Matsushita, Sing Bing Kang, Stephen Lin, Heung-Yeung Shum, and Xin Tong. Lighting interpolation by shadow morphing using intrinsic lumigraphs. In *Proceedings of Pacific Conference on Computer Graphics and Applications 2002*, pages 58–65, 2002.

[58] Wojciech Matusik, Hanspeter Pfister, Addy Ngan, Paul Beardsley, Remo Ziegler, and Leonard McMillan. Image-based 3D photography using opacity hulls. *ACM Transactions on Graphics*, 21(3):437–437, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[59] Wojciech Matusik, Hanspeter Pfister, Remo Ziegler, Addy Ngan, and Leonard McMillan. Acquisition and rendering of transparent and refractive objects. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 267–278, 2002.

[60] Yasushi Mishima. Soft edge chroma-key generation based upon hexoctahedral color space. U.S. Patent 5,355,174, 1993.

[61] Tomoo Mitsunaga, Taku Yokoyama, and Takashi Totsuka. Autokey: Human assisted key extraction. In *Proceedings of ACM SIGGRAPH 1995*, pages 265–272, 1995.

[62] Eric N. Mortensen and William A. Barrett. Intelligent scissors for image composition. In *Proceedings of ACM SIGGRAPH 1995*, pages 191–198, 1995.

[63] 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, 2001.

[64] Michael T. Orchard and Charles A. Bouman. Color quantization of images. *IEEE Transactions on Signal Processing*, 39(12):2677–2690, December 1991.

[65] Pieter Peers and Philip Dutré. Wavelet environment matting. In *Proceedings of the 14th Eurographics Symposium on Rendering*, pages 157–166, 2003.

[66] Fabio Pellacini, Parag Tole, and Donald P. Greenberg. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics*, 21(3):563–566, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[67] Pietro Perona and Jitendra Malik. Scale space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[68] Lena Petrovic, Brian Fujito, Lance Williams, and Adam Finkelstein. Shadows for cel animation. In *Proceedings of ACM SIGGRAPH 2000*, pages 511–516, 2000.

[69] Thomas Porter and Tom Duff. Compositing digital images. In *Computer Graphics (Proceedings of ACM SIGGRAPH 1984)*, pages 253–259, 1984.

[70] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing (2nd ed.)*. Cambridge University Press, 1992.

[71] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: texturing and synthesis. *ACM Transactions on Graphics*, 21(3):501–508, 2002. Special issue: Proceedings of ACM SIGGRAPH 2002.

[72] Richard J. Qian and M. Ibrahim Sezan. Video background replacement without a blue screen. In *Proceedings of International Conference on Image Processing (ICIP 1999)*, volume 4, pages 143–146, 1999.

[73] Alex Reche, Ignacio Martin, and George Drettakis. Reconstruction and interactive rendering of trees from photographs. *ACM Transactions on Graphics*, 23(3), 2004. Special issue: Proceedings of ACM SIGGRAPH 2004. To appear.

[74] Richard Rickitt. *Special Effects: the history and technique*. Virgin Books, 2000.

[75] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut — interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3), 2004. Special issue: Proceedings of ACM SIGGRAPH 2004. To appear.

[76] Mark A. Ruzon and Carlo Tomasi. Alpha estimation in natural images. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, pages 18–25, 2000.

[77] Kosuke Sato and Seiji Inokuchi. Three-dimensional surface measurement by space encoding range imaging. *Journal of Robotic Systems*, 2(1):27–39, 1985.

[78] Silvio Savarese, Holly Rushmeier, Fausto Bernardini, and Pietro Perona. Shadow carving. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 2001)*, pages 190–197, 2001.

[79] Arno Schödl, Richard Szeliski, David H. Salesin, and Irfan Essa. Video textures. In *Proceedings of ACM SIGGRAPH 2000*, pages 489–498, 2000.

[80] Jeremy A. Selan. Merging live video with synthetic imagery. Master's thesis, Cornell University, 2003.

[81] Jonathan Shade, Steve Gortler, Li-wei He, and Richard Szeliski. Layered depth images. In *Proceedings of ACM SIGGRAPH 1998*, pages 231–242, 1998.

[82] Mikio Shinya and Alain Fournier. Stochastic motion – motion under the influence of wind. *Computer Graphics Forum*, 11(3):119–128, 1992.

[83] Mikio Shinya, Takeaki Mori, and Noriyoshi Osumi. Periodic motion synthesis and fourier compression. *The Journal of Visualization and Computer Animation*, 9(3):95–107, 1998.

[84] 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 Transactions on Graphics*, 23(2):143–162, 2004.

[85] Emil Simiu and Robert H. Scanlan. *Wind Effects on Structures*. John Wiley & Sons, 1986.

[86] Alvy Ray Smith. Alpha and the history of digital compositing. Technical Report Microsoft Technical Memo 7, August 1995.

[87] Alvy Ray Smith. Image compositing fundamentals. Technical Report Microsoft Technical Memo 4, August 1995.

[88] Alvy Ray Smith and James F. Blinn. Blue screen matting. In *Proceedings of ACM SIGGRAPH 1996*, pages 259–268, 1996.

[89] Stefano Soatto, Gianfranco Doretto, and Ying Nian Wu. Dynamic textures. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 2001)*, pages 439–446, 2001.

[90] Jos Stam. *Multi-Scale Stochastic Modelling of Complex Natural Phenomena*. PhD thesis, Department of Computer Science, University of Toronto, 1995.

[91] Jos Stam. Stochastic dynamics: Simulating the effects of turbulence on flexible structures. *Computer Graphics Forum*, 16(3):159–164, 1997.

[92] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *Proceedings of ACM SIGGRAPH 1993*, pages 369–376, 1993.

[93] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Transactions on Graphics*, 23(3), 2004. Special issue: Proceedings of ACM SIGGRAPH 2004. To appear.

[94] Meng Sun, Allan D. Jepson, and Eugene Fiume. Video input driven animation (VIDA). In *Proceedings of IEEE International Conference on Computer Vision (ICCV 2003)*, pages 96–103, 2003.

[95] Shijun Sun, David Haynor, and Yongmin Kim. Motion estimation based on optical flow with adaptive gradients. In *Proceedings of International Conference on Image Processing (ICIP 2000)*, volume I, pages 852–855, 2000.

[96] Richard Szeliski, Shai Avidan, and P. Anandan. Layer extraction from multiple images containing reflections and transparency. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000)*, pages 246–253, 2000.

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

[98] Martin Szummer and Rosalind W. Picard. Temporal texture modeling. In *IEEE International Conference on Image Processing (ICIP 1996)*, volume 3, pages 823–826, 1996.

[99] Kar-Han Tan. *Visual Objects and Environments: Capture, Extraction, and Representation*. PhD thesis, University of Illinois at Urbana-Champaign, 2003.

[100] Kar-Han Tan and Narendra Ahuja. Representation for image structure and its application to object selection using freehand sketches. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, pages 677–683, 2001.

[101] Kar-Han Tan and Narendra Ahuja. Selecting objects with freehand sketches. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 2001)*, pages 337–344, 2001.

[102] Jerry Tessendorf. Simulating ocean water. *SIGGRAPH course notes*, 2001.

[103] Philip H. S. Torr, Richard Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(3):297–303, March 2001.

[104] Bruce A. Wallace. Merging and transformation of raster images for cartoon animation. In *Computer Graphics (Proceedings of ACM SIGGRAPH 1981)*, pages 253–262, 1981.

[105] Bruce A. Wallace. Automated production techniques in cartoon animation. Master's thesis, Cornell University, 1982.

[106] John Y. A. Wang and Edward H. Adelson. Representing moving images with layers. *IEEE Transactions on Image Processing*, 3(5):625–638, 1994.

[107] Yizhou Wang and Song Chun Zhu. Modeling textured motion: Particle, wave and sketch. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 2003)*, pages 213–220, 2003.

[108] Gregory J. Ward. Measuring and modeling anisotropic reflection. In *Computer Graphics (Proceedings of ACM SIGGRAPH 1992)*, pages 265–272, 1992.

[109] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of ACM SIGGRAPH 2000*, pages 479–488, 2000.

[110] Yair Weiss. Deriving intrinsic images from image sequences. In *Proceedings of IEEE International Conference on Computer Vision (ICCV 2001)*, pages 68–75, 2001.

[111] Yair Weiss and Edward H. Adelson. Perceptually organized EM: A framework for motion segmentation that combines information about form and motion. Technical Report MIT Media Lab Perceptual Computing Section TR 315, 1994.

[112] Yonatan Wexler, Andrew W. Fitzgibbon, and Andrew Zisserman. Bayesian estimation of layers from multiple images. In *Proceedings of European Conference on Computer Vision (ECCV 2002)*, pages 487–501, 2002.

[113] Yonatan Wexler, Andrew. W. Fitzgibbon, and Andrew. Zisserman. Image-based environment matting. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 279–290, 2002.

[114] Daniel N. Wood, Daniel I. Azuma, Ken Aldinger, Brian Curless, Brian Curless, Tom Duchamp, David H. Salesin, and Werner. Surface light fields for 3D photography. In *Proceedings of ACM SIGGRAPH 2000*, pages 287–296, 2000.

[115] Steve Wright. *Digital Compositing for Film and Video*. Focal Press, 2001.

[116] Shuntaro Yamazaki, Ryusuke Sagawa, Hiroshi Kawasaki, Katsushi Ikeuchi, and Masao Sakauchi. Microfacet billboarding. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 169–180, 2002.

[117] Kazutaka Yasuda, Takeshi Naemura, and Hiroshi Harashima. Thermo-key: Human region segmentation from video. *IEEE Computer Graphics & Applications*, 24(1):26–30, 2004.

[118] Douglas E. Zongker, Dawn M. Werner, Brian Curless, and David H. Salesin. Environment matting and compositing. In *Proceedings of ACM SIGGRAPH 1999*, pages 205–214, 1999.

Appendix A

ANALYSIS OF THE SHADOW COMPOSITING EQUATION

In Section 4.3, we devised our shadow compositing model based on the occlusion of a single light source. Here, we use a more general lighting framework to analyze the conditions in which our model is theoretically correct. For each wavelength, the composite color C at a pixel p can be written in the form of the rendering equation,

$$C = \int \rho_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i, \tag{A.1}$$

where ω_o is the fixed viewing direction for pixel p, $\rho_{\omega_o}(\omega_i)$ is the 2D BRDF for the viewing direction ω_o, and $\beta(\omega_i)$ represents the visibility of the incoming radiance $l(\omega_i)$ at the incident direction ω_i, and the domain of integration is the hemisphere above the visible scene point. The lit image L is obtained when $\beta(\omega_i) = 1$. Hence,

$$L = \int \rho_{\omega_o}(\omega_i)l(\omega_i)d\omega_i, \tag{A.2}$$

For compositing, we mean to keep the same lighting condition but replace the background with a scene of the *same geometry* with a different material. That is, we only intend to replace ρ with ρ' during compositing. Therefore, the composite color C^* for the new scene is

$$C^* = \int \rho'_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i. \tag{A.3}$$

Note that, because the scene geometry is assumed to be the same in this analysis, the domains of integration in equations (A.1) and (A.3), i.e., the hemispheres above the visible scene points, are the same. Ideally, we'd like to record the whole lighting environment so that we can correctly predict C^* from C. However, this is difficult if not impossible. Instead, during matting, we summarize the lighting condition with a number β representing the ratio of the incoming radiance to the maximum radiance. To simplify the discussion, we assume $S = 0$ in equation (4.2) and $S' = 0$ in equation (4.5) and these equations become

$$C = \beta L, \tag{A.4}$$

$$C' = \beta L'. \tag{A.5}$$

Since S and S' only represent the baseline radiance that we record in the shadow images for both scenes, it won't affect our analysis. Under this assumption, dividing L on both sides of equation (A.4) and replacing C and L with equation (A.1) and equation (A.2), we have

$$\beta = \frac{C}{L} = \frac{\int \rho_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i}{\int \rho_{\omega_o}(\omega_i)l(\omega_i)d\omega_i}. \tag{A.6}$$

Substituting equation (A.6) for β in equation (A.5), we obtain

$$C' = \beta L' = \frac{\int \rho_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i}{\int \rho_{\omega_o}(\omega_i)l(\omega_i)d\omega_i} \int \rho'_{\omega_o}(\omega_i)l(\omega_i)d\omega_i. \tag{A.7}$$

Ideally, the composite color C' using the shadow composting equation and the estimated β will equal C^*, but this is not true in general. Nonetheless, we can show that the composite color C' does equal C^* for the following cases.

(1) **Single point light source**. For a point light source of intensity l_p at direction ω_p, the incoming radiance $l(\omega_i)$ equals $l_p\delta(\omega_i - \omega_p)$ and

$$C' = \frac{\rho_{\omega_o}(\omega_p)\beta(\omega_p)l_p}{\rho_{\omega_o}(\omega_p)l_p}\rho'_{\omega_o}(\omega_p)l_p = \rho'_{\omega_o}(\omega_p)\beta(\omega_p)l_p = C^*. \tag{A.8}$$

This can be extended to the case when ρ_{ω_o} and ρ'_{ω_o} are both approximately constant within a solid angel subtended by an area light source.

(2) **Lambertian materials**. For Lambertian materials, the BRDF is constant: $\rho_{\omega_o}(\omega_i) = \rho$ and $\rho'_{\omega_o}(\omega_i) = \rho'$. Therefore,

$$\begin{aligned} C' &= \frac{\rho \int \beta(\omega_i)l(\omega_i)d\omega_i}{\rho \int l(\omega_i)d\omega_i}\rho' \int l(\omega_i)d\omega_i \\ &= \rho' \int \beta(\omega_i)l(\omega_i)d\omega_i = C^*. \end{aligned} \tag{A.9}$$

Although pure Lambertian materials are rare, most materials have primarily diffuse components, and these components can be modeled by equation (4.2).

(3) **BRDF scaling**. If we assume that the source and destination BRDF's are the same up to a scale factor, i.e., $\rho'_{\omega_o}(\omega_i) = k\rho_{\omega_o}(\omega_i)$, then we have

$$\begin{aligned} C' &= \frac{\int \rho_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i}{\int \rho_{\omega_o}(\omega_i)l(\omega_i)d\omega_i} \int k\rho_{\omega_o}(\omega_i)l(\omega_i)d\omega_i \\ &= k\int \rho_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i \\ &= \int \rho'_{\omega_o}(\omega_i)\beta(\omega_i)l(\omega_i)d\omega_i = C^*. \end{aligned} \tag{A.10}$$

This case could arise when performing background edits. For example, we could replace a background image of white paper with paper painted with text.

The above analysis assumes an ideal per-channel camera spectral sensitivity function, $\varphi(\lambda) = \delta(\lambda - \lambda')$, where λ' is the only wavelength the camera responds to for that channel. However, in reality, the response of a camera is integrated over a spectrum. Hence, for a general spectral sensitivity function, the camera observation becomes

$$C = \int\int \varphi(\lambda)\rho_{\omega_o}(\omega_i, \lambda)\beta(\omega_i)l(\omega_i, \lambda)d\omega_i d\lambda, \tag{A.11}$$

and the results for the latter two cases (2) and (3) do not hold anymore. However, if $l(\omega_i, \lambda)$ is separable, i.e., $l(\omega_i, \lambda) = l(\omega_i)l(\lambda)$, then the result for case (2) holds. ($l(\omega_i, \lambda)$ is separable when all the light sources are of the same kind.) That is, our shadow compositing model is correct if the background is diffuse and the lights are of the same kind. This probably explains why the result of Figure 4.7(a) looks realistic.

VITA

Yung-Yu Chuang was born in 1971 in Tainan, Taiwan. He received a Bachelor of Science and a Master of Science both in Computer Science and Information Engineering from National Taiwan University in 1993 and 1995, respectively. He came to Seattle in August 1998, attending the University of Washington. In 2004, after six good years of study, he was awarded a Doctor of Philosophy in Computer Science and Engineering. He will move back to Taiwan, where he will join National Taiwan University as an assistant professor in the department of computer science and information engineering.