Compositing 3D Objects into Digital Images with Global Illumination

Compositing 3D Objects into Digital Images with Global Illumination

Ergun Akleman
Texas A&M University

SYNTHESIS LECTURES ON COMPUTER GRAPHICS #13

MORGAN & CLAYPOOL PUBLISHERS

ABSTRACT

Digital Compositing: This book presents a simplified mathematical approach to solve problems involving digital compositing for augmenting images. Digital compositing is the art and science of combining computer generated elements with real-world imagery either in the form of still images or video, which is critical part of contemporary moviemaking. Digital compositing process often requires inclusion of global illumination, which is an essential to obtain believable digital composites.

Compositing 3D Objects into Digital Images: In this book, I will focus on only the type of digital compositing applications, in which 2D depictions of artificial 3D objects are introduced into a scene. This type of digital compositing is mainly used in on-line or off-line augmented reality applications. On-line augmented reality applications are usually interactive processes that is used to include artificial objects to the real scene in real time. Off-line augmented reality applications are mainly used in movie industry, this is used include unusual characters and objects to the scene such as Dinosaurs in Jurassic Park. The methods I represent in this book originated to speed-up inclusion of global illumination in off-line and it can eventually be used in on-line applications.

Focus of the Book: Since the newly introduced object is an artificial one, we can have a complete control how it is depicted. Moreover, it is easy to obtain any depictions from any viewpoints under any transformation. All of these are essential for obtaining global illumination effects by creating any reflected and refracted versions of the newly introduced object. We can also easily produce the shadows of the objects and include it in the compositing. Unfortunately, there is not much theoretical work in this direction since classical compositing problems such as halo reduction or color correction becomes trivial in this case. On the other hand, as I will show later, simple inclusion of global illumination requires a new way to reconstruct shaders and shapes, which is therefore interesting.

Digital Compositing with Global Illumination: Digital compositing with global illumination involves (1) reconstruction of camera parameters from a set of images, (2) reconstruction of light orientation and energy of environment from a set of images, (3) reconstruction of shapes of the objects from a set of pictures, (4) reconstruction of transparency and specularity of an object from a set of pictures, (5) recovery of BRDF from images, and (6) recovery of texture for material samples from images. In this book, I assume that we do not have any additional data beyond simple low-dynamic range photographs and I demonstrate that all of these can simply be reconstructed from such simple photographs using simple mathematical operations along with simplified shape representations.

KEYWORDS

Digital Compositing, Augmented Reality, Global Illumination, Image Based Lighting and Rendering

Contents

Preface

Expressive Uniqueness of Entertainment: The Media and Entertainment market, such as motion pictures, television programs and commercials, games and publishing, is one of the largest markets in the world. One distinguishing characteristic of the media and entertainment industry is its strong emphasis on "non-realism". Most of the successful movies, games or paintings owe their success to their uniquely expressive —or, in other words, unrealistic— attributes from their stories to their visual elements. This strong reliance on expressive uniqueness requires development of custom solutions during the production stage.

Digital Compositing: One of the key elements of current production methods is digital compositing, which allows economical production of complex scenery since modeling and rendering problems are reasonably simplified. On the other hand, digital compositing requires a significant amount of human resources. In this book, I present methods that can speed up development of custom solutions.

Digital Compositing Course at Texas A&M University: This book developed from my Digital Compositing course that I have been teaching in the Visualization Program at Texas A&M University since 2000. The course introduces the mathematical and artistic principles of digital compositing. Students learn digital compositing, image based lighting, modeling and rendering, which is basically the art and science of combining CG elements with real-world imagery either in the form of still images or video. This is an essential and critical part of contemporary movie-making.

Types of Compositing: Compositing is a very general term that includes a wide variety of artistic applications from combining real objects with real photographic backgrounds to artificial images with artificial backgrounds. Each one of these areas is important. For instance combining artificial images with artificial backgrounds can be used to speed up the rendering process, helping to obtain desired images with minor modifications.

Focus of the Book: I only focus on image augmentation applications, i.e. combining artificial images with given background images. I assume that we have almost no additional information and image can be as simple as a low-dynamic range photograph. As I demonstrate in the book, we do not really need camera parameters. I provide practical methods to unearth "qualitatively acceptable" information about shapes and materials by analyzing these background images. Although the methods to obtain shapes and materials are not

fully automatic, they are really straightforward. They can eventually be automatized for online augmented reality applications.

Goal of the Book: My goal is to provide an intuitive and simple framework for finding "qualitatively acceptable" shapes and materials. This framework will mostly be based on vector and Barycentric algebra commonly used in computer aided geometric design. I will avoid matrices as much as possible. I believe matrix algebra is great for the development of automatic methods. However, matrices are not that helpful in developing an intuition which is essential to create compositing with hands-on interaction.

Mathematical Foundations: Despite its strong focus on practical and artistic aspects of compositing, this book is not a practical compositing book. The foundations are still highly mathematical. This mathematical foundation is essential to justify the practical methods I discuss in the book. Although it is possible to skip some mathematical details and use basic methods provided in compositing, I strongly suggest following up the mathematical reasoning. It will be helpful to make critical decisions especially for the cases that are not covered in book.

Theoretical Contributions: I provide three main theoretical contributions in this book. The first is that I represent colors as points and vectors in homogeneous coordinates. This representation allows us to make practical approaches to be theoretically consistent. Second important contribution is the introduction of qualitative consistency to digital compositing. That is also critical to theoretically support some of practical claims in the book. Finally, the third contribution is the introduction of masked-3D and mocked-3D shapes. They simplify the compositing process by better representing imperfections of real shapes.

Ergun Akleman
August 2017

Acknowledgments

My Students in Digital Compositing Classes: I am thankful to all the students who took Digital Compositing since 2000. Many of them are currently working in special effect and animation companies such as ILM, Pixar, Dreamworks, BlueSky and ReelFX. I am especially thankful to my Teaching Assistants over the years starting from Cody Starr, Gary Bruins, Jaemin Lee, Siran Liu, and Zhao Yan. Gary Bruins was the first to include reflection in his composites. Jaemin Lee helped to formalize the class by taking test-bed photographs and carefully collecting additional data.

My Former Thesis Students: I am also thankful to my MS and PhD students who completed or are about to complete their thesis on related topics: Youyou Wang and Ozgur Gonen on Qualitative Rendering and Mock3D modeling [277, 280, 282]; Jaemin Lee on practical digital compositing [191, 191]; Jonathan Kiker and Michael Stanley on non-realistic compositing [184, 263]; Katherine Calkins, Clara Chan, Sarah Eisinger, Siran Liu, Zhao Yan, and Yuxiao Du on non-realistic rendering and shading [62, 96, 98, 99, 123, 131, 202, 203, 308]; Scott Meadows and Jeff Smith on Non-realistic camera [211, 213, 256, 257].

My Colleagues: I am lucky to have wonderful colleagues in departments of Visualization and Computer Science and Engineering. I am particularly thankful to my Colleagues Don House and Richard Davison to give me input for this book. Especially Don House's insightful critics helped me to improve overall structure and even the title of the book.

My Editors: I am also thankful Brian Barsky and Michael Morgan to motivate me writing this book. Without their consistent push, many of these ideas would have never been published in a book format.

Ergun Akleman
December 2017

CHAPTER 1

Introduction and Motivation

Compositing 3D Objects into Digital Images with Global Illumination Using Qualitative Scene Reconstruction: In this book, I present qualitatively acceptable scene reconstruction methods that can particularly be used in compositing of 3D objects into digital images with global illumination. In current practice, global illumination effects can only be included through a laborious process in the post-production. My approach will provide an intuitive and simple framework to develop compositing systems that can provide efficient tools to include global illumination in compositing 3D objects into digital images. This approach can eventually be used for automatic construction of "qualitatively acceptable" scenes to obtain visually acceptable global illumination even for real-time interactive augmented reality applications.

(a) *A Photograph used as the Background*

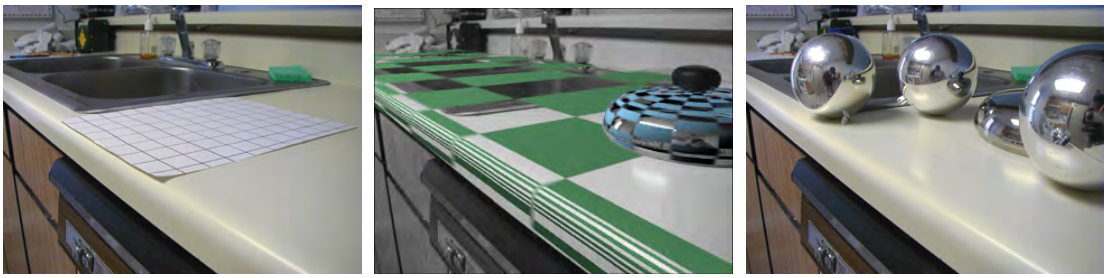
(b) *Augmenting photograph with rolling pin with no global illumination.*

(c) *Augmenting photograph with rolling pin with global illumination*

Figure 1.1: *An example of that demonstrate importance of global illumination for augmenting reality. Note the reflection of computer generated rolling pin on pot lid. The image in Figure 1.1c is a frame from an animation that is created in a laborious post production process by Gary Bruins in the PI Akleman's Digital Compositing class from early 2000's.*

Real-Time Interactive Compositing of 3D Objects into Digital Images The real time compositing of 3D Objects into digital images id called Augmented Reality (AR) today. The concept of Augmented Reality, as a real-time interactive experience, appeared 25 years ago by a William Gibson novel, called Virtual Light [153]. Despite the significant technical advances in real-time interactive augmented reality, rendering the 3D objects with realistic global illumination is still not practical in real-time scenarios. Today's real-time interactive augmented reality experiences provide visuals that are similar to the image

shown in Figure 1.1b. My goal in this book is to present practical methods that can lead global illumination in real-time by providing visuals that are similar to the image shown in Figure 1.1c.



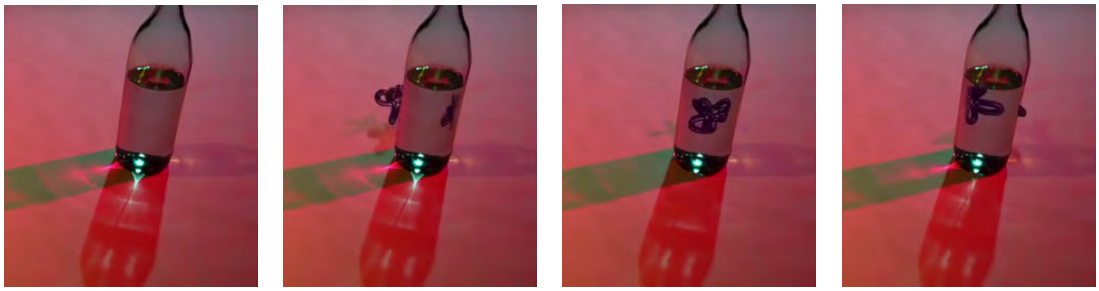
(a) *Determining position and orientation of the tabletop.* (b) *Reconstruction of Shapes and Positions of Objects.* (c) *Reconstruction of Shapes and Positions of Lights.*

Figure 1.2: *To construct the composited in Figure 1.1, there is a need to reconstruct light positions and intensities (a) and the shapes of counter-top and the lid (b). These 3D reconstructions usually have registration problems and make it hart to obtain believable global illumination effects. In this book, I demonstrate it is not necessary to use realistic 3D reconstruction to obtain believable global illumination effects.*

Compositing of 3D Objects into Digital Images in Post-Production: Global illumination as a part of post-production compositing of 3D objects into digital images has actually been around for a long time. In fact, Jurassic Park, which is the first movie that includes post-production compositing of 3D objects with global illumination effects, appeared 25 years ago. In Jurassic Park, we can see physically plausible reflections and shadows of computer generated (CG) dinosaurs [1] [134]. This book actually provides a theoretical framework to obtain global illumination effects in real-time interactive augmented reality environments with a high quality global illumination effects that are analogous to the quality of effects in Jurassic Park. The approach presented in this book can lead the development of compositing systems that can significantly speed up inclusion of global illumination.

Compositing of 3D Objects into Digital Images with Global Illumination in Post Production: Global illumination in compositing of 3D objects into digital images is a common post production process in current movie industry. To obtain realistic scenes, the global illumination effects, i.e. impact of any Computer Graphics (CG) object over the real scene and impact of the scene over Computer Graphics (CG) object must be included

[1] Post-production compositing of 3D objects has been widespread in movie production to reduce the cost of movie production since it was introduced by the film director George Melies more than 125 years ago [149]. The advance of Computer Graphics along with digital compositing made such image manipulations more popular in movie production [253, 254]. Jurassic Park is one of the first movies that provide global illumination effects [237].



(a) *A detail of a photograph that includes three colored lights and refractive objects.*

(b) *A frame from animation that demonstrate beginning of the appearance of refraction.*

(c) *Another frame from animation that demonstrate further change in caustic along with refraction.*

(d) *Another frame from animation that demonstrate change in caustic along with refraction.*

Figure 1.3: *An example that demonstrate compositing 3D objects into a digital image with global illumination using the qualitative scene reconstruction methods described in this book. Despite the fact that the original scene includes colored lights and refractive objects, using the methods discussed in this book, this can be achieved with relative ease.*

in final rendering. Unfortunately, this is a laborious process that requires a significant amount of computing and human resources (See Figures 1.1 and 1.2 as an example from approximately 20 years ago). For real time augmented reality applications, there is also a need for simple and effective methods for the computation of global illumination effects in real time.

(a) *Contact Shadow.*

(b) C^1 *Discontinuous Shadow.*

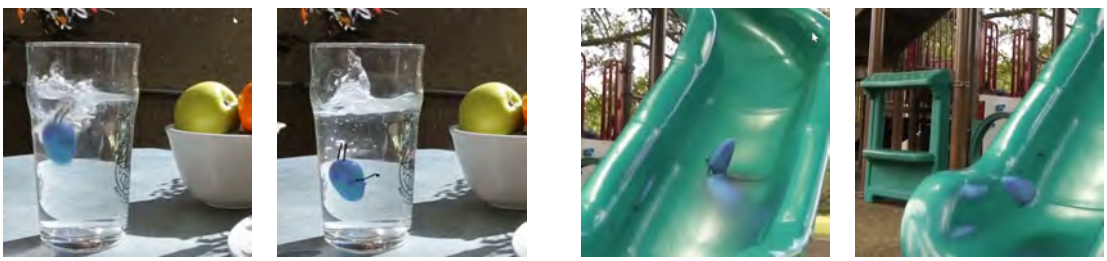
(c) *Distant Shadow.*

Figure 1.4: *Three distinct examples of shadows that demonstrate the importance of shadow in understanding position of the new object. Note that contact shadows cannot have constant value due to radiosity effects such as color bleeding or ambient occlusion. In* C^1 *discontinuous shadows, registration problems to match discontinuous regions are crucial. Distant and simple shadows are relatively simple, but their position may not be consistent.*

Global Illumination Effects: There are mainly three types of global illumination effects: Shadows, reflection and refraction. (1) Any Computer Graphics (CG) object introduced a real scene must block some illumination and cause shadows. (2) If there exist a real reflective object such as mirror, CG object must be reflected on real object. (3) If there

exist a transparent object, CG object should be both reflected and refracted, and cause caustics. Moreover, the reflection and refraction effects should be combined using physically plausible Fresnel operation.

The need for Global Illumination: Any new CG object introduced into a scene changes global illumination by changing visual appearance of all other objects. If we do not incorporate global illumination effects, the resulting scenes do not appear to be physically plausible, seamless or consistent. For instance, any new object introduced in the scene can cast shadows to the other objects, reflected and refracted by other objects and cause caustics as shown in Figure 1.3. Figure 1.4 provides a much simpler example that demonstrates different types of shadows and their importance in the perception of the reality. The new CG can also be reflected by mirrors and/or can be seen through transparent objects, and it can also create caustics, a few of which can be seen in Figure 1.3.



(a) *An alien diving into a glass of water.* (b) *The alien enjoys riding a slide.*

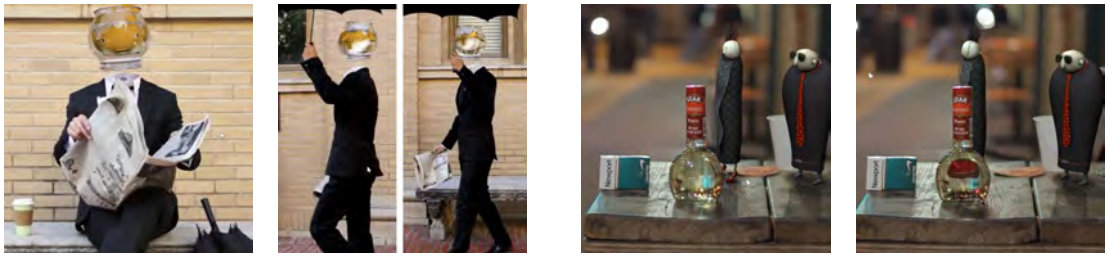
Figure 1.5: *Frames from animations created in a recent digital compositing classes. These animations are completed in a very short time using minimal computational resources based on the qualitative reconstruction methods developed over the years.*

Discussion of the Example: Figure 1.1a shows an example of background image that will be composited image with a foreground image of a rolling pin. Putting together two images that are rendered with the same camera parameters in the style of augmenting reality, as shown Figure 1.1b, is not really useful. Figure 1.1c shows an alternative composited image with global illumination. As shown in this example, creating appropriate shadow and reflection images and combining them into background image is critical for obtaining a realistic compositing. Note that in this case the process consists of a series of operations: production of shadow of rolling pin, production of reflection of rolling pin, production of an appropriate image of rolling pin. Each of these operations requires appropriate reconstruction of positions, shapes and material properties of all objects and lights in the background scene,

Difficulty in Reconstruction: Reconstruction of reasonably good representation of real objects in the photographs is required to obtain high quality global illumination effects as

shown in Figure 1.2b. Unfortunately, the true 3D reconstruction solutions as demonstrated in Figure 1.2b is not an option, since the real objects are almost always imperfect. When we use 3D reconstructions, there will always be some relatively minor reconstruction errors that cause significant disturbance in visual perception. Moreover, we need the whole scene information as demonstrated in Figure 1.2c to use image based lighting, which may not always possible. It is, therefore, global illumination is currently done in post production stage with a significant involvement of humans.

Need for a New Reconstruction Approaches: To provide speed-up the process of inclusion of global illumination, there is a need to circumvent the reconstruction problems by using new reconstruction approaches that can provide visually acceptable and physically plausible global illumination effects. If we want to provide global illumination real-time interactive augmented reality applications, we further find solutions that does not require for human involvement. In this work, I present qualitatively acceptable scene reconstruction methods that can effectively be speed up the inclusion of global illumination in digital compositing. These methods can later be extended for real-time augmented reality applications.



(a) *A Magritte character with an aquarium head with large fish. In this case aquarium is real and fish is CG.*

(b) *A cartoon character walking behind a glass.*

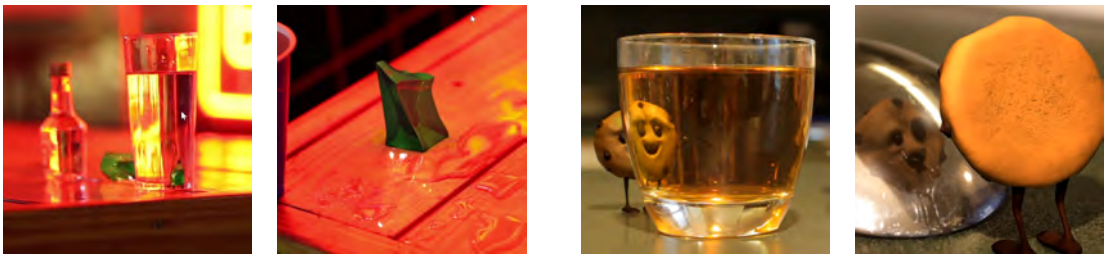
Figure 1.6: *More example frames from animations created in digital compositing classes.*

The Roots of New Reconstruction Approaches: The new reconstruction approaches stem from the Digital Compositing course that I have been teaching in the Visualization Program at Texas A&M University since 2000. Figure 1.3 demonstrates frames from a composited animation that is created with relative ease using the qualitative scene reconstruction methods described in this book. The course introduces the mathematical and artistic principles of post-production (or off-line) digital compositing of 3D objects with global illumination, which is an essential and critical part of contemporary movie-making. Although post production work allows dedication of significant amount of off-line resources, it is still critical to improve productivity by reducing the student work and computational time. Over almost 20 years period, I have, therefore, developed methods to significantly

improve productivity (See Figures 1.5, 1.6 and 1.7 to see animation frames from recent class project that are created very efficiently using minimal amount of resources[2].)

Qualitatively Acceptable Reconstructions: The qualitatively acceptable reconstructions are really expert methods to unearth "qualitatively acceptable" information about shapes and materials that are developed by analyzing images. Although the methods to obtain shapes and materials have not been automatic yet, they are really straightforward. They can, later, be automated for real-time augmented reality applications just by providing minimal amount of additional information such as depth, which can be readily available now with time-of-flight cameras. A counter intuitive result is that depth information also include embedded perspective transformation and it is even possible to avoid camera parameters.

Focus of the Book: In this book, I provide an intuitive and simple framework for finding "qualitatively acceptable" shapes and materials. This framework is based on Barycentric and vector algebra that is commonly used in computer aided geometric design. An important aspect of our framework is that we will use Barycentric and vector algebra not just for spatial domain, but also for solving problems in image space as operation on colors. We also turn 3D shapes of the objects into image-space representations, which we call Mock-3D shapes. Having an algebraic framework in image space for colors and shapes will help us to find qualitative mathematical solutions for reconstruction problems. Figure 1.8 provides a visual representation of this process as a projection from 3D space into image-space. This approach of using an algebra in image space can later help to employ deep learning techniques in shape reconstruction and compositing for automatic reconstruction.



(a) *A jello enjoying the nightlife in Texas A&M's Northgate while causing caustics and refractions.*

(b) *A cookie walking around a kitchen while being reflected and refracted.*

Figure 1.7: *More example frames from animations created in digital compositing classes. As shown in these examples, our world is full of reflective and refractive objects. If we do not appropriately include global illumination effects, the augmented reality experiences do not necessarily feel augmented or real.*

[2]Most of the students who created these animations are currently working in special effect and animation companies such as ILM, Pixar, Dreamworks, BlueSky and ReelFX

Theoretical Contributions: This book provides three main theoretical contributions. The first important contribution is the introduction of qualitative consistency. That is also critical to theoretically support some of our practical claims on working directly with image space. The key idea in this book is to develop an image space approach that provides qualitative acceptable results without worrying about physical accuracy. The second contribution is that we use Barycentric and vector algebra to operate on colors and images. This approach allows us to make practical approaches to be theoretically consistent. Finally, the third contribution is the introduction of mocked-3D shapes into compositing. They simplify the compositing process by better representing imperfections of real shapes.

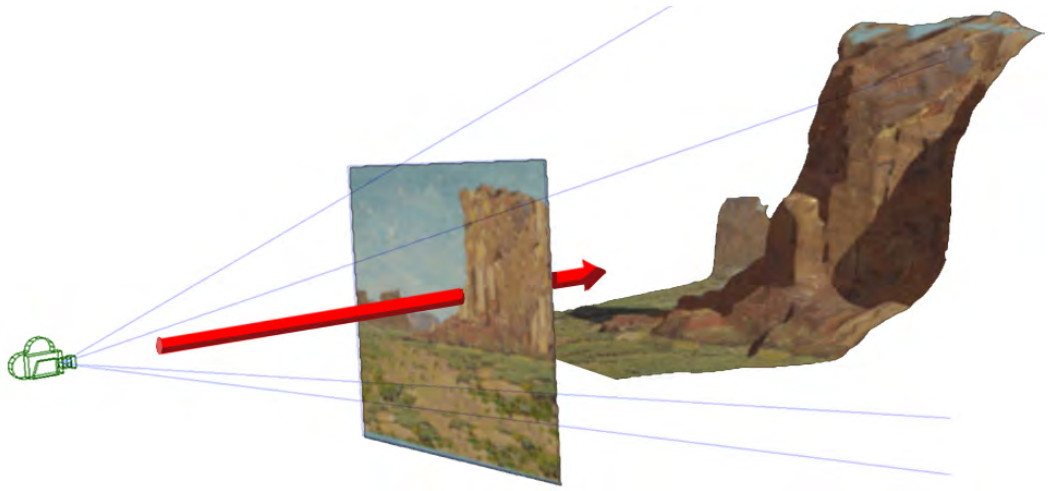


Figure 1.8: *This projection demonstrates the basic premise of this book: Composition in image-space. We turn all 3D shapes of existing objects into images such as depth maps and normal maps by projecting them into image space. Similarly, we turn all materials into a set of texture images projected into image space. Shaders, then, simply become compositing operators that are implemented as algebraic operations on images. As a result, inclusion of global illumination effects turn into manipulation of these control images.*

Importance of Case Studies: The key part of learning the approach is working on case studies. In my digital compositing classes, students first learn how to handle global illumination by dealing with a set of carefully chosen extreme cases. These cases are motivated by common experiences such as outside under strong sun light, inside under strong sun light, outside night illumination, and inside night illumination. Each case is further classified based on existence of diffuse, reflective and refractive objects in the scene. Students develop expertise by handling these cases. Then, students use this experience to create an animation from a story they have developed. Since they have already build the expertise, they finish the final works very fast as shown in Figures 1.5, 1.7 and 1.6.

CHAPTER 2

Preliminaries & Previous Works

Introduction In this book, I demonstrate that using Barycentric formulation for the reconstruction of depictions and Masked-3D/Mock-3D shapes for shape reconstruction can significantly improve the efficiency of workflow in digital compositing process and help to include global illumination easily. In this book, I mainly focus on three types of global illumination effects:

Shadowing: Any Computer Graphics (CG) object introduced a real scene will block some illumination and cause shadows. Shadow is a type of global illumination effect since it cannot be computed only local shading information. In this book, I discuss two types of scenes: (1) Scenes with directional lights and (2) scenes with multiple colored lights.

Reflection: Any CG object that is composited into a photograph of a reflective object such as mirror should be reflected. In this book, I consider both flat and curved mirrors.

Refraction: Any CG object that is composited into a photograph of a transparent object should be both reflected and refracted. Moreover, these effects should be combined using physically plausible Fresnel operation. I also discuss how to incorporate the impact of newly added object into caustics created by refractive objects.

2.1 DIGITAL COMPOSITION

Introduction: In this section, I will briefly introduce the concepts related digital compositing. This is just useful to provide some definitions to explain what will be covered in this book.

Binary Compositing Operations: Compositing, as a broad term, means putting a set of images together to obtain a new image. The term compositing is used as a generic name for [usually non-commutative] binary operations that combine any two images into one [234, 253] as

$$I = I_1 \otimes I_0, \tag{2.1}$$

where I, I_1 and I_0 are images and $\otimes$ is an operation works on images . Since compositing operations are usually not to be commutative, we differentiate I_1 and I_0 by calling them

as foreground and background images respectively. Some operations like *Over* are closely associated with compositing [234, 253], however any binary operation that works on image can be considered a compositing operation.

General Digital Compositing: In this book, I view compositing as a more general operation that combines a 2D depiction of a new 3D object into an existing (or original) 2D depiction of a 3D scene, which is usually called an image [88, 89, 92, 181, 190, 294]. In this generalized case, we cannot simply write a single equation such as the one in Equation 2.1. The generalized operation requires creation of a set of images that are combined by a series of equations. Moreover, we need to determine camera parameters and viewpoints, matching lighting and material properties, and shapes of objects to construct consistent images. It is, therefore, each compositing problem can be unique and the process of compositing requires a great deal of human expertise and involvement along with mathematical and scientific knowledge. In this book, I present a theoretical framework that can simplify the decision making process.

The Audience of this book: I expect the audience is familiar with 3D computer graphics. This book is particularly interesting for two types of audience: Technical directors and software developers. Technical directors (or aspiring technical directors) who deal (or wants to deal) with compositing in movies can benefit from this book since the methods presented in the book significantly speed up the compositing process. These methods can also be used in real time applications. There is a need for the development of software that can automate these methods for real time applications. I, therefore, this that researchers and software developers who wants to include global illumination into real-time augmented reality applications will also be interested in improving and implementing the methods presented in this book.

Knowledge needed for this book: The 3D graphics knowledge I expect from readers are somewhat basic. They should know how to render an image of an object. They should be able to define camera, lights and material. I assume they should be able to set up reasonable camera parameters and viewpoint for given image. As I will discuss later, exact camera parameters and viewpoint is not necessary for inclusion of global illumination. I also assume the reader can create lights and light shaders. On the other hand, the light positions and actual intensities do not have to be precise to obtain qualitatively consistent results. The most of the book is dedicated to shader development. Although, these shaders are extremely simple, it is better to have some experience with shader programming. Since existing user friendly shading interfaces cannot provide the kind of the control you need for these applications.

2.2 FOUR TYPES OF COMPOSITING

Introduction: Augmented reality of one of the four types of digital compositing, in which virtual objects are introduced to real scenes. To define these four types of composting and, in particular, augmented reality, we need first classify objects and scenes by differentiating real and artificial.

Classification of Objects and Scenes: The original 2D depiction of a 3D scene can be a computer generated image (CGI), a photograph, a painting or an illustration. Similarly, the 2D depiction of a newly introduced 3D object can be a computer generated image, a photograph, a painting or an illustration. These 2D depictions can be representations of (1) either real (2) or artificial (or virtual) scenes. The main differences between the two types of scenes can be summarized as follows:

Real Scenes or Objects: We have an extremely limited knowledge about real scenes or objects. Our main access to real scenes is only through their 2D depictions. Photographs are of this type; i.e. they are 2D depictions of real scenes or objects.

Artificial Scenes or Objects: We have complete knowledge of artificial/virtual scenes or objects. We also know how the 2D depictions are created. Representational paintings, illustrations and computer generated images (CGI) are of this type; i.e. they are 2D depictions of artificial/virtual scenes and objects.

Four Types of Compositing: Based on this classification, it is possible to classify compositing operations in four categories based on types of the original scene and added objects as follows: (1) real into real, (2) real into artificial, (3) artificial into real, and (4) artificial into artificial. Although I focus on artificial in reality for augmenting reality in this book, each one of these four compositing categories have important applications and there exists a significant amount of research for each case. I, therefore, briefly describe history of non-augmented reality compositing types.

2.2.1 BRIEF HISTORY

Real into Real/Artificial: Introduction of a 2D depiction of a real object into an existing (or original) 2D depiction of a real scenes is the oldest and the most common type of compositing. Real into artificial is also widely used in movies to create virtual backgrounds (virtual sets) to reduce the cost of movie production. George Melies is most likely the first person who used both "real into real" and "real into artificial" compositing in movies [149]. Since George Melies' work, real into real/artificial compositing has been widespread in movie production. Even before the advance of Computer Graphics, "real into real/artificial" compositing techniques such as Chroma Key Compositing (also known as Blue or Green Screen) has been popular in movie industry. In fact, the concept of alpha for

digital compositing is introduced by Alvy Ray Smith inspiring from chroma key compositing [253, 254].

The Concept of Transparency/Alpha: The concept of alpha is one of the key inventions for digital compositing [253]. Over operation in the form of the Equation 2.1 is developed based on alpha and it used to extract an object from given image. Another important invention is the introduction of associative colors and associative over operation that turn the operation $\otimes$ in the Equation 2.1 into an associative operator [234]. In this way, when more than two images are composited, the result become independent of the order of the implementation of operations. This is especially useful for volume rendering in compositing multiple slices [122, 125]. [320]

Halo Removal: An important problem with Chroma Key Compositing is that the resulting images usually have a halo around the extracted object. Halo happens because of (1) incorrect alpha values in boundary regions and (2) the mismatch the original background color (Blue or Green) and new background image. Obtaining good alpha values around the boundary regions such as hair is particularly difficult. Therefore, a wide range of semi-automatic methods are developed to solve this problem [196, 264]. All these methods start with a three color map and by computing alpha only in the gray region, it is possible to obtain reasonable results by removing most of halo.

Color Correction: Another important problem with real into real/artificial compositing is to match illumination of foreground and background objects. Debevec has developed most of the ideas related to color matching [116, 118, 120, 241]. He and his team in Institute for Creative Technologies at University of Southern California has developed several light stages to relight foreground objects to match with background. Their systems are widely used in movie industry to create higher quality real into real/artificial compositing. His contributions to this field is so overwhelming, I simply provide a direct link to his website [117].

Global Illumination in Real into Real/Artificial Compositing: For global illumination, we need all possible views of the foreground object. For instance, for shadow we need an additional view from light position. Similarly, we need additional views for reflection and refraction. In the future, most likely computer graphics researchers will solve these problems to obtain more realistic real into real/artificial compositing by including global illumination. There are many developments that can help in this direction. On the other hand, in the current technology level it is straightforward to include global illumination compositing with artificial objects as demonstrated in this book.

Artificial into Artificial: The flexibility of introduction of a 2D depiction of an artificial object into an existing (or original) 2D depiction of an artificial scene is useful in movie production. Therefore, artificial into artificial compositing is widely used to speed

up rendering [125] in animated films. It is also used to control look-and-feel of final results. The idea is extremely simple. We render objects separately and combine them later. This process can speed up production process since color corrections can directly be done in post processing without a need for re-rendering. Artificial into artificial is also not focus of this book.

Artificial into Real (Augmenting Reality): In this book I focus on introduction of a 2D depiction of an artificial object into an existing (or original) 2D depiction of a real scene. This type of compositing is used for augmenting reality. For instance, in movie industry this is used to include unusual characters and objects to the scene such as Dinosaurs in Jurassic Park. This problem is simple from purely digital compositing point of view since classical digital compositing problems such as halo reduction or color correction becomes trivial in this case. On the other hand, inclusion of global illumination requires a new way mathematical thinking to handle shaders and shapes to speed up and/or automate the process. In this book, I focus on providing simple solutions to these global illumination problems.

2.3 GLOBAL ILLUMINATION IN AUGMENTING REALITY

The need for Global Illumination in Composition: Compositing problem in augmenting reality is complicated since any object introduced into a scene changes global illumination by changing visual appearance of all other objects. If we do not incorporate global illumination effects, the resulting composites do not appear to be physically plausible, seamless or consistent. For instance, any new object introduced in the scene can cast shadows to other objects, can create caustics, can be reflected by mirrors and/or can be seen through transparent objects as shown in Figure 1.1.

Composition Types that can allow Global Illumination: The previous discussion of compositing types should make it clear that we need more than one depiction of the newly introduced object for global illumination. Unfortunately, compositing with global illumination is very difficult for real into real and real into artificial compositing since we usually have a single depiction of newly introduced object. On the other hand, global illumination is possible with artificial to real compositing since we can always render new depictions of artificial objects that can be useful for global illumination.

Advantage of Artificial Objects: The main advantage of dealing with artificial objects is that we can have a complete control over depiction of artificial objects. Moreover, we can obtain any depiction of the objects from any given viewpoint using any transformation. In other words, obtaining global illumination effects is made possible using artificial objects as newly introduced objects instead of real ones. In this way, the global illumination problem simplifies reconstruction of depiction method and reconstruction of the background scene.

Difficulty in Reconstruction: Reconstruction of reasonably good representation of real objects in the photographs is required to obtain high quality global illumination effects. Unfortunately, the true 3D reconstruction solution as discussed in previous chapter is not always an option, since the real objects are always imperfect. When we use 3D reconstructions, there will always be some relatively minor reconstruction errors that cause significant disturbance in visual perception. Moreover, we need the whole scene information to use image based lighting, which may not always possible. In this book, I, therefore, present methods to circumvent these reconstruction problems using qualitatively acceptable reconstructions of shapes, materials and lights.

CHAPTER 3

Qualitative Reconstruction

Introduction In this chapter, I introduce the term "qualitative reconstruction" . The key idea in this book is providing qualitative acceptable results without worrying about physical accuracy. This idea is, in fact, not entirely new. Qualitative methods has been widely used since the beginning of Computer Graphics in one form and another. This formalization helps us to work directly on collected data and we can simply avoid physically based methods. In the next section, I will provide a formal introduction of qualitative consistency.

3.1 QUALITATIVE CONSISTENCY

Qualitatively Consistent Results: One of the key ideas in this book is qualitative consistency. I base all the digital compositing framework on qualitative similarity concept introduced by Forbus et al [148]. Let $Q_x : \mathfrak{D} \rightarrow \mathbb{R}$ and $Q_y : \mathfrak{D} \rightarrow \mathbb{R}$ denote functions $x = Q_x(p)$ and $y = Q_y(p)$ where $p \in \mathfrak{D}$ denote any given domain and $x, y \in \mathbb{R}$ denote real numbers. They, then, say Q_x and Q_y are qualitatively proportional to each other if there exists a monotonically increasing mapping $y = f(x)$: For all $p_1, p_2 \in \mathfrak{D}$ if $x_2 = Q_x(p_2) > x_1 = Q_x(p_1)$, $f(x_2) = y_2 = Q_y(p_2) \geq f(x_1) = y_1 = Q_y(p_1)$, then we can call Q_x and Q_y qualitatively proportional to each other. One advantage of this formulation, there is no need to explicitly derive the monotonically increasing function f. To demonstrate Q_x and Q_y are qualitatively proportional, we can simply demonstrate the following inequality is correct for all p_1 and p_2:

$$\frac{Q_x(p_2) - Q_x(p_1)}{Q_y(p_2) - Q_y(p_1)} \geq 0$$

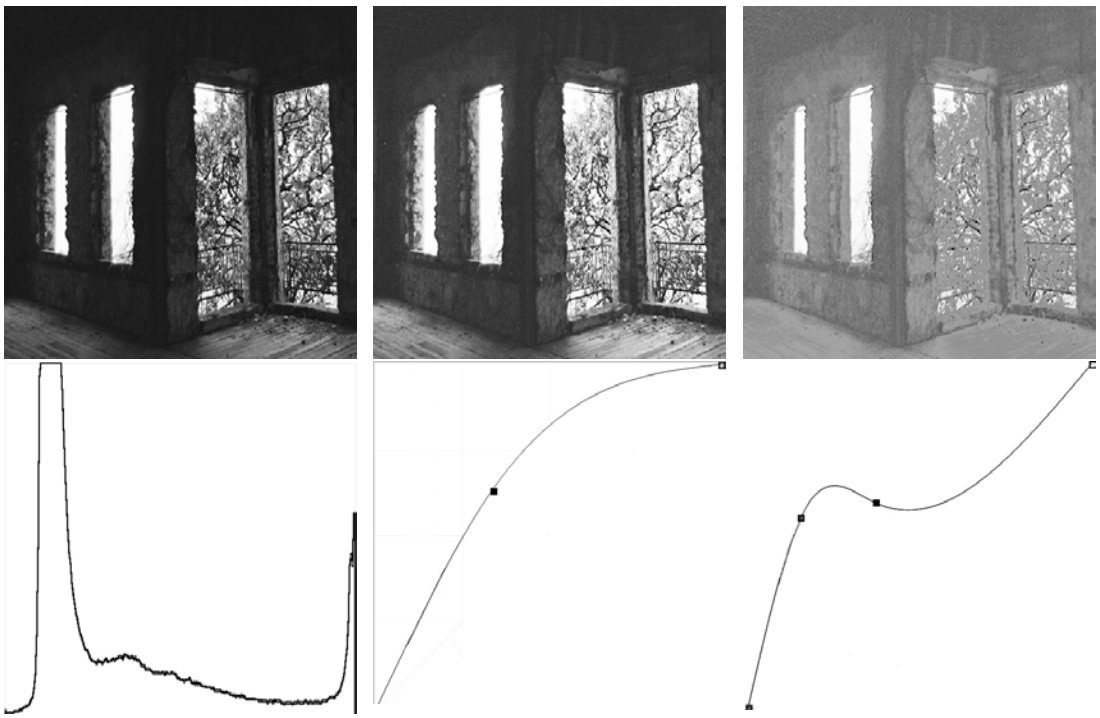
The symbol $\propto_+$ is used to relate two qualitatively proportional entities to each other as

$$Q_y \propto_+ Q_x.$$

Relativistic Perception: This formalization is especially useful to evaluate qualitative similarities in rendering and post processing and it is in sync with human visual perception reserach. As shown by researchers such as Yilmaz, Motoyashi and Adelson [216, 309, 310, 311] and artists such as Albers [75], the color perception is relativistic process. In other words, it makes sense to check positive correlation between relative differences.

Examples of Qualitative Consistency and Inconsistency: The Figure 3.1 shows examples of qualitatively consistent and inconsistent images. Note that the original pho-

tograph in Figure 3.1a and the manipulated photograph in Figure 3.1b looks acceptable although color values are not the same. On the other hand, the manipulated photograph in Figure 3.1c looks strange since the order of gray values are changes based on a transformation that is is not monotonically increasing.



(a) *A gray level picture by photographer Tanju Akleman and its histogram.*

(b) *The original photograph manipulated by a monotonically increasing function shown below.*

(c) *The original photograph manipulated by a function that is not monotonically increasing.*

Figure 3.1: *An example that demonstrate qualitative similarity using photo manipulation.*

3.2 MONOTONICALLY INCREASING FUNCTIONS

Monotonically Increasing Functions From $S \in \Re$ to $[0,1]$ To construct monotonically increasing functions it is often useful to restrict and/or convert the parameters to the range 0 to 1. This conversion is easy since any subset of real numbers can always be mapped onto $[0,1]$ and there exist several practical methods for this conversion. Here, I present a few such monotonically increasing functions.

Clamp Function: The simplest and one of the most common monotonically increasing function is *clamp*, which is defined in Renderman as,

$$y = Clamp(x, x_0, x_1) = \begin{cases} 0 & \text{if} \quad x_0 \leq x, \\ \dfrac{x - x_0}{x_1 - x_0} & \text{if} \quad x_0 < x < x_1, \\ 1 & \text{if} \quad x \leq x_1, \end{cases} \tag{3.1}$$

This function is, of course, is monotonically increasing only $x_0 \leq x_1$. An example of the use of clamp is photographing process, which is a type of HDR to LDR conversion. Another example of clamp function is the conversion of cosines in Gooch shading [161], in which $x_0 = -1$ and $x_1 = 1$.

Gamma Correction: It is easy to show that a γ corrected LDR (or HDR) image is qualitatively similar to the original LDR (or HDR) image since x^γ is monotonically increasing function for any $x \in \Re^+$ respectively.

Logistic (or Sigmoid) Functions: Logistic Functions that are often used in Machine learning algorithms also based on qualitative similarity concept. It is also easy to show that the these functions, which are in the form of

$$f(x) = \frac{1}{1 + e^{-kx}},$$

are also a monotonically increasing functions that can be used to convert real numbers to $[0, 1]$.

Any Tone Mapping for HDR to LDR conversion: Tone mapping algorithms are also based on qualitative similarity concept. There exist many non-linear tone mapping methods to convert numbers in high dynamic range to low dynamic range. All tone mapping methods, in essence, are monotonically increasing. For instance, Gradient Domain methods clearly preserve and control relativistic properties [143] [1].

Z Buffer Transformations: Depths and distances can also be significantly large. Therefore, before Z buffering, we convert distance or depth, i.e. z information into the range of $[0, 1]$. Since distance, i.e. z, is always a positive number, we usually using the following monotonically increasing transformation

$$z' = \frac{1}{z + 1}.$$

In this case, when the distance z goes to $infty$, z' goes to 1 and when $z = 0$ z' is also 0.

[1] Since we can always convert $[0, 1]$ to any real number with a monotonically increasing transformation, in this book I do not have to have a special treatment of high dynamic range. I, therefore, do not use tone mapping in the case studies in the book. However, there is no reason tone mapping cannot be used for this purpose.

Bas Reliefs: A low relief or bas-relief is a a relief sculpture with a shallow overall depth such as reliefs on coins. The original shape is distorted by scaling. Front view also include embedded perspective transformation. Wang has shown that monotonically increasing changes in depth defines monotonically increasing transformation on $\cos\theta$ [277]. In other words, the diffuse reflection on bas-reliefs are qualitatively similar to the original shapes. This is most likely the reason that the bas-reliefs make sense from the front view.

3.3 USING QUALITATIVE SIMILARITY FOR OUR ADVANTAGE:

Observation: As discussed in the previous section, almost all data we deal with already obtained by some type of monotonically increasing transformations. We do not normally know the transformation. Therefore, it is easier to work directly on data.

My Approach in a Nutshell: I turn every problem into a simple interpolation problem, in which we interpolate given extreme values. Extreme values for color can be the darkest or the lightest colors, for shapes, they can be boundaries. Since we interpolate extreme values, we can guarantee to obtain correct registration. Assume that we interpolate two extreme values x_0 and x_1 with the following Barycentric line equation as

$$x = x_0(1 - t^\gamma) + x_1 t^\gamma$$

where $t \in [0, 1]$ and γ is any positive real number. In this example, γ corresponds qualitative similarity term, which we do not necessarily know. On the other hand, we know extreme values x_0 and x_1. As far as we obtain them correctly, the value of the γ is not that essential and we can simply choose $\gamma = 1$.

Implications: With this approach, we only need to focus on extreme values that are critical for registration. We assume that we can always redistribute the rest using some monotonically increasing transformation. This approach makes the whole process extremely simple. In this book, I always use Barycentric interpolation equations to guarantee that (1) we always obtain extreme values, and (2) the computed results are always in the convex hull of extreme values. The second property is particularly important to quickly obtained acceptable results by guaranteeing to avoid unusual values.

3.4 QUALITATIVELY CONSISTENT GLOBAL IMPACTS

Two Impacts of Newly Introduced Object: The introduction of new object to an existing scene is expected to impact it in two different ways: (1) The appearance of the newly introduced object is impacted by the original scene. (2) The appearances of all objects in the original scene is impacted by the newly introduced object. We use extreme

values that are already existed in the image to compute the impact, which leads qualitative consistency.

Qualitative Consistency of Impacts: In this book, I discuss essential impacts that are key to obtain a depiction that is qualitatively consistent with the original depiction of the original scene after the introduction of the new object. For successful compositing, both types of impacts should produce qualitatively consistent depictions. In my approach to obtain consistency becomes easy since I assume light energy and perspective transformation is already embedded into extreme values. Then, we simply interpolate these extreme values. In other words, instead of dealing with significant number of interacting control variables, we deal with few number of extreme values that can provide qualitatively consistent impacts with ease.

Two Conditions to obtain Qualitatively Consistent Depictions: Based on the previous discussion, I observe that the following two conditions must hold to obtain qualitatively consistent depictions: (1) The appearance of the newly introduced object must be qualitatively consistent with overall illumination in the scene. (2) The appearances of all objects must change qualitatively consistent with expected impact of newly introduced objects. Both can be achieved by guaranteeing to obtain extreme values using simple interpolations.

A Few Examples of Qualitative Consistency: Since the key concept for compositing is qualitative consistency, let me provide a few examples to demonstrate what I mean with consistency. Existing lights in the scene must illuminate the object the same way they illuminate the other objects in the scene for consistency. The objects already existed in the scene must cast shadows onto the newly introduced object the same way they cast shadow to other objects. If the newly introduced object is reflective object, other objects and lights must be reflected by the newly introduced objects the same way the other reflective objects reflect. This few examples of consistency requirements also demonstrate the difficulty of solving general consistency problem. Interpolation of extreme values provides a simple solution to obtain consistent results.

3.5 QUALITATIVELY SIMILAR MATERIALS

Qualitative Similarity in Expressive Depictions: In the case of shading models (and shader), we can consider the domain $\mathfrak{D}$ consists of surface positions, surface normals, light directions. The shader parameters we use in computer graphics are like Q_y or Q_x that provide shader parameters for any given set of surface positions, surface normals and light directions. Now, assume that there exists a shader parameter Q_x that correspond "so-called physical reality", we can then call shader Q_y is qualitatively acceptable if and only if $Q_y \propto_+ Q_x$. This property explain how artists create visually acceptable images

even when the images are not quantitatively correct. As I discuss earlier, we extensively use this property in practice to obtain reasonable results by mapping parameters using monotonically increasing functions.

Shaders and Shading Languages: The concept of shader and shading languages are developed to obtain such a widely different types of depictions. Cook laid the foundations of the procedural shader concept with the concept of *Shade Trees*[110], which quickly evolved into the *Renderman* shading language [77, 111, 168, 273]. The main goal of shading languages is to provide users easy control over the reflection functions to obtain desired look-and-feel. The conceptual flexibility of the procedural shader, which can be developed with functional operators, has made *Renderman* an industry standard since the 1990's [128]. The shader concept is now ubiquitous in graphics, as it is central to the modern GPU architecture, which is structured around a sequence of programmable shaders [201, 208, 238, 244].

Obstacles to Obtain Desired Look: This problem has not usually been acknowledged, since there exists a false impression that obtaining a desired background is not that difficult. It is easy to understand where this impression comes from. There exists a large number of publicly shared shader examples for a wide variety of styles. Creation of a version of an existing shader is not that difficult. People can always create a new shader by making minor changes to a publicly shared shader that provides a style they like. It is also possible to obtain an interesting style by just randomly playing with functions. On the other hand, to obtain the qualitatively similar style, we need to match overall look of background. This is a very different problem that cannot typically be solved by tweaking existing shaders.

Formalization of Depiction Reconstruction: We recently developed general mathematical approaches for the development of the shading models for depiction reconstruction [61]. These approaches, which are based on Barycentric algebra that is widely used in Computer Aided Geometric Design, provide simple ways to develop models that can help to obtain look-and-feel of a given 2D depiction. Using Barycentric algebra, in fact, we obtain meaningful equations, such as Bézier or B-Splines, that can provide desired results with a set of control points. Reconstruction problem simply reduces to identification of control points. Therefore, once the shading models are constructed/decided, it is relatively straightforward to provide consistent results by identifying extreme points that can be used as control points. In this book, I simply use n-linear Barycentric functions such as Bilinear or Trilinear functions depending on the number of independent variables. In our case, the independent variable are simply light positions. An important property of this approach is that it is straightforward. With relatively limited information, it is possible to provide qualitatively similar look.

3.6 QUALITATIVE SIMILAR SHAPES

The Need for Shape Reconstruction: Since global illumination requires object-to-object interactions, we need to have virtual representations of shapes and positions of all objects including lights in the photographs, paintings and illustrations to obtain these global illumination effects. I call these virtual shapes "proxy geometry" since they are not really visible. Their impact becomes indirectly visible through shadows and reflections.

Qualitative Reconstruction with Masked-3D and Mock-3D Shapes: There is, therefore, a need for a reconstruction methodology to convert real objects into virtual representations that can be used for computing object-to-object interactions. I, therefore, introduced the concept of masked-3D shapes in my digital compositing course, at least ten years ago. I, with Youyou Wang, recently demonstrated that it is also possible to represent 3D shapes using simple 2.5D reconstructions of real scenes, which we called Mock-3D scenes [277] (See Chapter 6 for details). Both of them require for some minor but conceptual changes in existing compositing operations to simplify inclusion of global effects in digital compositing. I will also demonstrate that these proxy shapes can be considered

Barycentric Interpolation in Shape Reconstruction: I assume that qualitative reconstructions are also based on Barycentric interpolation of extreme values. In this case, extreme values are 2D positions of maximum, minimum and saddle points of the shapes and the 2D silhouette boundaries. By correctly identifying the extreme values and interpolating them, we can guarantee that there exist a monotonically increasing transformation from 3D shape to 2.5D shapes. The conceptual requirement of Barycentric formulation also critical to stay in convex hull of extreme points and do not create additional maximum or minimum points that may cause visual problems. Of course, from a minimal information such as a single image, we can only identify 2D silhouette boundaries as extreme points. The rest of the extreme points are just rough estimations.

CHAPTER 4

Mathematical Foundations: Colors and Shading

Introduction: In this chapter, I present mathematical foundations for colors and shading that are required for digital compositing. These foundations that are based on the mathematical framework that supports computer aided geometric design. It is particularly useful for practical problem solving in compositing. In fact, I use the same type of entities and operations to represent colors and develop shading models. For digital compositing, it is important to develop simple models that can easily be reconstructed to obtain qualitatively consistent results with depictions of background scenes.

4.1 COLOR-POINTS AND COLOR-VECTORS

You could say: Otherwise, the reader is lost without knowing the space you are working in or even how an additive operation would work.

Colors as Points and Vectors: I recently realized that it is better to consider color as either points or vectors in homogeneous coordinates. If we want to develop our models based on physics, we have to have addition operation. On the other hand, point addition on homogeneous coordinates is not really an addition operation. As will be shown below in this chapter, addition in homogeneous coordinates is really a Barycentric operation disguised as an addition operation. Therefore, if we want to develop our models based on physics, we need to represent colors as vectors since addition is closed only over vectors. On the other hand, if our goal is just to faithfully represent underlying data, our models do not have to be based on physics. Without addition, we can still obtain physically plausible results with Barycentric operations.

Definitions: For color-points and color-vectors, I will basically use the point and vector notation. Let $\vec{C}$ denote to color vectors and $\boldsymbol{C}$ denote to color points, then without loss of generalization we can treat colors as 3D points and vectors that consists of three – red, green and blue – components or "color channels". Their first letter, i.e. R, G, B, and α denotes red, green, blue and alpha channels as scalars. C or c denote any channel of a color vector or point. The channels of in color-points, $\boldsymbol{c}$'s, are any positive real numbers. On the other hand, the channels of color vectors, $\vec{C}$, are any -positive or negative- real numbers.

Closure Properties: These two entities, i.e. $\vec{C}$, and $\boldsymbol{C}$ are closed under different set of operations. There are also a special subset of $\boldsymbol{C}$ that is closed under over operation. Therefore, we need to be careful which operations we are using. Using just three dimensions is not really a restriction, we can easily extend the dimension without changing the rest of theoretical framework.

Examples:

- $\vec{C}_i = ((R_i, G_i, B_i)$ is a known color-vector with $\vec{C}_i \in \Re^3$..
- $\vec{C} = (R, G, B)$ is a variable color-vector with $\vec{C} \in \Re^3$.
- $\boldsymbol{C}_i = (\alpha_i R_i, \alpha_i G_i, \alpha_i B_i, \alpha_i)$ is a known color-point with $\boldsymbol{C}_i \in [0, \infty)^4$.
- $\boldsymbol{C} = (\alpha R, \alpha G, \alpha B, \alpha)$ is a variable color-point with $\boldsymbol{C} \in [0, \infty)^4$.

Spectral Radiance as a Vector: In all computer graphics equations such as classical rendering equation [179], we allow adding spectral radiance terms since the summation is really happening physically. It also makes sense to multiply diffuse or specular reflection terms with incoming spectral radiance terms since the reflection is simply a ratio of the incoming radiance. However, when we start to solve problems in mathematical domain, adding comes with subtraction. This forces us to take difference of two colors in solving problems. Therefore, the terms we use in many of our equations naturally correspond to vectors in $(-\infty, \infty)$ since $[0, \infty)$ is not closed under subtraction. In other words, when we use colors in equations that are not closed under space of points, we need to use notation $\vec{C}$ to clarify that they are not really points anymore. As mentioned before, the current version of Renderman, a popular shader programming language, it is allowed to use negative colors [77, 111, 168, 273].

Conversion to Points: When we deal with vectors, we need to make sure to convert color vectors to point vectors by adding a point $\boldsymbol{C}_0$ to all vector colors once the computation is completed. Since vector plus a point defines a point, this is formally acceptable. However, this is still tricky since we have to choose $\boldsymbol{C}_0$ in such a way that all numbers are turned into positive. In this conversion, $\boldsymbol{C}_0$ corresponds to origin of the color space. In current practice, negative numbers is removed by clamping negative values or using absolute value. However, such non-linear operations are not acceptable for digital compositing since they can significantly alter appearance of colors.

Colors as Points: I, therefore, use color points with Barycentric algebra for the development of shading models in digital compositing applications. These shading models are not necessarily physics based. The development of these models resemble to statisti-

cal approaches since we simply use data along with our physical knowledge. A significant advantage of this approach, we can handle more complicated cases using simple models. Moreover, our models provide physically plausible results for photographs since we build our models based on the information available in the images. Another advantage of this approach is that it can work even for expressively depicted paintings and illustrations.

Dynamic Range of Color Points: Another tricky issue is the dynamic range of color space. Dynamic range of a set of colors is defined as $D = C_{max}/C_{min}$ where C_{max} and C_{min} maximum and minimum scalar energy in any wavelength. The terms low and high dynamic range is misleading since they refer actual dynamic range of physical entities such as our recording or display devices, in which C_{min} can never be absolute zero. On the other hand, in mathematical representations we can always choose $C_{min} = 0$. To avoid we assume C_{min} is smallest non-zero number we can represent. Under this assumption, it is easy to show that homogeneous coordinates provides higher dynamic range than regular coordinates since the each term is actually multiplication of two terms as αC. If both α and C has the same dynamic range D, their multiplication has a dynamic range of D^2, which is significantly better.

Additional Discussion: As a conceptual representation, there is really no difference between $[0, 1)$ and $[0, \infty)$ since these two spaces are isometric to each other. Although we sample these domains and turn them into a discrete set of numbers, we can still assume that the domain is continuous by reconstructing missing regions using some kind of interpolation. Moreover, since color terms are still given as αC, can still be large. Therefore, in this work, I sometimes use $\boldsymbol{C} \in [0, 1]^4$ to represent colors. However, if we use $[0, 1]^4$,we also need to be careful with operations to guarantee to stay in $[0, 1]^4$.

4.1.1 OPERATIONS OVER COLOR-VECTORS

Operations over Colors; Vector vs. Points: Most people still think colors in shading computations as positions, however in our shading computations or in our image manipulations software such as Photoshop or GIMP, we treat them as vectors. As explained in the previous section, vectors are not introduced intentionally. They are just by product of historical legacy coming from physically based shading computations, which requires to include color addition. In this section, I demonstrate some additional problems caused by non-native operations over vectors. This problem can be resolved if we treat them as real vectors and allow negative numbers.

Operations over Color Vectors: Since color vectors are really like vectors, their components can be both positive or negative. If a shading language that treats colors as vectors allows both positive and negative values such as Renderman, the system can be used safely without causing any unexpected/unintuitive results as far as users understand that these

are really "vectors". We can still use all operations we use with vectors such as addition, subtraction, matrix multiplication, or rotation. Matrix multiplication and rotations can be used to obtain affects such shift in the wavelength. Although, I do not see any intuitive use of dot product or cross product, they can be useful for some effect that I do not see now. Classical Barycentric operations are also defined on color vectors. The only problem using vectors once all computations are finished, we need to convert them to points.

The Most Common Representation: Most shading languages and image manipulation programs treats colors as points. Unfortunately, they restrict them in $[0,1]^4$. This creates issues in the identification of operations to solve compositing problems efficiently. Before discussing the compositing operations, let me briefly present how colors and images are treated in image manipulation programs such as Photoshop and GIMP. In such programs, images consist of layers, which are really images of same size. Every layer image consists of a set of pixels and every pixel has its color and opacity α. Using classical integer representations of colors, maximum number of each channel is usually given as 255, but it really corresponds to 1.0. Compositing operations are applied to corresponding pictures in two consecutive layers. Without loss of generalization, we can call these layers, background and foreground. Now, we can discuss operations.

Structure of Existing Compositing Operations: Let two corresponding pixels in these two layers are given $(\alpha_0 C_0, \alpha_0)$ and $(\alpha_1 C_1, \alpha_1)$, where former is the pixel from background layer and latter is the pixel from foreground layer with αC and α denote premultiplied color of one channel and opacity of a given pixel. All image operations are currently implemented in the following form in image processing programs

$$\boldsymbol{C} = (w\alpha_1 F(\alpha_0 C_0, \alpha_1 C_1) + (1 - w\alpha_1)\alpha_0 C_0, w\alpha_1 + (1 - w\alpha_1)\alpha_0)$$

where the binary operation F really defines the operation and w provides an additional control usually with a slider. The simplest of these operations is called normal, which is actually "over" operator [1]. In over operator, $F(C_0, C_1) = C_1$. Since over operator is Barycentric and never exceeds to 1, this is closed over $\boldsymbol{C} \in [0,1]^4$.

Existing Compositing Operations over $\boldsymbol{C} \in [0,1]^4$: The choice of type of function F defines compositing operations. I list some basic ones below, you can discover rest by comparing before and after an operation in your favorite image compositing program. I

[1] Note that associative over is formulated in a different way to provide associativity.

give the names usually used in compositing software in parenthesis:

1. $F = \alpha_1 C_1$: Over (Normal);
2. $F = \alpha_0 C_0 \alpha_1 C_1$ Multiplication (Multiply);
3. $F = 1 - \alpha_0 C_0 \alpha_1 C_1$ Exclusion (Exclusion);
4. $F = max(\alpha_0 C_0, \alpha_1 C_1)$ Maximum (Lighter Color);
5. $F = min(\alpha_0 C_0, \alpha_1 C_1)$ Minimum (Darken);
6. $F = |\alpha_0 C_0 - \alpha_1 C_1|$ Subtraction (Difference):
7. $F = \alpha_0 C_0 - \alpha_1 C_1$; if $(F \leq 0)$ $F = 0$ Subtraction (Subtract), and;
8. $F = \alpha_0 C_0 + \alpha_1 C_1$, if $(F \geq 1)$ $F = 1$ Addition (Lighter Color).

As discussed earlier, the first five operations can safely be used since they are closed over $\boldsymbol{C} \in [0,1]^4$, although there are small details we need to be careful such as multiplication is really a matrix multiplication. On the other hand, the last three operations are really problematic to be directly used in compositing.

Issue with Subtraction and Addition: Note that truncate, clamps and absolute value in these three operations are introduced to be sure that the range of the function F is always between 0 and 1. Otherwise, resulting color-points cannot be closed under these operations. As discussed earlier, it is expected to have such a problem with subtraction operation. Difference and Subtract are really two quick solutions to this problem. There is nothing we can do about them except using vectors and avoiding these operations over points. On the other hand, the problem with addition (also called linear dodge) operation is not expected since it is a Barycentric operation on points. In fact, this operation is closed over points.

Dynamic Range Problem: Let us demonstrate with an example of adding two red colors as: $(1.0, 0.5, 0.5, 1) + (1.0, 0.5, 0.5, 1) = (2.0, 1.0.1.0, 2.0) \equiv (1.0, 0.5.0.5, 1.0)$ If we apply formal operation the canonical form is the same as the original as expected. On the other hand, if we do not allow numbers larger than 1.0 during computation, we end up with white colors by truncating all number larger than one to one as $(1.0, 0.5, 0.5, 1) + (1.0, 0.5, 0.5, 1) \longrightarrow (1.0, 1.0.1.0, 1.0)$ This can be viewed as a mathematical dynamic range problem, which is conceptually similar to physical dynamic range problem. But unlike physical case, this is not produced because of limitations of our devices. It is produced by our choice of representation. We can easily avoid this problem by allowing the whole range $[0, \infty)$, which is easy to allow by simply using floating range arithmetic during computations. There are always some minor computational inaccuracies caused by discretization, however, these errors do not cause wrong results as gross as turning a red color to white color.

Conclusion: In composting, there is really no magic bullet solution. If we allow the whole range $[0, \infty)$, we can produce α values larger than one, this can later cause not-legal

Over operation, which can cause negative numbers. This really summarizes the nature of the problem. We cannot solve compositing problems with such pseudo-operations that introduce uncontrollable non-linearities. Unless practitioners become aware of algebraic properties of underlying representations, the operations over them can always cause unexpected results. The words like linear dodge or difference or subtract appears to simplify the life of the practitioners, but, in fact, such intuitive looking names further hide the nature of the problem. Practitioners, therefore, end up with ad hoc solutions that result in very complicated models. In the next chapter, I discuss how to circumvent these issues to obtain desired colors using the concept borrowed from geometric modeling.

CHAPTER 5

Image Space Shader Design with Parametric Equations

Introduction: Parametric shapes are useful not only to design 3D shapes but also to design shaders. The information provided in this chapter is well-known in Computer Aided Geometric Design community. I included this chapter to make the book self-contained since this is critical to intuitively understand shader equations, which we use throughout this book.

Parametric Equations for Shader Design: In our framework, shaders are designed as parametric curves, surfaces, volumes or hyper-surfaces that are projected in 3-color-space. I provide a discussion of the structure of the parametric equations in this chapter. I also discuss some critical issues in designing shaders to reconstruct depictions.

The number of Parameters: In shader design, It is important to reduce the dimension as much as possible by identifying the number of independent parameters that can cause some visible effect. The number of independent parameters, in general, can be quite high since almost every material provide some of every possible effect such as specular reflection or subsurface scattering, which are linearly independent. It is therefore critical to identify which effects are really negligible and remove the parameters that correspond those effects.

Linearly Dependent Illumination: Another key issue, related to reduction of number of parameters, is to remove linearly dependent illumination. If we treat each light as a parameter, each light increase the number of parameters, i.e. the dimension of the shader. Unfortunately, in current computer graphics practice shadows are embedded in diffuse shading by truncating negative part of $\vec{C} \bullet \vec{C}_i$. This truncation turn every light linearly independent from each other and we can add significant amount of lights by hoping to get better shading. In later chapters that discuss lights, I demonstrate that we do not need many lights to obtain desired shading distribution. We only need a minimum number of lights to obtain all qualitative different visible shadows.

Shader Coefficients: Another critical issue to design shaders is the coefficients that correspond to control vertices [1] of parametric equation. These control vertices can be four types:

[1] Control vertices are called control points in usual terminology

1. Color-Vectors: $\vec{C}_i$;
2. Transformed Color-Vectors: $S(\vec{C}_{i,0})\vec{C}_{i,1}$;
3. Color-Points: $\boldsymbol{C}_i$; and
4. Transformed Color-Points: $S(\boldsymbol{C}_{i,0})\boldsymbol{C}_{i,1}$;

where $S(\vec{C}_{i,0})$ is a 3x3 scaling matrix and $S(\boldsymbol{C}_{i,0})$ is a 4x4 scaling matrix that are used to represent reflection properties. As discussed earlier, depend on cases these can be any 3x3 or 4x4 matrix. Using scaling matrix simplifies the problem since we can define transformation matrix simply with a single color-vector or a single color-point. I prefer to use color-points or transformed color-points. On the other hand, it is also possible to use color-vectors and transformed color-vectors. But, as I explained before, using vectors needs more care to remove negative numbers without introducing non-linearities.

5.1 PARAMETRIC SHAPES AS SHADERS

Parametric Shapes: Unfortunately, we cannot write a generic equation for parametric shapes that can handle every dimension. The number of parameters actually defines the dimension. For instance, $\boldsymbol{C}(t) = R(t), G(t), B(t)$ is a curve, but $\boldsymbol{C}(s,t) = R(s,t), G(s,t), B(s,t)$ is a surface in color space. An advantage of parametric representation is that we can control the shapes by restricting the domain of the parameters. In this case, vector vs. Barycentric operations play important role to define equations that are useful in different domains. In this chapter, I provide a few essential equations of such parametric shapes that I use frequently throughout the book. An important property of these shader equations of the parametric shapes is that they are actually compositing operations over images (See Figure 5.1 for an example).

5.2 PARAMETRIC CURVES

Parametric Rays and Lines with Vector Equations: This is given as

$$\boldsymbol{C}(t) = \boldsymbol{C}_0 + \vec{C}_1 t.$$

In this equation, if $t \in (-\infty, \infty)$, this equation gives us line, if $t \in [0, \infty)$, this gives us a ray, f $t \in [0, 1]$, this is a line segment. I do not prefer this equation since it does not lend itself well for compositing because of the vector term $\vec{C}_1$, which can have negative components as explained in section 5.6.

Parametric Line Segments with Barycentric Equations: There are two ways to define line segments in Barycentric form. The first form is given as

$$\boldsymbol{C}(t) = \boldsymbol{C}_0(1-t) + \boldsymbol{C}_1 t.$$

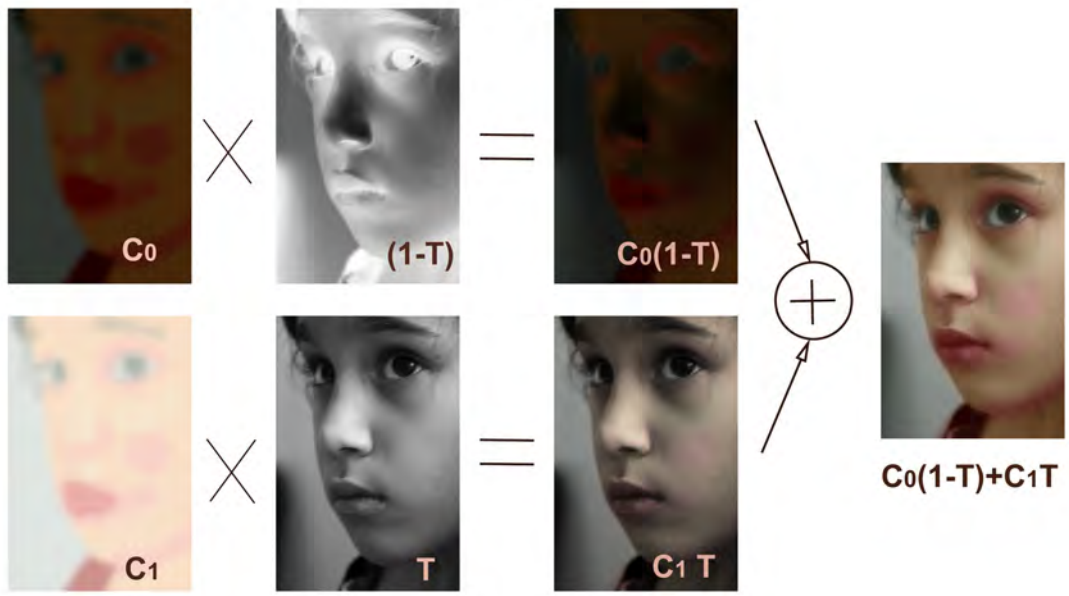


Figure 5.1: *An example of Barycentric line-segment shader in the form of* $\boldsymbol{C}(t) = \boldsymbol{C}_0(1-t) + \boldsymbol{C}_1 t$*.. Note that in this case,* $\boldsymbol{C}_0$ *and* $\boldsymbol{C}_1$ *are different for each pixel, which defines a color image and* t *is computed for each pixel, which defines a black & white image. These images can easily be obtained original image. An important advantage of these decomposition of the parts is that we can easily re-illuminate the image as shown in Figure 5.2*

This equation is Barycentric if and only if $t \in [0, 1]$. Figure 5.1 visually demonstrate how this shader equation works as a compositing operator[2]. Note that t can be considered a rendering under white light for white material. In Figure 5.1, I obtained t by simply turning the original photograph into a B&W image. In this case, t image includes a wide variety of effects such as specular highlights, local shadows and subsurface scattering.

Manipulation with Barycentric Equations: A significant advantage of this approach, it is easy to manipulate the final image just by altering t image. The result will still be consistent with original look as shown in 5.2. In this particular case, I added shadow coming from Venetian blind, by drawing darker strips in t image. Note that for realism, the shadow of Venetian blinds must conform the shape of the face. Therefore, we need proxy geometry to make manipulation consistent with the shapes of original objects, such as shadows conforming the shape of the occluded object. I will discuss simple methods to construct proxy geometry in the next chapter.

[2]Here, C_0 and C_1 are simple hand drawn images. Note that how important the dark looking $c_0(1-t)$ term to obtain the look of color photography.

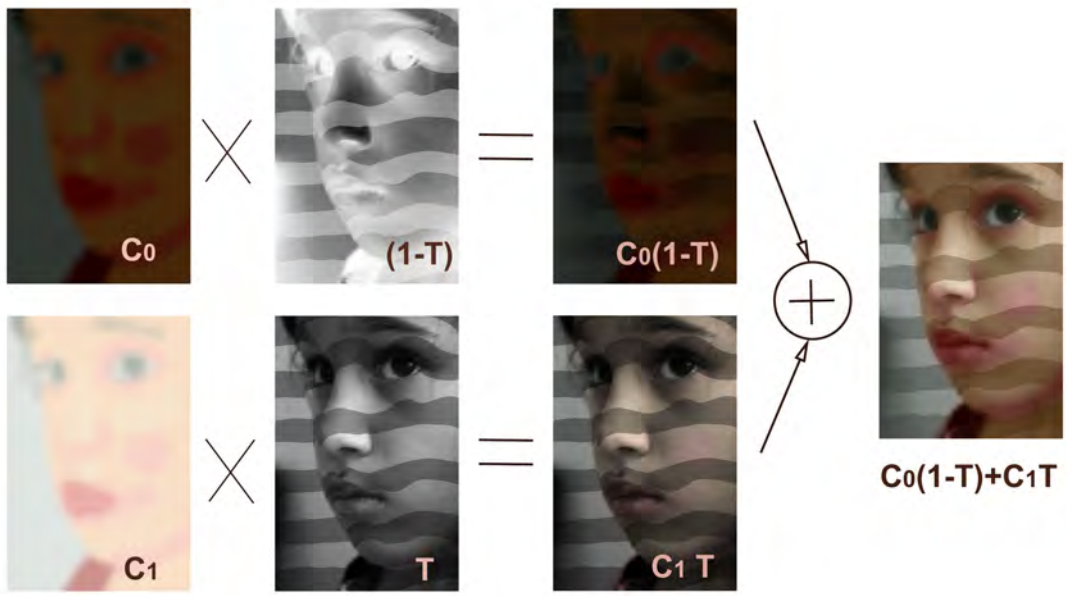


Figure 5.2: *An example that demonstrate updating Barycentric shaders is easy. In this case, I re-illuminated the image as shown in Figure 5.1 by adding shadows produced by some hypothetical Venetian blind. Note that to make shadows correct, we need some proxy shapes, which will be explained in the next section.*

Parametric Lines with Non-Barycentric form: It is possible to use the Barycentric equation as a non-Barycentric form by choosing t outside of 0 and 1 range, in which $t \in (-\infty, \infty)$ gives us a line; and $t \in [0, \infty)$ gives us a ray.

Another Barycentric form: Based on our extension, there is actually another Barycentric form is simpler and it is given as

$$\boldsymbol{C}(t) = \boldsymbol{C}_0 + \boldsymbol{C}_1 t$$

where $t \in [0, \infty)$. This appears to be a ray equation, but it is not since $\boldsymbol{C}_1$ is a color point, as described in the previous section. It is easy to show that this equation is in Barycentric form. Let both $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ be in canonical form, then the actual weights for Barycentric form are $\omega_0(t) = \frac{1}{1+t}$ and

$$\omega_1(t) = \frac{t}{1+t}.$$

In this case, $\omega_0(t) + \omega_1(t) = 1$; $\omega_0(0) = 1$ and $\omega_1(0) = 0$. More interestingly, when t goes to ∞, $\omega_0(t)$ approaches to zero and $\omega_1(t)$ approaches to one. This is very useful property, since we do not have to be careful with the value of t and the result will always be inside of convex hull defined by these two points. Only issue is that we do not exactly interpolate

$\boldsymbol{C}_1$. In this book, I do not use this form, but it is a useful one dealing with, in particular, HDR data.

5.3 SMOOTH CURVES AND SURFACES IN BARYCENTRIC FORM

Higher Degree Curves and Surfaces Barycentric Form: Higher degree curves and surfaces are easy to obtain using basis functions that satisfy partition of unity. More importantly, any parametric rational or irrational polynomial or piecewise polynomial can be converted to a form that satisfies the partition of unity property [80]. In addition, the convex hull property, which comes with partition of unity, is particularly useful in practical shader development applications, since it provides an intuitive control mechanism for obtaining desired results. For instance, a Barycentric equation for surfaces can be given as follows:

$$\boldsymbol{C} = \sum_{i=0}^{M_i}\sum_{j=0}^{M_j} \Omega_{i,j}(u,v)\boldsymbol{C}_{i,j}, \quad \text{where} \quad \forall\, \Omega_{i,j}(u,v) \geq 0 \quad \text{and} \quad \sum_{i=0}^{M_i}\sum_{j=0}^{M_j} \Omega_{i,j}(u,v) = 1,$$

To obtain functions with partition of unity property is very easy recursively starting from a simple basis functions that have already satisfy partition of unity property. For instance, the following basis functions also satisfy partition of unity for all K values that are positive integers

$$\Big(\sum_{i=0}^{M_i}\sum_{j=0}^{M_j} \Omega_{i,j}(u,v)\Big)^K.$$

In fact, basis functions for Bézier curves and surfaces are obtained using this operation.

Rational Bézier Curves in Barycentric Form: The functions $\Omega_0(t) = (1-t)$ and $\Omega_1(t) = t$ are basis functions for degree-1 Bézier curves. Basis functions for higher degree Bézier curves can simply be obtained as

$$(\Omega_0(t) + \Omega_1(t))^K = ((1-t) + (t))^K$$

Let $B_{K,i}$ denote ith basis function of Kth degree Bézier curve with $i = 0, 1, \ldots, K$. Then, $B_{K,i}$ can be computed as

$$B_{K,i} = \frac{K!}{(K-i)!\; i!}(1-t)^{K-i}t^i$$

Then corresponding rational Bézier curve is defined using K number of control points (i.e. coefficients) $\boldsymbol{C}_i$ as follows:

$$\boldsymbol{C}(t) \;=\; \sum_{i=0}^{K} B_{K,i}\boldsymbol{C}_i.$$

The curve is rational since it is defined on points given in homogeneous coordinates. The shape of this rational Bézier curve is intuitively defined and it is a good example of human-mathematics interface (HMI). Although, the equation itself is still a polynomial, coefficients now make sense. In fact, they are, therefore, called control points. The rational Bézier Curves interpolate its end-points $\boldsymbol{C}_0$ and $\boldsymbol{C}_K$, the directions of the tangent vectors at the end-points as $(\boldsymbol{C}_1 - \boldsymbol{C}_0)$ and $(\boldsymbol{C}_K - \boldsymbol{C}_{(K-1)})$, we can also obtain other derivatives in a similar way.

Transformation Independence: An important property of all parametric equations that satisfy partition of unity over points, such as Rational Bézier Curves, is that they are translation, scaling, rotation and perspective projection independent. In other words, if we apply translation, rotation, scaling, or perspective projection operations to all control points, the shapes of resulting curves, surfaces or volumes also translate, rotate, scale and project. Therefore, we can safely represent shapes with control points. Moreover, the operations on control points result in shapes that are intuitively corresponding the original shapes. This independence from transformation property is also useful for shader development.

Non-Uniform and Rational B-Spline (NURB) Curves in Barycentric Form: Similarly, Non-Uniform B-Splines can be obtained by a recursive operation that can produce basis functions that can satisfy partition of unity. The degree zero B-spline basis functions, which are are defined as,

$$N_{0,i}(t) = \begin{cases} 1 & \text{if } t_i \leq t < t_{i+1} \\ 0 & \text{otherwise.} \end{cases}$$

satisfy partition of unity and each t_i is called a knot where $i = 0, 1, \ldots, M$. Moreover, higher degree B-spline basis functions that are defined Cox-de Boor recursion formulation

$$N_{K,i}(t) = \frac{t - t_i}{t_{i+K} - t_i} N_{i,K-1}(t) + \frac{t_{i+K+1} - t}{t_{i+K+1} - t_{i+1}} N_{i+1,K-1}(t),$$

satisfy the partition of unity. The equation of the non-uniform B-spline curve is given as

$$\boldsymbol{C}(t) \;=\; \sum_{i=0}^{K+M} N_{K,i}\, \boldsymbol{C}_i.$$

The curve is again rational since it is defined on points given in homogeneous coordinates. The degree zero and one B-Splines are particularly useful for the development of shaders since they can interpolate control points exactly the way we want.

5.4 PARAMETRIC SURFACES

Parametric Planes Using Vector Equations: These shapes are defined by two linearly independent color vectors $\vec{C}_1$, $\vec{C}_2$ and a color point $\boldsymbol{C}_0$ as

$$\boldsymbol{C}(u,v) = \boldsymbol{C}_0 + \vec{C}_1 u + \vec{C}_2 v.$$

For $(u,v) \in [0,1]^2$, this gives a parallelogram. $(u,v) \in (-\infty,\infty)^2$, this gives a plane that is perpendicular to color $\vec{C}_0 = \vec{C}_1 \times \vec{C}_2$. In other words, this is another formulation of the implicit plane $\vec{C}_0 \bullet (\boldsymbol{C} - \boldsymbol{C}_0)$. As we will later, the shapes that can be represented by both types of equations are useful to solve some problems.

Triangles Using Barycentric Equations: Triangles that interpolate three color points $\boldsymbol{C}_0$, $\boldsymbol{C}_1$, and $\boldsymbol{C}_2$ that are given in three different Barycentric forms and they are not in the same line. The first and most common form is given as:

$$\boldsymbol{C}(u,v) = \boldsymbol{C}_0(1-u-v) + \boldsymbol{C}_1 u + \boldsymbol{C}_2 v.$$

To satisfy Barycentric rule $0 \le u+v \le 1$, $0 \le u \le 1$ and, $0 \le v \le 1$. The second form, which is a version of bilinear form, is given as

$$\boldsymbol{C}(u,v) = (\boldsymbol{C}_0(1-u) + \boldsymbol{C}_1 u)(1-v) + \boldsymbol{C}_2 v.$$

One advantage of this form, we do not have to be that careful to satisfy Barycentric rule. As far as $0 \le u \le 1$ and $0 \le v \le 1$, Barycentric rule is satisfied. Another advantage is that it is suitable for adding specular reflection exactly the places we want with complete control as shown in Figure fig:ch004operation3. The third and simplest triangular form is given as

$$\boldsymbol{C}(u,v) = \boldsymbol{C}_0 + \boldsymbol{C}_1 u + \boldsymbol{C}_2 v$$

where $(u,v) \in [0,\infty)^2$. This particular form is simplest since we do not have to be careful with parameters to stay in triangle. However, the distribution of the parameters inside of the triangle is complicated. When u goes to ∞, the equation approaches to $\boldsymbol{C}_1$; v goes to ∞, the equation approaches to $\boldsymbol{C}_2$; and more interestingly both u and v approaches to ∞, the equation approaches to the middle point of $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$. On the other hand, this form provides more natural specular highlights by eliminating dark regions as shown in Figure 5.4. Regardless of which form we use, if the three points are not in the same line, we always obtain a triangle, which will give a larger variety of colors.

Bilinear Surfaces Using Barycentric Equations: Bilinear surfaces that interpolates four points $\boldsymbol{C}_{00}$, $\boldsymbol{C}_{01}$, $\boldsymbol{C}_{11}$, and $\boldsymbol{C}_{10}$ can also be given more than one Barycentric forms. The first and most popular form is given as

$$\boldsymbol{C}(t) = \boldsymbol{C}_{00}(1-u)(1-v) + \boldsymbol{C}_{10}u(1-v) + \boldsymbol{C}_{11}uv + \boldsymbol{C}_{01}(1-u)v.$$

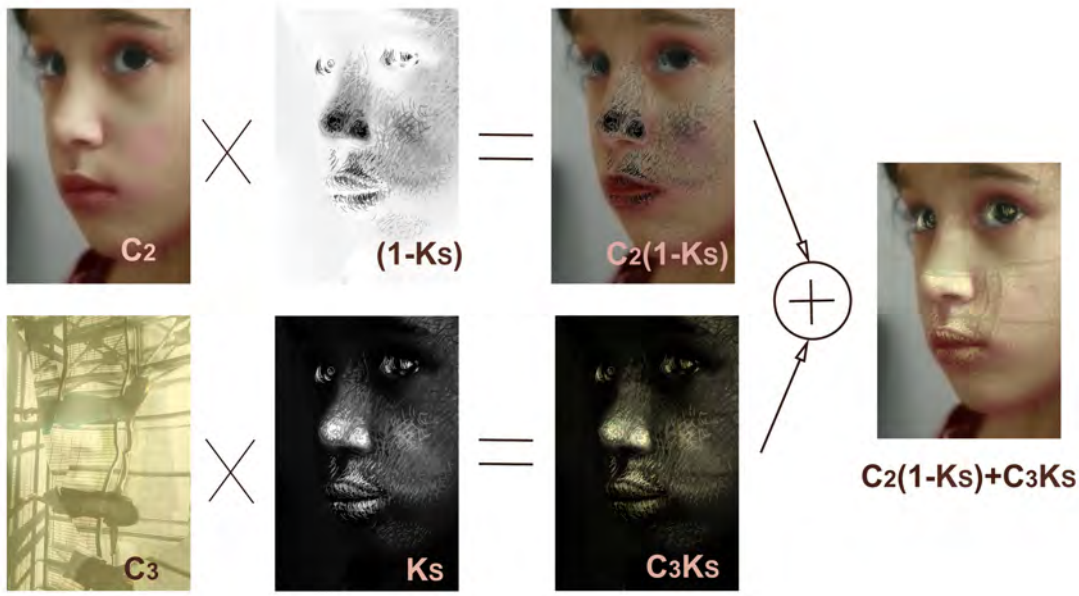


Figure 5.3: *An example that demonstrate a triangular Barycentric shader in bilinear form, where $C(T) = (\boldsymbol{C}_0(1-u) + \boldsymbol{C}_1 u)$ is the image in Figure 5.1 with $u = T$ and and $v = K_s$ specularity coefficient. In this case, I added specular reflection of the environment to the image as shown in Figure 5.1 by mixing it an image that represent specular reflection of the environment, called C_2. Note that to update C_2 to add specular reflection of newly added object, we again need some proxy shapes. Designing proxy shapes will be explained in the next section.*

To satisfy Barycentric rule $(u, v) \in [0, 1]^2$. The second form is given as

$$\boldsymbol{C}(t) = \boldsymbol{C}_{00} + \boldsymbol{C}_{10}u + \boldsymbol{C}_{11}uv + \boldsymbol{C}_{01}v.$$

where $(u, v) \in [0, \infty)^2$. The domain of this form is again simple and do not require any special care. Note that, in this case when both u and v goes to ∞, $\omega_{1,1}(u, v) = \frac{uv}{1+u+v+uv}$ approaches to one when all other ω terms approaches to zero. Both equations, therefore, define a bilinear shape with vertex positions $\boldsymbol{C}_{00}$, $\boldsymbol{C}_{01}$, $\boldsymbol{C}_{11}$, and $\boldsymbol{C}_{10}$.

Rational Bézier Surfaces in Barycentric Form: This can be generalized to surfaces using basis functions for triangles and quadrilaterals. For instance, if we choose $\Omega_{0,0}(u, v) = (1 - u - v)$, $\Omega_{1,0}(u, v) = u$ and $\Omega_{0,1}(u, v) = v$, then $(\Omega_{0,0}(u, v) + \Omega_{1,0}(u, v) + \Omega_{0,1}(u, v))^K$ gives us basis functions DeCasteljau (triangular) surfaces. Similarly, if we choose $\Omega_{0,0}(u, v) = (1 - u)(1 - v)$, $\Omega_{1,0}(u, v) = u(1 - v)$, $\Omega_{0,1}(u, v) = (1 - u)v$, and $\Omega_{1,1}(u, v) = uv$, then $(\Omega_{0,0}(u, v) + \Omega_{1,0}(u, v) + \Omega_{0,1}(u, v) + \Omega_{1,1}(u, v))^K$ gives us basis functions of Bézier (tensor-product) surfaces. Using this approach, it is easy to obtain higher

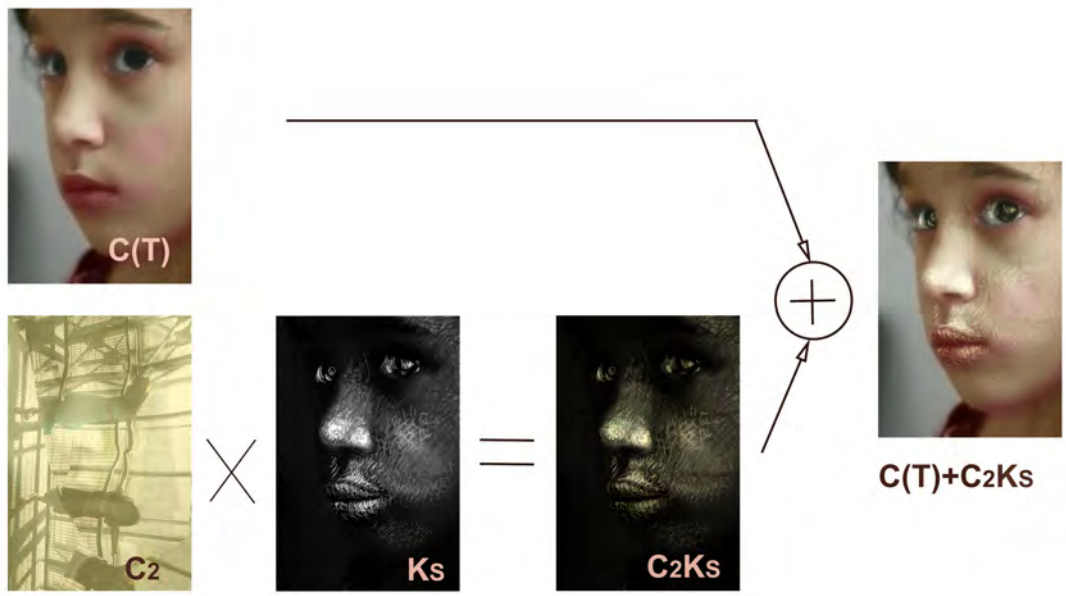


Figure 5.4: *An example that demonstrate another triangular Barycentric shader in linear form,* $\boldsymbol{C}(T, K_s) = \boldsymbol{C}(T) + \boldsymbol{C}_2 K_s$ *where* $C(T) = (\boldsymbol{C}_0(1-u) + \boldsymbol{C}_1 u)$*, which is the image in Figure 5.1 with* $u = T$ *and and* $v = K_s$ *specularity coefficient. In this case, the environment reflection and* K_s *parameter is the same as the ones shown in Figure 5.3 and* $+$ *is a Barycentric operator over color points. On the other hand, the dark regions of reflections are removed since we are simply adding color-points.*

dimensional smooth structures. Using methods similar to Bézier formulation, we can obtain B-spline surfaces, volumes and other higher dimensional shapes.

5.5 HIGHER DIMENSIONS

Generalization to Higher Dimensions: Depending on how many independent variables we have and the structure of extreme values based on these values, we can use a wide variety of high-dimensional parametric equations that represent high-dimensional shapes. Although with colors we usually deal with 3 dimensions, our shapes can simply be projections of higher dimensional shapes. They are especially useful since we may want to interpolate more than four points. This is particularly important for shaders. The whole problem is to define, as discussed above, a well defined domain for parameters. Most useful domain for the interpolation of 2D positions of an arbitrary number of points is mean value coordinates [146]. The linear and bilinear equations can easily be generalized to obtain equations of their higher dimensional counterparts.

Simplices Using Barycentric Equations: Simplices are higher dimensional extensions of lines and triangles. A M-simplex is a M-dimensional polytope, which is defined as the convex hull of its M+1 points. From the definition it is clear that an M-simplex interpolates its M+1 end-points denoted by $\boldsymbol{C}_i$, where $i = 0, 1, \ldots, M$. Simplices can again be given in three in there different Barycentric forms. The first and most common form is given as:

$$\boldsymbol{C}(t) = \sum_{i=0}^{M} \boldsymbol{C}_i \; t_i.$$

In this case it is really hard to satisfy the specific Barycentric rule in which $\sum t_i = 1$. The second form, which is a high dimensional version of bilinear form, is given as iteratively as follows:

$$\begin{aligned} \boldsymbol{p'}_1(t) &= \boldsymbol{C}_0(1 - t_1) + \boldsymbol{C}_1 t_1 \\ &\ldots \\ \boldsymbol{p'}_i(t) &= \boldsymbol{p'}_{i-1}(1 - t_i) + \boldsymbol{C}_i t_i \\ &\ldots \\ \boldsymbol{C}(t) &= \boldsymbol{p'}_{M-1}\,(1 - t_M) + \boldsymbol{C}_M \; t_M \end{aligned}$$

One advantage of this form, we do not have to be that careful to satisfy Barycentric rule. As far as $0 \leq t_i \leq 1$, Barycentric rule is satisfied. This gives us an additional advantage of controlling hierarchy, which is particularly important for shaders. The third and simplest form is given as

$$\boldsymbol{C}(t) = \sum_{i=0}^{M} \boldsymbol{C}_i \; t_i.$$

where $t_i \in [0, \infty)^2$. This particular form is simplest since we do not have to be careful with parameters to stay in M-simplex. Homogeneous coordinates automatically provide Barycentric property. Another advantage is that this equation is naturally commutative like natural summation. Commutativity is essential for applications such as computing photon density estimations [166] since the parameters are produced in no particular order in the rendering system. However, the distribution of the parameters inside of the M-simplex is complicated. Although an M-simplex is an M-dimensional object, since we only interpolate points in 3D, the resulting object is a complex polyhedron in 3D, which is a projection of M-simplex to 3 space. Another problem with the third form is that it does not interpolate the control points. So, in current practice, I suggest to use second form to represent simplices.

Hypercubes Using Barycentric Equations: Using generalization of bilinear form, once can creates hypercubes. For instance, hexahedrons with bilinear faces can be described by tri-linear equations. These can also be given more than one Barycentric forms. Handling

parameter space is also easier since they can stay inside of hypercubes in the form of either $[0,1]^M$ or $[0,\infty)^M$. Using higher dimensional forms could be necessary in some shading applications, in which linear formulations provided by simplex form do not have sufficient power. In the examples of this book I at most need to use Bilinear or tri-linear forms.

Others: It is also possible to have forms that can provide other shapes. For instance, if we combine a bilinear form with a simplicial form we can obtain a pyramid shape with a quadrilateral base. Similarly, combining a generic 2D polygon described by mean value coordinates [146] with an additional dimension, we can obtain general prisms or pyramids. In practice, I do not believe that we will ever need any of these relatively complicated shaders and I think hyper-cubes and simplices will be sufficient for most applications. In the following examples, I also show that it is better to use color-points and Barycentric forms.

5.6 EXAMPLES

Introduction: In this section, I give two examples that demonstrates that if we want to use color-vectors, we need to allow negative numbers to solve the problems with color-vectors. Color-points, on the other hand, do not have to be negative to solve the problems as shown in these examples. In this book, therefore, I work mainly with color-points and Barycentric operations over color-points. Since I use Barycentric operations, I can ignore the term w conveniently since it is always 1 under Barycentric operations. This section also provides a practical approach to choose extreme values, i.e. extreme color-points.

Example 1: Figure 5.5 demonstrates one example that requires using negative colors to obtain close match. If we choose to use a ray equation in the form of by focusing on only one light as

$$\boldsymbol{C}(t) = \boldsymbol{C}_0 + \vec{C}_1 \;\; t$$

where the parameter $t = 0$ means corresponding shading point is in shadow for the given light and parameter $t = 1$ means corresponding shading point is illuminated by the light. $\boldsymbol{C}_0$ corresponds to emitted radiance and $\boldsymbol{C}_1$ transformed illumination color. Since this is really a very simple algebraic equation, we can easily solve it for the photo in Figure 5.5.1 and we obtain:

$$\boldsymbol{C}(t) = (214, 66, 90, 1) + (-14, 66, 119, 0) \;\; t$$

where we consider that x, y and z correspond to red, green, and blue color channels respectively. Note that red channel of vector $\vec{C}_1$ turn out to be negative number to obtain exact interpolation. This is not a theoretical problem since we deal is "relative-color" instead of "absolute-color". Negative numbers are direct result of using relative entities instead of absolute. However, most users cannot easily comprehend this fact. It is also hard to develop

(a) *An unaltered photograph.* (b) *Detail around shadow boundary.*

Figure 5.5: *This figure shows an unaltered of Mr. Potato head illuminated by three colored lights and color values in one of the shadow boundary region. As shown in the detail image, the red channel in the shadow region is brighter than in the non-shadow region.*

an intuition with negative colors. Fortunately, it is possible to resolve this issue by using a Barycentric equation of a line segment in the form of

$$\boldsymbol{C} = \boldsymbol{C}_0 \ \ (1-t) + \boldsymbol{C}_1 \ \ t$$

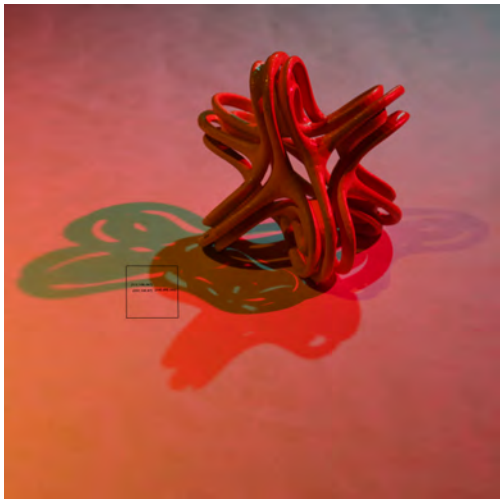
In this case, again the parameter $t = 0$ means corresponding shading point is in shadow and parameter $t = 1$ means corresponding shading point is illuminated. This is even simpler equation to solve than before. If we solve it for the photo in Figure 5.5.1, we obtain:

$$(r, g, b, 1) = (214, 66, 90, 1) \ \ (1-t) + (200, 132, 209, 1) \ \ t$$

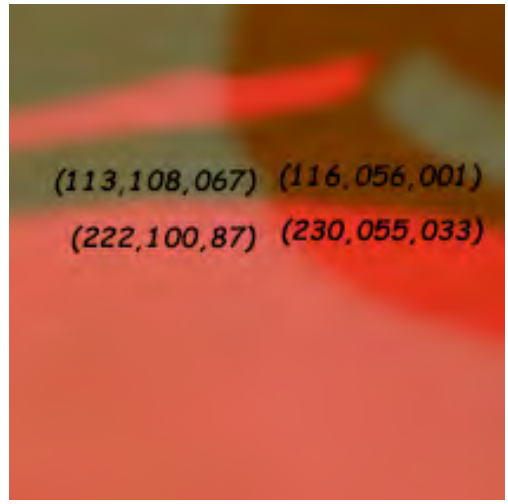
In this case, all coefficients are positive since we only deal with points. These two equations are not really different as it can be seen by replacing $\boldsymbol{C}_1$ with $\boldsymbol{C}_1 = \boldsymbol{C}_0 + \vec{C}_1$. Using points is important only for guaranteeing to use positive numbers as coefficients of the equations. Note that $\boldsymbol{C}_0$ and $\boldsymbol{C}_1$ are actually extreme values.

Example 2: Figure 5.6 demonstrates a more complicated example that again requires using negative colors to obtain close match. In this case, in the region we are interested in there are shadows created by two light sources. If we choose to use classical shading equation, we have to use a plane equation in the form of:

$$\boldsymbol{C}(u, v) = \boldsymbol{C}_0 + \vec{C}_1 \ \ u + \vec{C}_2 \ \ v$$



(a) *An unaltered photograph.*



(b) *Detail around shadow boundary.*

Figure 5.6: *This figure shows another unaltered photograph and color values in shadow boundaries. In (3) green channel in one shadow regions and red channel in another shadow region are brighter than green and red channels in the illuminated region. To obtain the same colors, we need negative colors if we use vector algebra.*

where the parameter u corresponds to light 1 and parameter v corresponds to light 2. $u = 0$ means corresponding shading point is not visible by light 1. $u = 1$ means corresponding shading point is visible by light 1. $v = 0$ means corresponding shading point is not visible by light 2. $v = 1$ means corresponding shading point is visible by light 2. Based on this information, we can solve it for the photo in Figure 5.6.1 using sample points $u = 0, v = 0$, $u = 1, v = 0$ and $u = 0, v = 1$ and ignore $u = 1, v = 1$, by hoping that these points are on the same plane.

$$\boldsymbol{C}(u, v) = (116, 56, 1, 1) + (-3, 52, 66, 0)\ \ u + (114, -1, 32, 0)\ \ u$$

Note that red channel of color vector $\vec{C}_1$ and green channel of color vector $\vec{C}_2$ turned out to be negative number to obtain exact interpolation of three points. Now, if the three points are in the same plane, $u = 1, v = 1$ should give us $(222, 100, 87)$. If we compute it, we find:

$$\boldsymbol{C}(1, 1) = (116, 56, 1, 1) + (-3, 52, 66, 0) + (114, -1, 32, 0) = (227, 107, 99) \neq (222, 100, 87)$$

This result may appear to be reasonably close, however, these four colors obviously are not in the same plane. In other words, in this case we deal with an additional issue to negative colors. The linear model cannot represent this case. Fortunately, it is again possible and easy to resolve this issue by using a Barycentric Bilinear equation in the form of

$$\boldsymbol{C}(u, v) = \boldsymbol{C}(0, 0)\ \ (1 - u)(1 - v) + \boldsymbol{C}(1, 0)\ \ u(1 - v) + \boldsymbol{C}(1, 1)\ \ uv + \boldsymbol{C}(0, 1)\ \ (1 - u)v$$

where $\boldsymbol{C}_{0,0} = \boldsymbol{C}(0,0) = (116, 56, 1, 1)$, $\boldsymbol{C}_{1,0} = \boldsymbol{C}(1,0) = (113, 108, 67, 1)$ $\boldsymbol{C}_{1,1} = \boldsymbol{C}(1,1) = (222, 100, 87, 1)$ $\boldsymbol{C}_{0,1} = \boldsymbol{C}(1,1) = (230, 55, 33, 1)$ In this case, we perfectly interpolate the colors of four sample positions and all coefficients are positive since we only deal with points. Note that in this case $\boldsymbol{C}_{0,0}$, $\boldsymbol{C}_{0,1}$, $\boldsymbol{C}_{1,1}$, and $\boldsymbol{C}_{1,0}$ are actually extreme values.

CHAPTER 6

Image-Space Shape Reconstruction for Compositing

The Need for Shape Reconstruction: Since global illumination requires object-to-object interactions, we need to have virtual representations of shapes and positions of all objects including lights in the photographs, paintings and illustrations to obtain the global illumination effects. We call these virtual objects "proxy geometry" since they are not really visible. Their impact becomes indirectly visible through shadows and reflections. As discussed earlier, it is impossible to reconstruct the true 3D shapes of these virtual objects. In this chapter, I demonstrate that there exists alternative image-space representations that can efficiently be used as proxy geometry. It is also easy to construct these 2.5D image space shape representations. The Figure 6.1 shows that it is possible to obtain physically almost-correct results with extended normal maps are used as proxy refractive objects. Figure 6.2 shows an image space shape representation that provide both opaque and transparent regions. As we will see later, a single normal map is sufficient to provide object thickness for transparent objects, which is necessary to obtain realistic looking refraction effects.

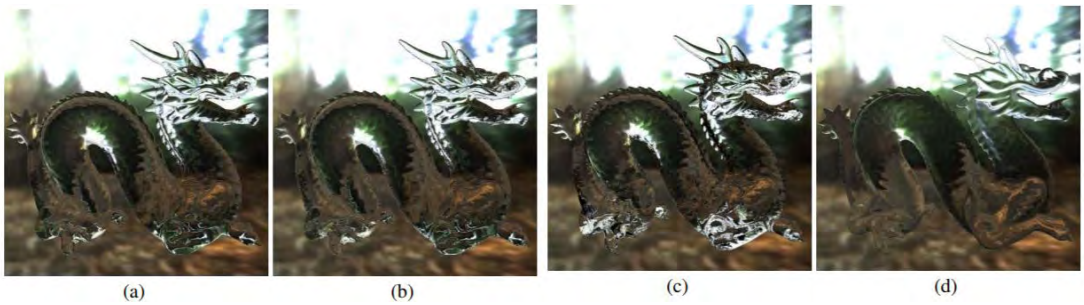


Figure 6.1: *This example demonstrate that with qualitative image space representations it is even possible to obtain physically almost-correct results. In this example, only (c) is true 3D refraction obtained by by ray tracing a 3D shape [221]. The rest of them are obtained by image-space refraction by using Mock-3D shapes (See [221, 266, 296, 297] for more examples)*

Qualitative Shape Reconstruction with Masked-3D: For our applications, there is a need for a simple reconstruction methodology to convert real objects into virtual representations that can be used as proxy geometry for computing object-to-object interactions. In my Digital compositing class, make students use the concept of masked-3D shapes since 2008. Masked-3D shapes significantly improved the workflow of proxy geometry creation. Basically, masked-3D shapes consists of simple 3D or 2.5D objects that are masked. Unfortunately, this method is hard to use in automatic reconstruction. I, instead, propose to use even simpler shape representations based on a formalization of image-space approaches [296, 297, 298], which I call Mock-3D shapes [277].

(a) *Normal Map.* (b) *Transparency Map.* (c) *Composited Results.*

Figure 6.2: *An example that demonstrate mixed diffuse and transparent objects using Mock-3D. This particular image also demonstrate that these approach can also be used in very unusual setting even in Non-Photorealistic rendering since it is not strictly based on Physics.*

Qualitative Shape Reconstruction with Mocked-3D objects: Many researchers including myself demonstrated that it is also possible to use image-space reconstructions of real objects, Mock-3D objects, as proxy geometry [177, 220, 250, 277, 296, 297, 298]. Mocked-3D objects does not require any changes in existing graphics pipeline while significantly simplify inclusion of global effects [1].

Depth Maps vs. Normal Maps: As shown in Figures 6.5 and 6.7, we consider two types of Mock-3D shapes: Depth Maps and Normal Maps. We observe that Depth Maps seems to be more useful to represent primarily diffuse or mirror objects [65] and Normal Maps are more useful to represent refractive objects as proxy geometries [277, 282, 307]. An additional transparency map is sufficient to define transparency amount. It is also easy to include the deformation caused by index of refraction and thickness using normal maps

[1] Note that depths and distances can be significantly large. In these cases, we, in computer graphics, usually use monotonically decreasing perspective-motivated mapping $z' = 1/(z+1)$. As discussed earlier, this mapping makes large distances goes to zero and 0 distance goes to 1. Wang & Akleman has shown that such monotonically decreasing change in depth defines monotonically decreasing transformation on $\cos\theta$ [277] and it can safely be used for shading. Moreover, many researchers including Akleman & Wang have shown that it is possible to obtain global illumination effects including subsurface scattering using such simple models [65, 221, 266, 277, 282, 296, 297].

[307]. In conclusion, we propose to reconstruct diffuse and/or mirror objects using depth maps and refractive objects with normal maps.

6.1 INTRODUCTION

Summary: In this chapter, I discuss proxy shapes that are used to represent shapes of the real objects in the scenes. As discussed in the previous section, to manipulate images by adding shadows or reflections, we need to have proxy shapes that can be used to re-illuminate the original objects in the scene. I also provide information how to reconstruct these shapes so that they can easily be used for practical compositing. The key global impacts of newly added objects to the existing background is that their shadows need to be casted on objects of background scene and their reflection and refraction should appear if the existing objects in the scene are reflective or transparent. To obtain acceptable results the shadows must conform the shape of the objects and reflections and refractions must provide appropriate deformations.

Proxy Shapes for Casting Shadows and Computing reflections & Refractions: We, therefore, need a somewhat actual geometry to make shadows appear to be conform the shapes of the background objects. As discussed in the previous section, I observe that it is better to avoid real 3D shapes as proxy geometry to represent objects in background scene. Instead, I propose to use either Mock-3D shapes or Masked-3D shapes. Mock-3D shapes simply provide height fields and object boundaries. On the other hand, Masked-3D shapes are projected simple 3D shapes that are masked using α maps.

Offline vs. Online: Height fields are particularly useful since they can directly be reconstructed as depth maps by using any depth camera such as Kinect [176] or directly from motion [233]. Therefore, for real time applications, height fields are particularly useful. The other ones such as masked-3D or normal maps are useful for offline applications where we do not have much information of the shapes.

Camera Parameters: An important implication of using masked-3D and mock-3D shapes, we can simply ignore camera parameters since the projections to create image space representations already provide the camera position and field of view. In other words, camera parameters are already embedded in image space representations. Of course, newly introduced artificial object needs to be rendered using a reasonably correct camera, but camera parameters do not have to be precise. The camera parameters also impacts the reflections and refractions, however, as we discuss later it is not really that important. having camera parameters embedded image space representations is very important since it suggests that there is really no need to have precise camera reconstruction to obtain qualitatively acceptable global illumination. Therefore, I completely ignore camera calibration

and classical reconstruction problems of computer vision in this book. There already exists extensive amount of research and publications on those problems such as [232, 270, 319].

6.2 MASKED-3D OR MOCK-3D AS PROXY SHAPES

Imperfect Shapes of the 3D Objects: The real 3D objects are not perfect. For instance, a mathematically perfect sphere or cube never exist in 3D space. Every real object has some imperfections. To include global illumination effects such as shadows, it is important to model discontinuities which exist in the boundaries.

Difficulty with Unknown Transformations: Even if we faithfully reconstruct the 3D shape of the object, it is still hard to use it in compositing since it is hard to obtain the exact transformations that can produce the exact silhouettes that cause shadow discontinuities, which we discusses in chapter 10.

Need for Mock-3D and Masked-3D Shapes: It is, therefore, important to provide exact silhouettes that can produce shadow discontinuities, which will be important to obtain believable shadows. Masked-3D refers simple 3D shapes that are masked in 2D to obtain exact silhouettes. On the other hand, I use the term Mock-3D to refer shapes that appear to be 3D but not really 3D. These Mock-3D shapes can be used for placeholders of real objects as proxy geometry in digital compositing application.

6.3 MODELING WITH MASKED-3D SHAPES

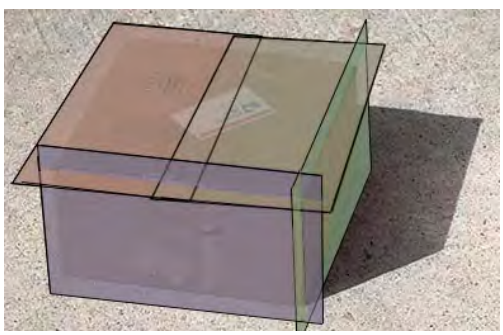
Masked-3D Shapes for Compositing with Photographs: For compositing with photographs, Masked-3D shapes are useful since we can use a rough approximation of the original shapes by simple shapes such as planes or spheres and just mask them to match silhouette edges that are important for compositing. To get best results, for each smooth surface visible in the scene, we need to create a 3D shape and mask the boundaries in 2D. The advantage of masked-3D shapes is that they are easy to model using simple shapes and masks. They can later to turn either a depth map or normal map for rendering.

Example of Masked-3D Shapes: Let us assume that we have a photograph of a cardboard box as shown in Figure 6.3. Even if the box is not even slightly beaten up, the boundaries and edges are not perfectly straight as shown in Figure 6.3b. Although, rectangular prism is the most logical choice for representing such a box, if we use a rectangular prism it is impossible to match discontinuous regions correctly. It is, therefore, easier to represent this particular shape using overlapping planes as shown in Figure 6.4a. We mask the boundaries later in 2D to obtain imperfect edges of the cardboard box as shown in Figure 6.4b.

(a) *A photograph of a cardboard box.*

(b) *Straight red lines shows that the boundaries are not straight.*

Figure 6.3: *An example demonstrating that even very simple objects do not have perfect straight discontinuities in silhouette and boundary regions. Even if the boundary regions were perfect, it would still be impossible to match the orientations.*



(a) *Four planes in 3D to represent visible faces of cardboard box.*

(b) *Four planes masked in 2D to represent boundaries.*

Figure 6.4: *An example demonstrating how masked simple 3D shapes can be used to represent discontinuities. The normals to these planes do not have to be exact, since we only need these planes for casting shadows. Even if we needed to use these planes for shading purposes, the results will not be significantly different since small perturbations of the normal vector do not change shading significantly.*

Masked-3D Shapes as Image-Space Representations: Masked-3D shapes can originally be represented by polygonal meshes with boundaries. The shape boundaries in 3D defines an approximate version of 2D $\mathcal{D}$ and vertex positions provides a discrete height field. Since we eventually mask each shape, these 3D shapes can be considered as layers. These simple 3D shapes can be constructed in any 3D modeling program and turn into images by rendering them as either height or normal maps.

6.4 REPRESENTATION WITH MOCK-3D SHAPES

Definition of Mock-3D Shapes: Mock 3D shapes are image space representations that can simply be considered as functions over a given domain. The most useful ones are height fields or depth maps (See Figure 6.5 for some visual examples) that can formally be defined as follows: Let $\mathcal{D} \in \Re^2$ and $F(x, y)$ denote a function from $\Re^2$ to $\Re$. Using this function, a mock 3-D shape is simply defined by an implicit inequality as

$$S = \{(x, y, z) \ | \ F(x, y) \leq z \ \text{ and } \ (x, y) \in \mathcal{D}\}.$$

In other words, for representing Mock-3D shapes we need two entities: a function and a subset of 2D space.

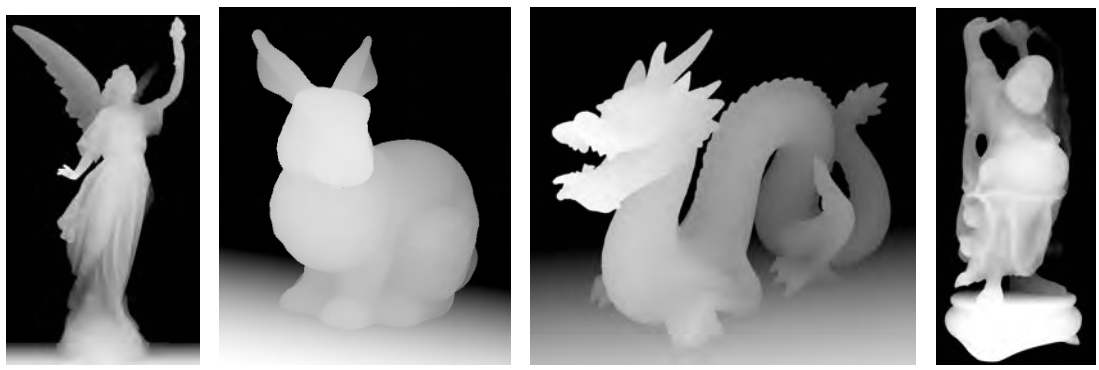


Figure 6.5: *Examples of depth maps as Mock-3D shapes. These images are created by Fermi Perumal from actual 3D polygonal models [235].*

Mock-3D Shapes as Bas-reliefs: Mock-3D shapes conceptually resemble most to bas-reliefs. Bas-reliefs are sculptures that can be viewed from many angles with no perspective distortion as if they are just images. In other words, perspective transformation is embedded in bas-relief sculptures [286]. Similar to bas-reliefs, the perspective transformation is embedded in $F(x, y)$. However, in bas-reliefs $\mathcal{D}$ is usually is a rectangle. In Mock-3D case, $\mathcal{D}$ can have any 2D shape. In this case, normal vector can simply be computed using ∇F as follows

$$(n_0, n_1, 1) = \left(\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, 1\right).$$

Since actual normal vector $\vec{N}$ must be a unit vector, we obtain

$$\vec{N} = (N_0, N_1, N_2) = \left(\frac{n_0}{\sqrt{n_0^2 + n_1^2 + 1}}, \frac{n_1}{\sqrt{n_0^2 + n_1^2 + 1}}, \frac{1}{\sqrt{n_0^2 + n_1^2 + 1}}\right).$$

Mock-3D Shapes as Height Field Images: Height field images can be used even to obtain subsurface scattering as shown in Figure 6.6. For compositing purposes, they are advantageous to polygonal meshes with boundaries, since we can easily include details. A useful property of the images is that $\mathcal{D}$ can also be defined by using the opacity of image, which can even provide sub-pixel details. The opacity map can be created using existing Chroma Key Compositing methods [196, 264]. Moreover, heights can simply be provided by

Figure 6.6: *An example that demonstrate a depth map can even be used to obtain even subsurface scattering. In this case, we used the depth map of the Lucy sculpture shown in Figure 6.5. This particular rendering algorithm is, in fact, very simple (See [65, 235] for the details of algorithm.). There are, in fact, extensive work on such complex global illumination effects with mock-3D shapes such as subsurface scattering [248, 298] and caustics [248, 298].*

single color channel like bump maps. In this case, the height field $z = F(x, y)$ is considered to be given as a discrete field defined each pixel position using discrete set of values provided by corresponding color channel. We obtain the height function by interpolating these values at pixel samples. Since color values in LDR images are between 0 and 1, there is a need for an additional monotonically increasing mapping from color space to heights. These bump maps can be simply used by mapping to rectangular panels. An advantage of Mock-3D height field images is that they can be created using any image manipulation software such as Photoshop or GIMP by simply painting an image. The problem is that it is hard to feel the height just by entering color values.

Mock-3D with 3D Vector Field Images: Another type of proxy Mock-3D geometry is 3D Vector Field Images such as normal maps. Note that the function $F(x, y)$ that is used to define the height field does not have to be explicitly defined; it can be computed from vector field. Let $\vec{n} = (n_0, n_1, n_2)$ denote a normal vector. Then using three color channels of an LDR image, it is possible to describe a discrete normal field defined each pixel position as $n_0 = 2r - 1$, $n_1 = 2g - 1$ and $n_2 = 2b - 1$. We can then obtain a continuous functions by interpolating these discrete values given at pixel samples. Since, $\vec{N}$ is supposed to be unit vector, it is expected that $n_0^2 + n_1^2 + n_2^2 = 1$. Therefore, not every image defines a normal map. Some examples of normal maps that can define proxy geometries are shown in Figure 6.9. In this case, $\mathcal{D}$ can still be defined by using the opacity of image and provide sub-pixel details.

Mock-3D with 2D Vector Field Images: An additional type of proxy Mock-3D geometry is 2D Vector Field Images. True 3D normal vectors are not really necessary for compositing applications, since we do not need to compute local shading for proxy shapes; it is already given by the original/background image. Instead, we need height field to be able to compute global effects that are introduced by newly introduced object(s) such as shadow or reflection. To describe height fields, the first step is the conversion of normal map into a 2D Vector field that provides gradient of a height field in every (x, y) position by assuming that the view direction is z. It is possible to work directly these 2D vector fields to obtain global effects such as shadow, refraction and refraction without reconstructing a height field [277, 282]. Let $\vec{N} = (N_0, N_1)$ denote the 2D vector field, then the two components of this vector field is simply computed as

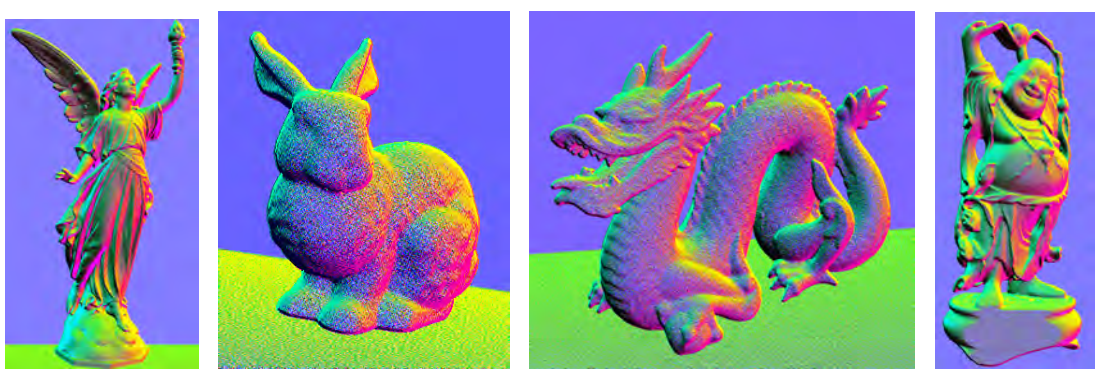


Figure 6.7: *Examples of normal maps as Mock-3D shapes. These images are created from depth map images as gradient fields shown in Figure fig:ch020depthmapexamples with gradient transformation.*

$$N_0 = \frac{n_0}{n_2} \text{ and } N_1 = \frac{n_1}{n_2}$$

Note that N_0 and N_1 can be any real number now and we can view $\vec{N}$ as a function

$$\vec{N} : \mathcal{D} \leftrightarrow \Re^2$$

Gradient Fields: One property of vector fields is that they are not necessarily conservative, i.e. if the vector field is not a gradient field in the form of ∇F, there is no solution for height field $F(x, y)$ (See gradient field normal maps in Figure 6.7). Therefore, the normal maps are not necessarily correspond to conservative 2D vector fields. There is, therefore, no guarantee that existence of a unique height field corresponding to normal maps that are produced by existing methods. Fortunately, this theoretical problem can be considered an advantage for some compositing applications. Since vector fields can represent impossible shapes [277, 282], this property is especially useful if we deal with images that include impossible objects.

6.5 MODELING OF MOCK-3D SHAPES

6.5.1 HEIGHT FIELDS IMAGES

Extended Bas-Reliefs: To turn bas-reliefs into Mock-3D shape, each object in the original scene need to be represented by single bas-relief and boundaries need to be explicitly defined. They can simply be represented as bump maps with opacity. Each bump map can simply be mapped into a quad or a single 2-manifold with boundary [47]. It is possible to view each piece as a "layer" just to be consistent with standard terminology used in image manipulation since they will appear like layers to the most casual users.

Figure 6.8: *Examples that demonstrate normal maps can be used to obtain ambient occlusion and shadow [277]. In this case, we used the normal map obtained from the Lucy depth map shown in Figure 6.5. Note that the normal maps obtained as gradient fields can be noisy. It is, therefore, better to model them directly [177].*

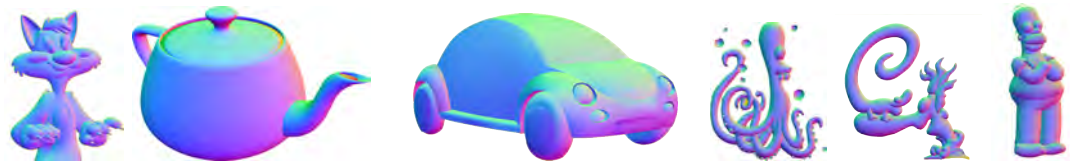
Modeling Impossible Looking Shapes with Bas-Reliefs: Z-depth deformations are introduced by Gershon Elber to represent impossible looking objects. This approach is

particularly useful in this case since it can deform the objects without changing the visible parts [133]. Z-depth deformations are useful since they can turn impossible shapes into "possible" structures. Gradient of such height fields will still be conservative while it visually appears non-conservative. Z-depth deformations can also effectively provide local layering [210], which is important to include cases that cannot be handled by simple ordering such as knots, links and handshakes.

6.5.2 MODELING VECTOR FIELDS

Mock-3D Normal Maps: Normal maps became popular as soon as they are introduced in 1998 [108]. Although,they are mainly used as texture maps to include details to polygonal meshes, they are closely related depth maps since they can be obtained simply as as gradients of depth similar to bump maps [147]. Therefore, any mock-3D shape given by depth can be turned into a normal map as shown in Figure 6.7. However, gradient fields can be noisy as shown in Figure 6.8. It also does not make much sense to create normal maps when we already have height fields. It is, therefore, logical to directly model normal maps independent of depth maps as first observed by Johnston [177].

Modeling Normal Maps: Johnston developed a sketch based system, called Lumo, to model normal maps by diffusing 2D normals in a line drawing [177]. The advantage of his approach is that we can simply model mock-3D representation from a silhouette drawing. The resulting representation also embeds perspective information into the normal map [277]. During 2000's, only a few groups investigated the potential use of normal maps as a shape representation such as [87, 220, 293]. Sun et al. [265] introduced Gradient Mesh to semi-automatically and quickly interpolate normals from edges, and Orzan et al. [223] calculate a diffusion from edges by solving the Poisson equation. Finch et al. [144] build thin-plate splines which provide smoothness everywhere except at user-specified tears and creases. The underlying splines are used to interpolate normals. Wu et al. [295] proposed shape palette, where user can draw a simple 2D primitive in the 2D view and then specify its 3D orientation by drawing a corresponding primitive.



(a) *Two models created by using Lumo [177].*

(b) *A normal map created by using Crossshade [250].*

(c) *Normal maps with quad dominant 2-Manifold mesh modeling system [157].*

Figure 6.9: *Some examples of normal maps generated using sketch based modeling programs.*

Recent Works on Modeling Mock-3D Normal Maps: Recently, a new interest emerged in using Normal Maps for Modeling Mock-3D Normal Maps. Recently, Shado et al. [250] developed CrossShade, another sketch based interface to design complicated shapes as normal maps. They use an explicit mathematical formulation of the relationships between cross-section curves and the geometry. The specified cross-section is used as an extra control point to control the normals. Vergne et al. [276] introduces surface flow from smooth differential analysis, which can be used to measure smooth variations of luminance. Therefore, the author also propose to drawing the shadows and other shading effects. Gonen has also developed a method to construct normal maps using quad dominant 2-Manifold mesh modeling approach that provide more control [157]. Based on the recent results, we can claim that modeling normal maps are easier than modeling height fields directly. To utilize normal map modeling systems for modeling height fields, it is possible consider to convert normal maps into height fields. In the next section, I discuss problems associated with this type conversion and existing approaches.

6.5.3 CONVERSION OF VECTOR FIELDS TO HEIGHT FIELDS

Conversion of 2D Vector Fields to Height Fields: Vector fields are not necessarily come from gradients of height fields. Therefore, they are not necessarily conservative. If the field is non-conservative, there is no corresponding height field and, as a result, we can have a mathematical representation that does not correspond to any real shape (See Figure 6.10c for impossible shapes). Normal maps can even be hand-drawn and unlikely to be conservative (See Figures 6.10a and 6.10b as examples). Unless the corresponding 2D vector field is conservative, a unique constructable geometry does not exist [143]. Fortunately, existing computational methods to model normal maps are designed in such a way that they usually produce conservative vector fields. If the 2D vector field is conservative, conversion is expected to be straightforward by simply taking an integral. On the other hand, these methods do not really guarantee that the results a always conservative. If the vector field is not conservative, then there are two options for conversion.

(a) *Hand-drawn mock-3D normal maps that represent paintings.*

(b) *Hand-drawn mock-3D normal maps that represent objects.*

(c) *Impossible shapes as mock-3D normal maps.*

Figure 6.10: *Examples of hand-drawn and impossible normal maps.*

Approximate Height Fields: Even when the 2D vector field is not conservative, it is possible to find an approximate height field by solving Poisson equation [143]. It is, therefore, always possible to obtain a reasonably acceptable height field corresponding a normal map. For instance, Sýkora et al. [267] developed a user-assisted method to convert normal maps into Bass-Reliefs that can provide and receive correct shadows. One problem with this approach is that it can produce height fields do not necessarily correspond the shape of the original objects.

No Need for Conversion for Global Effects: Our observation is that the conversion from vector fields to height fields is not really necessary for practical applications. Normal maps as well as depth maps can directly be used in computing global illumination effects without any need for conversion. In the next chapter, I briefly explain how to use these two representations to compute global illumination effects.

CHAPTER 7

Computing Global Effects with Mock-3D Shapes

Introduction: For computing global effects introduced by newly added objects, we need the 3D positions and normal vectors for any given pixel in the image and eye position. Since mock-3D representations can provide a good approximation of both of these entities, if the discontinues are perfectly designed, it is possible to obtain acceptable shapes for shadows, reflections and refractions. In this chapter, I briefly present algorithms to compute shadows, reflections and refractions of newly added objects.

Computing Shadows: To identify shadow regions, all we need is the height (or z-depth) information for each pixel. By checking if there is any newly added object between this 3D position and light source, we can identify if that particular pixel is fully or partially occluded by the newly added object. Based on amount of occlusion, we can change the color of the pixel to create illusion of shadow. Changing the color correctly is essential to obtain consistent shadow, which I explain in later chapters and case studies.

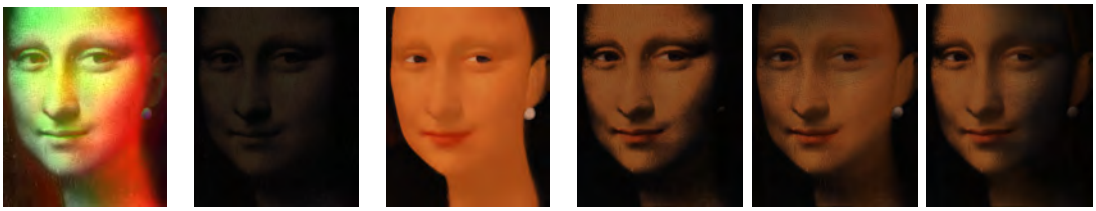
Computing Reflections and Refractions: For reflection and refraction, in addition to position information we also need normal information to compute refracted and reflected ray directions. We also need eye position. Fortunately, as far as reflections are refractions are qualitatively correct, it is acceptable for humans. We do not need precise normals and positions to obtain qualitatively acceptable reflection and refraction. This is understandable since refraction depends on not only the surface normal, but also index of refraction and thickness of refracted object. Like shadows, discontinuities are more important since they introduce qualitative changes. Therefore, as far as we introduce discontinuities of normals and positions in right places, the results look acceptable.

Embedding Camera Parameters: An important property of these proxy shapes is that camera parameters are embedded into masked-3D or mock-3D shape representations. We can simply ignore camera parameters and assume we have parallel projection. This assumption works well as demonstrated all the examples in the book. This is partly because we do not need precise eye position to obtain qualitatively correct reflections or refractions except for planar mirrors. If the image is not cropped, eye position is always on the line passing through the center of the image. Our experience is that almost any point works on that line works for non-planar objects. We, therefore, can us parallel projection by

assuming that the eye is in infinity. In other words, we simply use single view direction that is perpendicular to image plane for all pixels. For planar mirrors, there is a need for a reasonable estimation of eye position. But, it still does not have to be exactly correct.

7.1 COMPUTING SHADOWS

Computing Shadows with Depth Maps: Shadows computation is straightforward with depth maps. The position of a shading point in transformed coordinates is always given as $\boldsymbol{p}_S = (i, j, F(i, j))$, where (i, j) is the pixel position and $F(i, j)$ comes from depth map. Since this is after perspective transformation, which also transform actual depth with a transformation such as $z' = z/(1 + z)$, we also need to transform newly added object and light positions to determine if the shading point is in shadow. For this transformation, we need to provide an estimation of the eye position. This does not have to be exactly correct. By assuming that image is not cropped and center of the image is $(0, 0, 0)$, the eye position can be given as $(0, 0, e)$, where e is a positive number. Note that e simply determine field of view. If we know what kind of the lens is used to take the picture, we can always roughly estimate e. Using this information, we can simply apply perspective transform to the all the vertex positions of the newly added object and light positions. In transformed space, we make occlusion test. If the shading point is occluded, we simply turn contribution from that particular light source into zero and recompute shader. In practice, as I will explain in later chapter, this is even simpler. We do not need to reconstruct the shader parameters of the objects. We can simply include shadow using color multiplication.



(a) *Proxy Geometry: Normal Map.* (b) *Dark Image.* (c) *Light Image.* (d) *Three Renderings.*

Figure 7.1: *An example of manipulating an image by recomputing shading and shadows for a moving light source.*

Computing Shadows with Normal Maps: Shadows computation is not that straightforward with normal maps. We know normal direction in shading position, but we do not know height. Therefore, to compute shading position We simply draw a line from shading point to light. By assuming we know the height in boundary, we take the discrete integral by walking from the boundary point along the line pixel by pixel using a simple line drawing

algorithm. This integral gives us a height value. Then, the shading position in transformed coordinates is again given as $\boldsymbol{p}_S = (i, j, z')$, where (i, j) is the pixel position and z' is the one computed by our discrete integral. We still assume that this is a height obtained by a perspective transformation. Therefore, we also transform newly added object and light positions to determine if this shading point is in shadow. We again make occlusion test in transformed space. If the shading point is occluded, we again recompute contribution from that particular light source and update colors based on shading equations (See Figure 7.1 for example.).

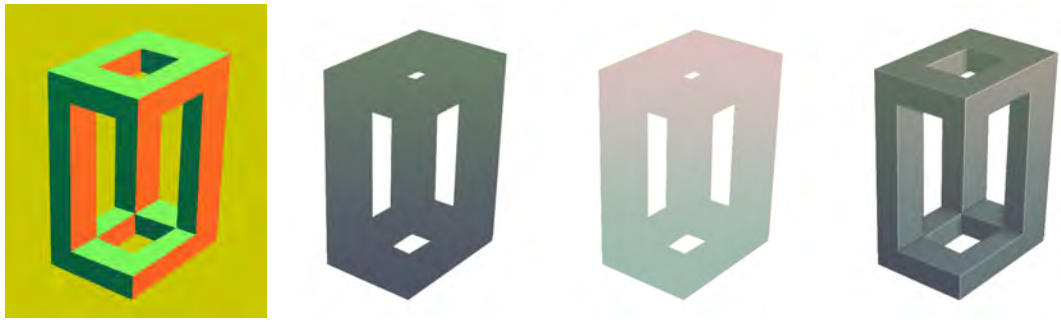
(a) *Proxy Geometry: Normal Map.* (b) *Dark Image.* (c) *Light Image.* (d) *Three Renderings.*

Figure 7.2: *An example of manipulating an image by recomputing shading and shadows for a moving light source.*

Discussion: For photographs, depth maps seems to be better choices to render shadows. On the other hand, for compositing with paintings or drawings, normal maps seems to be better choice as proxy geometry since they can handle impossible or incoherent shapes as shown in Figures 7.1, 7.2 and 7.3. Each of these cases uses a linear Barycentric shader that interpolates two extreme images, one dark and one light. An additional property of Mock-3D shapes is that the occlusion tests can be done directly in image space if the newly added object or the light is completely in the image. We can, then, simply draw a line from shading point to light and walk through along the line pixel by pixel using a simple line drawing algorithm and check of there is any intersection with the newly added object. On the other hand, since the newly added object can still cast a shadow when it is outside of the image boundaries, it is easier to use classical ray-tracing that can handle all cases.

7.2 COMPUTING REFLECTION

Preliminaries: Let $\vec{N}_S = (a, b, c)$ and $\vec{E} = (0, 0, 1)$ denote the unit normal of the shading point and the unit eye vector respectively. Note that since perspective transformation is already embedded, we have to use $\vec{E} = (0, 0, 1)$. Now, remember that the reflection vector



(a) *Proxy Geometry: Normal Map.* (b) *Dark Image.* (c) *Light Image.* (d) *One Rendering.*

Figure 7.3: *An example of rendering an impossible object. If an image includes impossible objects, we can even cast shadow over them using normal maps.*

$\vec{R}$ is computed as follows [147]:

$$\vec{R} = -\vec{E} + 2\left(\vec{N}_S \cdot \vec{E}\right) \vec{N}_S.$$

Using this equation, we compute reflection direction as

$$\vec{R} = (2ac, 2bc, 1 - 2c^2).$$

To compute reflection of newly added object, all we need to do is to compute intersection of the ray starting from the position of the shading point with newly added object. If there is no intersection, nothing changes in the image. If there is an intersection, we simply update reflection term in shading equation. The only difference between depth maps and normal maps is the position of the shading point.

Reflection with Depth Maps: Remember that the position of a shading point in transformed coordinates is always given as $\boldsymbol{p}_S = (i, j, f(i, j))$, where (i, j) is the pixel position and $f(i, j)$ comes from depth map.

$$\vec{N}_S = (a, b, c) = \frac{\nabla F}{|\nabla F|} = \frac{\left(\frac{\delta F}{\delta x}, \frac{\delta F}{\delta y}, 1\right)}{\frac{\delta F}{\delta x}^2 + \frac{\delta F}{\delta y}^2 + 1}$$

Approximations of $\frac{\delta F}{\delta x}$ and $\frac{\delta F}{\delta y}$ can easily be computed using emboss filter in x and y directions [1]. Therefore, adding reflection with depth maps are really straightforward.

[1] As discussed before, this operation can be noisy, but the noise can somewhat be reduced by changing kernel of the emboss filter.

Reflection with Normal Maps: In this case, $\vec{N}_S = (a, b, c)$ is given, however, we need to estimate $\boldsymbol{p}_S$. One approach is, to take the discrete integral by assuming we know the height in boundary. The other and even simpler approach, which works in practice, just to choose $\boldsymbol{p}_S = (i, j, 0)$. The error in position just shifts the the reflection, but it gives desired effect. Therefore, using normal maps as proxy geometry is even simple. We simply shoot a ray starting from $(i, j, 0)$ in the direction of $(2ac, 2bc, 1 - 2c^2)$. If there is an intersection, we simply update reflection term in shading equation.

7.3 COMPUTING REFRACTION

Preliminaries: Let again $\vec{N}_S = (a, b, c)$ and $\vec{E} = (0, 0, 1)$ denote the unit normal of the shading point and the unit eye vector respectively. Moreover, let η_1 and η_2 denote indices of refraction first and second medium respectively. The transmission vector $\vec{T}$, then, is computed as follows [147]:

$$\vec{T} = \frac{-1}{\eta}\vec{E} + \left(\frac{Cos\theta_i}{\eta} - \sqrt{\frac{Cos\theta_i^2 - 1}{\eta^2} + 1}\right)\vec{N},$$

$$\text{where } \theta_i \text{ incident angle and } Cos\theta_i = \vec{E} \cdot \vec{N} \text{ and } \eta = \frac{\eta_2}{\eta_1}$$

Using this equation, we can compute transmission direction.

Discussion: The most important parameter in getting deformation with refraction is not really η as most people think. If the object is thin, we do not see any deformation even if η is significantly different than 1. For deformation, the most important parameter is the distance the light travel inside of the transparent object. For our compositing applications, the most important observation is that this distance is zero in silhouette boundaries. Therefore, for our applications, the areas that corresponds the extreme cases are silhouette boundaries. We need to make sure that there is no distortion in these boundaries.

Thickness parameter c: To control the visual results of the transmission process, I suggest to provide another thickness parameter c and we assume the light goes back the first medium after travelling c amount inside of the second medium. Then, we can assume it goes back initial direction, which will be $-\vec{E}$. In other words, To include refraction of newly added object, all we need to do is first to compute the position of a new point

$$\boldsymbol{p}'_S = \boldsymbol{p}_S + c\,\vec{T}$$

then check if there is an intersection with newly added object along the ray starting from the position of $\boldsymbol{p}'_S$ in the direction of $-\vec{E}$. If there is no intersection, nothing changes in the image. If there is an intersection, we simply update refraction term in shading equation. The only difference between depth maps and normal maps is again the position of the shading point. Another important issue is to provide the value of c.

Reconstruction of Thickness parameter c: As discussed earlier, the key issue is to make sure that c value is zero at the silhouette boundaries. This is actually the reason I called thickness parameter c. If the perspective transformation is embedded into the geometry, the c component of $\vec{N}_S = (a, b, c)$ is always zero. Note that in the silhouette boundary $a^2 + b^2 = 1$, therefore $c = 0$, in which we do not want any distortion in refraction. Therefore, we can simply use c component of $\vec{N}_S = (a, b, c)$ as thickness parameter. The biggest distortion usually happen when $a^2 + b^2 = 0$, i.e. $c = 1$. The only problem with this approach is that some parts of the object must be thin even when $a^2 + b^2 = 0$. This is where 2D vector field comes to rescue. Note that since $\vec{N}_S$ is a unit vector, two values a and b are sufficient to describe the vector. The third value c is really redundant. Therefore, we can simply use it as a separate parameter. For usual normal maps, existing c value already works as we have discussed.

(a) *Proxy Geometry.* (b) *Opaque Region.* (c) *B&W regions shows Fresnel Parameter.* (d) *Transparent & Translucent Refraction mixed with Reflection using Fresnel.*

Figure 7.4: *An example that demonstrate the use of extended normal maps. Note that the parts of the bottle have different thickness values, with which we can control the amount of distortion caused by refraction. This particular figure also demonstrate multiple effects: from the use of object thickness parameter c to the Fresnel parameter that depends on indices of refraction η_1 and η_2, mixing refraction and reflection with Fresnel, and transparency vs. translucency.*

Refraction with Depth Maps: For depth maps, we can compute c as a gradient. However, in this particular case, silhouette boundaries needs to be zero. Therefore, we need to take gradient non-transformed space. In other words, if we used a transformations such as $z' = z/(1 + z)$, then it means $F(x, y) = z/(1 + z)$. Then, we need to compute gradient of $z = -F(x, y)/(1 - F(x, y))$. This discussion shows that depth maps are not perfectly suitable to represent shapes such as half-filled bottle shown in Figure 7.4. We, of course, add an additional parameter to include thickness.

Refraction with Normal Maps: Normal maps fit perfectly to represent thickness as discussed earlier. Therefore, normal maps are more suitable than depth maps in the reconstruction of proxy shapes for transparent objects.

7.4 COMPUTING FRESNEL

Preliminaries: Let again $\vec{N}_S = (a, b, c)$, $\vec{E} = (0, 0, 1)$, and $\vec{T}$ denote the unit normal of the shading point and the unit eye vector, and unit transmission vector respectively and let η_1 and η_2 denote indices of refraction first and second medium respectively. Fresnel equation is really a Barycentric mixing of reflection and refraction components. It is already included in any transparent region for every pixel as

$$\boldsymbol{C}(i, j) = \boldsymbol{C}_{B_o}(1 - f_o) + \boldsymbol{C}_{F_o} f_o$$

image in the where $0 \geq f_o \geq 1$ is the original Fresnel parameter, and C_{B_o} and C_{F_o} are original refracted background and refracted foreground colors respectively. Our estimation of the Fresnel parameter f_e also come from proxy shapes and their estimated index of refraction as follows:

$$f_e = \left(\frac{Cos\theta_T - \eta Cos\theta_i}{\eta Cos\theta_T + \eta Cos\theta_i} \right)$$

where θ_i incident angle and $Cos\theta_i = \vec{E} \cdot \vec{N}$; θ_t transmission angle and $Cos\theta_t = \vec{T} \cdot -\vec{N}$ and $\eta = \frac{\eta_2}{\eta_1}$

Figure 7.4c shows Fresnel parameter for three different η values. Now, assume that we have computed reflected and refracted contributions of new added objects as $\boldsymbol{C}_R$ and $\boldsymbol{C}_T$. There are four possible cases: (1) No new contribution; (2) New reflection; (3) New refraction; and (4) both reflection and refraction are new. I will next analyze these four cases:

1. **No new contribution:** In this case, we simply use $\boldsymbol{C}_o(i, j)$.

2. **New reflection:** In this case, we recompute the color as $\boldsymbol{C}'(i, j) = \boldsymbol{C}(i, j)(1 - f_e) + \boldsymbol{C}_R f_e = (\boldsymbol{C}_{B_o}(1 - f_o) + \boldsymbol{C}_{F_o} f_o)(1 - f_e) + \boldsymbol{C}_R f_e$. Note that this not correct even if $f = f_e = f_o$. Correct answer should be $\boldsymbol{C}'(i, j) = \boldsymbol{C}_{B_o}(1 - f) + \boldsymbol{C}_R f_e$ since $\boldsymbol{C}_R$ must now replace $\boldsymbol{C}_{F_o}$. On the other hand, this is not a problem for large f values since $\boldsymbol{C}_{F_o}$ term is practically eliminated by multiplication factor $(1 - f)$.

3. **New refraction:** In this case, we recompute the color as $\boldsymbol{C}'(i, j) = \boldsymbol{C}_T(1 - f_e) + \boldsymbol{C}(i, j) f_e = \boldsymbol{C}_T(1 - f_e) + (\boldsymbol{C}_{B_o}(1 - f_o) + \boldsymbol{C}_{F_o} f_o) f_e$. Note again that this not correct even if $f = f_e = f_o$. Correct answer should be $\boldsymbol{C}'(i, j) = \boldsymbol{C}_R(1-) + \boldsymbol{C}_{F_o} f$ since $\boldsymbol{C}_T$ must now replace $\boldsymbol{C}_{B_o}$. On the other hand, this is again not a problem for small f values since $\boldsymbol{C}_{F_o}$ term is practically eliminated by multiplication factor f.

4. **Both reflection and refraction are new:** In this case, we simply replace $\boldsymbol{C}(i, j)$ by $\boldsymbol{C}'(i, j) = \boldsymbol{C}_T R(1 - f_e) + \boldsymbol{C}_R f_e$.

7.5 ADDITIONAL INFORMATION

Caustics: I discuss caustics as a part of case studies in chapter refch070. It is mainly related to shadows since newly added diffuse object cast shadow on caustic region.

Translucent and Glossy Objects in Image: If objects in the image are translucent or glossy, this is still refraction or reflection effect as shown in Figure 7.4d. We simply get randomized samples to create translucent or glossy look as widely used distributed ray tracing [112].

Transparent, Reflective or Emissive type Newly Added Objects: In our case studies, I do not deal with these type of objects. They basically add direct or indirect illumination to the scene, which is different than diffuse type of newly added objects that can mainly occlude existing lights. Despite this difference, the basic idea is the same. We simply recompute light contribution and using our Barycentric shader, we recompute colors. However, as we would see in case studies, shader reconstruction is simpler if the newly added objects are diffuse and only cast shadows.

CHAPTER 8

Reconstruction of Directional Lights

Introduction In this chapter, I briefly discuss idealized lights. In particular, I focus on directional lights. I also introduce a new light, which I call hemispherical light to include ambient-like information easily.

Lights as Interfaces: I view virtual lights as interfaces for image creation. Their positions, directions, shapes and illumination are parameters that can be changed to control final results. If we want to match existing illumination in the background scene, we need to match some of these parameters to reasonably consistent image. We, of course, do not to use a complicated interface unless it is completely necessary. For instance, it does not make sense to use the position and shape of the Sun if we are dealing with a simple outdoor scene under sunlight. For all practical purposes, the lights coming from Sun can be considered parallel. It is not just sunlight, for any lights that is sufficiently away from the scene, we can ignore position and shape of the light, and we can consider light rays are parallel rays. There is another light type that can be represented only with a single direction: hemispherical lights. These type of lights can represent ambient illumination caused by sky.

8.1 CLASSIFICATION OF SIMPLEST IDEALIZED LIGHTS

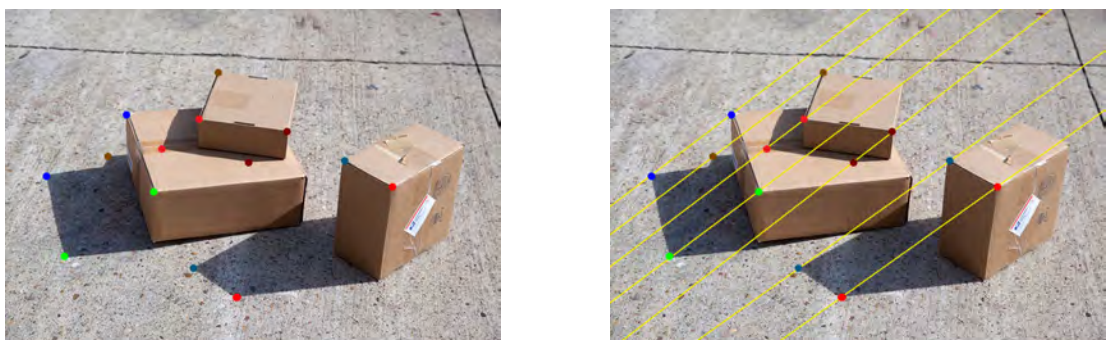
Directional Lights: Directional lights are idealized lights that produces parallel rays. We can use the results we obtain in the Section 8 to reconstruct direction of directional lights such as sunlight. In other words, the projections of rays obey perspective rules. If the rays are not parallel to $z = 0$ plane, they projection intersects in a vanishing point, that corresponds direction of incoming rays of directional light.

Hemispherical Lights: For digital compositing, I introduce the concept of hemispherical light. This light provides a simple model to represent of scattered sunlight that causes the sky illuminate any object from every direction. Although sky color is not constant, for a simple model we can assume that sky is an a hemisphere shaped area light with constant illumination in every direction. Since sky is really large, we can assume that this hemisphere has an infinite size. In other words, it does not have any center point and it can be represented only using directions. Such a hemisphere can be represented only by a single

direction that provide orientation of the north pole. If we can identify direction of north pole, we can again use a very simple model to estimate ambient illumination

8.2 RECONSTRUCTION OF DIRECTIONAL LIGHTS

Identification of Shadow Rays: Only problem is to identify rays. Fortunately, they are easy to find by using the boundaries of shadows as shown in Figure 8.1a. Although, it can be hard to find correspondences automatically, most human can see correspondent sharp points that creates shadows and their shadow positions. When we connect correspondent points with lines, we obtain projection of shadow rays as shown in Figure Figure 8.1b. Since these rays are almost parallel to each other, we can conclude that these rays are parallel to $z = 0$ plane. We can conclude that the 3D direction of these rays are simply $(\cos\theta, \sin\theta, 0)$, where we can compute $\cos\theta$ and $\sin\theta$ from the the direction of 2D lines in the photograph.



(a) *Shadow Correspondences.* (b) *Shadow Rays.*

Figure 8.1: *An example of identification of direction of the directional lights.*

Intersecting Shadow Rays: In some camera orientations, rays will not be parallel to $z = 0$ plane. In this case, the projected rays intersect in a vanishing point (A, B) in 2D. In the first sight, this does not seem to be enough to compute ray direction. Unfortunately, we know that the projected rays lies in $z = f$ plane. Therefore, actual 3D position of the vanishing point is (A, B, f). We also know that this vanishing point is a representation of a unit vector, which are given as (a, b, c) where

$$A = \frac{fa}{c} \quad B = \frac{fb}{c} \quad \text{and} \quad a^2 + b^2 + c^2 = 1.$$

Using these three equations, we can reconstruct light direction $n_L = (a, b, c)$ as:

$$a = \frac{A}{\sqrt{A^2 + B^2 + f^2}} \quad b = \frac{B}{\sqrt{A^2 + B^2 + f^2}} \quad \text{and} \quad c = \frac{f}{\sqrt{A^2 + B^2 + f^2}}$$

This equation also shows that when A and B goes to ∞, $(a, b, c) \longrightarrow (\cos\theta, \sin\theta, 0)$. This is really all the information we need to reconstruct directional lights.

Incident Energy of Directional Lights: It is really easy to compute incident energy coming to a planar surface patch: it is simply the area visible to directional light. Let n_S denote surface normal and n_L denote light direction (inverse of direction of the incident light ray), then the ratio of the area visible by directional light to the original area of the surface can be computed as

$$n_L \bullet n_S = \cos\theta$$

where θ is the angle between the unit vectors n_L and n_S. $\cos\theta$ is not a percentage since it is can be negative. In computer graphics, we simply truncate negative values.

Illumination Control: We can better control the illumination using a monotonically increasing function $F : [-1, 1] \longrightarrow [0, 1]$ to convert $\cos\theta$ into percentages. I suggest to use the following clamp function F as an illumination control function:

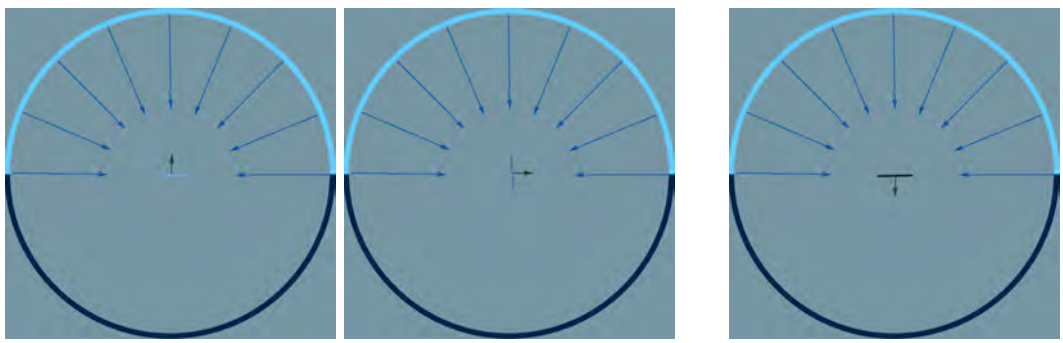
$$t = F(n_L \bullet n_S)_{[\theta_0,\theta_1]} = \begin{cases} 0 & \text{if} \quad \cos\theta_0 \leq \cos\theta, \\ \dfrac{n_L \bullet n_S - \cos\theta_0}{\cos\theta_1 - \cos\theta_0} & \text{if} \quad \cos\theta_0 < \cos\theta < \cos\theta_1, \\ 1 & \text{if} \quad \cos\theta \leq \cos\theta_1, \end{cases} \tag{8.1}$$

where $\cos\theta_0 \leq \cos\theta_1$ and t is called diffuse illumination parameter. This function can provide cartoon shading for $F(n_L \bullet n_S)_{[\theta_0,\theta_0]}$, i.e. $\cos\theta_0 = \cos\theta_1$; classical computer graphics equation for $F(n_L \bullet n_S)_{[0,\pi/2]}$, i.e. $\cos\theta_0 = 0$ and $\cos\theta_1 = 1$; and Gooch shading for $F(n_L \bullet n_S)_{[0,\pi]}$, i.e. $\cos\theta_0 = -1$ and $\cos\theta_1 = 1$ [161]. Any power of this illumination control function F such as F^γ where γ is any real number can also be used to further control illumination distribution. I prefer to start with $F(n_L \bullet n_S)_{[0,\pi]}$ since it can provide full parameter domain that can provide better control in shading level.

Reconstruction of Incoming Light Energy: Let C_L denote incoming energy of the light ray that hits a surface element. C_L is a function that provides incoming energy for every wavelength. Range of C_L is really all positive numbers. Without loss of generalization we can restrict range of C_L to $[0, 1]$ since we can represent any range of positive numbers by mapping to $[0, 1]$. This assumption simplifies our shading process since we only deal numbers between 0 and 1. One issue is that it is not really possible to reconstruct C_L directly from diffuse reflectance in images illuminated by directional lights. Therefore, we do not really reconstruct C_L and we assume that it is embedded in material color as we see in the next section.

8.3 RECONSTRUCTION OF HEMISPHERICAL LIGHTS

Ambient Occlusion: Reconstruction of ambient illumination in a real scene is extremely difficult. I introduce concept of hemispherical lights to estimate ambient illumination. This model is particularly useful in the scenes that are directly illuminated by sunlight. The scene will also illuminated by scattered sunlight that is coming from every direction. This scattered light gives a subtle illumination that presents itself especially in shadow regions as we will see later. It is important to include this effect to obtain a consistent look-and-feel.



(a) *100%.* (b) *50%.* (c) *0%.*

Figure 8.2: *Hemispherical Light model can be considered as an infinite sphere. Only half of this infinite sphere emits light and light emission is constant everywhere. Only the orientation of a surface patch can specify how much light can reach the patch. I drew the surface patch in the center of the sphere just for demonstration purposes. Since the sphere is infinite, only direction matters and position of surface patch can be ignored.*

Mechanism of Hemispherical Light: Figure 8.2 presents Hemispherical Light as an infinite sphere. Half of this infinite sphere emits light. The other half does emit any light. Orientation of surface patch defines half much light can the surface patch. This can be computed as an integral of visible emission region. I purposely do not provide an explicit equation since this is just a conceptual model. Figure 8.2 demonstrate the key points of the idea. The amount of light reaching a surface patch must be a monotonically decreasing function the angle between surface patch normal and ground plane normal. Therefore, reconstruction of Hemispherical Light is simply the reconstruction of the normal of the plane that correspond the ground.

Incoming Ambient Energy of Hemispherical Lights: Based on the discussion in the previous paragraph, it is easy to identify incoming ambient energy using, again, dot product. Let n_S denote surface normal and n_G denote normal of the ground plane, then the incoming ambient energy that can be computed as the ration of hemispherical area that is visible to the surface patch is a monotonically increasing function of the following

parameter

$$n_L \bullet n_G = \cos\psi$$

where ψ is the angle between the unit vectors n_L and n_G. $\cos\psi$, again, is not a percentage since it is can be negative. We now need to convert this parameter using a monotonically increasing function.

Ambient Illumination Control: We can again control the ambient illumination using the monotonically increasing function used before as

$$a = F(n_L \bullet n_S)_{[\theta_0,\theta_1]},$$

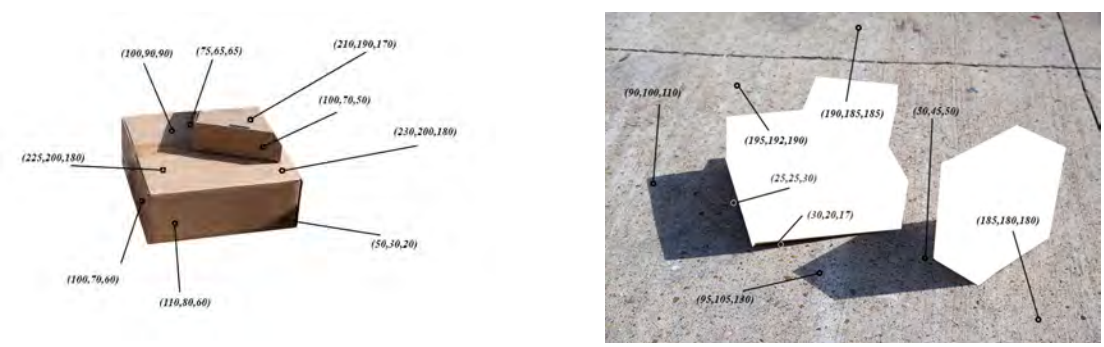
where a is an ambient illumination parameter. In this case, we can again choose θ_0, θ_1 values to obtain control over illumination. I, again, prefer to move the control to shading by using $\theta_0 = 0$ and $\theta_1 = \pi$.

CHAPTER 9

Reconstruction of Diffuse Materials

Introduction In this chapter, I briefly discuss diffuse materials and their reconstruction. Diffuse reflection is an idealized reflection model in which the reflected energy is the function of only incoming illumination, surface orientation and material properties. It does not depend on eye position.

Diffuse Materials: Diffuse materials are those that only demonstrate diffuse reflections. These are, of course, idealized materials that does rarely exist in real life. As very careful examination of real materials can demonstrate that most real materials provide some reflectance that depends on eye position. Fortunately, eye dependency effects are so subtle that we can ignore them in many cases and we can call a material is diffuse if eye dependency effects are not easily detectable in photographs or paintings.



(a) *Cardboard Paper.* (b) *Concrete.*

Figure 9.1: *Two example of diffuse materials under illumination of a directional light. The numbers shows averaged LDR colors in given positions. We use average colors using a nxn kernel to better avoid local imperfections in the image. Our goal is to develop a simple model that can provide these particular colors using only diffuse illumination parameter t and ambient illumination parameter a.*

Modeling Diffuse Reflections: One advantage of diffuse materials is that it is possible to use relatively simple models to reconstruct diffuse reflection. In this chapter, I demonstrate that it is possible to significantly simplify the reconstruction by using a Barycentric algebra to construct the models. In this chapter, I demonstrate how to develop such models

using the two real materials shown in the previous chapter (see Figure 8.1). I first analyze these two examples in the next section.

9.1 ANALYZING IMAGE TO RECONSTRUCT DIFFUSE MATERIALS

Two Real Materials: Figure 9.1 shows two types of materials in Figure 8.1: Cardboard paper and Cement. In the image, I identified color values of different regions. In each case, we can classify regions as "illuminated" and "shadow" regions. As shown in these two examples, the color values in illuminated planar regions, which are the regions where $t >> 0$, are essentially constant. On the other hand, the color values in shadow regions, which are the regions where $t \approx 0$, are changing significantly.

Surface	Top	Top	Side
Illumination	Illuminated	In Shadow	In Shadow
Angles	$\cos\theta_1 = 0.8$ $\cos\psi_1 = 1$	In Shadow $\cos\psi_1 = 1$	In Shadow $\cos\psi_0 = 0$
Parameter Assignment	$t = 1$ $a = 1$	$t = 0$ $a = 1$	$t = 0$ $a = 0$
Colors	$(220, 200, 180)$	$(100, 90, 90)$	$(50, 30, 20)$

Table 9.1: Cardboard Diffuse Material. The most useful is shadow areas since they can provide the reflected colors for $t = 0$.

Color Change in Shadow Regions: There is always a significant color changes in shadow regions since the ambient illumination, which is the dominant term in these regions where $t \approx 0$, rapidly change from contact points to shadow boundaries. In regions closer to contact points, neighboring objects cover a significant portion of the sky by occluding most of the illumination coming from our conceptual hemispherical light. Therefore, the ambient

illumination in these regions is usually very small. We can assume that $a \approx 0$. On the other hand, ambient illumination takes its largest values in the shadow boundaries. These regions are those that are the most distant from nearby neighboring objects. Therefore, they are the least occluded by those objects. Although some part of sky is always occluded, we can safely assume that in these regions $a >> 0$ if these shadow boundaries are sufficiently away from neighboring objects.

Surface	Top	Top	Bottom
Illumination	Illuminated	In Shadow	Contact Shadow
Angles	$\cos\theta_1 = 0.8$ $\cos\psi_1 = 1$	In Shadow $\cos\psi_1 = 1$	In Shadow $\cos\psi_0 = -1$
Parameter Assignment	$t = 1$ $a = 1$	$t = 0$ $a = 1$	$t = 0$ $a = 0$
Colors	$(195, 192, 190)$	$(195, 105, 120)$	$(25, 20, 17)$

Table 9.2: Concrete Diffuse Material. The most useful is again shadow areas.

Three key Regions: This discussion demonstrates that there are three key regions to analyze to develop a model, also called shader in computer graphics, for reconstruction of a diffuse material (see Tables 9.1 and 9.2):

The Most Illuminated Regions: These are the regions where t values are largest, hopefully closer to 1. These correspond to brigtest regions of diffuse object. Unfortunately, in neither of our two examples materials, the plane that would be the most illuminated by given directional light is not provided. Therefore, we need to use the largest value, which could be reasonably close to $t = 1$.

Shadow Boundaries: These are the regions where $t \approx 0$ and a values are largest, hopefully closer to 1. In this case, we use the fact that the shadow boundaries are sharpest under

directional light. We need to take samples in shadow part and among those samples the largest color values corresponds to higher a values.

Contact Shadows: These are the regions where $t \approx 0$ and $a \approx 0$. As we show in these examples, colors in these regions are not completely black. In other words, even when there is no ambient illumination coming from our conceptual "hemispherical light", some illumination reaches these tight areas.

Interpolation: The colors in this three regions provides us extreme cases that can be interpolated to obtain a simple model. However, this information is not useful to reconstruct incoming light energy as discussed in the next section. To develop a simple model, we need to ignore colors of diffuse and ambient illumination.

9.2 TOWARDS A SIMPLE MODEL OF DIFFUSE REFLECTION

Ignoring Color of Directional Light: Let C_L and C_M denote color of a light ray hits a surface element and color of surface element respectively. Then, we compute the diffusely reflected energy as tC_LC_M. It is important to note that although we call them colors, these two entities are not the same type. C_L is a function that represents incoming energy for every wavelength and C_M is direction-independent reflectance function that represents the percentage of reflected energy for every wavelength. In other words, range of C_L is really all positive numbers, on the other hand, the range of C_M is just $[0, 1]$. As discussed earlier, without loss of generality we can assume that the range of C_L is also $[0, 1]$. This assumption simplifies our work since we can directly work on Low Dynamic Range images. Under this assumption, tC_LC_M is also a energy (or color) and defined as a function from wavelength to $[0, 1]$. One advantage of this formulation, we can completely ignore C_L and we can use the pre-multiplied term, $C_{M_0} = C_LC_M$, as the color of the object, which we can directly estimate from images. Then, the diffuse illumination parameter t alone can define reflected energy.

Ignoring Color of Hemispherical Light: Similarly, Let C_H and C_M denote the emitted color of hemisphere and color of surface element respectively. Then, we compute the diffusely reflected energy of Hemispherical light as aC_LH_M. Similarly here, we can completely ignore C_H and we can use $C_{M_1} = C_HC_M$ as the ambient color of the object, which we can directly estimate from images. Similarly, the ambient illumination a alone can define reflected ambient energy.

Vector Based Model: Our goal is to obtain a simple equation in the form of $C(t, a)$ by combining diffuse reflections of directional and hemispherical lights. From the Figure 9.1, we know some reflection colors as listed in Tables 9.1 and 9.2. One idea based on classical

Computer Graphics approach is to use an affine equation with two variables in the form of

$$C(t,a) = \vec{C_2}t + \vec{C_1}a + C_0$$

where $\vec{C_2}$ and $\vec{C_1}$ are vector colors and C_0 is a point in color space. Computing $\vec{C_2}$, $\vec{C_1}$, and C_0 is really easy by using our data. For instance, from Table 9.1, we estimate that

$$C(1,1) = (220, 200, 180), \quad C(0,1) = (100, 90, 90), \quad \text{and} \quad C(0,0) = (50, 30, 20).$$

Using simple algebra, we compute that

$$C_0 = C(0,0), \quad \vec{C_1} = C(0,1) - C(0,0), \quad \text{and} \quad \vec{C_2} = C(1,1) - C(0,1).$$

The equation clearly demonstrates that $\vec{C_1}$ and $\vec{C_2}$ are indeed vectors that are defined as difference of two points (of color). Using this we obtain simple models for shading cardboard and cement materials as follows:

$$\begin{aligned} \text{Cardboard:} \quad & C(t,a) = (120, 110, 90)t + (50, 60, 70)a + (50, 30, 20) \\ \text{Cement:} \quad & C(t,a) = (100, 87, 70)t + (70, 85, 103)a + (25, 20, 17) \end{aligned}$$

The problem with this model, in order to have the complete power of this equation, the vector colors should be able to take negative values since difference of two points can have negative values. As we see later, there exist many cases including photographs, in which we have to use negative values to obtain a desired look-and-feel. This is one of the dirty secrets in computer graphics industry. Significant amount human-time is spent just to obtain a desired look-and-feel using wide variety of techniques such as negative lights. I believe that computer aided design provides a simple solution to this problem. We simply use Barycentric functions.

Barycentric Model: The simplest Barycentric model is the following Bilinear form that interpolates three end-points as follows:

$$C(t,a) = (C(0,0)(1-a) + C(0,1)a)(1-t) + C(1,1)t$$

We can improve this model by adding a dummy term $C(1,0)$ as

$$C(t,a) = (C(0,0)(1-a) + C(0,1)a)(1-t) + (C(1,0)(1-a) + C(1,1)a)t$$

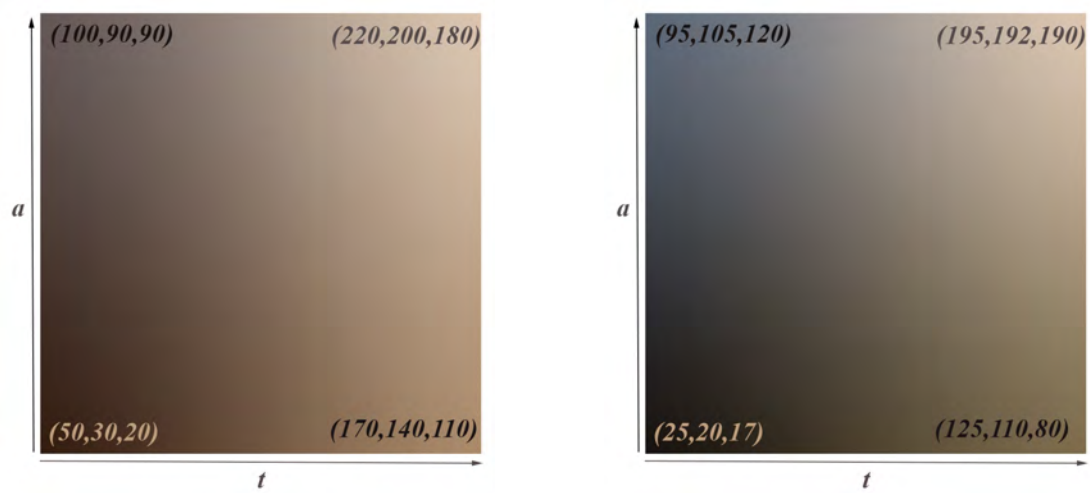
I call this particular term as dummy since it can never exist since there can be no case in which a point is fully illuminated with no ambient illumination. However, having these dummy terms can help further control the results. Moreover, if we choose

$$C(1,0) = C(1,1) - C(0,1) + C(0,0)$$

bilinear terms are eliminated and we obtain an affine equation as

$$C(t,a) = C(0,0)(1-a) + C(1,1)t + C(0,1)(a-t)$$

in which $C(1,0)$ is still formally a point. This property is nice for formalism, however, $C(1,0)$ can still have negative values. This is not really a problem. Since $C(1,0)$ does not effect visible results significantly, we can simply choose $C(1,0)$ using an appropriate color. Figure 9.2 shows two bilinear models for cardboard and concrete materials that are constructed using this approach. In these two figures, the triplets in the corners shows $C(0,0)$, $C(0,1)$, $C(1,1)$, and $C(1,0)$ control points (i.e. control colors) that are interpolated by the bilinear function.



(a) *Bilinear Cardboard Paper Shader.*

(b) *Bilinear Concrete Shader.*

Figure 9.2: *Two examples of shaders that can reconstruct Cardboard and Concrete materials based on the image.*

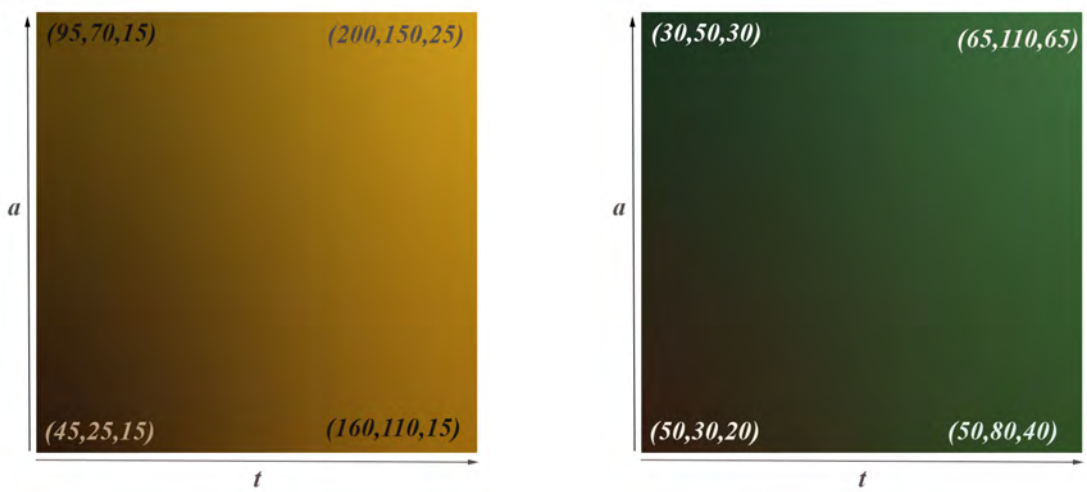
Using These Models in the Depiction of a New Objects: We obtain these models such that depictions of the new objects can be consistent with the rest of of the background image. These shading models are extremely simple and easy to implement with any existing shader language. If we have a good geometric representation of background scene, it is possible to improve the quality of results by computing ambient illumination term, a, using ambient occlusion instead of the hemispherical light. Using ambient occlusion instead of hemispherical light does not require any change in our shading model.

Texture Mapping: One additional issue is that materials are not homogeneous. Even when they visually look homogeneous, they may consist of many materials that are packed

together. This heterogeneity of materials can be represented by texture maps. These texture maps can be either procedurally defined such as Perlin Noise or provided by Images. Let a surface consists of two materials $C_0(t,a)$ and $C_1(t,a)$, $T(u,v)$ denote a texture mapped to a surface point (u,v) that gives us a percentage of two materials that are used in surface point (u,v), then, the heterogeneous material can be given as a Barycentric combination of two materials as

$$C(t,a) = C_0(t,a)(1 - T(u,v))C_0(t,a)T(u,v).$$

For heterogeneous multi-material surfaces that are mixture of more than two, it is easy to extend this formulation to a more general Barycentric combination. In the next section, I present two methods to create new materials from existing models.



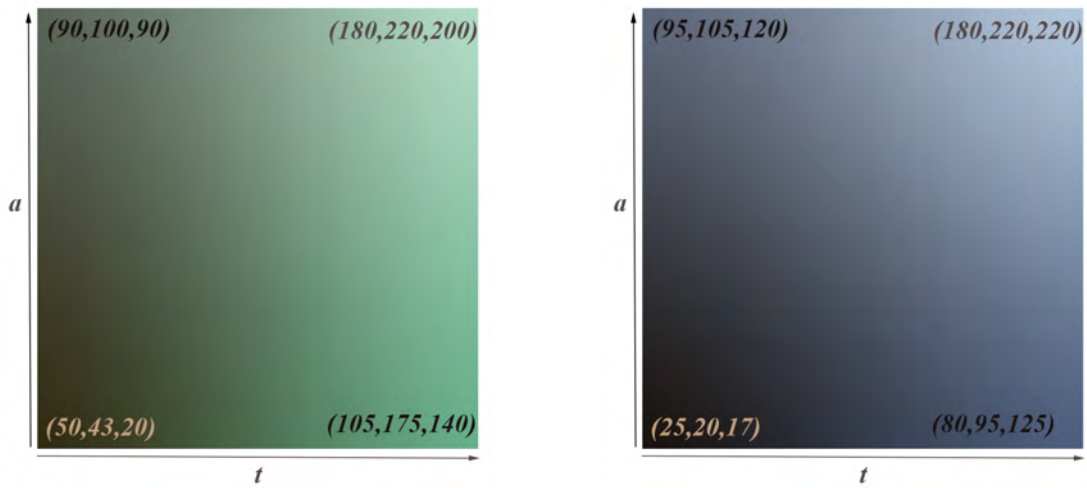
(a) *New Diffuse Yellow Material Shader.* (b) *New Diffuse Dark-Green Material Shader.*

Figure 9.3: *Two examples of new Bilinear shaders that are obtained by multiplying control colors,* $C(1,1)$, $C(1,0)$ *and* $C(0,1)$ *with a color* C_m.

9.3 CONSTRUCTION NEW SHADING MODELS

Creating Depictions of New Object: To create depictions of the new object, these reconstructed shading models are useful. We can simply change control colors to obtain new models and we can use these models to render newly added objects to the scene. In this process, the most important difficulty is to develop a meaningful method to change control colors. The main problem comes from the fact that the terms $C(0,1) = C_H C_M$ and $C(1,1) = (C_L + C_H)C_M$ contain hidden illumination information. Since we cannot really separate illumination part C_H and $(C_L + C_H)$ from reflection function C_M, we cannot

directly change reflection function to obtain new control colors. Therefore, we need to develop reasonable methods that can provide acceptable results.

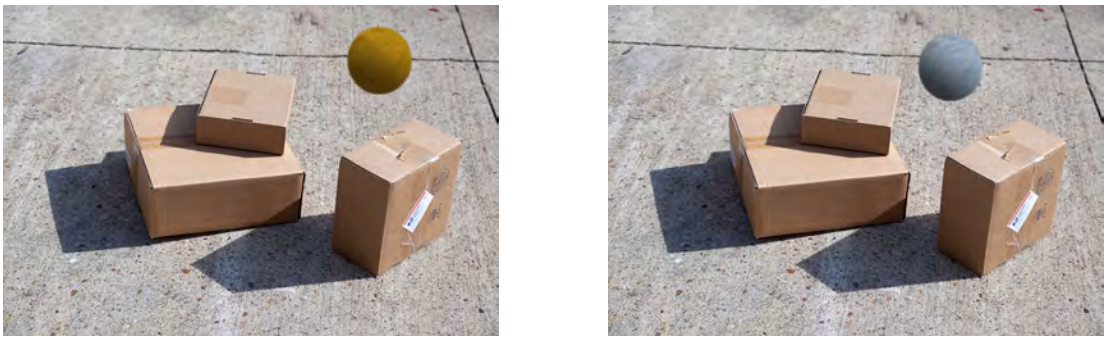


(a) *New Diffuse Green Material Shader.* (b) *New Diffuse Blue Material Shader.*

Figure 9.4: *Two examples of new Bilinear shaders that are obtained by changing hue & saturation of original control colors,* $C(1,1)$*,* $C(1,0)$ *and* $C(0,1)$*.*

Multiplication Method: This method is based on the observation that both $C(0,1)$ and $C(1,1)$ is multiplied by C_M. If we multiply both with with another color C_m as $C'(1,0) = C(1,0)C_m$ and $C'(1,1) = C(1,1)C_m$, we can safely assume that we obtained a new material $C'_M = C_M C_m$. The problem is that this process does not provide a guidance how to change colors $C(1,0)$ and $C(0,0)$. Based on the observation that $C(1,0)$ is just a dummy term, we can also multiply it C_m and assume that we obtained a new material. On the other hand, we cannot simply multiply $C(0,0)$. Since $C(0,0)$ is the darkest color, it is effected by any non-linear process coming from image creation process. Its value could be one of the darkest colors in the image. If we create a darker color, it can be disturbingly visible. Therefore, we can change its hue and saturation slightly, but it is better to keep its value approximately the same as before. Figure 9.3 shows two bilinear shading models that are obtained by this method.

Changing Hue & Saturation: Since shading comes mostly value of the color, we can simply change the hue and saturation of control colors of $C(1,1)$, $C(1,0)$ and $C(0,1)$ to obtain new shading models such as the ones shown in Figure 9.4. We can even change the hue and saturation of $C(0,0)$. Since its value does not change, this change does not cause a visual problem in rendered image.



(a) *Yellow Sphere.* (b) *Blue Sphere.*

Figure 9.5: *Two examples of hand drawn objects that are created using diffuse materials we have created. As it can be seen in these examples, although these colors are not present in the original image, the colors of the spheres appear to be consistent with the rest of the scene. Texture and soft silhouette edges also helped to get the consistency. Spheres do not have shadow. In the next chapter, I present strategies for compositing shadows. .*

Automatic Creation of Material Models: These two methods is general enough to automatically create new materials and produce more realistic heterogeneous multi-material surfaces than homogenous material model shown in Figure 9.2 that smooths out every detail that exists in concrete surface as shown in Figure 9.1.2.

Adding a New Object: This method is general enough to be used even for adding hand-drawn images. For instance, in Figure 9.5 both spheres are hand-drawn. I simply used the colors provided by the bilinear models to produce these images. As it can be seen in the figure, the spheres appears to be consistent with the rest of the scene, although these colors do not exist in the background image. I also made the silhouette of the spheres slightly transparent and added a slight texture to be consistent with the rest of the image. I ignored shadows to demonstrate importance of shadows, which is the simplest element of global illumination that is explained in the next chapter.

CHAPTER 10

Compositing Shadows

Introduction: Shadow is a global illumination effect since it is caused by occlusion of lights by all objects in the scene. Shadow is an important device to describe the position of newly added object. Figure 8.1, shows how the perceived position and size of a newly added object can be controlled by changing the shadow position. The object in Figure 8.1.1 is perceived to be significantly larger than and more distant from the camera than the object in Figure 8.1.2, although their actual sizes and positions in the image are exactly the same.

10.1 ANALYSIS OF SHADOWS FROM DIRECTIONAL LIGHTS

Shadow Types: Figure 10.1 also shows some important types of shadows. Figure 10.1a shows a contact shadow in which color of shadow changes in the shadow region because of occlusion of ambient illumination object itself. Inclusion ambient illumination is essential for acceptable perception of contact shadows. Figure 10.1b shows a shadow with C^1 discontinuity. We can also have shadow with C^0 discontinuities. The positions of these discontinuities must conform to shape of the occluded for acceptable perception of such shadow shapes with discontinuities. Figure 10.1c shows a distant shadow, in which there must be almost no ambient illumination change. However, shadow color is essential for acceptable perception.

(a) *Contact Shadow.* (b) C^1 *Discontinous Shadow.* (c) *Distant Shadow.*

Figure 10.1: *Three examples that demonstrate the importance of shadow in understanding position of the new object.*

Importance of Color in Shadow Perception: Figure 10.2 demonstrates the most important problems in shadow compositing: shadow colors that are significantly different than existing shadows in the scene. Since the color of the shadow is not consistent with

colors of the existing shadows, these shadows appears to be strange even though their positions and shapes are correct.

(a) *Contact shadow darker than other shadows in the scene.*

(b) *Discontinous shadow darker than pther shadows in the scene.*

(c) *Distant shadow that is darker than other shadows in teh scene*

Figure 10.2: *Three examples that demonstrate the importance of shadow in understanding position of the new object. As shown in the example, darker distant shadows look more unnatural than darker contact shdows.*

Importance of Conforming the Shadow to the Shape of Occluded Object in Shadow Perception: Figures 10.3a shows a shadow with a G^0 discontinuity in an area that can cause only G^1 discontinuity. Figure 10.3b shows a shadow with no discontinuity in an area that can cause discontinuities. If the positions and types of discontinuities are not correct or there are no discontinuities where there must be discontinuities, these inconsistencies cause the shadows to be perceived as "strange". In order to avoid these problems, the boundaries of the proxy geometries that receive shadow must exactly be correct. This problem is actually one of the key issues in real time shadow compositing for augmented reality. Figure 10.3c shows a similar case in which shadow shape does not conform the ground surface. When a shadow does not conform the shape of occluded object such as this one, it is also perceived to be strange.

(a) G^0 *discontinuous shadow, although occluded object is just* G^1 *discontinuous.*

(b) G^1 *continuous shadow, although occluded object is just* G^1 *discontinuous.*

(c) G^1 *discontinuous shadow on* G^1 *discontinuous object.*

Figure 10.3: *Three examples that demonstrate the importance of conforming the shadow to the shape of occluded object. In this case, occluded object has a* G^1 *discontinuous shape. In (a) and (b) discontinuities do not match. In (c), demonstrate that even when discontinuities match correctly, the result loo unnatural if shadow shape do not conform the surface of occluded object.*

Importance of Approximate Position of Shadow in Shadow Perception: Figure 10.4 shows another problem: a distant shadow in wrong position. That particular shadow in the image cannot be created any object in the scene based on the direction of the directional light that illuminate the scene. This particular scene looks strange, but it is possible since both the shadow of the yellow sphere and the object that cast the shadow can be outside of the image frame.

(a) *Non-conforming shadow.* (b) *Another Non-conforming shadow.* (c) *Shadow in wrong position.*

Figure 10.4: *Three examples that demonstrate the importance of approximate position of shadow. Shadow must be located approximately in the same direction of light. In none of these cases, this is correct. However, these images somehow look acceptable since those shadows might be casted by another object which is not inside of the image frame and the yellow sphere's shadow might be outside of the image frame.*

Wrong but Acceptable Shadow Shapes: Figure 10.5 demonstrate another interesting property in shadow compositing. In these examples, spherical object cast a rectangular shadow. We know that it is not possible but since the approximate positions of shadows are acceptable, colors are consistent with shadow colors of background scene and shadow shapes conforms to the shapes of occluded objects, these shadows appears to be visually acceptable. In fact, artists use this property to improve overall composition of the paintings.

(a) *Rectangular Contact Shadow.* (b) *Conforming Rectangular Shadow.* (c) *Distant Rectangular Shadow.*

Figure 10.5: *Three examples that demonstrate shape of the shadow is not important as far as (1) color of the shadow, (2) shadow shape conform the shape of the occluded object and (3) shadow position is correct.*

10.2 CLASSIFICATION OF SHADOW REGIONS

Introduction of Shadow Parameter: I introduce a new parameter to classify shadow regions in an image. Let sh denote the shadow parameter assigned to every pixel. If sh is not 1, that point is in shadow for a given light. $sh = 0$ means the shading point that corresponds to that pixel is completely occluded by newly introduced object. If $0 < sh < 1$, the shading point is partially occluded by new object. For every light type we need a shadow parameter. In this case, since we have only directional and hemispherical lights, we only need two shadow parameter: sh_t and sh_a.

The need for Proxy Geometry: To compute sh values for every pixel in the image, we need to assign a proxy geometry to every object in the background scene. Using this proxy background scene, we can compute a shadow parameter sh for every point using existing methods.

Classification of Shadow Regions: There are two types of shadow regions based on how they are computed: Shadows resulted from direct illumination; and (2) Shadows resulted from global illumination.

Shadows by Direct Illumination: These are the regions that can simply be identified from local information, in which $\cos\theta = n_L \bullet n_S < 0$. This computation provides only the self-shadow of locally convex regions. In other words, this can only provide self-shadows on locally-convex regions of newly introduced object. Therefore, this particular method is not applicable for shadow compositing. On the other hand, since I allow to choose t parameter using a monotonically increasing mapping, in my formulation t may not necessarily be labeled as 0 everywhere this region. We may still have to assign an sh parameter to the pixels that correspond to newly object to obtain consistent look.

Shadows by Global Illumination: These are the regions that can only be identified by considering other objects and non-convex parts of the same object as occluders. Again, there are two types of shadows resulted from global illumination based on the interaction between newly introduced object and proxy geometry: (1)shadows casted by newly introduced object on proxy geometry; and (2) shadows casted by proxy geometry on newly introduced object. The second requires better proxy geometry, therefore it is harder. Fortunately, we have to deal mostly the first type of shadows in digital compositing. However, we may sometime have to deal with the second type as shown in Figure 10.6.

10.3 COMPUTATION OF SHADOW PARAMETERS

Computing Shadow Parameter for Directional Light: There are two ways to compute the parameter sh by identifying existence of an occluder in the direction of light: (1) Ray Tracing and (2) Shadow mapping (See [154, 287] for tracing shadows and [290] for

(a) *No Shadow on New Object.* (b) *Shadow on New Object.*

Figure 10.6: *An example that demonstrates importance of shadows casted on the new objects by itself and proxy geometry. Self-shadow of the new object in this case must be changing smoothly since it is embedded into illumination. Ambient term is expected to be more prominent when object is closer to the ground.*

shadow mapping.). For idealized directional lights, shadow parameter sh is just a binary variable that can take the values either 0 or 1. In ray tracing, by sampling in a region that corresponds to a pixel, we can obtain any real number in $[0, 1]$. In shadow maps, we obtain an aliased shadow boundary regardless of the size of the shadow map. It is, therefore, necessary to filter shadow maps to obtain smooth boundary. Unfortunately, this creates very smooth boundary that does not look acceptable for directional lights. Therefore, I suggest to use ray tracer for directional lights to obtain sharp boundaries. It is also important to have at least $16x16$ jittered samples for each pixel to avoid aliasing and obtain a realistic estimation of occlusion (See [110, 240] for jittering). In this case, computation of shadow is really fast since we only check the effect of only one additional object.

Computing Shadow Parameter for Hemispherical Light: An additional illumination loss comes from occlusion of hemisphere by the newly introduced object. We can compute this effect using ambient occlusion [214]. However, we do not really have to be that accurate. As we have done earlier, we can estimate show effect using a simple method. In this case, I use the fact that we can harvest distance from occluder and occluded. Let f denote the distance between occluder and occluded, and let d denote approximate diameter of occluder object, then a cone given by the field of view d/f occlude a part of the hemisphere. In this case, shadow will be a monotonically increasing function of f/d.

CHAPTER 11

Case Study: Compositing into Diffuse Scenes Illuminated by Directional Lights

11.1 INTRODUCTION

(a) *Original Photograph of Brick.* (b) *Compositing with a High Genus Object.*

Figure 11.1: *An example of compositing into photographs of diffuse scenes illuminated by a directional light. Note that (1) matching discontinuities in shadows to original shape boundaries and (2) matching colors of CG shadow to colors of original shadows. Both are essential to obtain successful composite. This composite image is crated by Yuxiao Du.*

General Definition of the Problem: In this chapter, I discuss how to create seamless composites of animated computer generated (CG) objects with the photographs of scenes that are illuminated again by sunlight as shown in Figures 11.1a and 11.2a. This problem is different from previous case study in Chapter 6 only in the sense that the newly added shapes are supposed to be complicated Computer Generated shapes, which cannot simply be drawn by hand. For newly added objects, I ask students to use complicated high-genus shapes that are created our own software TopMod [26, 28]. Moreover, they are supposed to create an animation of the object. Therefore, they are forced to use a modeling, rendering and animation software such as Maya. In this particular case, the background is still a

photograph of a real scene illuminated by directional light. However, instead of cardboard boxes, we have slightly more complicated objects.

(a) *Original Photograph of Old Wooden Box.*

(b) *Compositing with a High Genus Object.*

Figure 11.2: *Another example of compositing into photographs of diffuse scenes illuminated by a directional light. Note again that (1) matching discontinuities in shadows to original shape boundaries and (2) matching colors of CG shadow to colors of original shadows. This composite image is crated by Yolanda Chen.*

The Goal of the Project: Many of issues related compositing with the photographs shown in Figures 11.1a and 11.2a can easily be solved based on the mathematical framework I have covered in the previous chapters. On the other hand, there are still some interesting problems that needs to be handled carefully once we deal with 3D modeling, animation and rendering systems. One of the important issues is that those programs are not developed to produce images with a great deal of control. As discussed in earlier chapters, current shading framework provides a great deal of flexibility, but, not much of intuitive control. This becomes an important issue to obtain shading consistent with background. There are also other issues that comes from qualitative nature of the problem. For instance, students do not need to spent significant amount of time for the identification of camera parameters, shape reconstruction, identification of light positions and intensities. Our goal is really to make the whole process error tolerant in a such a way that errors in camera parameters, geometry of proxy shapes, light positions and intensities do not make qualitative impact in final compositing.

Additional Issues that Need to be Addressed: In this project, I chose to include animation of a complicated object since (1) the shadows of animated objects can cover a large area moving from one object to another by changing shapes and colors, (2) the shapes of shadows are also complicated that can have holes and handles. Therefore, we need a more generalized solution that can work wherever the object moves. Our goal is to obtain visual realism despite the fact that our newly added object cannot be perceived as "real". In other

(a) *Shadow of a moving object 1.* (b) *Shadow of a moving object 2.* (c) *Shadow of a moving object 3.*

Figure 11.3: *These three frames of an composited animation by Sam Hatfield demonstrates that the shadow of the newly added object can be casted on any original object when the newly added object is animated. Multiplicative inverse method can simplify the work on this case since it is independent of materials for the surfaces of the same orientation. However, for the surfaces of different orientation such as side of the brick in this case, we need to assign different multiplicative factor.*

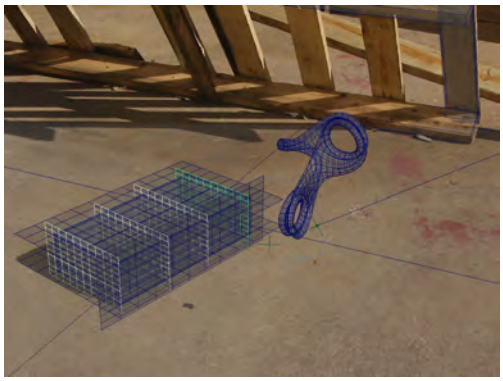
words, the shape and motion of our newly added object do not implicitly help to accept the resulting composite as "real". Therefore, students must focus on purely visual consistency to obtain visual realism.

11.2 BUILDING PROXY GEOMETRY

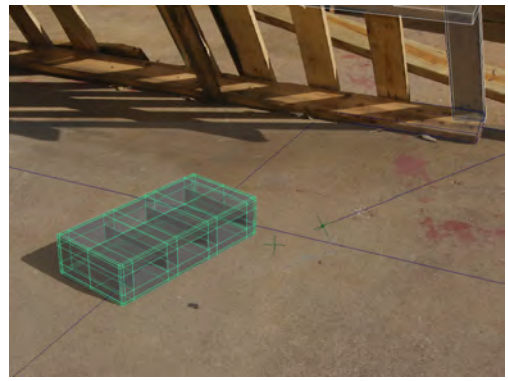
Masked-3D Planar Shapes as Proxy: In both of the scenes, the main objects, the brick and wooden box, are simple shapes that consist of three sets of parallel planes. As discussed earlier, these shapes can be build using masked planes. The main disadvantage of using planes, we may need too many masked-3D objects. For instance, we need at least 9 planes to represent the wooden box in Figure 11.2a. Note that reconstruction of the small white plate over the wooden box need an additional 3 planes. If we do not represent sides of that particular board we do not get correct shadow discontinuities. We also need additional planes for ground, two walls and the large wooden plate on the right side. One important issue having such a large number of independent shape is that current compositing software rely mainly image operations. We first need to render each planar piece separately. We, later, mask and composite them to obtain final image. Once we set up the system, it can work perfectly with animation. However, for most people who are not very methodical it is hard to make initial set-up and naming each plane correctly to remember their respective roles later.

Reduction of Number of Masked-3D Shapes: A simplifying approach we use is to represent these shapes as subdivisions surfaces with some sharp boundaries. In this way, it is possible to obtain non-straight edges with less number of masked-3D shape. In this particular case, bilinear interpolation for subdivision can be sufficient. For most objects

such as the wooden box or the brick two or three masked-3D objects can be sufficient. It, therefore, becomes easier to handle the bookkeeping for most people.



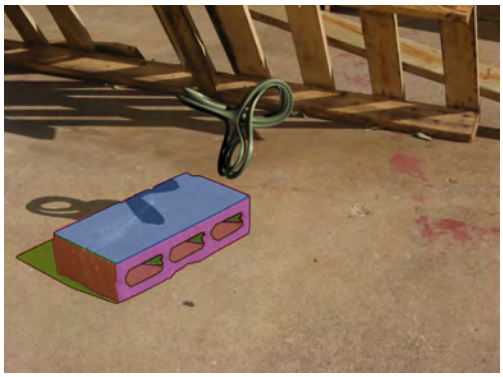
(a) *Masked-3D Subdivided Planes.*



(b) *A Single Masked-3D Subdivision Surface.*

Figure 11.4: *Two examples of masked-3D proxy shapes.*

Additional Information about Masking: In addition to handling discontinues, we also need to mask existing shadows. This process results in a set of conceptually different regions. For instance, Figure 11.5 shows the conceptually regions as (1) shadow of the brick on the ground, (2) top of the brick, (3) darker side of the brick, (4) brighter side of the brick with holes, (5) holes, (6) sides of the holes. Since we have proxy geometry already created, we can estimate normals in any point in these regions. In the regions that are already in shadow, i.e. regions 1 and 6, we do not apply multiplicative inverse. On the other hand, for regions 2, 3, 4 and 5, we apply multiplicative inverse method.



(a) *Masking Conceptually Different Regions.*

(b) *Final Shadow.*

Figure 11.5: *Two examples of masked-3D proxy shapes.*

11.3 ADDITIONAL NOTES ABOUT PHOTOGRAPHS

Visual Records for Light Reconstruction: It is useful to take visual references in the form of photographs or videos. Figure fig:ch025LightOrientation demonstrate one type of visual records that can be used to reconstruct light orientation. Using of the plate shown in Figure 11.6a is introduced by Jaemin Lee and I to reconstruct both camera parameters and light orientations [192, 193]. For close point lights, there is a need for using multiple sticks. Gazing Ball shown in Figure 11.6b is now industry standard since it provides all environment information through reflection of the whole scene [115, 116]. Using this information, it is also possible to provide environment illumination more accurate than ambient illumination estimation. Having several such visual records is helpful to confirm our initial findings about the scene.

(a) *Picture of a plate that can be used to find camera parameters and light direction. With additional sticks, it is also possible to identify light position, which is not necessary here.*

(b) *Picture of gazing ball that can also be used to identify light orientation. Reflection on the gazing ball can be used to compute environment illumination.*

Figure 11.6: *Two additional methods to find light orientation. Existing shadows in the original photo can also be used to estimate light orientation. However, having redundant information helps to confirm our estimation.*

Visual Records for Materials: It is also useful to collect additional visual records for materials. Additional visual information about materials that do not exist in the original photograph is useful to reconstruct shaders by confirming our assumptions. Until recently, we heavily relied on this information since we used the classical shaders. After the development of Barycentric shaders and based on the methods I described in earlier chapters, it is now easier construct materials that do not exist in the scene.

Taking Additional Notes: Even with a simple image set, we can discover some new details which we did not consider before. After a few weeks, it is hard to remember the details. I, therefore, suggest when you take a picture to be used in compositing, always

(a) *Picture of a plate that can be used to find camera parameters and light direction. With additional sticks, it is also possible to identify light position, which is not necessary here.*

(b) *Picture of gazing ball that can also be used to identify light orientation. Reflection on the gazing ball can be used to compute environment illumination.*

Figure 11.7: *Having visual records about additional materials that do not exist in the original photograph is useful to reconstruct shaders.*

try to get as much as additional information as possible. For instance, an invisible yellow wall may cause some part of image become slightly yellow. It is always helpful to make predictions about invisible regions and check those predictions with your data. Confirming such information can help to obtain better compositing. The original photographs shown in Figures 11.1a and 11.2a have taken more than 10 years ago by my then-TA Jaemin Lee. He is very methodical person who meticulously took additional notes about the scenes as shown below. These notes make these set of images more valuable since students can confirm their results. Moreover, Jaemin drew diagrams such as the ones shown in Figure 11.8. These diagrams are also helpful to remember the details later. Having notes still help us to make analysis by comparing our findings with the notes.

Notes: These are Jaemin's notes for Wooden Box Photograph. I strongly suggest to take similar notes to remember some details later. Most of this information is necessary based on the framework I presented in this book. On the other hand, it is always useful to have such redundant information to make confirmation of our framework.

Environment, Time and Location: This is a photograph of an environment that includes a Bluish rotten wood box/Concrete floor and wall. It was photographed at the loading dock behind the Meteorology Building at Texas A&M

University, College Station, TX at 10:45 AM, January 16th, 2003. It was a windy but clear day. During the photographing process there was only Sunlight. All images (unless otherwise specified) are photographed with a digital camera at F8.0 and 1/640 sec.

Object-1 Information: Distance b/w Camera and Plate(center) is 71".Diagonal distance b/w Camera and Plate(center) is 76". Plate Height from Ground is 20.25" and Size of the Plate measured inside of blue border is 22" x 22".

Object-2 Information: Height of the pink wood bar is 3".

Camera Information: Camera Height from Ground is 51.5". Camera Angle is about 20 degree tilted down. Film size(LCD screen) is W 1.5" x H 1.125". Zoom size is unknown (Canon digital still camera).

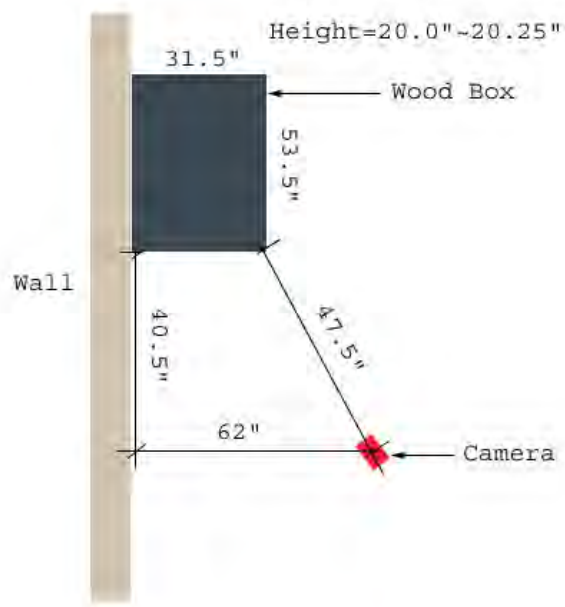


Figure 11.8: A diagram of the scene by Jaemin Lee.

CHAPTER 12

Reconstruction of Materials, Point and Spot Lights for Compositing

Physical Lights: In real life all lights are area lights, i.e. they are not infinitely small. On the other hand, for all practical purposes we can reconstruct them as directional, point or spot lights to solve certain compositing problems more easily.

Observable Effects of Real Lights: In this chapter, I demonstrate that it is not really necessary to obtain parameters of real lights to obtain reasonably good quality of composited images. Instead we can just reconstruct observable effects of the lights.

12.1 OBSERVABLE DROP-OFF EFFECTS

Small Colored Lights: In many indoor scenes, we have more than one colored lights illuminates the scene. We do not consciously notice the color differences since in practice the colors of lights around us are not usually saturated. However, these subtle color changes are important to obtain realistic looking results coming from such colored lights. I, therefore, analyze the effect first to demonstrate what kind of distribution we can usually expect.

From Physics to Mathematics: As we have done earlier, we will again start from some basic physical facts and turn it into simple mathematical equation. Let T denote the illumination of a single point light and we know that T is inverse proportional with the distance D between the light and illuminated (shading) point and $\cos\theta$ between light vector and normal vector at the shading point, therefore T can be computed as

$$T \propto \frac{\cos\theta}{D^2}.$$

Assume that we try to find incoming light energy distribution of a point light in a planar surface. It will take it's maximum value at the closest point to the light.

The Drop-off Equation: The basic setup is shown in Figure 12.1. Note that at the closest point, energy drops by $1/d^2$ only since $\cos\theta = 1$. As a result, we can use this point as maximum illumination on the plane as T_{max} and therefore $T_{max} \propto \frac{1}{d^2}$, by ignoring

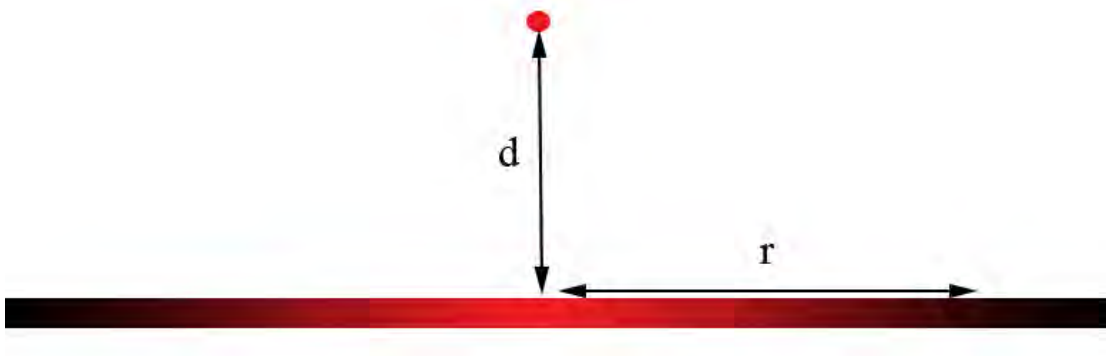


Figure 12.1: *This figure shows a basic setup for single point light and a plane.*

energy of the light source. In a point on that planar surface that is r distance from the maximum point, the energy drops by two factors. The distance D becomes $D = \sqrt{d^2 + r^2}$ and

$$\cos\theta = \frac{d}{\sqrt{d^2 + r^2}}.$$

and therefore,

$$T \propto \frac{\frac{d}{\sqrt{d^2+r^2}}}{d^2 + r^2} = \frac{d}{(d^2 + r^2)^{\frac{3}{2}}}.$$

Now, if we compute the drop-off from maximum as a proportion, we obtain following formula:

$$\frac{T}{T_{max}} = \frac{d^3}{(d^2 + r^2)^{\frac{3}{2}}} = \frac{1}{(1 + (r/d)^2)^{\frac{3}{2}}}$$

Let $x = d/r$, then for x goes to 0, r/d goes to ∞, as a result we can effectively obtain a Low Dynamic Range (LDR) equation that is qualitatively similar to original as

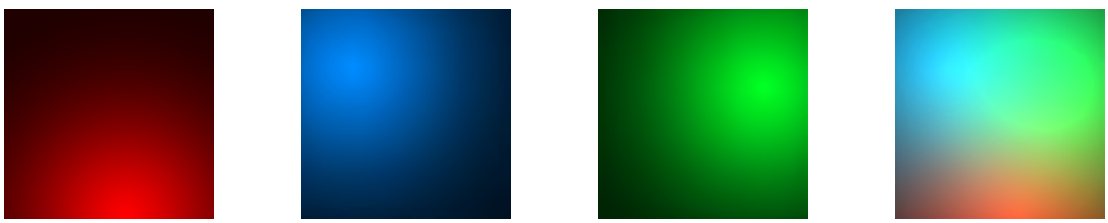
$$\frac{T}{T_{max}} \propto \frac{x^3}{(0.5x + 0.5)^3}.$$

Since the equation is LDR in every stage, using an LDR image processing program such as Photoshop we can create an image that represent the distribution of single light source on a plane as shown in Figure 12.2.

Unimportance of the Dropp-off Equation: Figures 12.2 and 12.3 demonstrate that it is not just not that essential to use any of these equations since we do not really know actual intensities of the light sources and their distances to the planar surface. Since light intensities and distances are correlated, it does not make sense to try to make better estimation of light intensities and distances. Color distribution on the surface is sufficient to obtain similar distribution.

(a) x. (b) $0.5 - 0.5x$. (c) x^3. (d) $(0.5 - 0.5x)^3$. (e) $\frac{x^3}{(0.5x+0.5)^3}$.

Figure 12.2: *An example that demonstrate all these equations creates images that qualitatively look the same. If we assume the initial energy and d are different, any of them are really acceptable.*



(a) *Red light.* (b) *Blue light.* (c) *Green light.* (d) *Addition of the three lights.*

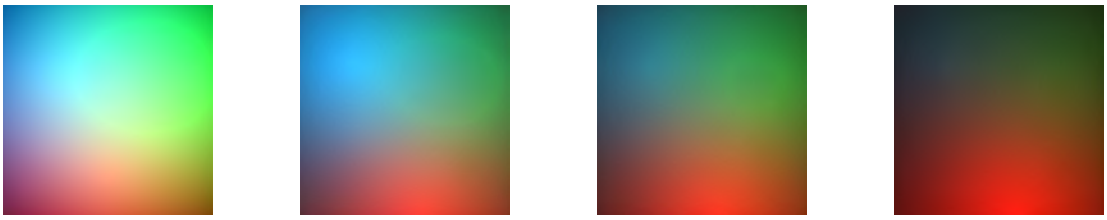
Figure 12.3: *An example that demonstrates three colored light with white material.*

Unimportance of the Material Properties: Comparison of Figures 12.4 and 12.5 further demonstrates that it is not important to estimate material properties since material properties, light intensities and distances work together to obtain final distribution. Since all of these parameters are correlated, it does not make sense to try to make better estimation of neither of them. We can only use statistical properties of color distribution on the surface to obtain similar distribution.

Shadow is more important than Shading: In the next section, I demonstrate that colors in both side of the boundaries of shadow regions are more important to than shading information. In those regions, the values of d and r of a given light are almost the same, however, in the shadow region intensity of one or more of lights go to zero. We can use this information, to develop a simple mathematical model that can provide impact of a newly added object.

12.2 SHADOWS REGIONS

Shadows Regions: Let us assume that we have n number of lights, whose effects are distinctly visible . In this case, around photograph of every real object we can identify 2^n number of distinct regions that are defined by n lights. This is because the shadow of each



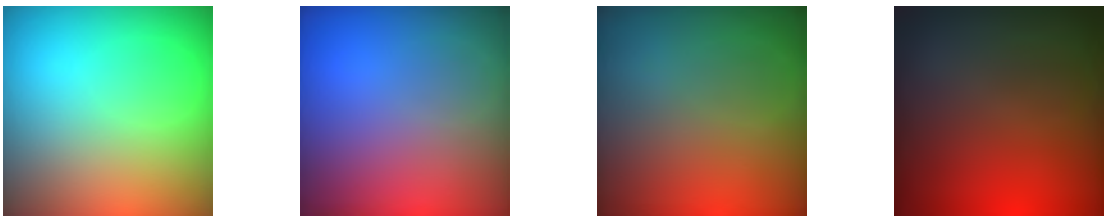
(a) *Replacing* $\frac{x^3}{(0.5x+0.5)^3}$ *term with* x *for all lights.*

(b) *Reducing intensity of green light by half.*

(c) *Reducing intensities of green and blue light by half.*

(d) *Reducing intensities of green and blue light by* 3/4.

Figure 12.4: *An example that demonstrates how much variety we can obtain from three colored light. Note that* x *provides almost identical image as* $\frac{x^3}{(0.5x+0.5)^3}$ *term. On the other hand, changing relative intensities of the maximum illumination by changing* d *or changing actual intensities of the lights can make more impact on final visual result.*



(a) *White material:* 100% *reflective in every wavelength.*

(b) *Purple material:* 100% *reflective in red and blue, and* 50% *reflective in green.*

(c) *Red material:* 100% *reflective in red, and* 50% *reflective in green and blue.*

(d) *Strongly red material:* 100% *reflective in red, and* 25% *reflective in green and blue.*

Figure 12.5: *An example that demonstrates the material properties effect the result exactly similar way of changing light intensities or distances. Note that these images are exactly the same as images in Figure 12.4 as expected.*

light separates a local region into two as illuminated and in-shadow. Let L_i denote a variable between 0 and 1, in which $L_i = 1$ means the region is illuminated, and $L_i = 0$ means the region is not illuminated by the light i, where $i = 0, 1, \ldots, n-1$. The values between zero and one represent partial illumination, and for all practical purposes we can ignore those boundary regions and assume each light defines two distinct regions. As a result, n distinct lights together define 2^n distinct regions.

Interpolating of 2^n color Points: In this case, to represent the total effect we have an n dimensional color equation in the form of $F(L_0, L_1, ...L_{n-1})$. We want this function interpolates 2^n number of color points. As we discussed earlier, each light produces a diffuse reflection that is function of d_i, r_i and light color / intensity $\vec{C}_{L_i}$ multiplied by a matrix that represents diffuse reflection property of the material. This diffuse reflection transformation

matrix is usually a scaling matrix, as $S(\vec{C}_M)$. The overall illumination is written in affine form as follows:

$$F(L_0, L_1, , ...L_{n-1}) = \sum_{i=0}^{n-1} L_i T(d_i, r_i) S(\vec{C}_M) \vec{C}_{L_i} + \vec{C}_A$$

As discussed earlier L_i is a number between 0 and 1 that is a boolean-like function of illumination. In other words, if the shading point is in shadow, then $L_i = 0$. If shading point is not in shadow $L_i = 1$. This formulation let us avoid all three parameters, i.e. $T(d_i, r_i)$, $\vec{C}_{L_i}$, and $S(\vec{C}_M)$, which we do not know anything about it. In this way, we can consider them as one lump-some parameter as $T(d_i, r_i)S(\vec{C}_M)\vec{C}_{L_i}$. Our goal is to estimate these parameters from images.

Estimating Parameters of Affine Model: Figure 12.6 shows a shadow data taken from an actual photograph of a scene that is illuminated by two lights. Using the table in Table 12.1, we can obtain the following equations:

$$\vec{C}_A = F(0,0) = \boldsymbol{C}_{0,0} = (87, 85, 90)$$

$$S(\vec{C}_M)T(d_0, r_0)\vec{C}_{L_0} = F(1,0) - F(0,0) = \boldsymbol{C}_{1,0} - \boldsymbol{C}_{0,0} = (25, 57, 122)$$

$$S(\vec{C}_M)T(d_1, r_1)\vec{C}_{L_1} = F(0,1) - F(0,0) = \boldsymbol{C}_{0,1} - \boldsymbol{C}_{0,0} = (91, 1, 5)$$

$$S(\vec{C}_M)\left(T(d_1, r_1)\vec{C}_{L_1} + T(d_0, r_0)\vec{C}_{L_0}\right) = F(1,1) - F(0,0) = \boldsymbol{C}_{1,1} - \boldsymbol{C}_{0,0} = (109, 67, 124)$$

Problems with Affine Model: The classical diffuse model in the previous paragraph suggests that an affine equation is sufficient to represent the illumination. However, as it can easily be confirmed from the results in the previous paragraph, these four color points are not on a plane most likely because of non-linearities introduced by photographing process. If we use the same affine equation with $n+1$ coefficients, it is possible to interpolate these 2^n color points if and only if these points are on an n-dimensional hyperplane. Unfortunately, the data we have collected demonstrate that these points will almost never be on a n-dimensional hyperplane. Therefore, using an affine equation we cannot guarantee to interpolate 2^n number of color points.

Approximate Modeling with Affine Models: An approximate solution to the above problem is to use linear regression by minimizing the error in interpolating 2^n color points. The result of linear regression provides an n-dimensional hyperplane that minimize L_2 error. By getting more sample for each case, we can get better approximation. Fortunately, there is an even easier solution to this problem.

Interpolation with n-Linear Barycentric Models: I discussed these models as bilinear ($n = 2$) and tri-linear ($n = 3$) Barycentric equations. In this chapter, I provide examples with three lights. I do not have an example with more than three lights that appears as distinct lights. In all practical cases, even the third light appears to be not as visually strong as others. Therefore, I expect that three is the largest n we expect in practical applications. More lights may start to appear as a combined area light. On the other hand, based on the discussion in the next section, it should be possible to generalize the method to n-Linear Barycentric models.

12.3 TWO LIGHTS AND BILINEAR MODEL

Two Lights: Figure 12.6 shows an example that demonstrates the venn diagram for two shadow regions. This data is obtained from a real photograph. We removed material properties in the second image by dividing every color to maximum color. The division is made based on the assumption that transformation matrix is just scaling. $(255, 255, 255)$ corresponds to the maximum illumination and reflection for $L_0 = 1$ and $L_1 = 1$. Unlike single distance light case, this assumption is not correct. However, in practice we can multiply this to obtain reasonable shadow colors that resemble original shadows even for different materials and distant regions.

Bilinear Barycentric Example: Based on the venn diagram, we can obtain partial illumination using Barycentric bilinear formula, which can simply be given as

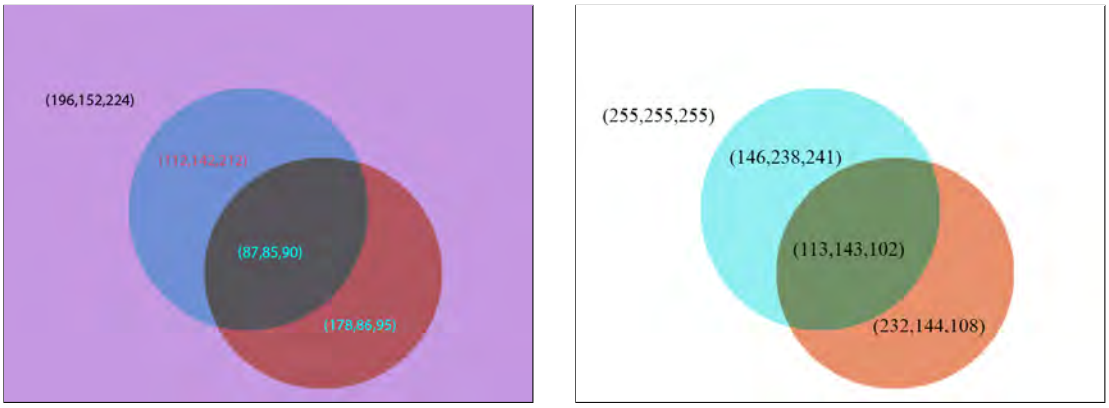
$$\begin{aligned} F(L_0, L_1) \quad = \quad & ((\boldsymbol{C}_{0,0,0}\,(1 - L_0) + \boldsymbol{C}_{1,0,0}\,L_0)\;\;(1 - L_1) + \\ & (\boldsymbol{C}_{0,1,0}\,(1 - L_0) + \boldsymbol{C}_{1,1,0}\,L_0)\quad L_1 \end{aligned}$$

and this bilinear formulation always guarantee to interpolate given four color points $\boldsymbol{C}_{0,0}$, $\boldsymbol{C}_{1,0}$, $\boldsymbol{C}_{0,1}$, and $\boldsymbol{C}_{1,1}$.

L_0	L_1	Colors to be interpolated
0	0	$F(0,0) = \boldsymbol{C}_{0,0} = (87, 85, 90)$
1	0	$F(1,0) = \boldsymbol{C}_{1,0} = (112, 142, 212)$
0	1	$F(0,1) = \boldsymbol{C}_{0,1} = (178, 86, 95)$
1	1	$F(1,1) = \boldsymbol{C}_{1,1} = (196, 152, 224)$

Table 12.1: Note that there is no affine function that interpolates these points. Fortunately, a bilinear function always exists.

Bilinear Multiplicative Function: Note that unknown terms $T(d_i, r_i)S(\vec{C}_M)\vec{C}_{L_i}$ should already be embedded into the equation of $F(L_0, L_1)$ through colors. Our goal is



(a) *A Venn diagram for shadow colors of two distinct lights in a local region.*

(b) *Same Venn diagram for shadow colors of two distinct lights by removing material effect.*

Figure 12.6: *An example that demonstrates the venn diagram for two shadow regions. The data is obtained from a real photograph. We removed material properties in the second image by assuming maximum illumination and reflection for $L_0 = 1$ and $L_1 = 1$.*

to eliminate them from the equation in a reasonable way. Since $\boldsymbol{C}_{1,1}$ supposed to include both of them and by assuming that there is a scaling operation between colors, we can divide all the colors to $\boldsymbol{C}_{1,1}$ to reduce the dependency to unknown terms. As a result, we obtain a new function that is less dependent on unknown terms as follows:

$$F'(L_0, L_1) = \boldsymbol{S}(\boldsymbol{C_{1,1}})^{-1} F(L_0, L_1). \tag{12.1}$$

This is still a bilinear function that interpolates given four color points $\boldsymbol{C}'_{i,j} = \boldsymbol{S}(\boldsymbol{C_{1,1}})^{-1}\boldsymbol{C}_{i,j}$. This operation converts $\boldsymbol{C}'_{1,1} = (1, 1, 1, 1) = (255, 255, 255, 255)$. I ignore $w = 255$ and use $\boldsymbol{C}'_{1,1} = (255, 255, 255)$ (See Table 12.2).

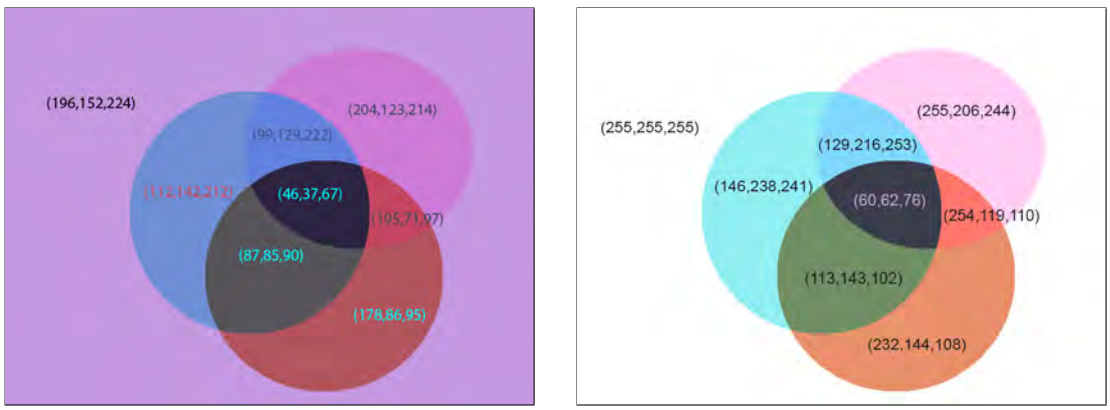
L_0	L_1	Colors to be interpolated
0	0	$F'(0,0) = \boldsymbol{C'}_{0,0} = (113, 143, 102)$
0	1	$F'(0,1) = \boldsymbol{C'}_{0,1} = (232, 144, 108)$
1	0	$F'(1,0) = \boldsymbol{C'}_{1,0} = (146, 238, 241)$
1	1	$F'(1,1) = \boldsymbol{C'}_{1,1} = (255, 255, 255)$

Table 12.2: Cardboard Diffuse Material. The most useful is shadow areas since they can provide the reflected colors for $t = 0$.

Shading with Bilinear Multiplicative Function: Using $F'(L_0, L_1)$, it is easy to obtain to create material and diffuse shade the newly added object. We first decide a material

property as a 4×4 matrix $\boldsymbol{M}$. We then compute two $B\&W$ rendering of the object using two distinct lights. These renderings only give us L_0 and L_1 for every visible point of newly added object. Then we simply compute $\boldsymbol{M}F'(L_0, L_1)$ as the color of the visible point. This approach turned out to be better to obtain reasonable results. Moreover, material properties and illumination distribution can easily be changed by changing $\boldsymbol{M}$ and by manipulating $F'(L_0, L_1)$ in a image manipulation software.

Shadowing with Bilinear Multiplicative Function: Shadowing is also easy using the multiplicative function. We simply render shadow regions as single $B\&W$ rendering for each light source using shadow of each light source on the proxy geometry which will give us L_0 and L_1 for every visible point. We then simply multiply background image with $F'(L_0, L_1)$.



(a) *A Venn diagram for shadow colors of three distinct lights in a local region.*

(b) *Same Venn diagram for shadow colors of three distinct lights by removing material effect.*

Figure 12.7: *An example that demonstrates the venn diagram for three shadow regions. The data is obtained from a real photograph. We removed material properties in the second image by assuming maximum illumination and reflection for $L_0 = 1$ and $L_1 = 1$.*

12.4 THREE LIGHTS AND TRILINEAR MODEL

Three Lights: Figure 12.7 shows an example that demonstrates the Venn diagram for three shadow regions using data that is obtained from a real photograph. We removed material properties in the second image by dividing every color to maximum color. The division is made based on the assumption that transformation matrix is just scaling. $(255, 255, 255)$ corresponds to the maximum illumination and reflection for $L_0 = 1$ and $L_1 = 1$. Unlike single distance light case, this assumption is not correct. However, in practice we can multiply this to obtain reasonable shadow colors that resemble original shadows even for different materials and distant regions.

Trilinear Barycentric Example: Based on the Venn diagram, we can obtain partial illumination using Barycentric trilinear formula, which can be given as

$$\begin{aligned} F(L_0, L_1, L_2) \quad = \quad & ((\boldsymbol{C}_{0,0,0}(1-L_0) + \boldsymbol{C}_{1,0,0}L_0) \;\; (1-L_1) + \\ & (\boldsymbol{C}_{0,1,0}(1-L_0) + \boldsymbol{C}_{1,1,0}L_0) \quad L_1 \;) \;\; (1-L_2) + \\ & ((\boldsymbol{C}_{0,0,1}(1-L_0) + \boldsymbol{C}_{1,0,1}L_0) \;\; (1-L_1) + \\ & (\boldsymbol{C}_{0,1,1}(1-L_0) + \boldsymbol{C}_{1,1,1}L_0) \quad L_1 \;) \quad L_2 \end{aligned}$$

This trilinear formulation always guarantee to interpolate given eight color points $\boldsymbol{C}_{i,j,k}$ where $(i, j, k) \in \{0, 1\}^3$. In this particular cases, colors are identified as shown in Figure 12.3

L_0	L_1	L_2	Colors to be interpolated
0	0	0	$F(0,0,0) = \boldsymbol{C}_{0,0,0} = (46, 37, 67)$
1	0	0	$F(1,0,0) = \boldsymbol{C}_{1,0,0} = (99, 129, 222)$
0	1	0	$F(0,1,0) = \boldsymbol{C}_{0,1,0} = (195, 71, 97)$
1	1	0	$F(1,1,0) = \boldsymbol{C}_{1,1,0} = (204, 123, 214)$
0	0	1	$F(0,0,1) = \boldsymbol{C}_{0,0,1} = (87, 85, 90)$
1	0	1	$F(1,0,1) = \boldsymbol{C}_{1,0,1} = (112, 142, 212)$
0	1	1	$F(0,1,1) = \boldsymbol{C}_{0,1,1} = (178, 86, 95)$
1	1	1	$F(1,1,1) = \boldsymbol{C}_{1,1,1} = (196, 152, 224)$

Table 12.3: Note that there is no affine function that interpolates these points. Fortunately, a trilinear function always exists.

Shading with Trilinear Multiplicative Function: Using $F'(L_0, L_1, L_2)$, it is easy to obtain to create material and diffuse shade the newly added object. We again decide a material property as a 4×4 matrix $\boldsymbol{M}$. We then compute two $B\&W$ rendering of the object using three distinct lights. These three renderings only give us L_0, L_1 and L_2 for every visible point of newly added object. We, then, simply compute $\boldsymbol{M}F'(L_0, L_1)$ as the color of the visible point. Material properties and illumination distribution can also be changed easily by changing $\boldsymbol{M}$ and by manipulating $F'(L_0, L_1)$ in a image manipulation software.

Trilinear Multiplicative Function: This is the same with two light case. We still do not know the terms $T(d_i, r_i)S(\vec{C}_M)\vec{C}_{L_i}$, which are already embedded into the equation of $F(L_0, L_1, L_2)$ through colors. To eliminate them from the equation in a reasonable way, we again divide all the colors to largest one, which is $\boldsymbol{C}_{1,1,1}$, by assuming that there is a scaling operation between colors. As a result, we again obtain a new function that is less dependent on unknown terms as follows:

$$F'(L_0, L_1, , L_2) = \boldsymbol{S}(\boldsymbol{C}_{\mathbf{1,1,1}})^{-1} F(L_0, L_1). \tag{12.2}$$

This is still a trilinear function that interpolates given eight color points $\boldsymbol{C}'_{i,j,k} = \boldsymbol{S}(\boldsymbol{C_{1,1,1}})^{-1}\boldsymbol{C}_{i,j,k}$. This operation, as before, converts $\boldsymbol{C}'_{1,1,1} = (1,1,1,1) = (255,255,255,255)$. I also ignore $w = 255$ and use $\boldsymbol{C}'_{1,1} = (255,255,255)$ (See Table 12.4).

Shadowing with Trilinear Multiplicative Function: Shadowing is also easy using the trilinear multiplicative function. We simply render shadow regions as single $B\&W$ rendering for each light source using shadow of each light source on the proxy geometry which will give us L_0, L_1 and L_2 for every visible point. We then simply multiply background image with $F'(L_0, L_1, L_1)$.

Shadowing with Trilinear Multiplicative Function: Shadowing is also easy using the trilinear multiplicative function. We simply render shadow regions as single $B\&W$ rendering for each light source using shadow of each light source on the proxy geometry which will give us L_0, L_1 and L_2 for every visible point. We then simply multiply background image with $F'(L_0, L_1, L_1)$.

L_0	L_1	L_2	Colors to be interpolated
0	0	0	$F'(0,0,0) = \boldsymbol{C'}_{0,0,0} = (60,62,76)$
1	0	0	$F'(1,0,0) = \boldsymbol{C'}_{1,0,0} = (254,119,110)$
0	1	0	$F'(0,1,0) = \boldsymbol{C'}_{0,1,0} = (178,86,95)$
1	1	0	$F'(1,1,0) = \boldsymbol{C'}_{1,1,0} = (265,206,244)$
0	0	1	$F'(0,0,1) = \boldsymbol{C'}_{0,0,1} = (129,216,253)$
1	0	1	$F'(1,0,1) = \boldsymbol{C'}_{1,0,1} = (232,144,108)$
0	1	1	$F'(0,1,1) = \boldsymbol{C'}_{0,1,1} = (146,238,241)$
1	1	1	$F'(1,1,1) = \boldsymbol{C'}_{1,1,1} = (255,255,255)$

Table 12.4: Note that there is no affine function that interpolates these points. Fortunately, a trilinear function again always exists. Note that red component of $\boldsymbol{C'}_{1,1,0}$ is 265, which is larger than 1 in normalized case. This particular term introduces additional non-linearity. In current systems, we need to choose it as 255.

12.5 AMBIENT TERM

Adding Ambient Effect: It is, of course, better to add another term for ambient illumination, which can be handled using more lights. Based on this general idea n-linear, it should be easy to add additional term. One issue is that we do not have that many samples samples to include ambient term in a n-linear function. Therefore, ambient can be added in a simplified form. For instance, to add ambient term in two and three light cases, we

simply write the equation in the following forms

$$F'(L_0, L_1)A + \boldsymbol{C'}_A(1 - A)$$

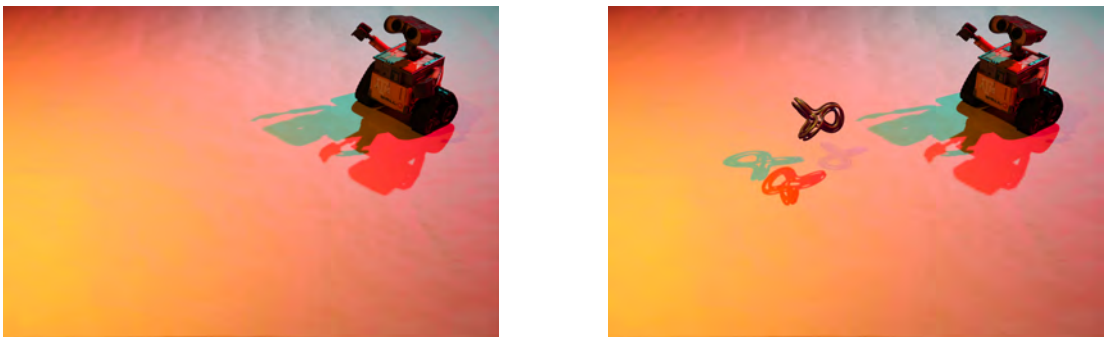
$$F'(L_0, L_1, L_2)A + \boldsymbol{C'}_A(1 - A)$$

where $\boldsymbol{C'}_A$ is normalized version of desired ambient color. The first equation defines a pyramid and the second equation defines a 4D hyper-pyramid, in which the top of the pyramid is the ambient color. In other words, only when ambient illumination becomes zero we see the ambient color. This formulation is important since it allows us to control ambient color precisely as we wanted.

CHAPTER 13

Compositing into Diffuse Scenes Illuminated by Multiple Colored Lights

Introduction: In this chapter, I discuss how to obtain a seamless composite of computer generated (CG) objects with the photographs of scenes that are illuminated by more than one colored light as shown in Figures 13.3, 13.2 and 13.1. I provide two case studies. To take photographs of these scenes, we used three lights in a photography studio. We created colors using color filters.



(a) *Original Photograph.* (b) *Compositing.*

Figure 13.1: *Another example of compositing into photographs of diffuse scenes illuminated by three colored lights. Note that in this case ground is simple plane. The main difficulty comes from matching colors of CG shadow to colors of original shadows. This composite image is crated by Qiao Wang.*

Motivation: One of the common misperceptions about shadows is that people think that shadows are colorless. In other words, we expect shadow colors to have no saturation. In practice, even if they are very dark, shadows has clearly defined hues. Unconsciously we can clearly differentiate between dark blue shadow vs. dark red shadow. It is important for students to develop a conscious understanding of existence of shadow colors. I, therefore, designed this particular case study as a project in my class to demonstrate students shadows

can be very rich in color. I constructed relatively simple scenes that include only a single plane and a single object to demonstrate the shadow effects of multiple colorful lights.

13.1 CASE STUDY 1

(a) *Original Photograph.* (b) *Compositing.*

Figure 13.2: *An example of compositing into photographs of diffuse scenes illuminated by three colored lights. Note that in this case ground is simple plane. The main difficulty comes from matching colors of CG shadow to colors of original shadows. This composite image is crated by Alyssa Pena.*

Case Study 1: Figure 13.1 shows an example of student projects using a photograph of a scene illuminated by three lights. I have discussed this particular data in the previous chapter. The main problem in this case, when the object moves the color of the shadows has to change in such a way we should not perceive shadows as wrong.

Proxy Shapes: This particular problem is also simple in terms of creation of proxy shapes since it is mostly a plane. The original 3D object is not that complicated. It can easily be represented as an approximate masked 3D shape.

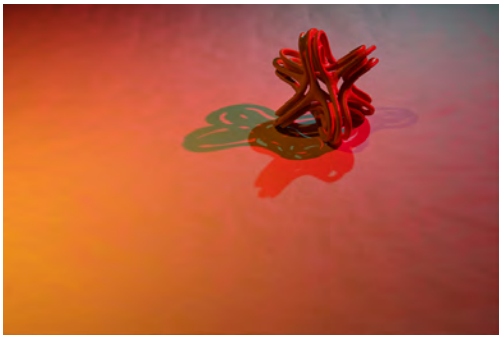
13.2 CASE STUDY 2

Case Study 2: Figure 13.1 shows a set of frames from a composited animation. These frames demonstrate how moving shadows appear using the method presented in the previous chapter. It also shows the need for handling intersecting shadows.

Proxy Shapes: This particular problem is still simple in terms of creation of proxy shapes since it is mostly a plane. The original 3D printed object is complicated however, it can simply be masked and represented as locally cylinder pieces.

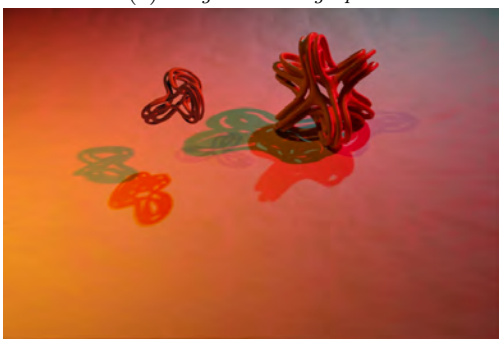
Moving Shadows: As artistically demonstrated by Joseph Albers in his highly influential book, Interaction of Color [76] in 1965, our color perception is relative to the colors around it. Huseyin Yilmaz, around the same time, developed theory of relativistic color perception [309, 311]. As also shown in this examples, if we keep the ratio approximately same, colors of shadows appears similar even when shadows move different regions even though colors are not the same as the original shadow colors.

Intersecting Shadows: The most important difficulty comes from handing interaction of real shadows with original shadows as shown in Figures 13.3. We need they are correctly masked and based on the interaction, their color is changed. For instance, when one original and newly created shadows corresponding to the same light is casted to the same point, there should be no change. On the other hand, based on the combination we need to apply different multiplication factors. In other words, the shadow mask must also include information about corresponding lights.



(a) *Original Photograph.*

(b) *A frame from compositing animation.*



(c) *Another frame from compositing animation.*

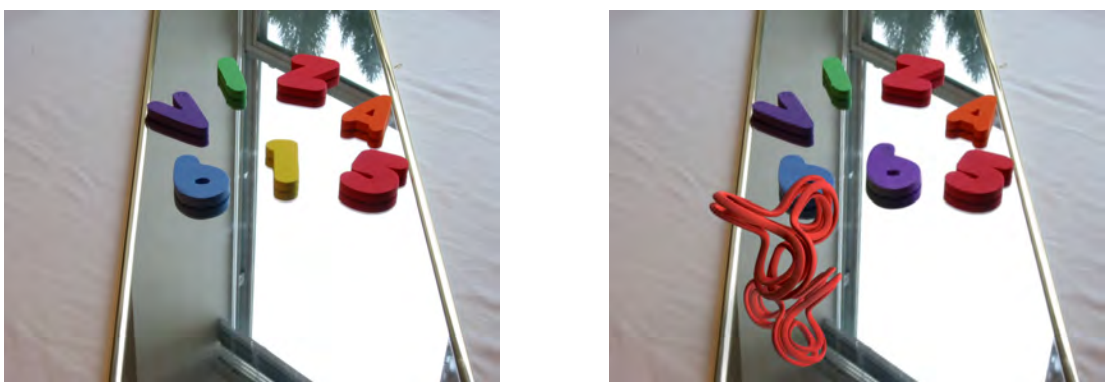
(d) *Another frame from compositing animation.*

Figure 13.3: *Another example of compositing into photographs of diffuse scenes illuminated by colored lights. Note that in this case ground is again simple plane. The difficulty comes from matching colors of CG shadow to colors of original shadows during animation. This composite image is crated by Yinan (Yvonne) Xiong.*

CHAPTER 14

Compositing into Scenes with Reflective Objects

Introduction: In this chapter, I discuss how to obtain a seamless composite of computer generated (CG) objects with the photographs of scenes that include flat mirrors as shown in Figure 14.1. I provide only one case study since this is really simplest case in which we only deal with a single plane.



(a) *Original Photograph.* (b) *Compositing.*

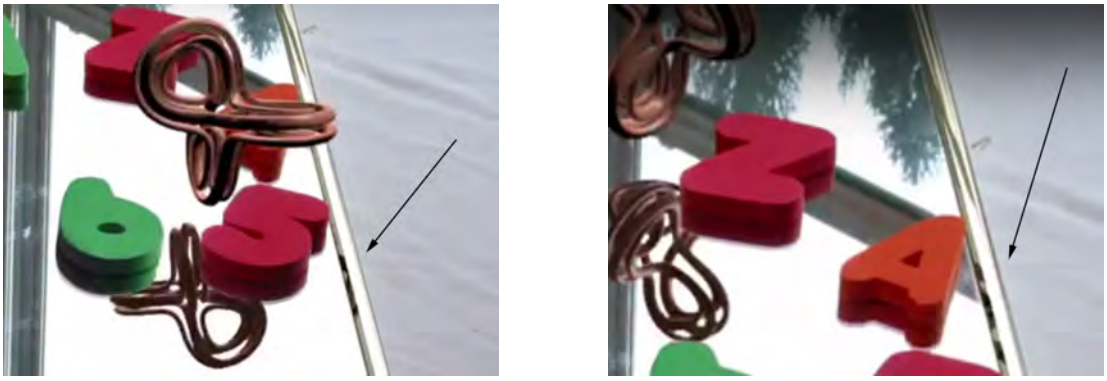
Figure 14.1: *An example of compositing into photographs of reflective objects. Note that in this case ground is simple plane. The main difficulty comes from matching colors of CG shadow to colors of original shadows. This composite image is crated by Oriyomi Adenuga.*

Description of the Case Study: As shown in Figure 14.1, the original photograph is very simple. When we took this photograph, there was no separate digital compositing class, called Viza 665. Therefore, the original letters spells computer animation course, called Viza 615. Therefore, one of the goals of the project to replace the number one with number six.

Reflection of the Light Sources: One of the challenge in this case is that the light coming from window is reflected by the mirror plane and that reflection needs to illuminate the newly added object. If we do not consider this additional light source the illumination

will not be correct. In other words, we need to create an additional area light that is reflection of the original light by the mirror plane.

Proxy Objects: The proxy object that represent mirror can simply be a masked plane. Moreover, the normal and position of the plane do not have to be precise to obtain acceptable results. A challenge is the border of the mirror since it is also reflective. This reflective surface can simply be represented again as a mashed cylinder. Note that cylinder reflect the whole space. The example shown in Figure 14.1 does not have that reflection in the borders. This problem might not appear important in a single static image. On the other hand, these minor reflections are important to obtain believable results during an animation since they quickly move on a cylindrical surfaces as shown in Figure 14.2.



(a) *A detail of composited reflection on a cylindrical surface.*

(b) *Moving reflection cylindrical surface.*

Figure 14.2: *An example of moving reflection on a cylindrical surface. The original composite image of these two detail images are crated by Megan Cook.*

Color Transformations: In this case, mirror is simple and there is no color change. Therefore, there is no need for color transformations.

CHAPTER 15

Reconstruction of Refraction

Introduction In this chapter, I discuss methods for reconstruction of refraction to obtain plausible refractions. Figure 15.1 shows a conceptual example that demonstrates only refraction, ignoring reflections, which also occurs on transparent objects due to Fresnel equation. We again use Masked-3D or Mocked-3D shapes for simple and effective reconstruction of refraction. We first need to identify parameters that can impact refraction before discussing of reconstruction of these two shape types.

(a) *Original bottle.* (b) *Foreground.* (c) *Replacement background.* (d) *Final composite.*

Figure 15.1: *A conceptual example demonstrating important elements of refraction. Figure 15.1a shows the original bottle that is replaced by masked-3D shapes.*

15.1 REFRACTION AS A TRANSFORMATION

Refraction as an Image Transformation: Refraction can be considered as an image transformation that deforms a given background image. We, therefore, can view a real transparent object as a transformation mapping. If we can replace real transparent objects with these "maps" that can intuitively provide control of refraction. Figure 15.1 demonstrates that the most important parts of refraction mapping are discontinuous regions. Smooth mapping is not that critical for visual similarity. As we have observed with shadow

compositing, discontinuities in the refractions of the newly added shapes must correctly match with discontinuities in the original image.

Parameters of Refraction Transformation: There are three main parameters that define refraction as a transformation. We, therefore, can think a real transparent object as a transformation. The key control parameters of this transformation are (1) normal vector, $\vec{n}$, to the surface that serves as an interface between two different isotropic media, such as water, glass, or air; (2) relative index of refraction η, which is given a ratio of the index of refractions of the two isotropic media. These two parameters are well understood in Computer Graphics literature. However, I noticed that there is another parameter, which is not well understood since it is just a byproduct that is automatically obtained through rendering. As it can be clear later in this chapter, it is impossible to control results without that additional parameter. In the next paragraph, I briefly discuss that parameter.

Thickness as a Parameter for Refraction Transformation: I noticed that a significantly important parameter for refraction transformation is the distance the light ray travel inside of isotropic media assuming that the light does not have total reflection inside of the medium. Let d denote this distance for a given light ray and for a given isotropic media. In a layman's terms, the bigger the d is, the more refraction mapping transforms background image. Although d is very complicated in general, In order to understand it's effect on refraction, we can simply ignore any complication and assume d is a single number for any given shading point. For further simplification of the discussion, I call d thickness parameter.

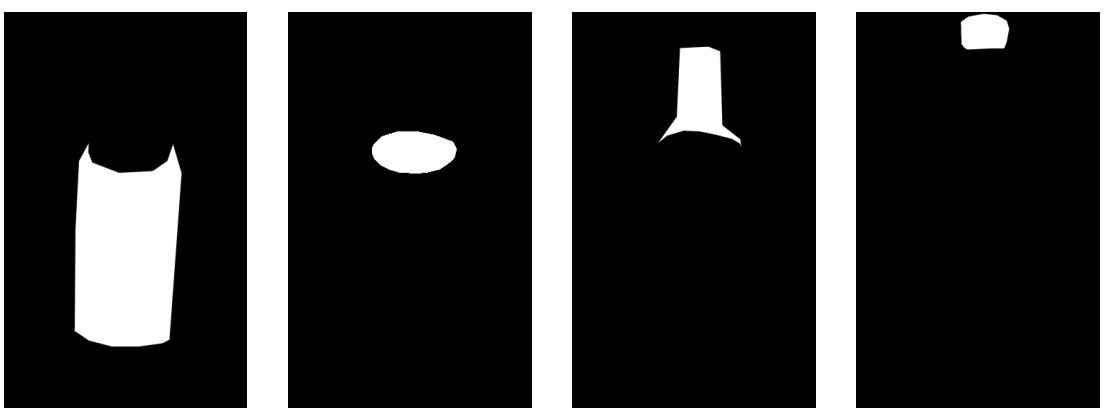
Importance of Thickness as a Parameter: To demonstrate the importance of thickness, assume the thickness of transparent object is zero, i.e. $d = 0$. This is equivalent of choosing index of refraction $\eta = 1$ even if actual index of refraction is not one. Regardless of normal vectors, $\vec{n}$'s, in the intersection of the two isotropic media, we will not see any deformation. In other words, If the thickness is small, index of refractions, η's, and normal vectors, $\vec{n}$'s, do not have any impact on refraction. Therefore, in such case, no significant deformations can be caused by refractions. This is the reason thin objects only appear transparent and the do not deform.

Causes for Discontinuities: Based on this discussion, it is possible to figure out how to introduce discontinuities. If refraction is viewed as a transformation, visual discontinuities occur in the places where the refraction transformation is not C^1 continuous. In real refractions, these discontinuities are caused by (1) in the regions where normal vectors, $\vec{n}$'s at the interfaces that separate two isotropic media, are C^1 discontinuous; (2) the regions where more than two isotropic media with different index of refractions η meet; (3) the thickness is C^0 discontinuous. The last one is particularly important. For instance, in the

bottle image the thickness C^0 discontinuous in the region from top of the water to the neck of the bottle 15.1.

Reconstruction of Discontinuities for Refraction: One problem is that we do not really define volumes of isotropic media in Computer Graphics. Our most common way of describing shapes are polygonal surfaces. Unfortunately, they do not necessarily define volume since polygonal surfaces could be either self-intersecting or manifolds with boundaries. As a solution, we use relative index of refraction, which is obtained by dividing two index of refraction. Relative index of refraction can be assigned to surfaces since it represents change in the interface between two isotropic medium. This is a very useful practical solution for rendering transparent surfaces. In this way, even if the polygonal surfaces have self-intersections or boundaries, our rendering works well. However, this is not useful for compositing since we have to be consistent with original discontinuities. Therefore, we cannot use a completely 3D approach as it is harder to match discontinuous regions.

Masked-3D vs. Mock-3D for Refraction: To avoid the unwanted discontinuities, we again use Masked-3D and Mocked-3D shapes. They also play an important role to exactly match discontinuous regions in refractions. In the rest of this chapter, I discuss how these two types of shapes can be used for obtaining compositing with refraction and I discuss advantages and disadvantages of the two types of shapes for refraction.



(a) *Mask for a cylinder.* (b) *Mask for a plane.* (c) *Mask for bottleneck.* (d) *Mask for collar of the bottle.*

Figure 15.2: *A conceptual example that demonstrate the minimum number of masks that is needed to obtain discontinuities with masked-3D shapes based on the original bottle in Figure 15.1a. Note that, bottleneck is really a simple thin object that does not refract much. We can therefore simply ignore to have a 3D model.*

15.2 MASKED-3D FOR REFRACTION

Preview: As discussed earlier, masked-3D shapes are obtained by masking projections of simple geometric objects. This approach not only simplifies modeling side, but also helps to match discontinuous regions correctly in images. For refraction, it is even better to further subdivide original shapes into simpler masked shapes. For instance, in the bottle model in Figure 15.1, the top of the water is simply a plane and there is no need to represent with a more complicated geometric shape. By masking this plane with a circular mask shown in Figure 15.2b, we simply obtain the top of the water surface. Note that this view, in reality, also includes thin glass. Fortunately, we can ignore it since thin transparent objects do not refract much.

Discontinuous Regions: Any C^1 discontinuity on surfaces that represent real objects can introduce additional visual discontinuities that may not exist in the photograph. It is easy to avoid this using masks and continuous surfaces. If we use masks and simple continuous objects such as planes, infinite cylinders and spheres, we can easily guarantee obtain discontinuities exactly at mask boundaries. The main problem with this approach is that the number of masks can be high. For instance, the conceptual example in Figure 15.2 demonstrates that we need at least four masks to handle conceptually different regions of bottle. This means that we also have to use at least four surfaces. Refracted images obtained by each one of them are appropriately cropped using the masks in Figure 15.2 as shown in Figure 15.3. In other words, this process requires significant number of layers, i.e. operations. Mock-3D shapes provides an alternative to represent the same process with fewer operations.

15.3 MOCKED-3D FOR REFRACTION

Efficiency of Mock-3D for defining Refraction Transformations: Mock-3D shapes can represent the refraction transformation using a single image. This is the key reason behind advantage of Mock-3D shapes as a reconstruction of real transparent objects.

Mock-3D for Refraction: Mock-3D shapes are either normal maps or height-fields with an alpha map that define its 2D, i.e. projected shape. To represent transparent objects with Mock-3D shapes, we need to add relative index of refraction, η, and thickness information, d, to every pixel. Since all three parameters, i.e. normal vectors, index of refractions and thicknesses are interdependent in impacting deformations, it is possible to use a single index of refraction for the whole shape. Then, we only need to provide normal vector and thickness information per pixel.

Mock-3D from Normal Vectors: Fortunately, we do not really need to have an additional parameter to encode thickness information for normal vectors. Since the normal

(a) *Refraction of the cylinder masked by 15.2a.*

(b) *Refraction of the plane masked by 15.2b.*

(c) *Refraction of the the neck of the bottle masked by 15.2c.*

(d) *Refraction of the the collar of the bottle masked by 15.2d.*

Figure 15.3: *A conceptual example that demonstrate masked reflections that are used to create composited image in Figure 15.1d.*

vectors are unit vectors, the z component can be computed from x and y components by observing z is always positive number. We can, therefore, use the the third component to provide thickness. Even if we do not use the z component for thickness, the actual z component of the normal vector provides a good estimation of the thickness.

Mock-3D with Height Fields: Height fields require just a single parameter to provide shape. If we can use an image as an height field, the other two components can be used to encode thickness. Even if thickness information is not given, we can simply compute normal map using gradient of height field and use its z component as a thickness information.

Mock-3D with Thickness: In other words, we can simply use the z value in normal maps as thickness as shown in Figures 15.4a and 15.4c. Figures 15.5a and 15.5b show how the thickness information in Figure 15.4a is used to obtain physically plausible deformations for refraction even in NPR setting. Similarly, Figure 15.4 provides examples how the thickness information in Figure 15.4c is used to obtain physically plausible deformations for refraction even in NPR setting. Note that the yellow region in Figure 15.4a does not deform background since $z = 0$. For such regions, reflection actually helps transparent feeling through Fresnel as discussed in the next chapter. In order words, to have interesting refractions, the objects must have a significant amount of volume.

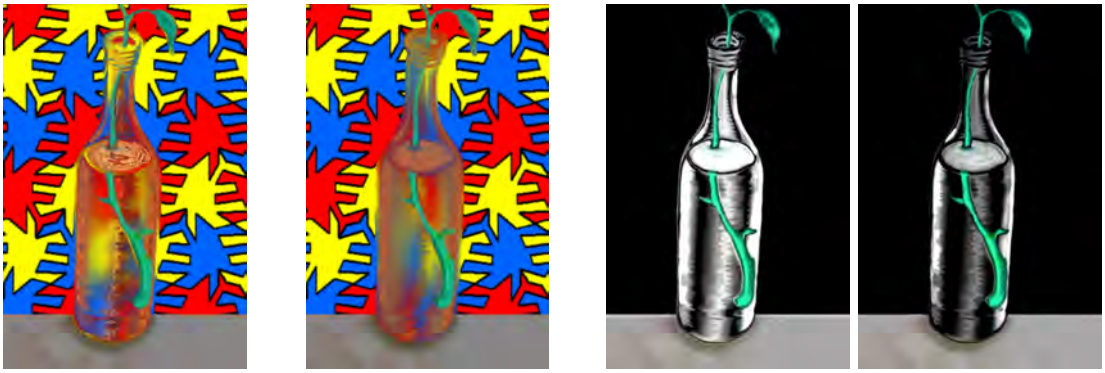
Transparent Mock-3D Shapes: Normal Maps with Thickness: In practice, these Mock-3D images can be used as a reconstruction of transparent 3D objects by providing sufficient 3D information to obtain acceptable refraction effects (See Figure 15.4). These

(a) *A hand-drawn Mock3D shape with thickness.* (b) *Corresponding non-transparent foreground.* (c) *Another hand-drawn Mock3D shape with thickness.* (d) *A corresponding non-transparent foreground.*

Figure 15.4: *Two examples that demonstrates hand-drawn Mock-3D shapes by painting images. These two particular Mock-3D shapes are created to obtain artistically inspired rendering results. Red and green components are simply painted based on desired normal vectors and blue component simply provides thickness information. The yellow region does not have thickness since blue is zero. It is therefore that regions does not refract. Actual transparent regions are provided by additional foreground images that provide alpha to define transparent regions shown as white in these images. This is just an example. In direct compositing applications, alpha channel is actually part of Mock-3d shape image and we completely ignore foreground since it already exists.*

images can be inserted in any digital paint system without any significant structural change. It is possible to obtain qualitatively convincing reflection and refraction effects with a minimal amount 3D information, such as normal as a vector field and thickness [277]. This 3D information, moreover, does not have to be complete or consistent for obtaining convincing composites. The images that provide 3D information can directly be obtained through 3D rendering by harvesting normal and thickness information. More important for 2D artists, they can be directly sketched or painted based on artists' intention using a 2D drawing software such as Lumo [177] or Crossshade [250].

Linearized Reflections & Refractions : We can also replace standard reflection and refraction effects with linearized transformations for art-directed intuitive control. These linearized transformations provide warping effects that are visually and qualitatively similar to 3D realistic rendering since they indirectly correspond to underlying physical phenomena of reflection and refraction. The linearized versions are also useful. Using linearized versions, we can completely eliminate discontinuities due to complicated interactions caused by total reflections. If they exist in original image, we can introduce them using discontinuities in normal or thickness.



(a) *Reflection and Refraction.*

(b) *Translucent and Glossy.*

(c) *Fresnel control by index of refraction.*

Figure 15.5: *An example of compositing with refraction that demonstrates the effect of thickness in a linearized refraction model. (a) shows reflection and refraction composited with our Fresnel function. (b) shows glossy reflection and translucent refraction combined with our Fresnel function. (c) directly shows Fresnel control with index of refraction using a black background and white environment map.*

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

Figure 15.6: *Four examples that demonstrate the effect of thickness in a linearized refraction model. (a) shows reflection and refraction composited with our Fresnel function. (b) shows glossy reflection and translucent refraction combined with our Fresnel function. (c) directly shows Fresnel control with index of refraction using a black background and white environment map.*

CHAPTER 16

Case Study: Compositing into Scenes with Transparent Objects

Introduction: In this chapter, I discuss how to obtain a seamless composite of computer generated (CG) objects with the photographs of scenes that include transparent objects as shown in Figure 16.4. This problem is really very simple based one methods discussed in Chapters 14 and 15 and combining results using Fresnel equation.

(a) *Original Photograph.*

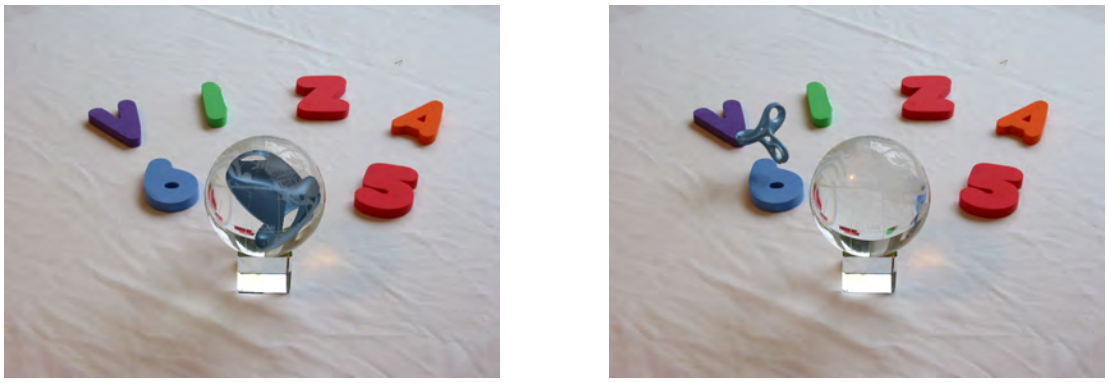
(b) *Mask.*

Figure 16.1: *Original photograph and mask for case 1.*

Case Studies: I provide two case studies. The first one is a glass sphere with some glossy and translucent regions (See Figure 16.1a). The second one is a glass vase filled with water along with some non-transparent regions (See Figure 16.3a).

16.1 CASE STUDY 1

Spherical Transparent Globe: In this case, we have a transparent spherical globe made from glass. The areas that correspond to land are sanded to obtain translucency (See Figure 16.1a). This problem is very simple since the whole shape can be represented by a



(a) *Compositing.* (b) *Compositing.*

Figure 16.2: *Example of frames from a compositing animation with transparent objects. In this case we need to handle warping caused by refraction when the CG object is behind the transparent object. The composite animation is crated by Dustin Han.*

single spherical object. We still need to have a mask to match the boundaries of the shape (See Figure 16.1b). It is also even easier to reconstruct this spherical globe with Mock-3D. We can simply obtain normal maps from the mask using an approach similar to Lumo [177]. Examples of composited frames are shown in Figure 16.2.

(a) *Original Photograph.* (b) *Compositing.*

Figure 16.3: *Original photograph and a compositing for case 2. In this case we need to handle warping caused by refraction when the CG object is behind the transparent object. The composite animation is crated by Pornubpan Hengtaweesub.*

Translucent Regions: Handling translucent regions are also simple. The simplest and fastest solution is to obtain exact reflection and refraction and then apply low-pass filtering to the regions defined by translucency masks. The second solution is to obtain translucency

and glossy through rendering and in masked areas use translucent and glossy rendering results. It is also possible to completely ignore that area since effect is very subtle.

16.2 CASE STUDY 2

Glass Vase: In this case, we have a simple shaped glass vase filled with water (See Figure 16.3a). As in bottle example, this problem can be decomposed into several pieces using Masked-3D shapes: (1) One upside down cone to represent part that is filled with water; (2) A planar surface to represent the top of the water; (3) Another thin conical piece for only glass part, which is used only for reflection since it does not refract much; (4) A toroidal surface for the rim of the vase. The rim area has a very high curvature and it can reflect the whole scene. This is particularly important during composited animation since we are expected to see the motion of the newly added object in the rim. We also need masks for foreground flower and ball inside of the water. Examples of composited frames are shown in Figure 16.4.

(a) *Compositing.* (b) *Compositing.*

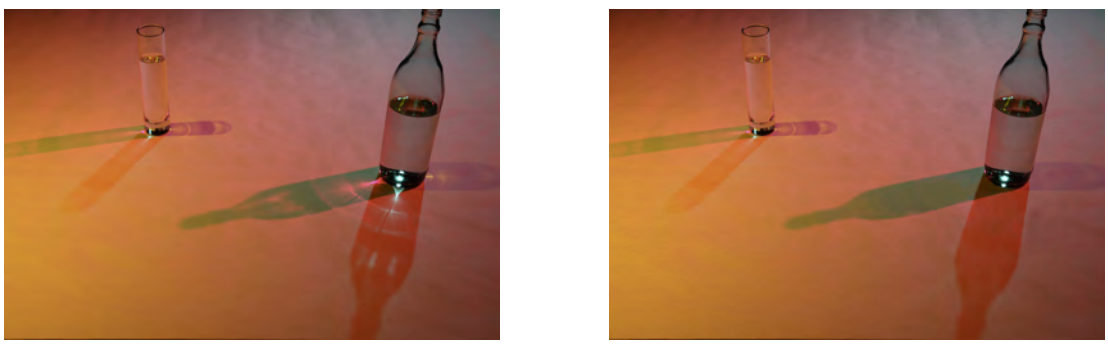
Figure 16.4: *Additional examples of frames from a compositing animation by Pornubpan Hengtaweesub.*

CHAPTER 17

Reconstruction of Caustics

Introduction In this chapter, I demonstrate that caustics also need to be updated with the inclusion of a new object. I have not recognized this issue until recently. The original photographs I used for compositing have only subtle caustic effects, which may not be visible a casual observer. However, the set of additional photograps of transparent shapes under colored studio lights, which was taken by Shyam Kannapurakkaran took, demonstrated strong caustic effect, which are harder to ignore for compositing.

Motivation Shyam's photographs, because of contrast created by studio lighting, clearly demonstrate caustic effects. When we use these photographs it was clear that the newly added object will have to change existing caustics. I, therefore, needed to find a simple solution to include the effect of newly added object over caustics (See Figure 17.2). Composting with caustics turned out to be also important for subtle caustic effects by improving compositing results (See Figure 17.3).



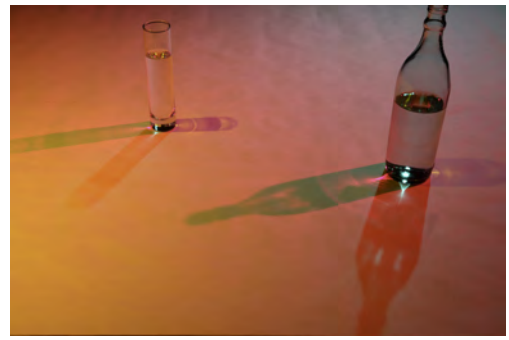
(a) *Original Photograph:* $\boldsymbol{I_0}$. (b) *Caustics Removed Image:* $\boldsymbol{I_1}$.

Figure 17.1: *Original and caustic removed images.*

17.1 METHODOLOGY

Observations: Our method is based on the observation that a newly included diffuse object will cast a shadow that changes caustic region of a transparent object. If any point is in shadow, caustics will simply be replaced by shadow as if that part of the original object is diffuse. Our advantage is that it is not possible for human observer to correctly estimate

(a) *Shadow of newly added object:* $\boldsymbol{I_S}$.



(b) *Compositing Result:* $\boldsymbol{I_F} = \boldsymbol{I_0}\ \boldsymbol{I_S} + (\mathbf{1} - \boldsymbol{I_S})\ \boldsymbol{I_1}$.

Figure 17.2: *An example of inclusion of impact of the newly added object on strong existing caustics.*

the shape of the shadow of the newly introduced object since lights passing through the transparent object are bent.

Our Method: Our method consists of two stages. In the first stage, we remove caustics from original images by painting over caustics as shown in Figures 17.1b and 17.3b. In the second stage, we compute a weighted average of original image with caustic image using shadow of newly added diffuse object.

The First Stage: The caustic removed image could easily be painted using dark regions in caustic image. One issue is that dark regions in caustic region can be darker than actual shadow. One advantage of this method is that the caustics removed images do not have to be perfect. As far as it looks like the original transparent objects are replaced by corresponding diffuse objects casting shadow, it is sufficient for our purpose. The shadow is particularly visible in Figure 17.1b. For subtle caustics, such as the one shown in Figure 17.3a, it is harder to paint corresponding shadow image(see Figure 17.3a). However, since subtle caustics are anyway hard to see, mistakes in paintings are more acceptable in these cases.

The Second Stage: Let $\boldsymbol{I_1}$, $\boldsymbol{I_0}$, $\boldsymbol{I_S}$ and $\boldsymbol{I_F}$ denote the original image, the caustic removed image, and shadow image respectively. Then, the final compositing images shown in Figures 17.2b and 17.3d can be computed by using following equation as

$$\boldsymbol{I_F} = \boldsymbol{I_0}\boldsymbol{I_S} + (\mathbf{1} - \boldsymbol{I_S})\boldsymbol{I_1}$$

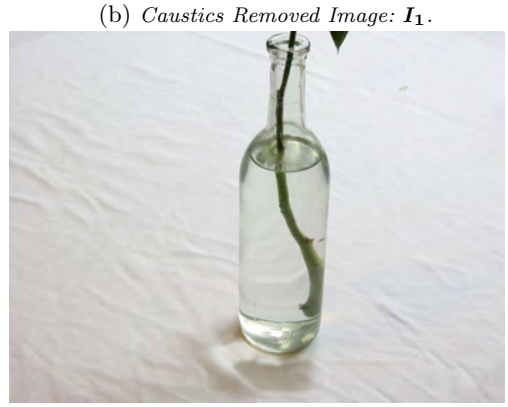
where $\mathbf{1}$ is a completely white image.

Additional Comments: The same method can also work if the newly added object is transparent. In that case, $\boldsymbol{I_S}$ will be replaced by a caustic image. The only issue that needs

(a) *Original Photograph:* $\boldsymbol{I_0}$.

(b) *Caustics Removed Image:* $\boldsymbol{I_1}$.

(c) *Shadow of newly added object:* $\boldsymbol{I_S}$.



(d) *Compositing Result:* $\boldsymbol{I_F} = \boldsymbol{I_0}\ \boldsymbol{I_S} + (\boldsymbol{1} - \boldsymbol{I_S})\ \boldsymbol{I_1}$.

Figure 17.3: *An example of inclusion of impact of the newly added object on weak caustics.*

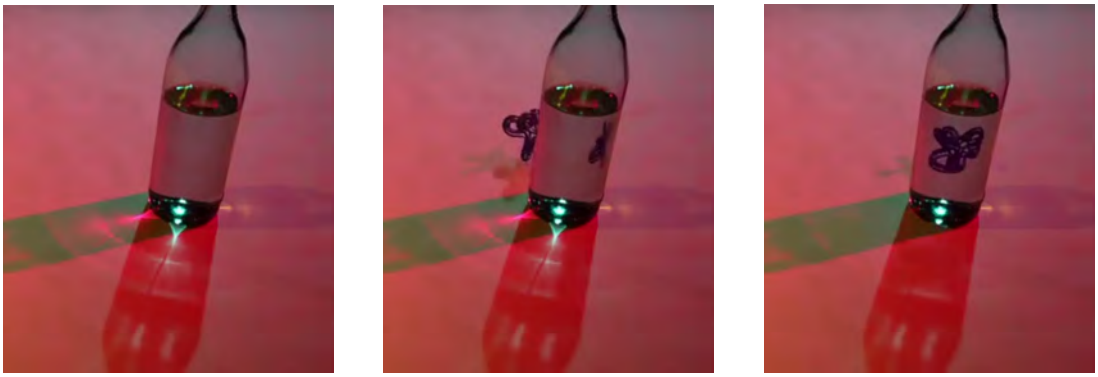
attention is that caustic image can be high dynamic. Therefore, the largest values need to be scaled down to 1.

CHAPTER 18

Case Studies: Compositing Caustics

Introduction In this chapter, I provide some compositing caustics examples done by my students in recent digital composting courses. Unfortunately, it is hard to appreciate caustics without seeing the object in motion. Therefore, I strongly suggest you look at class videos.

Case 1: Figures 18.1 and 18.2 shows cropped frames from a compositing animation with caustics created by Kelly Rogers. As shown in this example, the effect of newly added object over caustics is not clearly visible in single frame. However, the effect is clearly visible during animation since caustics are always changing. Figure 18.3 shows difference of original and composited frames to clearly demonstrate the change in the caustics region of the compositing animation. Another important detail in this particular case, Kelly chose to use modified shadow, which gives a little more natural impact.



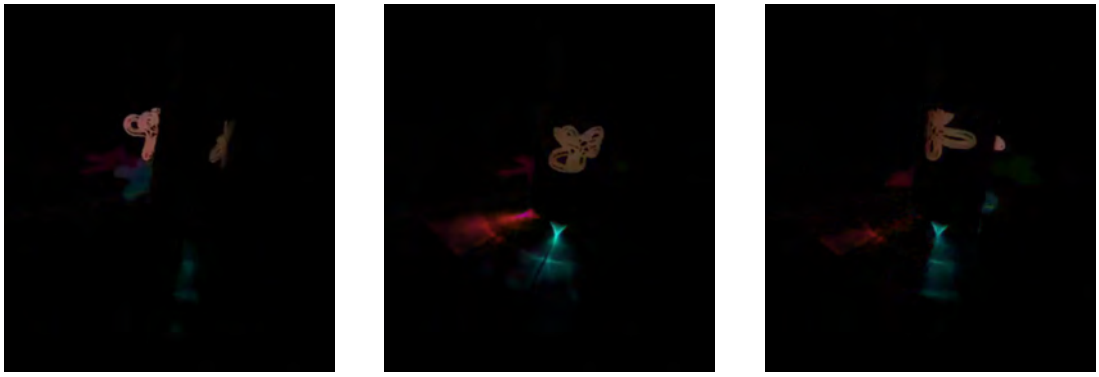
(a) *Original.* (b) *Compositing 1.* (c) *Compositing 2.*

Figure 18.1: *Example of frames from a compositing animation with caustics. The composite animation is created by Kelly Rogers.*

Case 2: Figures 18.4 and 18.5 shows cropped frames from a compositing animation with caustics created by Carl Van Huyck using the same original image. As shown in this example, the effect of newly added object over caustics is more visible in single frame

(a) *Compositing 3.* (b) *Compositing 4.* (c) *Compositing 5*

Figure 18.2: *More examples of frames from the compositing animation with Caustics by Kelly Rogers.*



(a) *Compositing 1 - Original.* (b) *Compositing 2 - Original.* (c) *Compositing 5 - Original*

Figure 18.3: *Difference images that demonstrate the impact of newly added object on caustics.*

as shadow. This is because Carl chose to use unmodified shadows. Although, unmodified shadows is not really correct, it still works in practice. Figure 18.6 shows difference of original and composited frames to clearly demonstrate the change in the caustics region of the compositing animation.

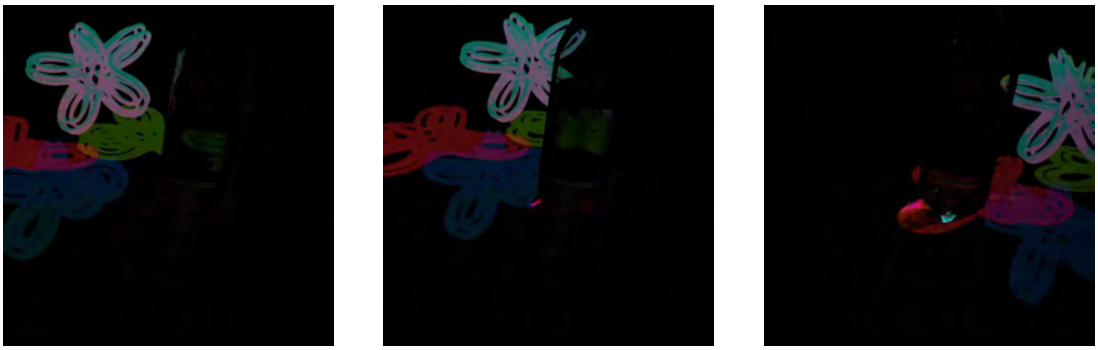
Case 3: Figure 18.7 shows cropped frames from a compositing animation with caustics created by Carl Van Huyck for another region of the same original image for a cylindrical glass. Figure 18.8 shows difference of original and composited frames.

(a) *Original.* (b) *Compositing 1.* (c) *Compositing 2.*

Figure 18.4: *Example of frames from a compositing animation with caustics. The composite animation is created by Carl Van Huyck.*

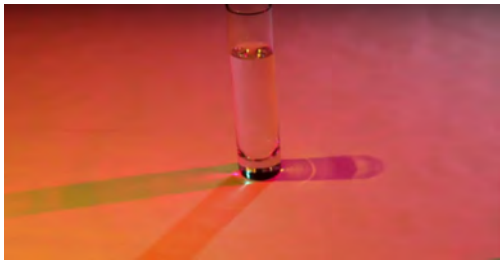
(a) *Compositing 3.* (b) *Compositing 4.* (c) *Compositing 5*

Figure 18.5: *More examples of frames from the compositing animation with Caustics by Carl Van Huyck.*



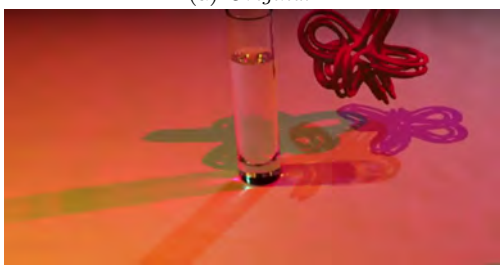
(a) *Compositing 1 - Original.* (b) *Compositing 2 - Original.* (c) *Compositing 5 - Original*

Figure 18.6: *Difference images that demonstrate the impact of newly added object on caustics.*



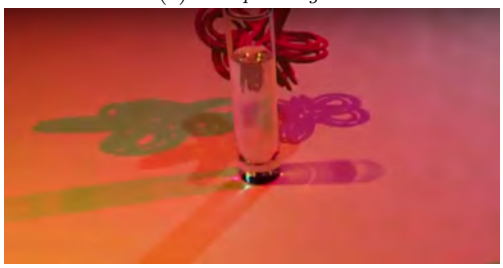
(a) *Original.*

(b) *Compositing 1.*



(c) *Compositing 2.*

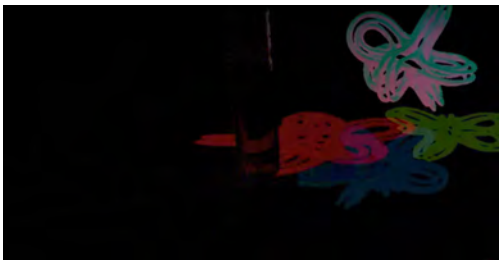
(d) *Compositing 3.*



(e) *Compositing 4.*

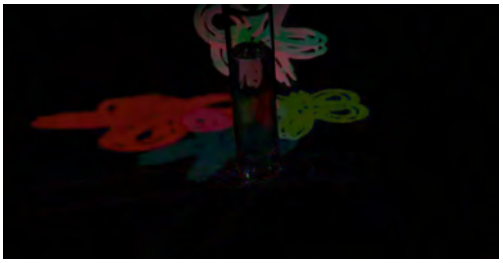
(f) *Compositing 5*

Figure 18.7: *More examples of frames from a compositing animation with caustics by Carl Van Huyck*

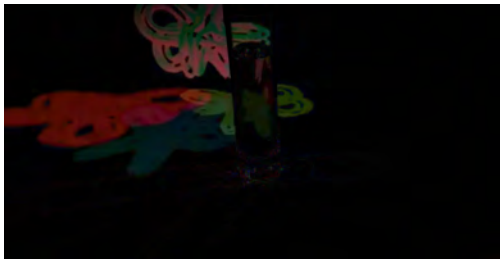


(a) *Compositing 1 - Original.*

(b) *Compositing 2 - Original.*



(c) *Compositing 4 - Original*



(d) *Compositing 5 - Original*

Figure 18.8: *Difference images that demonstrate the impact of newly added object on caustics.*

CHAPTER 19

Conclusion and Future Work

Conclusion In this book , I provide a theoretical framework for artificial to real compositing. In the book, I focused on only diffuse objects as newly added objects. However, newly added objects can be transparent and emit lights. My students also included such objects and this framework can be used to include such non-diffuse objects with some minor changes in overall framework.

Future Work: The theoretical framework presented in this book can be implemented to include global illumination into both artificial into real and artificial into artificial compositing. Both can be useful to speed up movie making processes to improve speed of rendering. We have developed bits and pieces of such a system as research projects. Development of such commercial products can provide effective human computer interfaces since this framework only requires 2D interaction.

Bibliography

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

[2] Tevfik Akgün, Irfan Kaya, Ahmet Koman, and Ergun Akleman. Spiral developable sculptures of ilhan koman. In *Bridges Donostia: Mathematics, Music, Art, Architecture, Culture*, pages 47–52. Southwestern College, 2007.

[3] Tevfik Akgün, Ahmet Koman, and Ergun Akleman. Developable sculptural forms of ilhan koman. In *Bridges London: Mathematics, Music, Art, Architecture, Culture*, pages 343–350. Southwestern College, 2006.

[4] Tevfik Akgün, Ahmet Koman, and Ergun Akleman. Paper sculptures with vertex deflection. In *Bridges London: Mathematics, Music, Art, Architecture, Culture*, pages 639–640. Southwestern College, 2006.

[5] E. Akleman. Twirling sculptures. *Journal of Mathematics and Arts*, 3:1–10, March 2009.

[6] E. Akleman. Architecture and sculpting. Website: http://www-viz.tamu.edu/faculty/ergun/research/architecture/, 2011.

[7] E. Akleman. Artistic depiction, non-photo realistic rendering and storytelling. Website: http://www-viz.tamu.edu/faculty/ergun/research/artisticdepiction/, 2011.

[8] E. Akleman. Implicit surface modeling. Website: http://www-viz.tamu.edu/faculty/-ergun/research/implicitmodeling/, 2011.

[9] E. Akleman. Rendering and visualization. Website: http://www-viz.tamu.edu/faculty/ergun/research/visualization/, 2011.

[10] E. Akleman. Topological modeling. Website: http://www-viz.tamu.edu/faculty/-ergun/research/topology/, 2011.

[11] E. Akleman. Research. Website: http://www.viz.tamu.edu/faculty/ergun/research, 2014.

[12] E. Akleman. Teaching. Website: http://www.viz.tamu.edu/faculty/-ergun/teaching/courses.html, 2014.

[13] E. Akleman. Visual storytelling course website. Website: http://www.viz.tamu.edu/courses/viza641, 2014.

[14] E. Akleman and J. Chen. Guaranteeing the 2-manifold property for meshes with doubly linked face list. *International Journal of Shape Modeling*, 5(2):149–177, 1999.

[15] E. Akleman and J. Chen. Insight for practical polygonal mesh modeling with discrete gauss-bonnet theorem. In *Proceedings of Geometry Modeling and Processing (GMP'06)*, pages 287–298. Springer Berlin Heidelberg, 2006.

[16] E. Akleman, J. Chen, Y. L. Chen, and Q. Xing. Cyclic twill-woven objects. *Computers & Graphics: Special Issue on Shape Modeling International (SMI) Conference 2011*, 35:623–631, June 2011.

[17] E. Akleman, J. Chen, F. Eryoldas, and V. Srinivasan. Handle and hole improvement by using new corner cutting subdivision scheme with tension. In *Proceedings International Conference on Shape Modeling and Applications (SMI'01)*, pages 32–49. IEEE, 2001.

[18] E. Akleman, J. Chen, and J. Gross. Paper strip sculptures. In *Proceedings of Shape Modeling International'2010*, pages 236–240. IEEE, June 2010.

[19] E Akleman, J. Chen, J. Gross, and Q. Xing. Extended graph rotation systems as a model for cyclic weaving on orientable surfaces. Technical Report - Computer Science Department, Texas A&M University - TR09-203, 2009.

[20] E. Akleman, J. Chen, and V. Srinivasan. A minimal and complete set of operators for the development of robust manifold mesh modelers. *Graphical Models Journal, Special issue on International Conference on Shape Modeling and Applications 2002*, 65(5):286–304, 2003.

[21] E. Akleman, J. Chen, Q. Xing, and J. Gross. Cyclic plain-weaving with extended graph rotation systems. *ACM Transactions on Graphics; Proceedings of SIGGRAPH'2009*, pages 78.1–78.8, August 2009.

[22] E. Akleman, J. Chen, Q. Xing, and J. Gross. Single cyclic plain-woven objects. *Proceedings of Shape Modeling International'2010*, pages 90 – 99, June 2010.

[23] E. Akleman, J. Chen, Q. Xing, and J. Gross. Twill-woven objects. In *Siggraph 2010, Late Breaking Talk and Poster, ACM Digital Library*, July 2010.

[24] E. Akleman, A. Kaur, and L. Green. Tiled texture mapping: What if miro painted a sphere. In *Proceedings of ISAMA: http://www-viz.tamu.edu/faculty/ergun-/research/textures/tiledtextures/tiledtextures.pdf*, 2008.

[25] E. Akleman and V. Srinivasan. Honeycomb subdivision. In *Proceedings of ISCIS'02, 17th International Symposium on Computer and Information Sciences*, volume 17, pages 137–141, November 2002.

[26] E. Akleman, V. Srinivasan, J. Chen, D. Morris, and S. Tett. Topmod3d: an interactive topological mesh modeler. In *Proceedings Computer Graphics International 2008, (CGI'08)*, pages 10–18, 2008.

[27] E. Akleman, V. Srinivasan, Z. Melek, and P. Edmundson. Semi-regular pentagonal subdivision. In *Proceedings of Shape Modeling International*, pages 110–118, 2004.

[28] E. Akleman, Vinod Srinivasan, Jianer Chen, Dave Morris, Stuart Tett, Eric Landreneau, Esan Mandal, Zeki Melek, and Other Students. Topmod3d software. TopMod3D software: http://www.viz.tamu.edu/faculty/ergun/research/topology/download.html, 2007.

[29] Ergun Akleman. *Pseudo-affine functions: a non-polynomial implicit function family to describe curves and sufaces*. PhD thesis, Georgia Institute of Technology, 1992.

[30] Ergun Akleman. Generalized morphing operations over implicitly represented shapes. In *Proceedings First Bilkent Computer Graphics Conference on Advanced Techniques in Animation, Rendering, and Visualization*, pages 143–149, 1993.

[31] Ergun Akleman. Interactive construction of smoothly blended star solids. In *Proceedings of Graphics Interface*, pages 159–167. Canadian Information Processing Society, 1996.

[32] Ergun Akleman. Ray-quadrics. In *Proceedings of Implicit Surfaces'96*, pages 89–98, 1996.

[33] Ergun Akleman. Making caricatures with morphing. In *ACM SIGGRAPH 97 Visual Proceedings: The art and interdisciplinary programs of SIGGRAPH'97*, page 145. ACM, 1997.

[34] Ergun Akleman. Implicit painting of csg solids. In *Proceedings of CSG*, volume 98, pages 99–113, 1998.

[35] Ergun Akleman. Implicit surface painting. In *Proceedings of Implicit Surfaces'1998*, volume 2, 1998.

[36] Ergun Akleman. Interactive computation of ray-quadric surfaces. In *Proceedings of Implicit Surfaces' 1998*, pages 105–114, 1998.

[37] Ergun Akleman. Computer aided sculpting: a multidisciplinary course in shape modeling. In *ACM SIGGRAPH 2006 Educators program*, page 11. ACM, 2006.

[38] Ergun Akleman. Designing symmetric high-genus sculptures. In *ACM SIGGRAPH 2006 Sketches*, page 73. ACM, 2006.

[39] Ergun Akleman. Symmetric sculptures. In *ACM SIGGRAPH 2006 Art gallery*, page 22. ACM, 2006.

[40] Ergun Akleman. Extended graph rotation systems and its applications to modeling 2-manifolds, woven surfaces and 3-manifolds. GD/SPM'2013, Presentation at the Minisymposium: Shaping Surfaces, 2013.

[41] Ergun Akleman and Jianer Chen. Progressive refinement with topological simplification: A theoretical framework. Technical Report.

[42] Ergun Akleman and Jianer Chen. Constant time updateable operations for implicit shape modeling. In *Proceedings of Implicit Surfaces'99*, pages 73–80, 1999.

[43] Ergun Akleman and Jianer Chen. Coupled modeling of solid textures and implicit shapes. In *Proceedings of Implicit Surfaces*, pages 89–95, 1999.

[44] Ergun Akleman and Jianer Chen. Generalized distance functions. In *Shape Modeling and Applications, 1999. Proceedings. Shape Modeling International'99. International Conference on*, pages 72–79. IEEE, 1999.

[45] Ergun Akleman and Jianer Chen. Regular meshes. In *Proceedings of the 2005 ACM symposium on Solid and physical modeling*, pages 213–219. ACM, 2005.

[46] Ergun Akleman and Jianer Chen. Regular mesh construction algorithms using regular handles. In *Proceedings of IEEE Int. Conference on Shape Modeling and Applications, (SMI 06)*, pages 171–181, 2006.

[47] Ergun Akleman, Jianer Chen, and Jonathan L. Gross. Block meshes: Topologically robust shape modeling with graphs embedded on 3-manifolds. *Computers & Graphics*, 46(0):306 – 326, 2015.

[48] Ergun Akleman, Jianer Chen, Shiyu Hu, and Jonathan L Gross. A topologically complete theory of weaving. Technical report, Technical Report-Computer Science Department, Texas A&M University-TR09-203, 2010.

[49] Ergun Akleman, Jianer Chen, and Burak Meric. Intuitive and effective design of periodic symmetric tiles. *Proceedings of ACM Multimedia*, 21(4):100–108, 2000.

[50] Ergun Akleman, Jianer Chen, and Burak Meric. Web-based intuitive and effective design of symmetric tiles. In *Proceedings of the 2000 ACM workshops on Multimedia*, pages 1–4. ACM, 2000.

[51] Ergun Akleman, Jianer Chen, and Vinod Srinivasan. A new paradigm for changing topology during subdivision modeling. In *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, pages 192–201. IEEE, 2000.

[52] Ergun Akleman, Jianer Chen, and Vinod Srinivasan. An interactive shape modeling system for robust design of functional 3d shapes. *Proceedings of ACADIA*, pages 248–257, 2001.

[53] Ergun Akleman, Jianer Chen, and Vinod Srinivasan. An interactive system for robust topological modeling of meshes. *Visual Proceedings of ACM SIGGRAPH'2001*, 2001.

[54] Ergun Akleman, Jianer Chen, and Vinod Srinivasan. A prototype system for robust, interactive and user-friendly modeling of orientable 2-manifold meshes. In *Shape Modeling International, 2002. Proceedings*, pages 43–50. IEEE, 2002.

[55] Ergun Akleman, Jianer Chen, Vinod Srinivasan, and Fusun Eryoldas. A new corner cutting scheme with tension and handle-face reconstruction. *International Journal of Shape Modeling*, 7(2):111–128, 2001.

[56] Ergun Akleman, Paul Edmundson, and Ozan Ozener. Smooth & fractal polyhedra. In *Bridges: Mathematical Connections in Art, Music, and Science*. Bridges, Winfield, Kansas, August 2004, 2004.

[57] Ergun Akleman, Paul Edmundson, and Ozan Ozener. A vertex truncation scheme to construct intriguing polyhedral shapes. In *Proceedings of Bridges: Mathematical Connections in Art, Music, and Science*, pages 117–124, 2004.

[58] Ergun Akleman, Jeff S Haberl, Frederic I Parke, Sajan Skaria, H John Halstead, and Malcolm Andrews. 4613 demonstration of multimedia electronic information enhancements for a handbook chapter cd-rom: Overview (rp-1017). *ASHRAE Transactions-American Society of Heating Refrigerating Airconditioning Engin*, 109(1):143–150, 2003.

[59] Ergun Akleman and L Hodges. Construction of convex and star shapes with yontsal functions. In *International Symposium on Computer and Information Sciences*, 1993.

[60] Ergun Akleman, Larry F Hodges, and Russell M Mersereau. Ray-affine functions: A general dual form to describe curves, surfaces, and volumes. Technical report, Georgia Institute of Technology, 1992.

[61] Ergun Akleman, S Liu, and Donald House. Barycentric shaders: Art directed shading using control images. In *Proceedings of Expressive 2016*. The Eurographics Association, 2016.

[62] Ergun Akleman, Siran Liu, and Donald House. Art directed rendering & shading using control images. In *ACM SIGGRAPH 2015 Posters*, page 43. ACM, 2015.

[63] Ergun Akleman, Zeki Melek, and Jeff S Haberl. Function-based flow modeling and animation. *The Journal of Visualization and Computer Animation*, 12(4):181–189, 2001.

[64] Ergun Akleman, James Palmer, and Ryan Logan. Making extreme caricatures with a new interactive 2d deformation technique with simplicial complexes. In *Proceedings of Visual'2000*, pages 165–170, 2000.

[65] Ergun Akleman, Fermi Perumal, and Youyou Wang. Cos θ shadows: an integrated model for direct illumination, subsurface scattering and shadow computation. In *ACM SIGGRAPH 2017 Posters*, page 38. ACM, 2017.

[66] Ergun Akleman and Jon Reisch. Modeling expressive 3d caricatures. In *ACM SIGGRAPH 2004 Sketches*, page 61. ACM, 2004.

[67] Ergun Akleman, Sajan Skaria, and Jeff S Haberl. Function based 2d flow animation. *Proceedings of Visual 2000*, pages 149–153, 2000.

[68] Ergun Akleman, Vinod Srinivasan, and Jianer Chen. Interactive rind modeling. In *Shape Modeling International, 2003*, pages 23–32. IEEE, 2003.

[69] Ergun Akleman, Vinod Srinivasan, et al. Topological mesh modeling. In *Bridges London: Mathematics, Music, Art, Architecture, Culture*, pages 661–662. Southwestern College, 2006.

[70] Ergun Akleman, Vinod Srinivasan, et al. Aesthetic beauty of rotation. In *Bridges Leeuwarden: Mathematics, Music, Art, Architecture, Culture*, pages 477–478. Southwestern College, 2008.

[71] Ergun Akleman, Vinod Srinivasan, and Esan Mandal. Remeshing schemes for semi-regular tilings. In *Proceedings of IEEE Int. Conference on Shape Modeling and Applications*, pages 44–50. IEEE, 2005.

[72] Ergun Akleman, Qing Xing, Pradeep Garigipati, Gabriel Taubin, Jianer Chen, and Shiyu Hu. Hamiltonian cycle art: Surface covering wire sculptures and duotone surfaces. *Computers & Graphics*, 37(5):316–332, 2013.

[73] Ergun Akleman and Cem Yuksel. On a family of symmetric, connected and high genus sculptures. In *Bridges London: Mathematics, Music, Art, Architecture, Culture*, pages 145–150. Southwestern College, 2006.

[74] Naif Alajlan, Mohamed Kamel, and Ed Jernigan. Detail preserving impulsive noise removal. *Signal Processing: Image Communication*, 19(10):993–1003, 2004.

[75] Josef Albers. *Interaction of color: unabridged text and selected plates*. Yale University Press, 1975.

[76] Josef Albers. *Interaction of color*. Yale University Press: Reprinted from 1965 original, 2006.

[77] Anthony A Apodaca, Larry Gritz, and Ronen Barzel. *Advanced RenderMan: Creating CGI for motion pictures*. Morgan Kaufmann, 2000.

[78] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. In *ACM Transactions on graphics (TOG)*, volume 26/3, pages 10–18. ACM, 2007.

[79] F O Bartell, EL Dereniak, and WL Wolfe. The theory and measurement of bidirectional reflectance distribution function (brdf) and bidirectional transmittance distribution function (btdf). In *1980 Huntsville Technical Symposium*, pages 154–160. International Society for Optics and Photonics, 1981.

[80] John C Beatty and Brian A Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1987.

[81] X. Bei, Yuksel C., Abur A., and E. Akleman. 3-d visualization of power system state estimation. *Proceedings of Melecon 2006, Benalmadena,Malaga, Spain*, May 2006.

[82] X. Bei, C. Y. Evrenosoglu, Abur A., and E. Akleman. Interactive evaluation of atc using a graphical user interface. In *Proceedings of IEEE Power Engineering Society General Meeting, Denver, CO*, pages 84–87, June 2004.

[83] Nancy Beiman. *Prepare to Board!: Creating Story and Characters for Animated Features and Shorts*. Taylor & Francis, 2012.

[84] Marcelo Bertalmio, Andrea L Bertozzi, and Guillermo Sapiro. Navier-stokes, fluid dynamics, and image and video inpainting. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–355. IEEE, 2001.

[85] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.

[86] Marcelo Bertalmio, Luminita Vese, Guillermo Sapiro, and Stanley Osher. Simultaneous structure and texture image inpainting. *Image Processing, IEEE Transactions on*, 12(8):882–889, 2003.

[87] H. Bezerra, B. Feijo, and L. Velho. An image-based shading pipeline for 2d animation. In *Computer Graphics and Image Processing, 2005. SIBGRAPI 2005. 18th Brazilian Symposium on*, pages 307–314, 2005.

[88] James F Blinn. Compositing. 1. theory. *Computer Graphics and Applications, IEEE*, 14(5):83–87, 1994.

[89] James F Blinn. Composting, part 2: practice. *Computer Graphics and Applications, IEEE*, 14(6):78–82, 1994.

[90] Bruce Block. *The visual story: creating the visual structure of film, TV and digital media*. CRC Press, 2014.

[91] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[92] Ron Brinkmann. *The art and science of digital compositing: techniques for visual effects, animation and motion graphics*. Morgan Kaufmann, 2008.

[93] Pradeep Buddharaju, Yuichi Fujiki, Ioannis Pavlidis, and Ergun Akleman. A novel way to conduct human studies and do some good. In *Proceedings of CHI'10 - Human Factors in Computing Systems*, pages 4699–4702. ACM, 2010.

[94] Pradeep Buddharaju, Yuichi Fujiki, Ioannis Pavlidis, and Ergun Akleman. A novel way to conduct human studies and do some good. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, pages 4699–4702. ACM, 2010.

[95] Brian Cabral and Leith Casey Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '93, pages 263–270, 1993.

[96] Katherine Calkins. Simulating thin films. Master's thesis, Texas A&M University, College Station, Texas, 1999.

[97] Saif Castaneda and Ergun Akleman. Shading with painterly filtered layers: a technique to obtain painterly portrait animations. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering*, page 8, New York City, NW, 2017. ACM.

[98] Ching (Clara) Chan. Two computational approaches for creating chinese painting. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/ETD-TAMU-2000-THESIS-C379, College Station, Texas, 2000.

[99] Ching (Clara) Chan, Ergun Akleman, and Jianer Chen. Two methods for creating chinese painting. In *Pacific Conference on Computer Graphics and Applications*, pages 403–412, 2002.

[100] Ching (Clara) Chan, Ergun Akleman, and Jianer Chen. Two methods for creating chinese painting. In *Pacific Conference on Computer Graphics and Applications*, pages 403–412, 2002.

[101] Frank Chance, John Truong, and E. Akleman. Modeling articulated hands. In *Proceedings of ISAMA 2008*, 2008.

[102] Gordon Chancellor and John Van Wyhe. Charles darwins notebooks from the voyage of the beagle. *Charles Darwin's Note-books from the Voyage of the Beagle*, 2009.

[103] Jianer Chen and Ergun Akleman. Topologically robust mesh modeling: concepts, data structures and operations. *Manuscript available at http://wwwviz. tamu. edu/faculty/ergun/research/topology/tr02a. ps. gz*, 2002.

[104] Y. L. Chen, E. Akleman, J. Chen, and Q. Xing. Biaxial textile weaving structure. In *Proceedings of ISAMA 2010 Art and Math Conference*, June 2010.

[105] Yen-Lin Chen, Ergun Akleman, Jianer Chen, and Qing Xing. Designing biaxial textile weaving patterns. In *Proceedings of ISAMA 2010; Art and Math Conference*, June 2010.

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

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

[108] Jonathan Cohen, Marc Olano, and Dinesh Manocha. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 115–122, 1998.

[109] Forrester Cole, Kevin Sanik, Doug DeCarlo, Adam Finkelstein, Thomas Funkhouser, Szymon Rusinkiewicz, and Manish Singh. How well do line drawings depict shape? In *ACM SIGGRAPH 2009 papers*, SIGGRAPH '09, pages 28:1–28:9, 2009.

[110] Robert L Cook. Shade trees. *ACM Siggraph Computer Graphics*, 18(3):223–231, 1984.

[111] Robert L Cook, Loren Carpenter, and Edwin Catmull. The reyes image rendering architecture. *ACM SIGGRAPH Computer Graphics*, 21(4):95–102, 1987.

[112] Robert L Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. *ACM SIGGRAPH Computer Graphics*, 18(3):137–145, 1984.

[113] Gábor Czédli and Anna B Romanowska. An algebraic closure for barycentric algebras and convex sets. *Algebra universalis*, 68(1-2):111–143, 2012.

[114] Richard Davison. *Real Drawing: Concepts, Constructions, Capriccios*. Blurb Publisihing, 2007.

[115] Paul Debevec. Image-based lighting. *IEEE Computer Graphics and Applications*, 22(2):26–34, 2002.

[116] Paul Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *ACM SIGGRAPH 2008 classes*, page 32. ACM, 2008.

[117] Paul Debevec. Paul debevec homepage. website: http://www.pauldebevec.com/, 2018.

[118] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 classes*, page 31. ACM, 2008.

[119] Paul E Debevec and Jitendra Malik. Recovering high dynamic range radiance maps from photographs. In *ACM SIGGRAPH 2008 classes*, page 31. ACM, 2008.

[120] Paul E Debevec, Camillo J Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 11–20. ACM, 1996.

[121] Julie Dorsey, Alan Edelman, Henrik Wann Jensen, Justin Legakis, and Hans Kohling Pedersen. Modeling and rendering of weathered stone. In *ACM SIGGRAPH 2006 Courses*, page 4. ACM, 2006.

[122] Robert A Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Siggraph Computer Graphics*, 22(4):65–74, 1988.

[123] Yuxiao Du and Ergun Akleman. Charcoal rendering and shading with reflections. In *ACM SIGGRAPH 2016 Posters*, page 32. ACM, 2016.

[124] Yuxiao Du and Ergun Akleman. Designing look-and-feel using generalized crosshatching. In *ACM SIGGRAPH 2017 Talks*, page 40. ACM, 2017.

[125] Tom Duff. Compositing 3-d rendered images. *ACM SIGGRAPH Computer Graphics*, 19(3):41–44, 1985.

[126] Tom Duff. Deep compositing using lie algebras. *ACM Transactions on Graphics (TOG)*, 36(3):26, 2017.

[127] Nelson Dunford, Jacob T Schwartz, William G Bade, and Robert G Bartle. *Linear operators*. Wiley-interscience New York, 1971.

[128] David S Ebert. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003.

[129] Alexei A Efros and William T Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 341–346. ACM, 2001.

[130] Elmar Eisemann, Sylvain Paris, and Frédo Durand. A visibility algorithm for converting 3d meshes into editable 2d vector graphics. *ACM Trans. Graph.*, 28(3):83:1–83:8, July 2009.

[131] Sarah Eisinger and Ergun Akleman. Teaching art and science of rendering and shading. In *ACM SIGGRAPH 2012 Dailies*, page 3. ACM, 2012.

[132] Sarah Eisinger and Ergun Akleman. Teaching art and science of rendering and shading. In *ACM SIGGRAPH 2012 Dailies*, page 3. ACM, 2012.

[133] Gershon Elber. Smi 2011: Full paper: Modeling (seemingly) impossible models. *Comput. Graph.*, 35(3):632–638, June 2011.

[134] Eric Enderton. Computer graphics for jurassic park. In *Compcon Spring'94, Digest of Papers.*, pages 456–457. IEEE, 1994.

[135] Gabriel Esquivel, Qing Xing, and E. Akleman. Twisted developables. *Proceedings of ISAMA 2011*, pages 100–108, June 2011.

[136] Gabriel Esquivel, Qing Xing, and Ergun Akleman. Twisted developables. *Proceedings of ISAMA 2011*, pages 145–153, 2011.

[137] Gabriel Esquivel, Qing Xing, Ryan Collier, Michael Tomaso, and Ergun Akleman. Weaving methods in architectural design. *Proceedings of Bridges 2011, (Connections between mathematics, art and science), Combria, Portugal*, pages 100–108, July 2011.

[138] C. Y. Evrenosoglu, Abur A., and E. Akleman. Three-dimensional visualization and animation of travelling waves in power systems. *Electrical Power System Research*, 77:876–883, May 2007.

[139] C. Y. Evrenosoglu, E. Akleman, and Abur A. Visualization of electromagnetic transients along transmission lines. In *Proceedings of IEEE Power Systems Conference and Exposition 2004, New York*, October 2004.

[140] C. Y. Evrenosoglu, E. Akleman, Abur A., and O. Ozener. Two methods for traveling wave visualization in multi-phase power systems: Painted transients and animated pipes. *Proceedings of International Power System Transients Conference Montreal, Canada*, June 2005.

[141] Cansın Y Evrenosoglu, Ali Abur, Ergun Akleman, and OO Ozener. Bewley diagrams revisited via visualization. *IEEE Transactions on Power Systems*, 24(3):1401–1407, 2009.

[142] Zeev Farbman, Gil Hoffer, Yaron Lipman, Daniel Cohen-Or, and Dani Lischinski. Coordinates for instant image cloning. *ACM Trans. Graph.*, 28(3):67:1–67:9, 2009.

[143] Raanan Fattal, Dani Lischinski, and Michael Werman. Gradient domain high dynamic range compression. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '02, pages 249–256, 2002.

[144] Mark Finch, John Snyder, and Hugues Hoppe. Freeform vector graphics with controlled thin-plate splines. *ACM Transactions on Graphics (TOG)*, 30(6):166, 2011.

[145] Graham D Finlayson, Steven D Hordley, and Mark S Drew. Removing shadows from images. In *Computer VisionECCV 2002*, pages 823–836. Springer, 2002.

[146] Michael S Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[147] James D Foley, Andries Van Dam, Steven K Feiner, John F Hughes, and Richard L Phillips. *Introduction to computer graphics*, volume 55. Addison-Wesley Reading, 1994.

[148] Kenneth D Forbus. Qualitative process theory. *Artificial intelligence*, 24(1):85–168, 1984.

[149] John Frazer. *Artificially arranged scenes: The films of Georges Méliès*. GK Hall, 1979.

[150] Pradeep Garigipati and Ergun Akleman. Duotone surfaces. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 99–106. Eurographics Association, 2012.

[151] Iliyan Georgiev, Jaroslav Krivánek, Tomas Davidovic, and Philipp Slusallek. Light transport simulation with vertex connection and merging. *ACM Trans. Graph.*, 31(6):192, 2012.

[152] Todor Georgiev. Photoshop healing brush: a tool for seamless cloning. In *Workshop on Applications of Computer Vision (ECCV 2004)*, pages 1–8. Citeseer, 2004.

[153] William Gibson. *Virtual light*. Spectra, 1994.

[154] Andrew S Glassner. *An introduction to ray tracing*. Elsevier, 1989.

[155] Albert Gleizes and Jean Metzinger. *Du" cubisme"*. RG Fischer, 1947.

[156] Dan B Goldman, Brian Curless, Aaron Hertzmann, and Steven M Seitz. Shape and spatially-varying brdfs from photometric stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1060–1071, 2010.

[157] Ozgur Gonen. *Quad Dominant 2-Manifold Mesh Modeling*. PhD thesis, Texas A&M University: Available Electronically From: http://oaktrust.library.tamu.edu/handle/1969.1/161442, College Station, Texas, 2017.

[158] Ozgur Gonen and Ergun Akleman. Sketch based 3d modeling with curvature classification. *Computers & Graphics 2012*, 36(5):521–525, 2012.

[159] Ozgur Gonen and Ergun Akleman. Sketching knots. In *ACM SIGGRAPH 2012 Posters*, SIGGRAPH '12, pages 24:1–24:1, 2012.

[160] Özgür Gönen, Ergun Akleman, and Vinod Srinivasan. Modeling d-forms. In *Bridges Donostia: Mathematics, Music, Art, Architecture, Culture*, pages 209–216. Southwestern College, 2007.

[161] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '98, pages 447–452, 1998.

[162] Wesley Griffin, Yu Wang, David Berrios, and Marc Olano. Gpu curvature estimation on deformable meshes. In *Symposium on Interactive 3D Graphics and Games*, I3D '11, pages 159–166, 2011.

[163] Lucye Guilbeau. The history of the solution of the cubic equation. *Mathematics News Letter*, pages 8–12, 1930.

[164] Mark Clayton Guillermo Vasquez de Velasco, Ergun Akleman. Living in a magazine: alternatives in the use of shared virtual reality. In *Conference of the European Association for Architechural Education, Las Palmas de Cran, Spain, 1997.*, 1997.

[165] Jeff Haberl and Ergun Akleman. Demonstration of the use of multimedia electronic information enhancements for a chapter handbook cd-rom: 3d modeling and animation. *ASHRAE Transactions*, 116(1), 2010.

[166] Toshiya Hachisuka, Wojciech Jarosz, Guillaume Bouchard, Per Christensen, Jeppe Revall Frisvad, Wenzel Jakob, Henrik Wann Jensen, Michael Kaschalk, Claude Knaus, Andrew Selle, et al. State of the art in photon density estimation. In *Acm Siggraph 2012 Courses*, page 6. ACM, 2012.

[167] Toshiya Hachisuka, Jacopo Pantaleoni, and Henrik Wann Jensen. A path space extension for robust light transport simulation. *ACM Transactions on Graphics (TOG)*, 31(6):191, 2012.

[168] Pat Hanrahan and Jim Lawson. A language for shading and lighting calculations. *ACM SIGGRAPH Computer Graphics*, 24(4):289–298, 1990.

[169] John Hart. Private conversation. *According to a market research firm 3D Graphics is only 8% of the whole graphics market. 2D graphics such as vector, image and video is 90% of the graphics market.*, 2008.

[170] Michiel Hazewinkel. *Lebesgue space in Encyclopaedia of Mathematics*. Springer Science & Business Media, 2013.

[171] Radu Horaud, Miles Hansard, Georgios Evangelidis, and Clément Ménier. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine vision and applications*, 27(7):1005–1020, 2016.

[172] Donald H House and Mayank Singh. Line drawing as a dynamic process. In *Pacific Conference on Computer Graphics and Applications*, pages 351–360, 2007.

[173] Shiyu Hu, Qing Xing, Ergun Akleman, Jianer Chen, and Jonathan Gross. Pattern mapping with quad-pattern-coverable quad-meshes. *Computers & Graphics*, 36(5):455–465, 2012.

[174] John F Hughes, Andries Van Dam, James D Foley, Morgan McGuire, Steven K Feiner, David F Sklar, and Kurt Akeley. *Computer graphics: principles and practice*. Pearson Education, 2014.

[175] T. Igarashi, S. Matsuoka, and H. Tanaka. A sketching interface for 3d freeform design. In *Proceedings of ACM SIGGRAPH'99*, pages 409–416, 1999.

[176] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, et al. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth

camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM, 2011.

[177] Scott F. Johnston. Lumo: illumination for cel animation. In *Proceedings of the 2nd international symposium on Non-photorealistic animation and rendering*, NPAR '02, pages 45–52, 2002.

[178] Matthew Justice and Ergun Akleman. A process to create dynamic landscape paintings using barycentric shading with control paintings. In *ACM SIGGRAPH 2018 Posters*, page 30, New York City, NW, 2018. ACM.

[179] James T Kajiya. The rendering equation. In *ACM Siggraph Computer Graphics*, volume 20/4, pages 143–150. ACM, 1986.

[180] Preyas Kamath, Ergun Akleman, and Andrew K Chan. Compression and accelerated rendering of volume data using wavelet transform. In *Proceedings SPIE - International Symposium on Optical Science, Engineering, and Instrumentation: 30 "Mathematical Modeling and Estimation Techniques in Computer Vision"*, volume 3457, pages 296–303. International Society for Optics and Photonics, 1998.

[181] Doug Kelly. *Digital Compositing in Depth: The Only Guide to Post Production for Visual Effects in Film with CD-ROM.* Coriolis Group Books, 2000.

[182] Johannes Kepler. *Mysterium cosmographicum.* Tbingen, 1596.

[183] Mladen Kezunovic, Ergun Akleman, Maja Knezev, Ozgur Gonen, and Satish Natti. Optimized fault location. In *Bulk Power System Dynamics and Control-VII. Revitalizing Operational Reliability, 2007 iREP Symposium*, pages 1–7. IEEE, 2007.

[184] Jonathan Kiker. Using local information for compositing cg into traditional art. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/ETD -TAMU -2009 -05 -325, College Station, Texas, 2009.

[185] Piotr Krawczyk. Futurist sculpting: modeling movement in 3d. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/5014, College Station, Texas, 2006.

[186] Yu-Kun Lai, Shi-Min Hu, and Ralph R Martin. Automatic and topology-preserving gradient mesh generation for image vectorization. In *ACM Transactions on Graphics (TOG)*, volume 28/3, page 85. ACM, 2009.

[187] Eric Landreneau, Ergun Akleman, and John Keyser. Interactive face-replacements for modeling detailed shapes. In *Geometric Modeling and Processing-GMP 2006*, pages 602–608. Springer Berlin Heidelberg, 2006.

[188] Eric Landreneau, Ergun Akleman, and Vinod Srinivasan. Local mesh operators: Extrusions revisited. In *Proceedings of IEEE Int. Conference on Shape Modeling and Applications*, pages 349–354. IEEE, 2005.

[189] Eric Landreneau, Ozan O Ozener, Burak Pak, Ergun Akleman, and John Keyser. Interactive rule-based design. In *Innovations in Design & Decision Support Systems in Architecture and Urban Planning*, pages 433–445. Springer Netherlands, 2006.

[190] Lee Lanier. *Professional digital compositing: essential tools and techniques.* John Wiley & Sons, 2009.

[191] Jaemin Lee. Practical image based lighting. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/ETD-TAMU-2003-THESIS-L436, College Station, Texas, 2003.

[192] Jaemin Lee and Ergun Akleman. A new image based ligthing method: Practical shadow-based light reconstruction. In *Proceedings of 17th International Symposium on Computer and Information Sciences, (ISCIS'02)*, pages 142–146. CRC Press, 2002.

[193] Jaemin Lee and Ergun Akleman. Practical image based lighting. In *Proceedings of Annual Conference of Association of Computer Aided Design in Architecture (ACADIA 2002)*, pages 279–288, 2002.

[194] Jaemin Lee and Ergun Akleman. Practical image based lighting. In *Proc. Annual Conference of Association of Computer Aided Design in Architecture (ACADIA 2002)*, pages 279–288, 2002.

[195] Anat Levin, Dani Lischinski, and Yair Weiss. A closed-form solution to natural image matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(2):228–242, 2008.

[196] Anat Levin, Alex Rav-Acha, and Dani Lischinski. Spectral matting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(10):1699–1712, 2008.

[197] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In *Computer Vision-ECCV 2004*, pages 377–389. Springer, 2004.

[198] Anat Levin, Assaf Zomet, Shmuel Peleg, and Yair Weiss. Seamless image stitching in the gradient domain. In *Proceedings of ECCV, Springer-Verlag, vol. IV*, pages 377–389, 2006.

[199] Hongbo Li, David Hestenes, and Alyn Rockwood. Generalized homogeneous coordinates for computational geometry. In *Geometric Computing with Clifford Algebras*, pages 27–59. Springer, 2001.

[200] Yin Li, Jian Sun, Chi-Keung Tang, and Heung-Yeung Shum. Lazy snapping. In *ACM Transactions on Graphics (ToG)*, volume 23/3, pages 303–308. ACM, 2004.

[201] Erik Lindholm, Mark J Kilgard, and Henry Moreton. A user-programmable vertex engine. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 149–158. ACM, 2001.

[202] Siran Liu. Chinese ink-and-brush painting with film lighting aesthetics in 3d computer graphics. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/154993, College Station, Texas, 2015.

[203] Siran Liu and Ergun Akleman. Chinese ink and brush painting with reflections. In *SIGGRAPH 2015: Studio*, page 8. ACM, 2015.

[204] Yutu Liu, Ergun Akleman, and Jianer Chen. Visual storytelling warehouse. Technical Report.

[205] Yutu Liu, Hernan Molina, and Ergun Akleman. Inout sculptures. In *Bridges Donostia: Mathematics, Music, Art, Architecture, Culture*, pages 247–252. Southwestern College, 2007.

[206] R. Malak, E. Akleman, N. Amato, D. Lagoudas, D. Hartl, and D. Mc Adams. Texas a&m reprogrammable origami. Website: http://origami.tamu.edu/, 2014.

[207] Esan Mandal, Vinod Srinivasan, and Ergun Akleman. Column modeling. In *ACM SIGGRAPH 2004 Sketches*, page 59. ACM, 2004.

[208] William R Mark, R Steven Glanville, Kurt Akeley, and Mark J Kilgard. Cg: a system for programming graphics hardware in a c-like language. *ACM Transactions on Graphics (TOG)*, 22(3):896–907, 2003.

[209] Wojciech Matusik. *A data-driven reflectance model*. PhD thesis, Massachusetts Institute of Technology, 2003.

[210] James McCann and Nancy Pollard. Local layering. *ACM Trans. Graph.*, 28(3):84:1–84:7, 2009.

[211] Scott Meadows and Ergun Akleman. Creating abstract digital paintings with painting camera technique. In *Information Visualization, 2000. Proceedings. IEEE International Conference on*, pages 250–254. IEEE, 2000.

[212] Scott Meadows and Ergun Akleman. Creating abstract digital paintings with painting camera technique. In *Information Visualization, 2000. Proceedings. IEEE International Conference on*, pages 250–254. IEEE, 2000.

[213] Scott Harrison Meadows. The painting camera: an abstract rendering system with camera-control images. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/1969 .1/ETD-TAMU-2000-THESIS-M42, College Station, Texas, 2000.

[214] Gavin Miller. Efficient algorithms for local and global accessibility shading. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 319–326. ACM, 1994.

[215] Jace Miller and Ergun Akleman. Edge-based intersected polyhedral paper sculptures constructed by interlocking slitted planar pieces. In *Bridges Leeuwarden: Mathematics, Music, Art, Architecture, Culture*, pages 259–264. Southwestern College, 2008.

[216] Isamu Motoyoshi, Shin'ya Nishida, Lavanya Sharan, and Edward H Adelson. Image statistics and the perception of surface qualities. *Nature*, 447(7141):206–209, 2007.

[217] Gero Müller, Jan Meseth, Mirko Sattler, Ralf Sarlette, and Reinhard Klein. Acquisition, synthesis, and rendering of bidirectional texture functions. *Computer Graphics Forum*, 24(1):83–109, 2005.

[218] A. Nasri, W. Bou Karam, and F. Samavati. Sketch-based subdivision models. In *Proceedings of the 6th Eurographics Symposium on Sketch-Based Interfaces and Modeling*, SBIM '09, pages 53–60, 2009.

[219] Martin E Newell, Richard G Newell, and Tom L Sancha. A new approach to the shaded picture problem. In *Proceedings of the ACM National Conference*, volume 443, page 50, 1972.

[220] Makoto Okabe, Gang Zeng, Yasuyuki Matsushita, Takeo Igarashi, Long Quan, and Heung-Yeung Shum. Single-view relighting with normal map painting. *Proceedings of Pacific Graphics*, pages 27–34, 2006.

[221] Manuel M Oliveira and Maicon Brauwers. Real-time refraction through deformable objects. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*, pages 89–96. ACM, 2007.

[222] Robert Olson. *Art direction for film and video*. CRC Press, 1998.

[223] Alexandrina Orzan, Adrien Bousseau, Holger Winnemöller, Pascal Barla, Joëlle Thollot, and David Salesin. Diffusion curves: a vector representation for smooth-shaded images. *ACM Trans. Graph.*, 27(3):92:1–92:8, 2008.

[224] Ozan Önder Özener, Ergun Akleman, and Vinod Srinivasan. Interactive rind modeling for architectural design. *International Journal of Architectural Computing*, 3(1):93–106, 2005.

[225] James Dean Palmer, Ergun Akleman, and Jianer Chen. Interactive deformation with triangles. In *International Symposium on Computer and Information Sciences*, page 147. CRC Press, 2002.

[226] Kedar A Patwardhan, Guillermo Sapiro, and Marcelo Bertalmio. Video inpainting of occluding and occluded objects. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–69. IEEE, 2005.

[227] Edwin A Peraza-Hernandez, Darren J Hartl, Katherine R Frei, and Ergun Akleman. Connectivity of shape memory alloy-based self-folding structures. In *Proceedings of 22nd AIAA/ASME/AHS Adaptive Structures Conference*, January 2014.

[228] Edwin Alexander Peraza Hernandez, Shiyu Hu, Han Wei Kung, Darren Hartl, and Ergun Akleman. Towards building smart self-folding structures. *Computers & Graphics*, 37(6):730–742, 2013.

[229] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Transactions on Graphics (TOG)*, 22(3):313–318, 2003.

[230] Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

[231] Lena Petrović, Brian Fujito, Lance Williams, and Adam Finkelstein. Shadows for cel animation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '00, pages 511–516, 2000.

[232] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters. *International Journal of Computer Vision*, 32(1):7–25, 1999.

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

[234] Thomas Porter and Tom Duff. Compositing digital images. *ACM Siggraph Computer Graphics*, 18(3):253–259, 1984.

[235] Fermi Niveditha Pramananda Perumal. An integrated model for direct illumination, subsurface scattering, specular highlights and shadow computation. Master's thesis, Texas A&M University. Available electronically from http : / /hdl .handle .net /1969 .1 /165958., College Station, TX, 2017.

[236] Entertainment PriceWaterhouseCoopers. Media outlook 2013. *Europe. London: PWC, October*, 2013.

[237] Stephen Prince. *true lies-perceptual realism, digital images, and film theory*. University of California Press, 1996.

[238] Kekoa Proudfoot, William R Mark, Svetoslav Tzvetkov, and Pat Hanrahan. A real-time procedural shading system for programmable graphics hardware. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 159–170. ACM, 2001.

[239] Paul Rademacher. View-dependent geometry. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 439–446, 1999.

[240] Ravi Ramamoorthi, John Anderson, Mark Meyer, and Derek Nowrouzezahrai. A theory of monte carlo visibility sampling. *ACM Transactions on Graphics (TOG)*, 31(5):121, 2012.

[241] Erik Reinhard, Wolfgang Heidrich, Paul Debevec, Sumanta Pattanaik, Greg Ward, and Karol Myszkowski. *High dynamic range imaging: acquisition, display, and image-based lighting*. Morgan Kaufmann, 2010.

[242] Daniel Robbins. Jean metzinger: At the center of cubism. *Jean Metzinger in Retrospect*, 1985.

[243] A. B. Romanowska and J. Smith. On the structure of barycentric algebras. *Houston, Journal of Mathematics*, 16(3):431–448, 1990.

[244] Randi J Rost, Bill M Licea-Kane, Dan Ginsburg, John M Kessenich, Barthold Lichtenbelt, Hugh Malan, and Mike Weiblen. *OpenGL shading language*. Pearson Education, 2009.

[245] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. In *ACM transactions on graphics (TOG)*, volume 27/3, pages 16–24. ACM, 2008.

[246] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Multi-operator media retargeting. In *ACM Transactions on Graphics (TOG)*, volume 28/3, pages 23–31. ACM, 2009.

[247] Vidya Setlur, Saeko Takagi, Ramesh Raskar, Michael Gleicher, and Bruce Gooch. Automatic image retargeting. In *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, pages 59–68. ACM, 2005.

[248] Musawir A Shah, Jaakko Konttinen, and Sumanta Pattanaik. Caustics mapping: An image-space technique for real-time caustics. *IEEE Transactions on Visualization and Computer Graphics*, 13(2):272–280, 2007.

[249] Musawir A Shah, Jaakko Konttinen, and Sumanta Pattanaik. Image-space subsurface scattering for interactive rendering of deformable translucent objects. *IEEE computer graphics and applications*, 29(1), 2009.

[250] Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. Crossshade: shading concept sketches using cross-section curves. *ACM Trans. Graph.*, 31(4):45:1–45:11, 2012.

[251] Mayank Singh and Donald H House. Pure contour drawings using dynamic lines. Technical report, Computer Science Department, Texas A&M University, TR-2009-4-5, 2008.

[252] Sajan Skaria, Ergun Akleman, and Frederic I Parke. Modeling subdivision control meshes for creating cartoon faces. In *Shape Modeling and Applications, SMI 2001 International Conference on.*, pages 216–225. IEEE, 2001.

[253] Alvy Ray Smith. Alpha and the history of digital compositing. *URL: http://www. alvyray. com/Memos/7_alpha. pdf, zuletzt abgerufen am*, 24:2010, 1995.

[254] Alvy Ray Smith and James F Blinn. Blue screen matting. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 259–268. ACM, 1996.

[255] Jeffrey Smith, Ergun Akleman, Richard Davison, and John Keyser. Multicam: A system for interactive rendering of abstract digital images. In *Proceedings of Bridges: Mathematical Connections in Art, Music, and Science*, pages 265–272, 2004.

[256] Jeffrey Smith, Ergun Akleman, Richard Davison, John Keyser, et al. Multicam: A system for interactive rendering of abstract digital images. In *Bridges: Mathematical Connections in Art, Music, and Science*, pages 265–272. Bridges Conference, 2004.

[257] Jeffrey Statler Smith. Multi-camera: interactive rendering of abstract digital images. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/341, College Station, Texas, 2003.

[258] Vinod Srinivasan and Ergun Akleman. Connected & manifold sierpinsky polyhedra. In *Proceedings of the ninth ACM symposium on Solid modeling and applications*, pages 261–266. Eurographics Association, 2004.

[259] Vinod Srinivasan, Ergun Akleman, and Jianer Chen. Interactive construction of multi-segment curved handles. In *Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on*, pages 429–430. IEEE, 2002.

[260] Vinod Srinivasan, Ergun Akleman, and John Keyser. Topological construction of 2-manifold meshes from arbitrary polygonal data. *Tamu College Station, Texas, USA*, 2004.

[261] Vinod Srinivasan, Esan Mandal, and Ergun Akleman. Solidifying wireframes. In *Proc. Bridges: Mathematical Connections in Art, Music, and Science*, pages 203–210, 2005.

[262] Vinod Srinivasan, Hernan Molina, and Ergun Akleman. Modeling high genus sculptures using multi-connected handles and holes. In *Bridges Donostia: Mathematics, Music, Art, Architecture, Culture*, pages 297–302. Southwestern College, 2007.

[263] Michael Leighton Stanley. Digital compositing with traditional artwork. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/2643, College Station, Texas, 2005.

[264] Jian Sun, Jiaya Jia, Chi-Keung Tang, and Heung-Yeung Shum. Poisson matting. *ACM Trans. Graph.*, 23(3):315–321, 2004.

[265] Jian Sun, Lin Liang, Fang Wen, and Heung-Yeung Shum. Image vectorization using optimized gradient meshes. *ACM Transactions on Graphics (TOG)*, 26(3):11, 2007.

[266] Xin Sun, Kun Zhou, Eric Stollnitz, Jiaoying Shi, and Baining Guo. Interactive relighting of dynamic refractive objects. *ACM Transactions on Graphics (TOG)*, 27(3):35, 2008.

[267] Daniel Sỳkora, Ladislav Kavan, Martin Čadík, Ondřej Jamriška, Alec Jacobson, Brian Whited, Maryann Simmons, and Olga Sorkine-Hornung. Ink-and-ray: Bas-relief meshes for adding global illumination effects to hand-drawn characters. *ACM Transactions on Graphics (TOG)*, 33(2):16, 2014.

[268] David Tall. Building theories: The three worlds of mathematics. *For the Learning of Mathematics*, pages 29–32, 2004.

[269] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (ToG)*, 21(3):665–672, 2002.

[270] Roger Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[271] Shih-Ting Tsai and Ming-Hsiu Mia Chen. The thinking of animation: Conflict of 3d and 2d. In *Proceedings of The Asian Conference on Ethics, Religion & Philosophy 2013*, pages 15–21, 2013.

[272] Authors Unknown. Pixar's animation process, 2015.

[273] Steve Upstill, Judi Achjadi, Asmoro Damais, Nian S Jumena, KRHT Hardjonagoro, Firman Ichsan, Perry Pieters, Rony Siswandi, Tara Sosrowardoyo, Iwan Tirta, et al. *The RenderMan Companion: A Programmer's Guide to Realistic Computer Graphics*, volume 226. Addison-Wesley, 1990.

[274] John van Wyhe. The complete work of charles darwin online. *Notes and Records of the Royal Society*, 60(1):87–89, 2006.

[275] Guillermo Vasquez de Velasco and Ergun Akleman. Arquitectos de virtualidad-nuevo mercado, nueva curricula (architects of virtuality-new market, new curricula). In *Proceedings of SIGRaDi'1998*. Sigradi, 1998.

[276] Romain Vergne, Pascal Barla, Roland Fleming, and Xavier Granier. Surface flows for image-based shading design. *ACM Transactions on Graphics*, 31(3):94–1, 2012.

[277] Youyou Wang. *Qualitative Global Illumination of Mock-3D Scenes*. PhD thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1, College Station, Texas, 2014.

[278] Youyou Wang and Ergun Akleman. Calligraphic cutting: extreme image resizing with cuts in continuous domain. In *ACM SIGGRAPH 2012 Talks*, page 49. ACM, 2012.

[279] Youyou Wang and Ergun Akleman. Calligraphic cutting: extreme image resizing with cuts in continuous domain. In *ACM SIGGRAPH 2012 Talks*, SIGGRAPH '12, pages 49:1–49:1, 2012.

[280] Youyou Wang and Ergun Akleman. Turning photographs into abstract expressionist paintings. In *ACM SIGGRAPH 2012 Posters*, page 66. ACM, 2012.

[281] Youyou Wang and Ergun Akleman. Turning photographs into abstract expressionist paintings. In *ACM SIGGRAPH 2012 Posters*, page 66. ACM, 2012.

[282] Youyou Wang, Ozgur Gonen, and Ergun Akleman. Global illumination for 2d artworks with vector field rendering. In *ACM SIGGRAPH 2014 Posters*, page 95. ACM, 2014.

[283] Youyou Wang, Ozgur Gonen, and Ergun Akleman. Global illumination for 2d artworks with vector field rendering. In *ACM SIGGRAPH 2014 Posters*, SIGGRAPH '14, pages 95:1–95:1, New York, NY, USA, 2014. ACM.

[284] Yu-Shuen Wang, Chiew-Lan Tai, Olga Sorkine, and Tong-Yee Lee. Optimized scale-and-stretch for image resizing. *ACM Transactions on Graphics (TOG)*, 27(5):118, 2008.

[285] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 479–488. ACM Press/Addison-Wesley Publishing Co., 2000.

[286] Tim Weyrich, Jia Deng, Connelly Barnes, Szymon Rusinkiewicz, and Adam Finkelstein. Digital bas-relief from 3d scenes. In *ACM SIGGRAPH 2007 papers*, SIGGRAPH '07, 2007.

[287] Turner Whitted. An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, page 4. ACM, 2005.

[288] Wikipedia. en.wikipedia.org/wiki/Conservative_vector_field, 2014.

[289] Wikipedia. en.wikipedia.org/wiki/Ascending_and_Descending, 2016.

[290] Lance Williams. Casting curved shadows on curved surfaces. *ACM Siggraph Computer Graphics*, 12(3):270–274, 1978.

[291] Philip Willis. Projective alpha colour. *Computer Graphics Forum*, 25(3):557–566, 2006.

[292] Georges Winkenbach and David H Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 91–100. ACM, 1994.

[293] H. Winnemoeller, A. Orzan, L. Boissieux, and J. Thollot. Texture design and draping in 2d images. *Computer Graphics Forum*, 28(4):1091–1099, 2009.

[294] Steve Wright. *Digital compositing for film and video.* Taylor & Francis, 2013.

[295] Tai-Pang Wu, Chi-Keung Tang, Michael S. Brown, and Heung-Yeung Shum. Shapepalettes: interactive normal transfer via sketching. *ACM Trans. Graph.*, 26(3):44.1–44.6, 2007.

[296] Chris Wyman. An approximate image-space approach for interactive refraction. *ACM transactions on graphics (TOG)*, 24(3):1050–1053, 2005.

[297] Chris Wyman. Interactive image-space refraction of nearby geometry. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 205–211. ACM, 2005.

[298] Chris Wyman and Scott Davis. Interactive image-space techniques for approximating caustics. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 153–160. ACM, 2006.

[299] Q. Xing and E. Akleman. Function based color retargeting. In *Proceedings of ISAMA 2010; Art and Math Conference*, June 2010.

[300] Q. Xing, E. Akleman, G. Taubin, and J. Chen. Surface covering curves. *Computational Aesthetics in Graphics, Visualization, and Imaging 2011 (CAE'2011) Poster*, August 2011.

[301] Q. Xing, G. Esquivel, E. Akleman, J. Chen, and J. Gross. Band decomposition of 2-manifold meshes for physical construction of large structures. *ACM SIGGRAPH'2011, Late Breaking Poster and Talk, ACM Digital Library*, August 2011.

[302] Qing Xing, Ergun Akleman, Gabriel Taubin, and Jianer Chen. Surface covering curves. In *Proceedings of the Eighth Annual Symposium on Computational Aesthetics in Graphics, Visualization, and Imaging*, pages 107–114. Eurographics Association, 2012.

[303] Qing Xing, Gabriel Esquivel, E. Akleman, Shi-Yu, Jianer Chen, and Jonathan Gross. Sculpting with vertex components. *Proceedings of ISAMA 2011*, pages 100–108, June 2011.

[304] Qing Xing, Gabriel Esquivel, and Ergun Akleman. Twisted d-forms: Design and construction of d-forms with twisted prismatic handles with developable sides. In *Proceedings of Bridges: Mathematical Connections in Art, Music, and Science*, pages 323–328, 2012.

[305] Qing Xing, Gabriel Esquivel, Ergun Akleman, Jianer Chen, and Jonathan Gross. Band decomposition of 2-manifold meshes for physical construction of large structures. In *ACM SIGGRAPH 2011 Posters*, pages 58:1–58:1. ACM, 2011.

[306] Qing Xing, Gabriel Esquivel, Ryan Collier, Michael Tomaso, and Ergun Akleman. Spulenkorb: Utilize weaving methods in architectural design. In *Proceedings of Bridges: Mathematical Connections in Art, Music, and Science*, pages 163–170, 2011.

[307] Yinan Xiong. Mock-3d web application: Interactive lighting, rendering and shading for 2d artwork. Master's thesis, Texas A&M University. Available electronically from http://oaktrust.library.tamu.edu/handle/1969.1/157944., College Station, TX, 2016.

[308] Zhao Yan. Painterly shading ocean surface. Master's thesis, Texas A&M University: Available Electronically From: http://hdl.handle.net/1969.1/, College Station, Texas, 2015.

[309] Hüseyin Yilmaz. Color vision and a new approach to general perception. In *Biological Prototypes and Synthetic Systems*, pages 126–141. Springer, 1962.

[310] Hüseyin Yilmaz. On color perception. *The bulletin of mathematical biophysics*, 24(1):5–29, 1962.

[311] Hüseyin Yilmaz. Perceptual invariance and the psychophysical law. *Perception & Psychophysics*, 2(11):533–538, 1967.

[312] Jinnah Yu, Ergun Akleman, et al. Video screening. In *Renaissance Banff: Mathematics, Music, Art, Culture*, pages 187–194. Bridges Conference, 2005.

[313] Jinnah Yu, Ergun Akleman, et al. Video screening. In *Renaissance Banff: Mathematics, Music, Art, Culture*, pages 187–194. Canadian Mathematical Society, 2005.

[314] S. Yu, Q. Xing, E. Akleman, J. Chen, and J. Gross. Pattern mapping with quad-pattern-coverable quad-meshes. *ACM SIGGRAPH'2011, Late Breaking Talk, ACM Digital Library*, August 2011.

[315] Cem Yuksel and Ergun Akleman. Rendering hair-like objects with indirect illumination. In *International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2005 Sketches: Los Angeles, California*, volume 31, 2005.

[316] Cem Yuksel and Ergun Akleman. Rendering hair with global illumination. In *ACM SIGGRAPH 2006 Research posters*, page 124. ACM, 2006.

[317] Cem Yuksel, Ergun Akleman, and John Keyser. Practical global illumination for hair rendering. In *Computer Graphics and Applications, 2007. PG'07. 15th Pacific Conference on*, pages 415–418. IEEE, 2007.

[318] Ergun Akleman Yutu Liu and Jianer Chen. Never-ending storytelling with discrete-time markov processes. In *Proceedings of Bridges: Mathematical Connections in Art, Music, and Science*, pages 85–92, 2012.

[319] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.

[320] Douglas E Zongker, Dawn M Werner, Brian Curless, and David H Salesin. Environment matting and compositing. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 205–214. ACM Press/Addison-Wesley Publishing Co., 1999.

Author's Biography

ERGUN AKLEMAN

I am a Professor in Visualization Department at Texas A&M University. I received my PhD in Electrical and Computer Engineering from Georgia Institute of Technology. I am also a professional cartoonist, illustrator and caricaturist who has published more than 500 cartoons, illustrations and caricatures.In the Visualization Sciences program, I teach a wide variety of topics in covering both artistic and scientific aspects of computer graphics. The topics I usually teach include 3D modeling, rendering, visual storytelling, image based lighting and digital compositing. All my courses are interdisciplinary by nature. I combine a studio approach with lectures. In my art and design courses, students also learn mathematics. In my technical computer graphics courses that are usually heavy in computer science and mathematics, students also learn aesthetics aspects of the subject.My research work is also interdisciplinary, usually motivated by aesthetic concerns. I have published extensively in the areas of shape modeling, image synthesis, artistic depiction, image based lighting, texture and tiles, computer aided caricature, electrical engineering and computer aided architecture. My most significant and influential contributions as a researcher have been in shape modeling and computer aided sculpting. My work on a topological mesh modeling. has resulted a powerful manifold mesh modeling system, called TopMod. By using TopMod, high genus manifold shapes can easily be constructed. I also have a significant body of work in subdivision modeling and implicit modeling. My future research goals include finding new ways for physical construction of large scale complicated shapes.