MERGING LIVE VIDEO
WITH SYNTHETIC IMAGERY

A Thesis

Presented to the Faculty of the Graduate School

of Cornell Univeristy

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

by

Jeremy Adam Selan

January 2003

© 2003 Jeremy Adam Selan

ALL RIGHTS RESERVED

ABSTRACT

Merging live action and synthetic imagery in a realistic manner is becoming increasingly important, and has become ubiquitous to filmmaking and television. However, current techniques to merge live and synthetic imagery typically generate low-quality images in real-time, or high-quality results though off-line processes. Being able to realistically merge live and synthetic imagery in real-time would generate many new applications, particularly in the virtual set, augmented reality, military, and entertainment industries.

This thesis presents a framework for the realistic, real-time merging of live action with synthetic imagery by analyzing the three major components of the process. First, we maintain the necessity of utilizing synthetic imagery that mimics the visual complexity of the real world. Second, we demonstrate that attention must be spent on acquiring live action that is visually compatible with the synthetic imagery, with the emphasis on matching specific illumination characteristics. Finally, we propose the use of a compositing mechanism that accounts for the missing visual interaction between the live and synthetic components, including occlusion, shadowing, and reflection.

We present the real-time implementation of two virtual set systems that adhere to these principles. An image-based renderer generates realistic imagery in limited interaction environments, while a software-based ray-engine simulates physically based, dynamic environments. The live and synthetic environmental illumination is matched by manipulating the histogram characteristics of the live video. Finally, we implement a compositing system that accounts for the visual interactions between the live and synthetic imagery, synthesizing inter-reflections and shadows using a silhouette reprojection technique.

BIOGRAPHICAL SKETCH

The author was born on August 17, 1979 in Chicago, Illinois, although he actually grew up next-door in the peaceful suburban village of Skokie. Upon his completion of Middleton Middle School, McCracken Junior High, and Niles North High School, our hero chose to head east for more schooling and join Cornell University's College of Engineering. After completing an Electrical Engineering degree in three years (with summers), he entered the Program of Computer Graphics.

To my parents and brother.

ACKNOWLEDGEMENTS

I must first thank Professor Donald P. Greenberg, who gave me the opportunity to join this wonderful program. I am truly in awe of his love for life. His passion is evident in everything he does; I can only hope to be as lucky in the future.

To the rest of the department, I thank you for your help, kindness, and for putting up with me all these years. Though I will now try to dole out specific thanks, please do not hold it against me if for some reason you are not mentioned particularly.

* Peggy Anderson for always being extraordinarily helpful and kind, and for introducing me to the joys of skiing.
* Mary Armstrong for not holding my messy nature against me.
* Kavita Bala for being a good sport as my 417 teacher.
* Martin Berggren for helping me learn the ropes at the PCG and for giving me access to so much cool stuff.
* Steve Berman and Rich Levy for helping to hold down the PCG E.E. fort.
* Jacky Bibliowicz for being the coolest Colombian.
* Mike Donikian for protecting this great country.
* Reynald Dumont for helping me with the original (infamous) hanging monkey demo.
* Phil Dutré for helping me with brainteasers.
* Randy "Pemith" Fernando for introducing me to the subtleties of indoor mini-golf.
* Sebastian Fernandez for helping to spawn my java game addiction.
* James "Jim" Ferwerda for his guidance in the perception study.

* Suanne Fu for he help with Art 372, and for being such a great foil to Spiff.
* Vikash Goel for all of his help with Art 372 (both years) and for maintaining a relaxing atmosphere around the lab. Stay chill my brother.
* Eric Haines for being approachable, cheerful, and helpful.
* Ryan Ismert for showing me the joys of Stellas.
* Adam Kravetz for being one chill dude.
* Henry Letteron for his eagerness.
* Hongsong Li for his all-around helpful spirit.
* John Mollis for keeping the PCG Blues tradition alive. Rock on John!
* Fabio Pellacini for being a great office-mate, an all-around smart guy, and a generally cynical dude.
* Moreno Piccolotto for playing the role of the laid-back PCG Italian.
* Stefan Roesch for the music.
* Hurf Sheldon for being an all-around fantastic and helpful person, and for not acting too upset (or surprised) when I would break something.
* Linda Stephenson for all of her advice and help.
* Will Stokes for keeping the Illini connection alive and well.
* Parag Tole for being so supportive of my endeavors.
* Bruce Walter for being the impetus behind the integration of video into RTGI, and for his healthy love of bagels.
* Steve Westin for his great assistance, and for being a good neighbor.

I would like to thank my minor advisor, Shimon Edelman, for being an understanding member of my committee. I would also like to thank all those who helped out with the implementation of my perception study: the Cooley Brothers & Bindman twins (sorry guys), Kristen Hamilton, Sarah Hoen, Prof. David Field, and all of my ever-so-helpful and unsuspecting subjects.

To my family, I want to say thanks for putting up with me all of these years, and for being supportive of all of my endeavors. Mom, thanks for your values. What else is there to care about other than education, travel, and food? Dad, your content attitude and love keep me going. Daniel, thanks for being my best friend.

Finally, I would like to thank the many people who did not directly help with this thesis, but whom made my time in Ithaca an enjoyable one. To all the Hangs, keep it up, io? And where is that new CD I keep hearing about? To the boyz at FHANK, may you never see a cold wet Jeremy again. To Joe, sorry for all the Björk, but compared to the B52s I think we're equal. To Kristen, thanks for everything. To everyone else, so long and thanks for all the fish. (Get it?)

The work in this thesis was supported in part by the National Science Foundation Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-8920219) and the generous support of the Intel Corporation.

TABLE OF CONTENTS

INTRODUCTION 1

Real-Time Rendering 5

A Brief Introduction to Compositing 8

MERGING LIVE AND SYNTHETIC IMAGERY IN THE FILM INDUSTRY 13

Recreating Realism 15

Compositing in the Visual Effects Industry 34

Inserting Real Objects into Virtual Worlds 38

MERGING LIVE AND SYNTHETIC IMAGERY IN THE TELEVISION INDUSTRY 40

Camera Tracking 44

Matte Generation and Compositing 52

Real-Time Image Generation 60

PERCEPTION OF LIGHTING ERRORS IN IMAGE COMPOSITING 62

Source Errors in Image Compositing 67

Procedure 74

Results 76

Conclusion 83

TWO VIRTUAL SET IMPEMENTATIONS 85

Two Approaches to Realistic Environment Generation 88

Controlling Live Action Image Characteristics 104

Compositing With Layer Interactions 107
Implementation 111
Results 119
CONCLUSION 129
BIBLIOGRAPHY 132

LIST OF FIGURES

CHAPTER 1

Figure 1.1: Top Ten Grossing Films 2

Figure 1.2: Three Components of Merging Live and Synthetic Imagery 3

Figure 1.3: Real-Time vs. Infinite Time Rendering Continuum 6

Figure 1.4: Introduction to Compositing 8

Figure 1.5: Compositing with Alpha Representation 10

Figure 1.6: Porter and Duff Compositing Operators 11

CHAPTER 2

Figure 2.1: The Motion-Picture Visual Effects Pipeline 14

Figure 2.2: Subdivision Surface Refinement 17

Figure 2.3: Edge Geometry in Lighting and Rendering 18

Figure 2.4: The Displaced Subdivision Surface 19

Figure 2.5: Procedural Modeling, Shrek 20

Figure 2.6: Procedural Modeling, L-Systems 21

Figure 2.7: Renderman Surface Shader 22

Figure 2.8: Physically Based Animation 24

Figure 2.9: Particle System, Twister 26

Figure 2.10: Direct Illumination vs. Global Illumination..... 28

Figure 2.11: Cinematic Lighting, Harry Potter 30

Figure 2.12: Cinematic Lighting, Stuart Little 32

Figure 2.13: Wrapped Diffuse Lighting 33

Figure 2.14: Rim Lighting, Stuart Little 34

Figure 2.15: Virtual Shadows in Compositing…… 35

Figure 2.16: Layer Compositing, The Perfect Storm …… 37

Figure 2.17: Live Footage into Virtual Environments, The Matrix …… 38

CHAPTER 3

Figure 3.1: Virtual Set Example …… 40

Figure 3.2: Virtual Set Breakdown …… 42

Figure 3.3: Component Overview …… 43

Figure 3.4: Camera Tracking …… 45

Figure 3.5: Camera Tracking, Physically Based …… 46

Figure 3.6: Camera Tracking, Radio Wave …… 48

Figure 3.7: Camera Tracking, IR…… 49

Figure 3.8: Camera Tracking, Pattern Recognition …… 51

Figure 3.9: Ultimatte Matte Generation …… 53

Figure 3.10: Chromakey vs. Ultimatte…… 54

Figure 3.11: LiteRing Matte Generation …… 56

Figure 3.12: Live Depth Information …… 57

Figure 3.13: Z-Cam …… 58

Figure 3.14: Garbage Mattes …… 59

Figure 3.15: Compositing with Depth-Of-Field …… 60

CHAPTER 4

Figure 4.1: Categories of Compositing Errors…… 62

Figure 4.2: Matching Composite Illumination, Light stage …… 65

Figure 4.3: Two Experimental Environments…… 69

Figure 4.4: Range of Chromaticity and Intensity Variations …… 71

Figure 4.5: Range of Directionality Variations …… 73

Figure 4.6: Results, Chromaticity Errors…… 77

Figure 4.7: Results, Intensity Errors …… 78

Figure 4.8: Results, Local Illumination Directionality Errors …… 79

Figure 4.9: Results, Cast Shadow Directionality Errors …… 82

CHAPTER 5

Figure 5.1: Planar Distortion in Virtual Set Compositing …… 87

Figure 5.2: Image Based Rendering, QuickTimeVR …… 91

Figure 5.3: The Lightfield Two-slab Parameterization …… 92

Figure 5.4: Image-based Virtual Set Environment Generation …… 94

Figure 5.5: Capturing A Lightfield Environment …… 95

Figure 5.6: Computing Novel Frustum Intersections …… 97

Figure 5.7: Resampling Texture Space …… 98

Figure 5.8: Two IBR Sampling Methods …… 99

Figure 5.9: Lightfield Image Synthesis Results …… 100

Figure 5.10: Three Global Illumination Algorithms …… 103

Figure 5.11: Accounting For Illumination Inconsistencies …… 106

Figure 5.12: Compositing With Shadows and Reflections …… 109

Figure 5.13: A Linear Matte Generation Algorithm …… 112

Figure 5.14: Results, Linear Keyer …… 114

Figure 5.15: IBR System Overview…… 116
Figure 5.16: RTGI System Overview, Full Frame …… 117
Figure 5.17: RTGI System Overview, Render Cache …… 118
Figure 5.18: Results, Image-Based Soft Shadowing…… 121
Figure 5.19: Results, Image-Based Shadowing and Reflection…… 122
Figure 5.20: Results, Comparison of Image-Based Methods …… 124
Figure 5.21: Results, Full Image-Based Virtual Set Simulation …… 125
Figure 5.22: Results, Single-Image Virtual Set Environment …… 126
Figure 5.23: Results, Render Cache Polygonal Integration …… 127
Figure 5.24: Results, Full Frame Polygonal Integration …… 128

Chapter 1

Introduction

Merging live action with synthetic imagery is becoming increasingly important in such disciplines as filmmaking, television, augmented reality, and gaming. In the motion-picture industry, merging live action and synthetic imagery has become the basis for almost all visual-effects. In the television industry, virtual sets (which merge live action with computer-generated scenes in real-time) are quickly becoming commonplace for newscasts and game shows. Furthermore, realistically merging live and synthetic action in real-time offers exciting new applications in areas such as augmented reality and gaming.

The potential importance of merging live and synthetic action can perhaps be best illustrated by observing how lucrative motion-picture visual effects have become. As shown in Figure 1.1, nine of the top-ten grossing movies ever released in the United

States are "effects films" [1]. We would argue that the success of these films is directly attributable to their stellar visual effects; thus concluding that the realistic merging of live action and synthetic imagery has an immense commercial appeal.

Figure 1.1: Merging live action with synthetic imagery is very lucrative, as evident in the motion-picture industry. Of the top-ten grossing movies to be released in the U.S., nine were nominated for Academy Awards in "Best Visual Effects". List compiled in March 2002 from [Best02].

The process of merging live and synthetic imagery can be separated into three main components: generating the synthetic imagery, acquiring the live-action, and merging the two together. In order to do this in a realistic manner, one must maintain the realism at each step in the process, illustrated in Figure 1.2.

[1] We define *effects films* as movies which have been nominated for the "Best Visual Effects" Academy Award™.

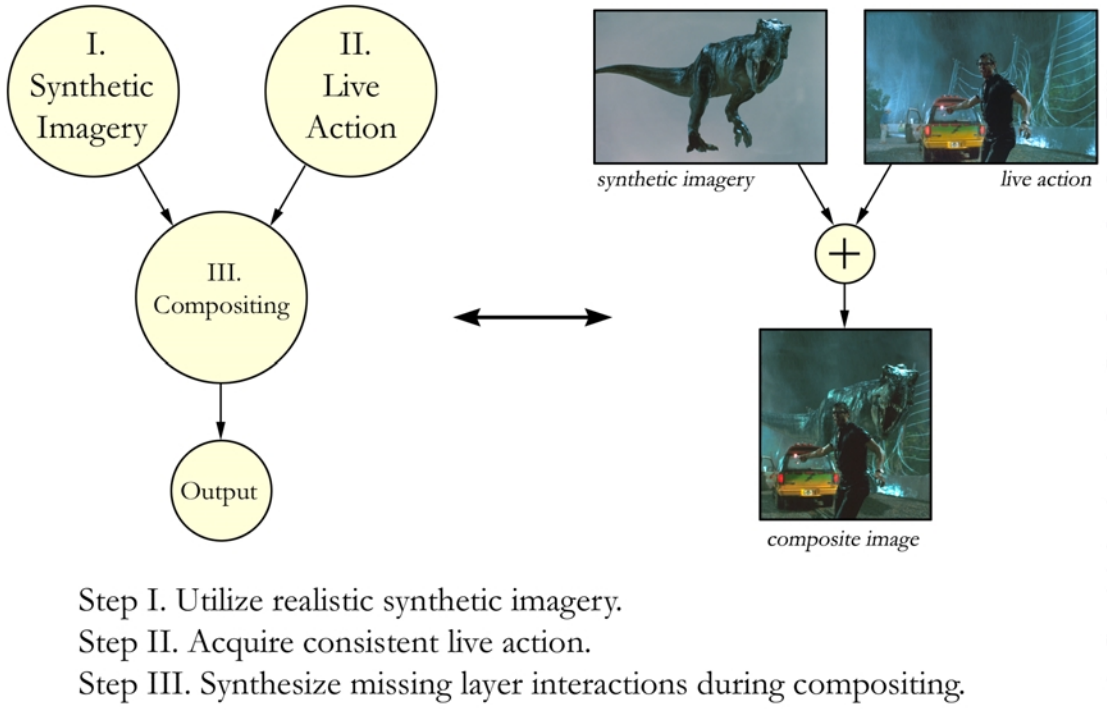


Figure 1.2: Merging live action and synthetic imagery in a realistic manner is dependent upon each aspect of the synthesis process (top left) and on fulfilling the three criteria listed above. Above we observe the merging of live and synthetic imagery in Jurassic Park (1993). Step I: The dinosaur has complex geometry and a sophisticated surface appearance, both contributing to its realistic appearance. Step II: The live action (specifically the illumination) is crafted in a manner to seamlessly blend with the synthetic imagery. Step III: During compositing the layer interactions (occlusions, shadowing, and reflections) are simulated, generating a unified composite output.

In this thesis we identify the following three components as being integral to the realistic merging of live and synthetic imagery:

I. Utilize realistic synthetic imagery.

II. Acquire realistic live action imagery in a manner that facilitates realistic compositing.

III. Leverage a compositing process that maintains the perception of realism, typically by recreating missing interactions between the live and synthetic elements.

Synthesizing imagery that mimics the visual complexity of the real world is a difficult task because geometric, illumination, material, and dynamic complexity are all important contributors to image realism. There is no single approach that always yields "realistic" synthetic imagery, characteristic of natural environments.

The motion-picture visual effects industry has had the most experience and success in synthesizing realistic imagery, and as such has popularized many of the techniques used in realistic image synthesis. For example, displacement mapping and subdivision surfaces allow complex, yet intuitive, surface representations. Sophisticated tools including image-based lighting and global illumination simulate realistic illumination environments. Procedural material definitions along with texture mapping yield highly realistic (though not predictive) light/surface interactions. Finally, procedural modeling algorithms automate the process of generating naturally complex geometry.

The second component to merging live and synthetic imagery is capturing the live action in a manner that facilitates realistic compositing. Numerous image characteristics including scene illumination (color balance), image resolution, depth-of-field, and film grain must be consistent between the live and synthetic imagery to yield a believable composite.

Finally, the live and synthetic elements are merged in a realistic manner. This requires using both a precise compositing process (free of artifacts) and adding additional image elements that account for the missing interactions. Accounting for the visual interactions

between the live and synthetic imagery (such as shadows, occlusions, and secondary reflections) is crucial to insuring a realistic result.

Although the film industry utilizes numerous methods to account for these scenarios, they are not typically compatible with our goal of real-time image synthesis. While the visual-effects industry typically allocates on the order of a few weeks for the completion of each shot, our goal is to do so in real-time.

Real-Time Rendering

Due to computation constraints, "realistic" animations are usually batch rendered, and viewed later. Allowable completion times for these realistic applications (such as feature-film visual effects work) are relatively unlimited, and individual frames can take hours to produce [Alex2001]. On the other hand, virtual sets, video games, 3d visualization, and virtual reality all emphasize real-time output. This thesis begins to bridge these extremes by creating systems with intermediate levels of both realism and performance. Figure 1.3 illustrates this approach.

To achieve the highest level of realism one must perform a rudimentary cost-benefit analysis, weighing the performance impact of a particular feature against the corresponding gains in image realism. In merging live and synthetic imagery one must consider multiple factors. For example, the choice of output media (video or CRT) has a great impact upon the acceptable rendering performance. Furthermore, there are differing definitions of what constitutes "real-time" performance. Frame rates between 24-60hz all may fulfill the "real-time" requirements of different systems. System latency is

also ill-defined. For many real-time applications, a lag of one second is acceptable, while for many it is not. We thus argue that the success of any targeted approach requires a precise specification of its particular display environment.

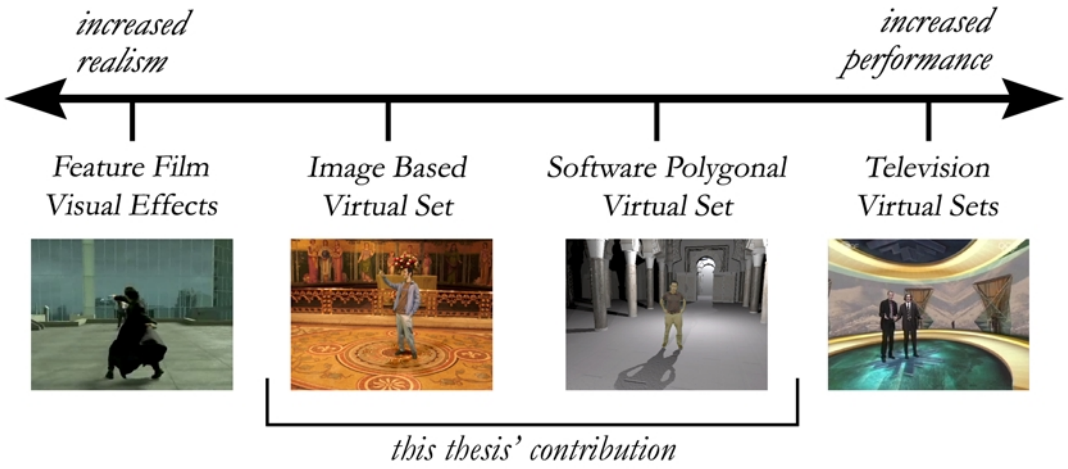


Figure 1.3: Realism and performance are at opposing ends of the continuum in merging live and synthetic imagery. This is because in order to achieve higher levels of performance and flexibility, one must typically sacrifice system features that contribute to image realism. This thesis attempts to find the middle ground, offering intermediate levels of realism and performance compared to the feature-film visual effects and virtual set industries.

Realistic real-time media thus have to make up for the drastically reduced time available for geometric complexity, materials, rendering, and simulation. Using the concept of the "time budget" [Alex2001] guides us to consider real-time rendering as an optimization task, with the goal being to allocate the limited computational resources to algorithms in proportions that maximize the overall visual fidelity. This is often implemented by utilizing multiple algorithms that perform the same task, yet offer contrasting levels of cost and performance. Gracefully switching between the alternatives to meet the given quality threshold for a set of system constraints is no doubt close to the optimal solution.

Our Approach

We implement a system for merging live action with synthetic imagery by considering the three system components (rendering synthetic imagery/ acquiring live action / realistic compositing), and designing a real-time implementation that balances the realism and performance trade-offs particular to each approach.

We present two rendering systems that address these motivations. For scenes in which the live camera is constrained to a limited set of viewpoints, an image based rendering algorithm is employed. If completely arbitrary viewpoints are required, a cluster of personal computers is utilized to render (in software) a real-time, globally illuminated environment.

To match live and synthetic image characteristics we first identify the characteristics that are most important (contribute the most to visual realism). After a review of the pertinent literature, we conclude that matching the illumination between live and synthetic imagery is the most important factor to consider. We ran a series of psychophysical experiments to further characterize the saliency of the different illuminations errors (brightness, chromaticity, and directionality) one typically encounters in image compositing.

Finally, we implement a real-time compositing system that accounts for the visual interactions between the live and synthetic imagery. We synthesize reflections and shadows using a silhouette reprojection algorithm, achieving our goal of realistic real-time compositing.

As the principles of compositing are the grounding for much of this thesis, we begin with an overview of the fundamentals of image compositing.

A Brief Introduction to Compositing

Image compositing is widely used in computer graphics to merge independently created visual elements into a single image. The origins of compositing date back to 1857, when Oscar G. Rejlander created a single image by combining different regions of 32 photographs [Brink99]. Although in Rejlander's era this "trick photography" was considered deceitful, in modern times it has become a common form of image manipulation. Digital image compositing is now ubiquitous in such areas as filmmaking [BerRed00] [Egstad98], television [Ultim02b], virtual sets [Hayashi98], and augmented reality [Debevec98].

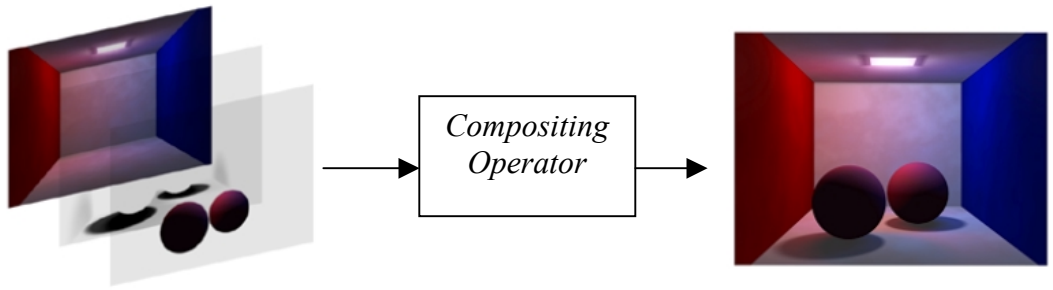


Figure 1.4: The "over" composite operator brings together multiple image layers, known as elements, into a final image (right). An additional layer for each element defines the opacity at each pixel (not shown). This additional information is referred to as the alpha channel in the graphics literature, and as the matte channel in the film and television industries.

Although analog compositing has been used in the film industry since the 1940s, the digital variant was introduced to computer graphics in 1984 when Porter and Duff

[PorDuf84] introduced a digital matting algebra. They defined a concise set of operators that merge image elements by leveraging an extra channel of image information. This extra channel, the alpha channel, defines the transparency at each pixel. Alpha values are defined within the range of zero to one, with an alpha of zero defining a fully transparent pixel, and an alpha of one defining a fully opaque pixel. Intermediate values denote partial transparency.

By extending the pixel's definition to include transparency information, one can composite images that include both partial occlusion and partial transparency. Partial occlusion is common in rendering, formed when edge pixels partially cover the background. Partial transparency is also common, particularly with volumetric effects such as fog.

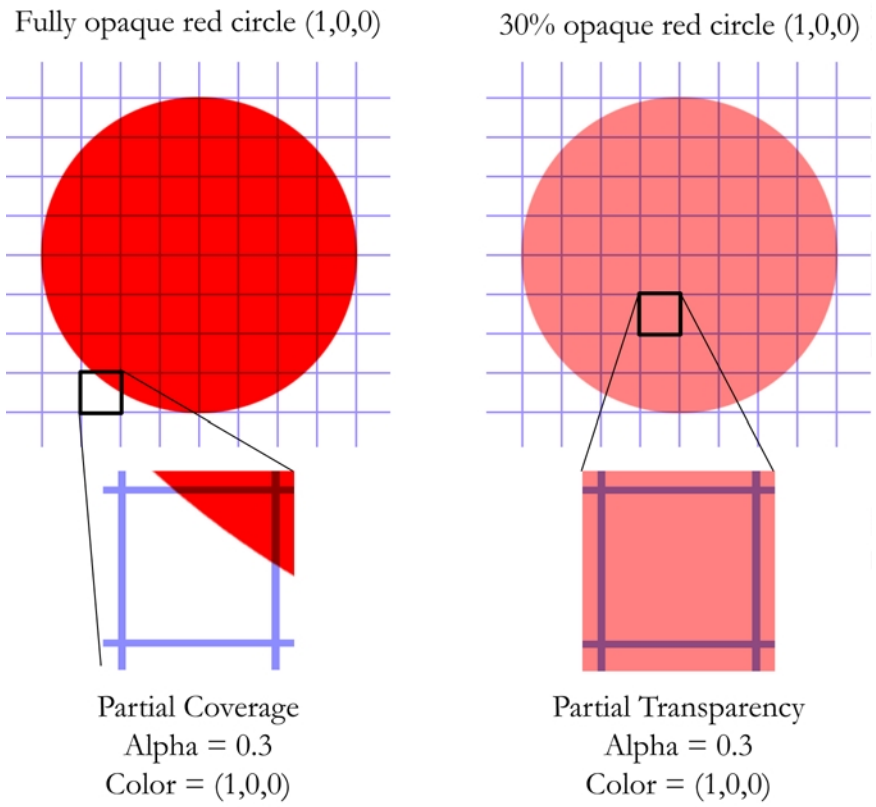


Figure 1.5: Pixels that partially occlude the background are assigned fractional values for alpha (left). Similarly, pixels that are partially transparent (even if they fully cover the pixel) are assigned fractional values for their alpha as well. This convention allows for compositing algorithms to appropriately handle both anti-aliased and transparent elements.

The Porter and Duff [PorDuf84] compositing operators are shown in Figure 1.6. Although many combinations of source material are defined (over, in, out, atop, XOR), the "over" operator is most commonly used [Berney00]. Although all of these functions operate on only two layers, an arbitrary number of elements can be edited using successive applications of the operators on pairs of elements.

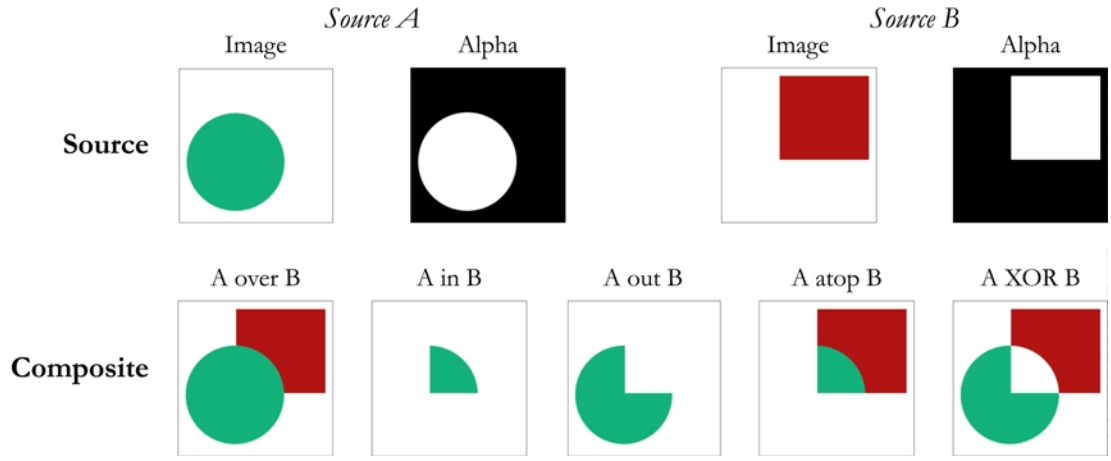


Figure 1.6: The common compositing operators defined by Porter and Duff {over, in, out, atop, XOR} are created by all the possible combinations of input regions.

The common formulation for the "over" operator is:

$$C_{out} = \alpha_{top} \cdot C_{top} + C_{bottom} \cdot (1 - \alpha_{top}) \cdot \alpha_{bottom}$$
$$\alpha_{out} = \alpha_{top} + \alpha_{bottom} \cdot (1 - \alpha_{top}) \quad ,$$

where C_{out} is the output color, C_{top} is the color of the top input, C_{bottom} is the color of the bottom input, and α_{out} is the output transparency of the composite. The output alpha is necessary when compositing multiple layers. Note that the output color is formed as a linear combination of the input colors, in proportion to the top-most alpha.

It is important to observe that the Porter and Duff compositing algebra does have limitations. Specifically, the current formulation relies on the assumption that the sub-pixel coverage of the input elements is uncorrelated. Though this assumption is reasonable when compositing a single pair of image elements that span multiple pixels, it breaks down when compositing large numbers of layers with sub-pixel detail. The

resulting error is observed as colors from fully occluded background objects incorrectly "bleed" onto the final image. Although a complete discussion of the issue is beyond the scope of this thesis, other techniques such as multisampling help mitigate, though not eliminate the problem. Despite this limitation, compositing is such a useful and enabling technique that virtually all image generation algorithms rely on some form of alpha-based compositing.

Computing the transparency information for physically acquired images is a necessity for realistic image compositing and is not trivial. Drawing from both art and science, the common matte generation techniques range from being fully automated (blue-screening), to completely manual (rotoscoping), and in between (Bayesian matting). These techniques (and others) will be discussed in Chapters Two and Three.

Organization

The remainder of this thesis is organized as follows: Chapter Two outlines the techniques used by the motion picture visual effects industry to realistically merge live and synthetic imagery. Real-time methods utilized by the television industry to perform a similar feat are discussed in Chapter Three. Chapter Four addresses the goal of merging visually consistent live and synthetic image element by quantifying the effect that illumination inconsistencies have upon composite realism. Chapter Five details the implementation of two virtual set systems, which balance the tradeoffs between realism and interactivity. We finally conclude with a summary of our contributions and avenues for further research.

Chapter 2

Merging Live and Synthetic Imagery in the Film Industry

The film industry transparently merges live and synthetic imagery by creating composite elements that match viewers' expectations of the real world. Although not immediately apparent, the viewer's expectations of reality do not always coincide with reality itself; and the difference is critical.[1] Necessitated by short research and development periods, the film industry often falls to an "ends-justify-the-means" approach to image generation, where any algorithm that yields good-looking results (regardless of its "correctness") is employed. Furthermore, the nature of film production also demands that the artists creating synthetic imagery have very precise control of the image pipeline, as complete control is necessary to guide the results towards the director's artistic vision.

The processes that merge live and synthetic imagery can best be analyzed in the visual effects production pipeline, as each step is delegated to a specific algorithm. Common to all the algorithms is the goal of achieving the most realism at the lowest cost (both computational and user time), subject to the constraint that the algorithm must allow for highly controllable results.

---

[1] [Apocada00] presents the notion of photosurrealism, whereby synthetic imagery is best realized in a manner that is more of a caricature of reality, than of reality itself. This approach is based on the observation that movies (unlike documentaries) record a manipulated reality that is accented to serve a story-telling goal.

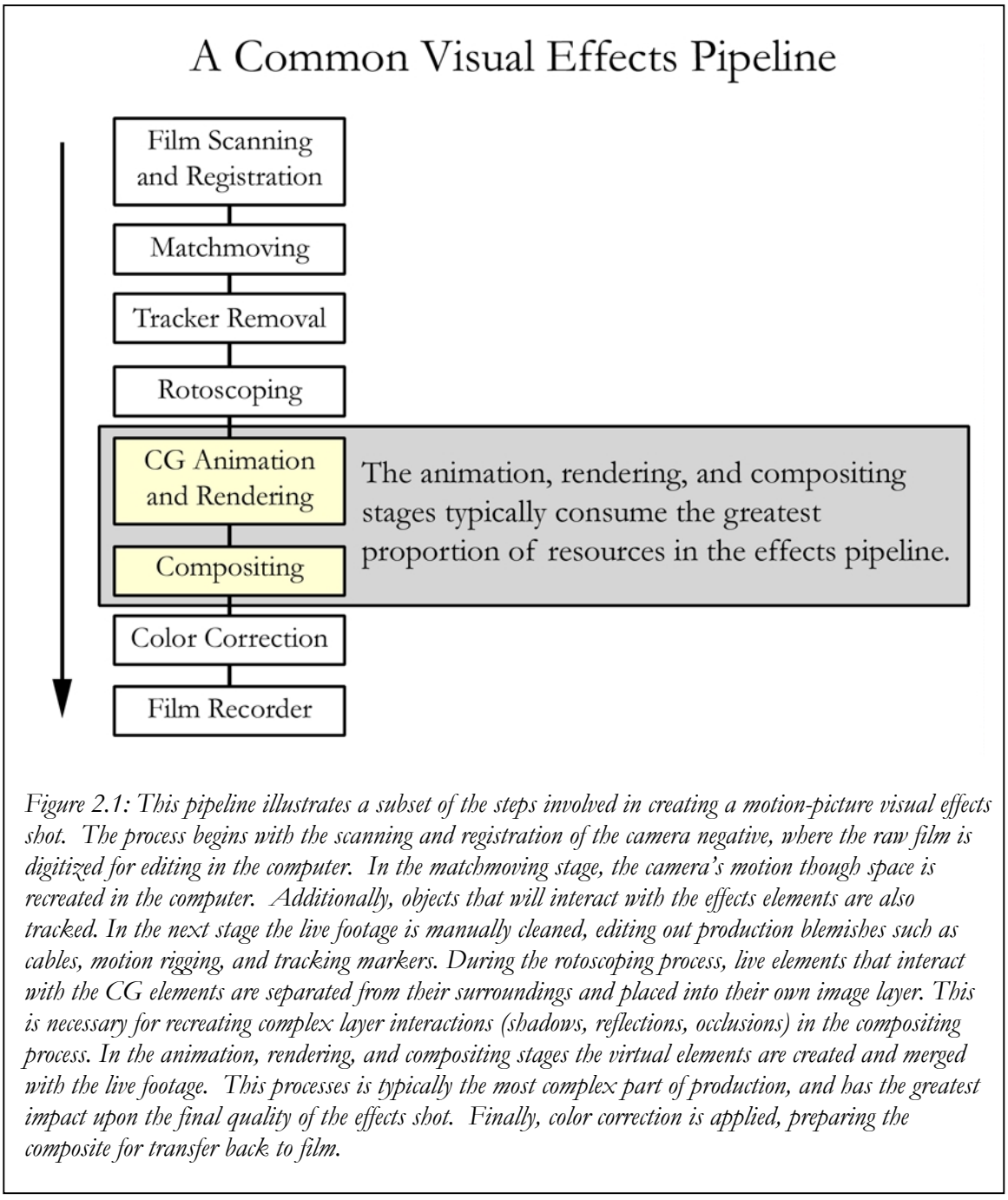


Figure 2.1: This pipeline illustrates a subset of the steps involved in creating a motion-picture visual effects shot. The process begins with the scanning and registration of the camera negative, where the raw film is digitized for editing in the computer. In the matchmoving stage, the camera's motion though space is recreated in the computer. Additionally, objects that will interact with the effects elements are also tracked. In the next stage the live footage is manually cleaned, editing out production blemishes such as cables, motion rigging, and tracking markers. During the rotoscoping process, live elements that interact with the CG elements are separated from their surroundings and placed into their own image layer. This is necessary for recreating complex layer interactions (shadows, reflections, occlusions) in the compositing process. In the animation, rendering, and compositing stages the virtual elements are created and merged with the live footage. This processes is typically the most complex part of production, and has the greatest impact upon the final quality of the effects shot. Finally, color correction is applied, preparing the composite for transfer back to film.

Though all of the steps shown in Figure 2.1 are necessary to accurately merge live action with computer graphics, the stages of animation, rendering, and compositing typically have the greatest impact upon the quality of the result. The remainder of this chapter will

focus on these stages. Additionally, for simplicity we will only discuss the process of merging computer-generated objects into otherwise live action environments. However, later in this chapter we will address the inverse task of placing live characters into a fully CG environment, and the different set of characteristics this problem poses.

Recreating Realism Through Animation and Rendering

As previously stated, the goal of animation and rendering is to create imagery that matches the viewer's expectations of the visual complexity in the real world. Effectively compositing live action with CG requires that the composited elements have coherent image characteristics. Thus, as the artist typically has more control over the CG elements, creating a realistic composite necessitates rendering elements that conform to the image characteristics of the already shot live action plate. Thus, we address the animation and rendering techniques used to match reality.

Geometric realism and complexity

The real world has an immense amount of geometric complexity, and some portion of this complexity needs to be incorporated into the modeling process to accurately recreate "realistic" objects. Unfortunately, most objects in the real world have a staggering amount of detail, with structure and variation at numerous levels of observation. This usually precludes the exact modeling of the objects in a scene, necessitating geometric approximations. Furthermore, even if one could recreate highly detailed geometry for entire scenes, this is not necessarily desirable as it would hobble rendering performance

and slow down user interactivity. A range of computer graphics techniques addresses this problem, including texture mapping, bump and displacement mapping, numerous advanced surface representations, and procedural modeling. These techniques all attempt to reduce the scope of the modeling and rendering tasks, by making simplifications based on observations of natural environments. For example, diffuse texture mapping allows the diffuse surface reflections to be stored in a compact, efficient data structure, as opposed to the physically correct method of using a unique illumination model for at point on the surface. However, the most popular method to cope with the immense geometric detail necessary to recreate natural objects is to leverage efficient geometric surface representations.

These geometric representations allow rendering engines to efficiently work on highly complex, curved surfaces and also enable artists to intuitively design complex objects. Though a number of efficient surface representations are in use today (NURBS, Metaballs, Polygons), subdivision surfaces have emerged as the preferred surface representation in the motion-picture effects industry.

Though a complete discussion of the alternative surface representation methods (NURBS, Metaballs, and Polygons) is beyond the scope of this thesis, they each have distinct disadvantages that limit their use. For example, although NURBS surfaces allow for the efficient storage of smooth surfaces, it is difficult to generate smooth parametric (UV) coordinates across multiple adjoining NURBS surfaces. Although for simple applications (such as texture mapping) disjoint UV coordinates can be worked around, in cinematic effects work (such as growing fur) irregular UV mappings are unacceptable.

Originally formulated by [Catmull78] and [Doo78], subdivision surfaces are continuous meshes that are synthesized by iteratively subdividing sets of control vertices. The degree of refinement can be optimized for specific applications (i.e. high refinement levels for rendering, low refinement levels for animating), and even varied across a static model.

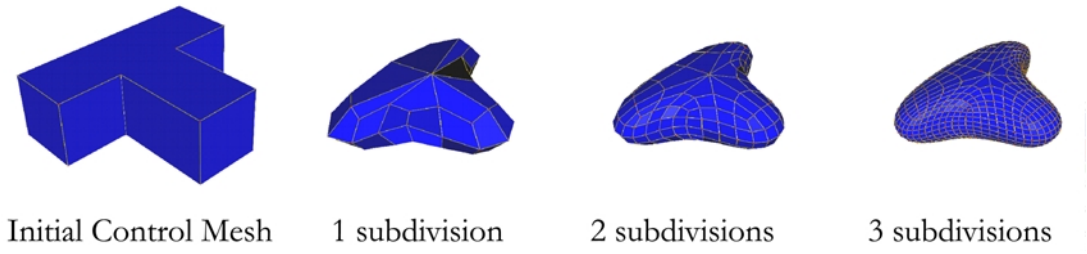


Figure 2.2: *Subdivision surfaces can be refined to arbitrary levels of precision. At each subsequent level of refinement, the mesh becomes smoother and more closely approximates its true form. The above object has been refined using the Catmull-Clark refinement process. Images from [Joy99].*

Subdivision surfaces are very powerful geometric representations because arbitrary surfaces can be stored using a small set of control points, as opposed to explicitly storing the fully tessellated polygonal mesh. Additionally, as the degree of the subdivision is progressive, the surfaces have practically infinite detail and are smooth at all zoom levels. This is not true for polygonal meshes, where close-ups will frequently show their faceted nature. Furthermore, interacting with complex subdivision surfaces is far easier than with polygons or NURBS, as with the latter users typically have far more control vertices to manipulate.

The efficient representation of curved surfaces is critical for recreating realistic geometry, as no objects have truly "sharp" edges or flat faces. Although accurately recreating curved edges may seem like a minor detail-- as they typically consume very little screen

space-- realistic edges are often crucial in realistic lighting, as they define the specular characteristics of the object.

Figure 2.3: The lighting is the same in both images, but the image on the right looks better because of the beveled (curved) edges. Although a highly tessellated polygonal mesh could have been used to represent the curved surface, subdivision surfaces are much more efficient. Images from [Birn00].

Two extensions to subdivision surfaces greatly extend their utility in modeling real-world objects. For objects that are truly not smooth, one can define breaks in the continuity of the first and second derivatives along edges [Sederberg98], allowing for the precise placement of creases. Another advance, [Lee00], details the displacing of subdivision surfaces, allowing fine details to easily be overlaid upon the otherwise smooth mesh. This technique of using displaced subdivision surfaces to store complex geometry is becoming more popular, and will likely continue to do so in the future. In fact, hardware manufacturers [ATI01] [NVIDIA02] have even begun supporting forms of displaced, subdivision surfaces directly on the video card.

Figure 2.4: Displaced subdivision surfaces allow for highly compact representations of complex surfaces. Left: Control mesh, rendered as flat shaded polygons. Center: A smooth subdivision surface is created by iteratively refining the control mesh. Right: Displaced subdivision surface, through the addition of a small displacement texture. Images from [Lee00].

However, even considering displaced subdivision surfaces, one of the major problems in modeling realistic scenes remains: how to efficiently model large scenes with the detail, clutter, and chaos characteristic of natural environments. For example, it is plausible that an experienced modeler could use displaced subdivision surfaces to model a single, highly realistic tree in a reasonable amount of time. However, if the goal were instead to realistically model the entire *forest*, completion would be implausible. In these situations, procedural modeling is an enticing alternative to the brute force approach.

Whereas traditional geometric modeling algorithms explicitly define an object through a vertex mesh, procedural-modeling algorithms generate geometry as needed on the fly using predefined rules and initial conditions. As such, the procedural modeler's task is not to sculpt a surface per se, but instead to only define the rules by which the surface can "sculpt itself". This approach lends itself to creating complex environments that exhibit high degrees of self-similarity such as grass fields, forests, and rocky beaches. However, with this powerful approach comes a downside: by algorithmically defining the geometry, it becomes highly difficult to uniquely edit the appearance of a single object, while leaving the surrounding objects unaffected.

Figure 2.5: The fields and forests were procedural generated for Shrek (2001). Modeling these objects in a traditional manner would have been practically impossible, given the enormous complexity and limited time allotted to creating the scene. Simply instantiating multiple copies of a common blade of grass also would not have been satisfactory, as each blade of grass and tree would be identical and not exhibit the variation present in natural environments.

Procedural modeling was introduced in 1968 [Lindenmayer68] to simplify the modeling of biological plant growth. Now known as "L-Systems", Lindenmayer's approach generated geometric models using three components: the definition of possible geometric states, an initital condition, and a closed set of rules defining a recursive procedure. By repeatedly applying the recursive procedure to the initial condition, a final model is iteratively built up. By specifying different initial conditions, an almost limitless variety of plants can be generated. Further extensions to this methodolgy [Prusinkiewicz96] are now common in motion picture modeling. For example, Pacific Data Images used a similar algorithmic approach to generate the grass and trees in Shrek. Trees are idealized as single curves during the animation process, with the complete geometric models generated only during rendering. Grasses and floweres are handled similarly. This type of

functionality is now becoming commonplace, and has led to the emergence of software packages based upon procedural modeling principles (of which Houdini™ is the most popular).

Figure 2.6: L-Systems are used to procedurally generate the model of a plant. On the left we see the recursion algorithm used to create new branches. In the center we see the procedurally generated mesh. On the right we see the rendered results of the algorithm. By modifying parameters used in the algorithm such as the angle between leaves, and the noise amplitude at each joint, almost limitless numbers of similar—though not identical—plants can be created. Images from Houdini™.

Material complexity and realism

While the task of modeling typically focuses on recreating objects on the macroscopic level, the clever use of materials is often used to recreate the physics of light interaction on lower levels. For example, "small" surface features such as tears, flaws, and cracks are not always modeled as geometry (as may be appropriate), but instead considered to be a material property along with the color and illumination models. Small procedural programs, known as surface shaders, define the fine-scale geometry as well as the "look" of the rendered surface. Although there are numerous shading languages, the most popular language in the motion picture industry is the Renderman API, defined by Pixar in 1989 [Pixar89].

Figure 2.7: All of the above objects are rendered using a single surface shader. The differing appearance of each model is a result of specifying unique surface parameters. It is important to note that nowhere in the material definition is there a photograph of "wood". The pattern is generated entirely through algorithmic methods, which allows for phenomenal power in guiding the final surface appearance. Images were rendered with Pixar's PRMAN, and found in [Apodaca99]

When matching virtual objects to natural environments, procedural material definitions are absolutely critical, as well written shaders offer phenomenal access to the factors that define the "look" of an object. This flexibility is of great value to the visual effects industry, as it allows for great amounts of time to be spent creating sophisticated shaders that can be reused as necessary, yet still customized to allow high levels of visual control in demanding situations. For example, as "wooden" objects are quite common in real life, creating realistic wooden materials for effects is also commonplace (as shown in Figure 2.7). If a simple texture map were used in the example, it would have been necessary to generate different maps for each object, or the repetition would have been obvious.

However, a single procedurally defined shader generated realistic, yet distinct results for each model.

Animation realism and complexity

Recreating realistic animation is one of the most difficult tasks in merging live and synthetic action, as the human mind is highly optimized for detecting natural movements from our every day experiences. In the main computer-graphics categories of animation: character animation, natural phenomenon (smoke, clouds, fluid flow), and Newtonian physics (things that bump, bend, collide, fall, and shatter), motion that goes against our expectations can be very jarring, and destroy the illusion of reality [Alex01].

Thus, the motion picture effects industry has developed specialized techniques that automate the process of creating highly realistic animation, while maximizing the controls afforded to the user. Specifically, physically based animation [Baraff90] simulates Newtonian physics, particle systems [Reeves83] are used to simulate natural phenomenon, and motion capture [Zordan02], inverse kinematics [Barzel88], and autonomous control methods [Reynolds87] are all used to efficiently animate characters. Once again, in most cases the artist sacrifices some control in guiding the results in order to accelerate the process as a whole.

As opposed to the traditional method of manually specifying keyframes, animations that rely on the laws of Newtonian physics can be computed automatically by iteratively applying the laws of physics over a series of discrete time steps [Baraff90] [Isaacs87]. By quantifying object properties such as elasticity, mass, and rigidity, the computer is able to compute object collisions, and then solve for the resulting positions and deformations.

As expected, this animation technique is well suited to simulating the motion of large numbers of objects under the influences of natural forces (gravity, wind, inter-object collisions, etc.).

Using physics to compute object motions can produce very realistic results, and remove much of the tedium involved in the animating process. However, the technique has its drawbacks. As these simulations are often very sensitive to the initial conditions, it is usually difficult to guide the simulation to a particular conclusion. Furthermore, due to quantization errors in typical algorithms (both in the temporal and spatial domains) the simulation techniques often produce unexpected results if the temporal and spatial step sizes are not small enough.

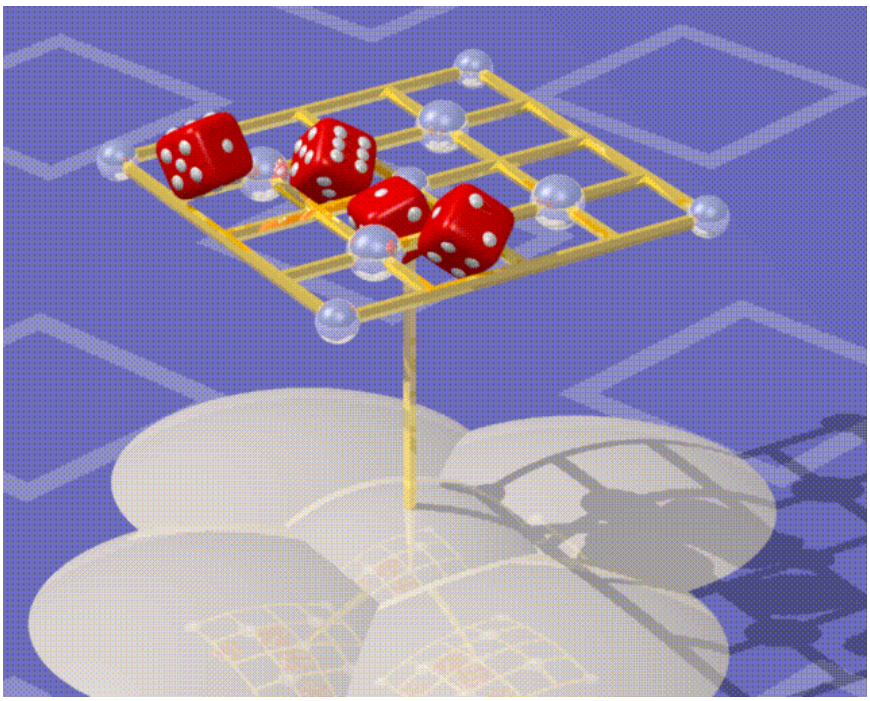


Figure 2.8: The motion of the dice have been computed procedurally, using the initial conditions and simulating the laws of motion. Unlike most modern simulation techniques, the solver used in the above simulation is analytic, and generates artifact-free results. Images from [Baraff90].

To improve interactivity using these techniques, the animation data can be distilled into a minimal set of key frames that recreate the simulated motion within a user-specified error bounds. The user can then manipulate these keyframes in the traditional manner [Reactor01].

Particle systems are commonly used to simulate a variety of natural phenomenon, such as smoke, clouds, and fluids. First introduced to aid in the production of Star Trek: Wrath of Khan, [Reeves83] described an animation system that generated coherent motions by iteratively applying the laws of physics to large sets of points. Each point has a defined position, velocity, and mass, which is updated automatically as the simulation progresses. By specifying forces that act on each point, and by defining properties including shape, size, lifetime, and transparency, emergent behaviors can appear over time. For example, in Figure 2.9 we can observe the particle system that was used to simulate the tornado effect in Twister (1996).

Figure 2.9: Twister (1996) simulated tornados using the emergent behaviors of particle systems. The shape of the funnel cloud is never explicitly modeled, but instead naturally emerges though the interactions of the particles with the vortex. The results are very convincing and accurately simulate many aspects of physical tornado's behaviors.

Yet another technique that allows animators to accomplish more work in less time is the use of rule based animation systems. As opposed to creating animation in the traditional linear fashion (one that always plays back from start to finish) the animator creates a set of short animations that serve as a palette of motions. The animator can then assign these motions (as well as transitions between them) to procedurally driven characters, which then function as autonomous beings.

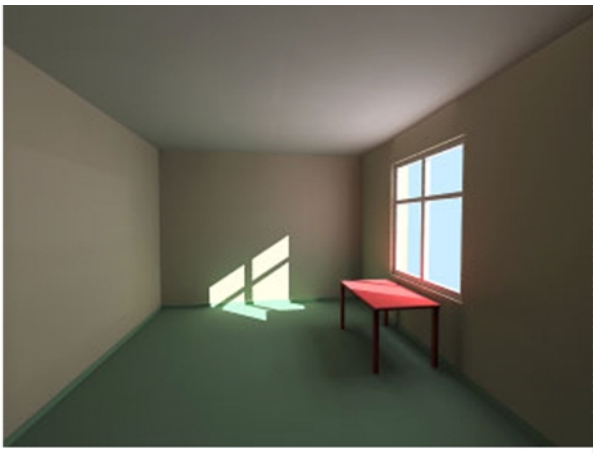
This system is especially useful in animating crowd behaviors [Reynolds87], where it is not often necessary to precisely define the action of each character, but only to govern the overall actions of the crowd. For example, in Lord of the Rings, a rule based animation system (called Massive) was used to animate large-scale battle sequences. While the front most characters were directly animated with motion capture data, the thousands of characters fighting in the background were autonomously animated, deriving their actions from a limited sort of "intelligence". Although the system's rules

had to be carefully designed to create the desired results, a great amount of time was saved over having to animate each character by hand.

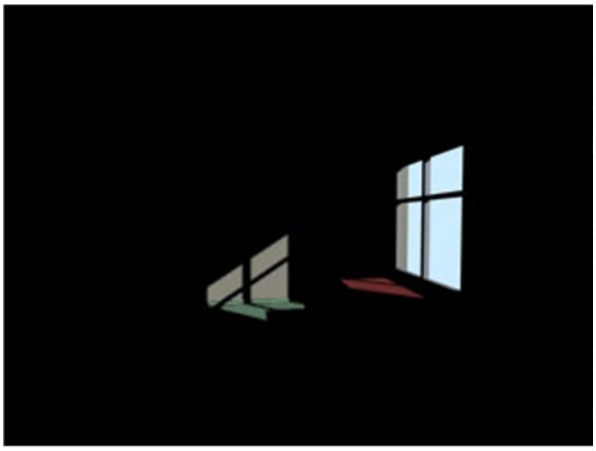
Illumination realism and complexity

One of the most difficult, yet important tasks in visual effects production is to illuminate virtual objects in a manner that is consistent with a real-world, studio-lit environment. Careful illumination is crucial to successfully merging live and virtual action, as lighting provides the visual "glue" that brings composite elements into a common visual setting. However, in the visual effects industry there are a few major obstacles that must be addressed to create successfully illuminated virtual objects. First, the global nature of light transport in the real world is often ignored in CG rendering. For example, the global illumination algorithms that accurately simulate light transport (path tracing, photon mapping, radiosity) are almost never utilized, as the industry almost exclusively uses the direct-only lighting component. Though this choice is a conscious one--allowing for more precise control over light-- it also greatly decreases the ease and realism of the lighting process. For example, whereas a typical scene in the real world may be illuminated by only a small number of light sources (i.e. the sun), many more lights are often needed to reproduce similar effects in directly lit CG environments (as seen in Figure 2.10).

full global illumination
(one light source)



direct lighting
approximation
(one light source)



direct lighting
approximation
(ten light sources)

Figure 2.10: Motion picture effects are typically rendered using only the direct contributions of the scene lighting. Although this method allows for intuitive control over lighting, it does not easily lead to realistic results. Compared with the global illumination solution (top), the directly illuminated scene (middle) is grossly inadequate. Only through the careful placement of ten lights (bottom), are we able to approach the desired look. Global illumination computed in Lightscape, images courtesy of Michael Scholz.

Another major challenge in lighting virtual objects for film is that simply recreating the lighting of the physical environment is not sufficient to yield acceptable results. One needs to consider that the real environment is not an arbitrary environment, but likely a precisely crafted, carefully lit studio location. The physical studio lighting is often highly manipulated, as it serves many purposes in the final image: to direct the viewer's eye, create depth, convey the time of day and season, to enhance the mood, and to reveal the character's personality and situation [Lowell99]. Virtual objects must be lit in a manner that is not only appropriate for the environment, but also serves to reinforce these lofty goals. It is thus apparent that designing the synthetic lighting for compositing virtual objects into feature film environments is a difficult and time-consuming process, yet critical to producing high quality composites.

The first step in illuminating virtual objects in a realistic manner is to use virtual lights that are motivated by light sources from the physical set. By recording the positions, intensities, and colors of the physical lights one can use this information as a starting point from which to illuminate the virtual objects. Also note that animated light sources (such as the flicker from a candle, or a lightning strike) need to be accurately reproduced in the virtual environment to create a convincing composite, as seen in Figure 2.11. However, recreating the physical studio's lighting simply serves as a starting point for the process. Once a good "look" is achieved, additional lights will need to be added to bring out the character (typically a rim light), or to guide the viewer's eye to another part of the image (typically through shadows).

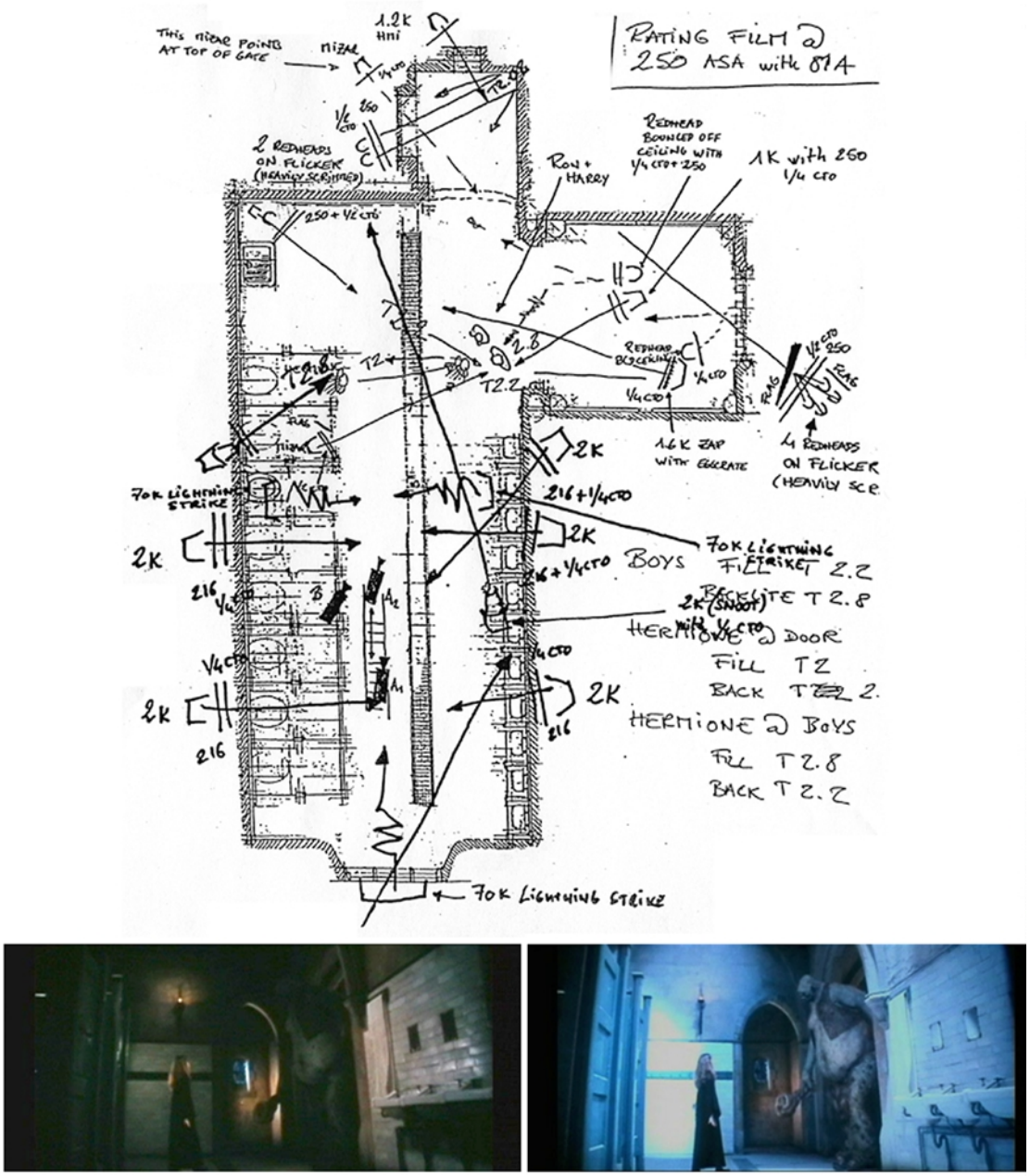


Figure 2.11: Effects artists go to extensive lengths to recreate the complex physical lighting on virtual characters. On top we see the chart used by the director of photography to record the position, intensities, and color temperatures of each light during the shooting of Harry Potter (2001). The key lights, fill lights, and rim lights can be easily identified. On the bottom we see the final composite of the scene: the girl and bathroom are real, while the troll is CG. Note that the illumination in the environment is animated, as lightning strikes (bottom right) are frequent. The virtual and CG lights need to be tightly synchronized to maintain the illusion of reality. Images courtesy of Sony Pictures Imageworks.

One new approach that has somewhat automated the process of matching lighting is to record two spheres (one mirrored, one diffuse) as placeholders for the virtual object during filming. Using the recorded images as the standard, one is then able to synthetically light the object, so that it matches the visual characteristics from the original scene (as seen in Figure 2.12). Typically, the white diffuse sphere is used to guide the diffuse lighting components, while the mirrored sphere is used to guide specular reflections. Furthermore, for highly reflective objects, the image of the mirrored sphere can function directly as a spherical environment map to yield highly realistic reflections. Spheres are an ideal choice to record the lighting for environments, as they encode the lit color for all possible surface normals visible to the camera. Furthermore, by using the exposed film as a guide, the artist is able to automatically account for characteristics such as film gamma, exposure settings, and colors gradings.

Figure 2.12: The computer can recreate the physical lighting by filming diffuse and mirrored spheres on the set (top). The white sphere records the diffuse shading at all foreword-facing normals, while the mirrored sphere functions as a guide for rendering specular surfaces. Images are from Stuart Little [Berney00].

To help artists quickly illuminate scenes in an intuitive manner, virtual lights are often employed that do not adhere to the physics of the real world. Though we have already shown that the lights employed in the effects industry are not fully realistic (as they typically do not consider indirect illumination), it is often useful to give them properties that are uniquely non-physical. For example, in real life all light sources cast shadows from occluding objects. However, in the realm of the computer artists can directly specify which lights, and which objects, are allowed to cast and receive shadows. Lights are even allowed to have negative intensities. That is, when these lights "illuminate" objects, the objects actually become darker than the rest of the scene! Although these example properties are clearly not based on a physical reality, their use can greatly

improve the ease at which a talented artist can achieve a desired effect. Below we see another extension to the traditional diffuse illumination model, one that allows for light to more easily "wrap" around the edges of objects. Wrap effects, often caused by rim lighting, are very important in creating strong edge highlights around characters, allowing them to stand out from the background. An example of wrapped lighting is shown in Figure 2.14.

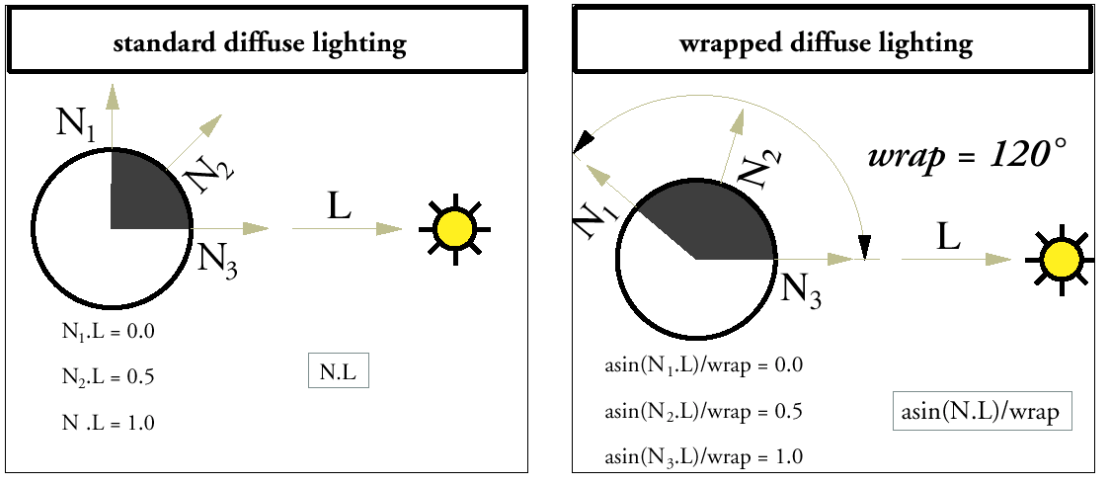


Figure 2.13: Standard diffuse lighting (from a point light) causes a sharp edge transition at the shading boundary between light and dark regions. By modifying the illumination model to allow for light to "wrap" beyond an object's silhouette, a softer shading falloff is created. Although this technique is not physically based, it does simulate the shading effect that a true area light would have caused. This model was used extensively on Stuart Little to help create strong edge highlights. Images from [Berney00].

Figure 2.14: The extensive use of rim lighting helps to create a sense of depth in the scene, and to separate the hero from the background. The lighting effect is motivated by cinematic principles, and would not have been present in the physical environments. Images from Stuart Little (1999).

Recreating Realism Through Compositing

Looking back to our model of the visual effects pipeline (Figure 2.1), when we are finished rendering the computer generated elements we need to merge them with the live action. Although the compositing process is largely transparent, by adding additional layers that simulate the physical visual interactions between layers, we can add another level of realism. Specifically, in the real world objects in a common environment interact by occluding, reflecting, or casting shadows on one another. Only by adding these additional elements into the compositing process can a truly realistic result be created. Furthermore, during the compositing process additional layer properties (tonal balance, histogram adjustments, highlights) can be modified in real-time, allowing for a final artistic touch-up on the overall sequence.

Synthesizing the proper interactions between composite elements is absolutely critical in making the final composite appear completely realistic. For example, if we consider only shadows and reflections, the virtual objects should cast shadows on real objects, and visa-versa. Furthermore, virtual objects should be reflected in the real objects, and visa-versa. If one does not appropriately account for these interactions during the compositing process, the realism will be lost.

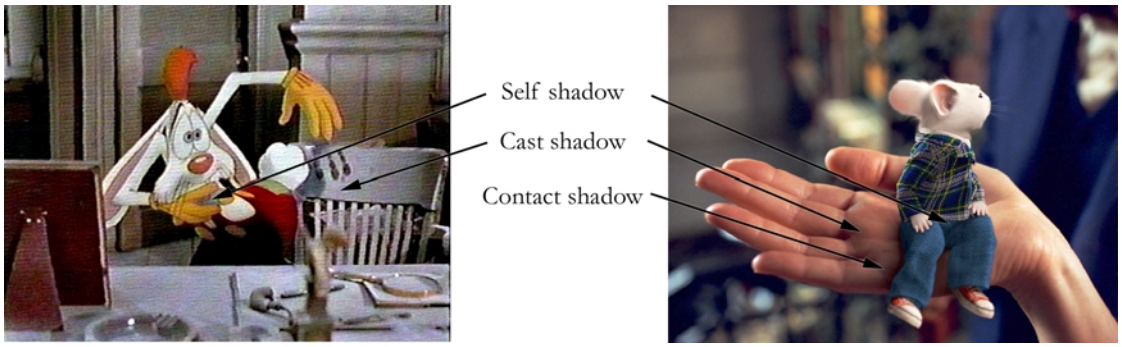
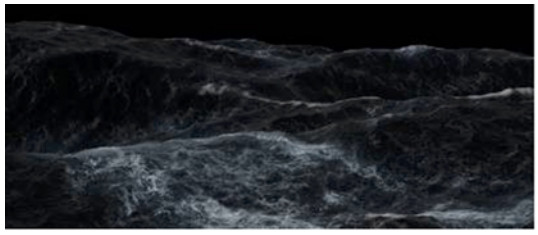


Figure 2.15: Adding the interactions (shadows / reflections) between elements are crucial to realizing a successful composite. Observe on the left that the cast shadow from the rabbit's hand serves as a visual "glue" to place the object into the live environment. Without the shadow, the rabbit would appear to float at an arbitrary depth behind the table. Similarly on the right, the cast and contact shadows serve to anchor the CG mouse to the actor's hand. Images from Who Framed Roger Rabbit (1988) and Stuart Little (1999).

One of the most important aspects of cinematic effects compositing is rendering with discrete, object-based image layers. For example, in Stuart Little individual layers were rendered for the ears, eyes, nose, whiskers, cloth, hands, muzzle, chin, shadows, reflections, and highlights! Separating the layers during the rendering process has two major benefits. First, if the director wants to modify a particular element's impact in the scene (such as reducing the intensity of a single shadow), the change can be made in real-time. Secondly, if the director decides that a part of the image had to be substantially

changed (requiring another rendering computation), only a small portion of the geometry would have to be rendered, greatly speeding up the editing process.

Other benefits from multi-layer rendering include being able to adjust the colors of each layer independently, and to apply per-layer depth of field and motion-blur effects. Because of the flexibility of the compositing process, virtually all modern motion picture effects are rendered to multiple layers, and only composited at the final stage in production (Figure 2.16). The only drawback of the technique is that data storage requirements are expanded. Although the storage requirements do not increase linearly with the number of layers (due to windowing optimizations and run-length encoding), using vast numbers of layers will increase the storage requirements. However, with large recent advances in hard disk space, this is becoming less of an issue.



Computer Generated Ocean

Computer Generated Boat

Computer Generated Fog

Computer Generated Fluid Flow

Final Composite

Figure 2.16: The Perfect Storm (2000), like most modern movies, rendered digital effects using multiple layers that were composited to create the final image (bottom). This process allowed the director a great amount of artistic flexibility at mixdown, as the composition, color, and interaction of each element could be quickly modified. Although only four elements are shown, many more were used to create the final shot.

The Inverse Problem: Real into Virtual

When compositing computer-generated elements into a real environment, the task is fairly straightforward, as one only needs to match the characteristics for a single object. However, the inverse task of merging live action into a fully CG environment is much more difficult, as one needs to recreate a fully photo-realistic environment from raw geometric elements. Furthermore, as motion-picture special effects typically occur in the context of traditional (live action) shots, it is often necessary to match the synthetic environment to the real version to which the audience has already become accustomed. This is a daunting problem, as a live character in an "almost real" synthetic environment will surely look out of place, and destroy the continuity of the film. To cope with this restriction, the motion picture industry typically leverages algorithms that simplify the process of recreating photorealistic environments.

Figure 2.17: Live footage (left) was composited into a synthetic environment (not shown) to generate the final composite (right). The synthetic environment was generated using an image-based reprojection technique, where multiple photographs of the physical environment were reprojected onto rudimentary geometry. This image-based rendering method is often preferred in effects production, as it allows for the precise correspondence of the physical and synthetic sets. Images from The Matrix (1999).

The most common approach to generating fully photorealistic reproductions of environments is to use image-based modeling and rendering techniques, whereby a virtual scene is reconstructed from a series of photographs. This approach has many benefits. First, image-based rendering decouples the geometric scene complexity from rendering performance, allowing photorealistic sets of arbitrary complexity to be rendered in reasonable amounts of time. Second, by leveraging image-based techniques that use the physical set as image inputs, effects artists can ensure that the final CG environment will accurately match up the original set. A number of movies have featured this approach, including the Matrix (1999), Harry Potter (2001), and Spiderman (2002).

Finally, we must observe that most, if not all of the techniques that are used in the motion-picture industry to merge live and synthetic imagery do not work in real-time, often taking hours to compute single images. In the next chapter, we focus on how the television industry attempts to solve this problem in real-time, and what limitations they accept to achieve sufficient results.

Chapter 3

Merging Live and Synthetic Imagery in the Television Industry

With the exponential increase in processing power, it is now possible to economically combine live video with synthetic imagery to create a virtual set. In essence, virtual set technology incorporates all of the compositing processes used by the film industry, along with camera tracking and real-time rendering. With this union of technology, one is now able to simulate the functionality of a full-fledged television studio entirely within the computer.

Figure 3.1: The virtual set is generated in real time and composited with the actors. Left: Video pre-composite element, Right: Final composite image. Images from WTTW Chicago, using a Vizrt hardware solution.

Virtual sets offer many advantages over traditional sets, which is why their popularity has been increasing in recent years. First, virtual sets can be designed and modified faster than real sets. For example, if a news program wishes to update the design of their physical set, the change would likely be time consuming and expensive. However, using virtual sets the design and modification is trivial. It has even been estimated that the cost of implementing a typical virtual set is one-tenth the cost of designing and building its real world equivalent [Kogler98].

Virtual sets also offer significant space savings. The physical spaces used to record virtual set programs are typically small environments, even though the final images can have the illusion of being shot in a large space. However, with traditional live television broadcasts the space requirements are often much greater.

Finally, although purchasing virtual set hardware is typically a greater investment than designing a single live set, the cost can be amortized across multiple shows and seasons, as the same green screen environment is versatile enough for all of their use. Storing multiple virtual sets as easy as purchasing additional hard drive space. Furthermore, using a single virtual set environment for multiple programs is additionally cost effective because of reduced space requirements, as traditional live television shows typically require dedicated studios.

However, despite the cost-savings of virtual sets, to date there has not been widespread adaptation by the mainstream networks. There are two key reasons for this. First, the image quality is not sufficiently high for the television networks. Most virtual set systems commercially available today have a "synthetic" look that is more fitting for computer games than live action. This can be partially explained by observing that most graphics

rendering hardware is designed with gaming in mind. Fortunately, the limitation of realism is a distinctly technological one and as faster rendering solutions become available in the next few years this obstacle will clearly be overcome. However, that still leaves the largest problem inherent in the technology. Virtual sets, by their very nature, greatly limit the interaction between the actors and their environment. Without the visual and haptic feedback that live environments naturally provide, virtual sets are likely to be used initially only in situations that require minimal interactions with the set, such as game shows and newscasts (see Figure 3.2).

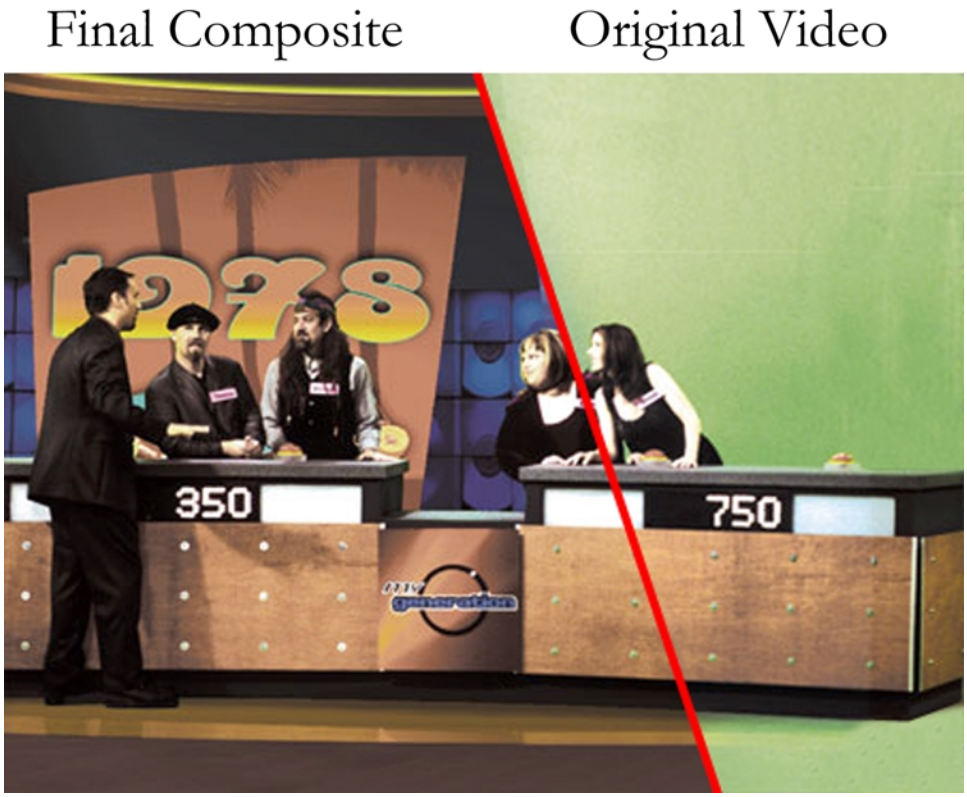


Figure 3.2: "My Generation", a short lived music quiz show on VH1, was the first game show to use a virtual set (1998). On the right, we see the actual scene as recorded by the camera. On the left, we see the final broadcast image, complete with the computer generated virtual set. Images courtesy of Vizrt.

When designing virtual set systems, real-time robust performance is a critical design consideration. If the system is not capable of keeping up with the demands of live television (60 fields/sec, at 720x486 resolution), it doesn't matter how good the image quality is—the system is not feasible! Therefore, when designing virtual set systems the rendering features are specified only after securing the stability and timing requirements. Very strict guidelines are placed on the complexity of the data path to guarantee the minimum frame rate. Scenes are typically limited to complexities well below those of gaming environments— perhaps on the order of a few hundred polygons for a single scene-- despite being based on the same graphics hardware. To cope with this limitation in geometric complexity virtual sets traditionally rely very heavily upon textures.

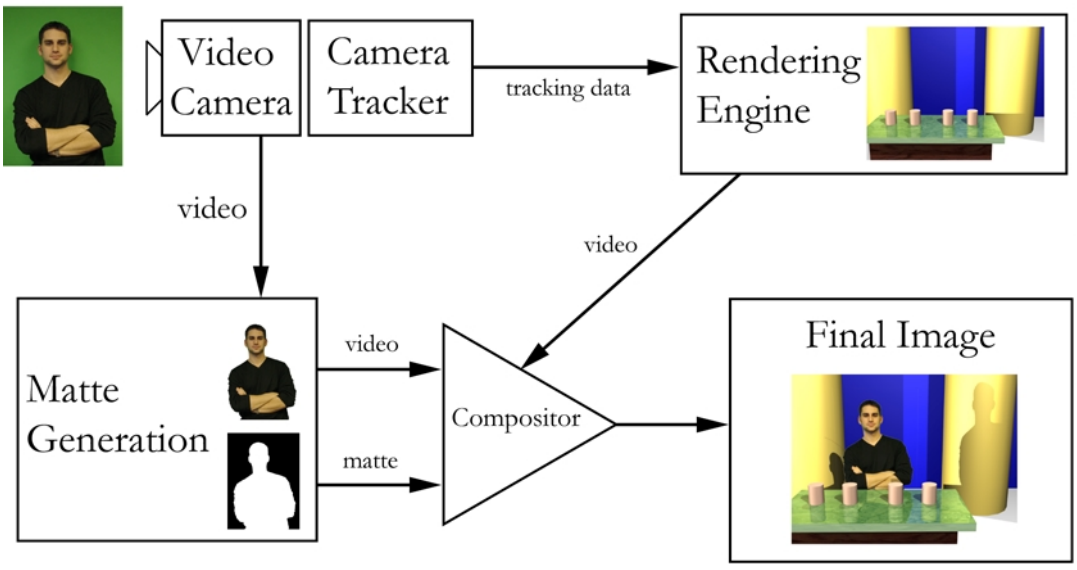


Figure 3.3: A traditional virtual set merges multiple video components, as shown above. A video camera captures the foreground subject against a single color environment. Using a form of camera tracking, the camera parameters are fed into a rendering subsystem that generates the synthetic background image from the correct perspective. A compositing system then places the live video feed "into" the rendered environment, maintaining a steady rate of 60 fields/second.

Because of the strict timing and reliability requirements, most current commercial virtual set systems are not software based, but tend to rely upon dedicated hardware components. Traditional virtual set implementations are all based around the following sub systems: camera tracking, matte generation, real-time image generation, and compositing. By examining each of these, we can survey common techniques that have emerged in the field.

Camera Tracking

To properly merge live action with a synthetic set, the set must be rendered from the correct perspective—the same perspective viewpoint as the original live camera. If the live video camera changes location and the synthetic background does not accurately follow, the video will appear to "float" over its environment, and the illusion of the virtual set will be destroyed. It therefore becomes necessary to accurately track the motions of the live camera with minimum latency and maximum accuracy.

There are many approaches towards camera tracking, and they all attempt in different ways to achieve the same goals: precision and accuracy. The accuracy needed from the tracking devices is substantial, and is therefore typically the most complicated, and least reliable component of the virtual set. At high zoom levels, a television camera will capture approximately five degrees of the visual field [Gibbs98]. Therefore, to keep jitter at pixel level accuracy, angular joint measurements need to be within .006 degrees. Considering that most mobile camera elements can rotate 360 degrees, this necessitates angular measurements that have error tolerances of below 0.002%! Errors in camera translations need to be held to similarly low values. To achieve these high levels of

accuracy along with robustness, many virtual set designs often employ multiple methods of camera tracking simultaneously, and use combined results to feed the rendering engine.

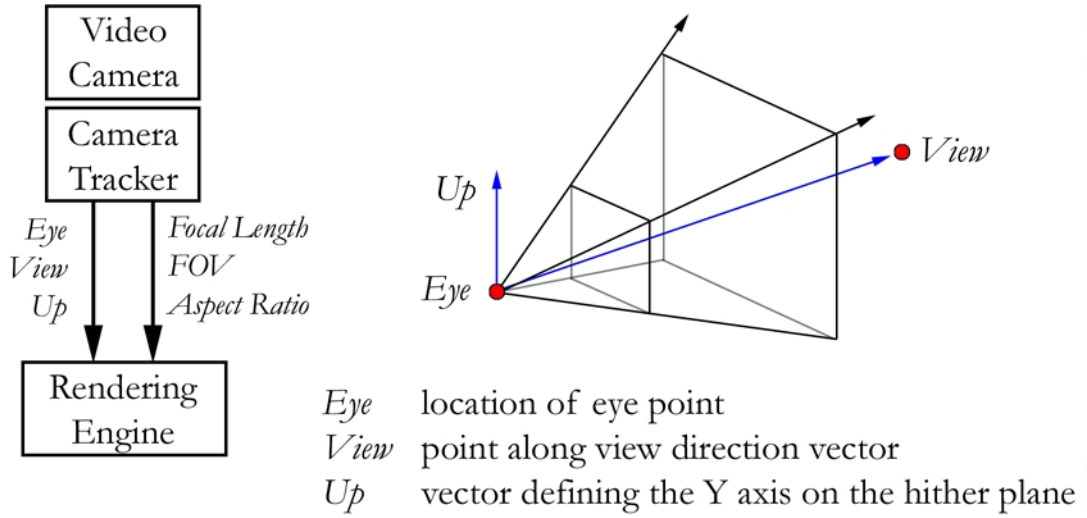


Figure 3.4: A real-time tracking system forwards to the rendering engine the physical measurements that define the camera's view frustum. In addition to recording the eye, view, and up vectors, the camera's aspect ratio, field of view, and focal length are necessary to completely recreate dynamic camera movements.

Ideally, for the computer to render the proper image from the camera's perspective, it would need to know all of the parameters from the camera's lens system. However, in practice it is common to only record the critical subset of the camera's parameters. First, the location of the camera's focal point in three dimensions needs to be measured. Next, the camera's orientation (also three degrees of freedom) needs to be recorded. Finally, information about the lens' zoom and focus is necessary. Although additional information such as iris settings, lens distortions, and white balance could be recorded, the current generation of virtual sets has no way to deal with this information, so they are typically ignored.

Mechanical / Actuated Camera Tracking

The first camera tracking systems that had the accuracy necessary for virtual sets were mechanical armatures connected to the camera, using strain gauges to measure the joint rotations. The rotation angles could then be transformed into the camera location by performing the vector sum of the joint lengths at the measured angles. These systems were offshoots of the first visual effects motion controlled cameras of the 1970s [Gibbs98].

The greatest advantage of these systems is that they have very high accuracy along with very low tracking delays, because the measurements are direct electromechanical measurements of the state of the system.

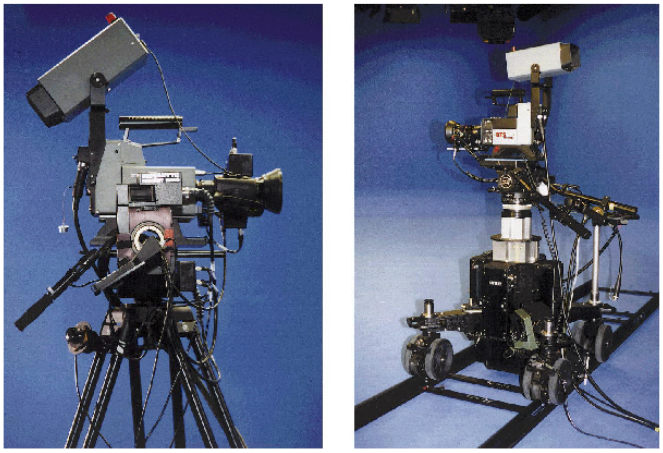


Figure 3.5: The Ultimatte memory head (left), with .01-degree angular resolution, was one of the first camera tracking systems to have the accuracy necessary for use on virtual sets. A Thomas System (right) is a mechanically actuated tracking system used to record eight degrees of freedom (three rotation, three translation, focus, zoom). Images courtesy of [Gibbs98].

However, mechanically based tracking suffers from limitations as well, which has led to the recent adoption of other systems. The largest limitation is that the cameras need to be physically linked to bulky devices at all times, which inhibits their maneuverability. These systems are not well suited to work over large areas, or in close, confined spaces. Another limitation is that although the devices are very accurate, they need to be carefully calibrated in order to maintain their precise relation to the computer's frame of reference. For example, if the camera were to "settle" into its mount even a little during use, the tracking data would be useless. Finally, most television productions use multiple cameras, which would require the large investment of purchasing multiple camera tracking systems.

Radio Wave Tracking

These tracking devices work as two components: a large, stationary magnetic coil generates a precisely defined magnetic field over the set, and smaller receivers detect these waves and use them to determine their positions and orientations. This approach overcomes the issue of physical flexibility that mechanical systems lack. Typically, the receivers are quite small and can be mounted on existing cameras, allowing flexible camera usage, even in hand held modes. Another major benefit of their small size is that these systems are well suited to a multiple camera environment.

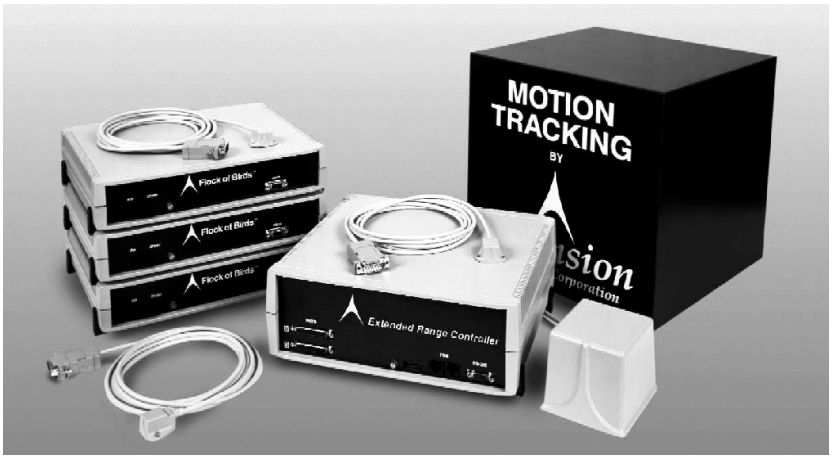


Figure 3.6: The "Flock of Birds" tracking system by Ascension Technology uses pulsed DC electric fields to track receivers through six degrees of freedom. Multiple receivers can be used simultaneously, allowing for very sophisticated tracking schemes. The typical range for this device is three meters.

However, these systems also have their own particular limitations. Metal objects or CRTs in close proximity to the detectors can interfere with the accuracy of the system, as they modify the magnetic field in their vicinity. Furthermore, the range of such systems is fairly limited, typically on the order of three to five meters. Once again, as with the mechanical tracking devices, over long periods of use (between calibrations) the system can drift, slowly decreasing the accuracy of the measurements.

Visual tracking

A variety of tracking systems have been developed based upon the idea of using high resolution charge coupled devices (CCDs) along with image processing techniques to track locations [Bishop01]. One of the most successful of these devices has been the HiBall Optical Tracker from the University of North Carolina [Welsh99]. This system requires installing an array of commodity infrared LEDs in the ceiling, which are individually pulsed at a high frequency (over 2khz) using a computer based controller.

Receiver modules consisting of six lenses in a dodecahedral configuration, six photodiodes, and a DSP processor record the distinct flashes and then use a single-constraint-at-a-time (SCAAT) algorithm [Welch96] to determine the pose (location and orientation) of the module in real-time.

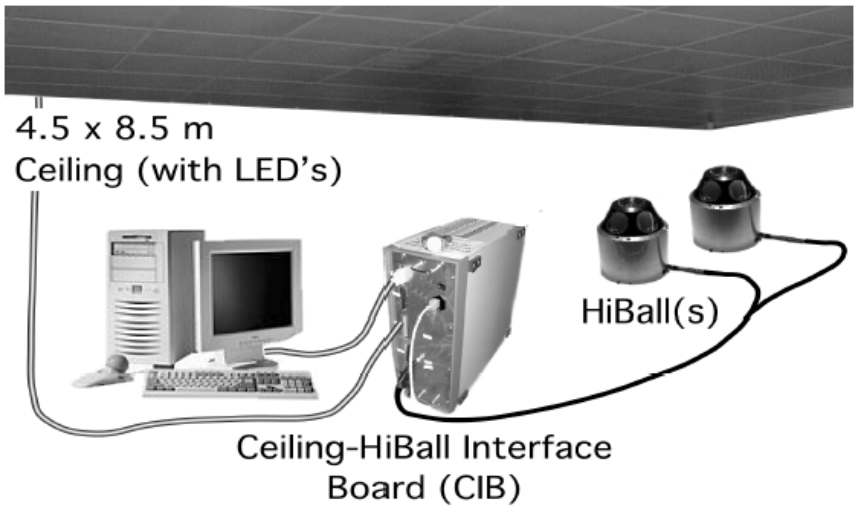


Figure 3.7: Ceiling mounted LEDs used in conjunction with a receiver module track the location of the hiball devices. The system uses an iterative single constraint at a time (SCAAT) algorithm to refine the location of the ceiling LEDs individually, allowing for very precise tracking along with relaxed setup requirements. Image courtesy of [Welsh99].

One of the advantages of the system is that throughout the measurement process the hiball tracking device continually updates its internal map of the LED positions. Therefore, even if the ceiling tiles shift or settle during usage, the system will quickly adapt to the new configuration. In this way, "drift" errors have been eliminated.

The other major advantage of this system is that as the ceiling tiles are inexpensive, the tracking space can be extended to practically arbitrary sizes, without a decrease in resolution. Spaces were tested up to 40'x40' feet, allowing for approximately ten times the area allowed under the radio based systems. Furthermore, this type of device has constant

precision over the tracking space, is largely unaffected by magnetic fields, and is largely immune to line-of-sight obstructions.

The system does however have its drawbacks. At the present time, the hiball tracker is limited to an angular resolution of .02 degrees [Welsh99]. Although this is sufficient for most virtual reality applications, this is not precise enough for virtual set use. The system is also currently limited to work only in indoor environments with moderate light levels, as the camera must be able to easily spot the LEDs. Finally, due to its low volume production, the receivers are expensive even when compared to other tracking systems. However, the technology is young, and as it becomes a commercial product and enjoys the cost reductions and stability increases associated with commercial production, it is likely to supplant other tracking techniques (such as radio).

Image Analysis

The most recently introduced tracking technique is revolutionary, as it completely does away with physical tracking solutions and instead uses an elegant, image analysis approach. Orad Technoloigies has created a system [Orad97] that replaces the traditional single colored screen with a specially designed, non-repeating patterned backing. The screen is composed of a grid of two colors which are similar enough that it is still possible to generate a good matte, though different enough that one can use their pattern to capture geometric information about the scene. The grid pattern in the background signal in analyzed using custom hardware to determine all of the camera parameters (location, orientation, and field of view and focus), from the single video frame, in real-time.

Figure 3.8: (Left) A non-repeating grid pattern upon the blue screen allows Orad hardware to track camera movements using a custom algorithm and a dedicated DSP solution. The tracking is precise and robust as long as a minimal section (three intersections) on the background is visible. (Right) the Orad DVP-300 performs the signal processing necessary to extract the real-time geometric information from the camera feed, and even fits in a standard 19" rack unit.

This new technique is extremely powerful because it rids the virtual set of all imposing tracking hardware, and allows studios to use the cameras they already own. Furthermore, a single video processing device can be used with an arbitrary number of cameras, simply by switching the input feed to the video processor at different times. One other major advantage is that the tracking data always has perfect synchronization with the corresponding video frame, as it is based that very same data. With other systems, namely IR and magnetic, the recorded positional information often does not exactly match existing frame boundaries as they are run asynchronously. Finally, with this system there is no degradation of tracking information (drift) over time, as the system continuously calibrates itself.

However, the Orad system still has some fundamental limitations. The largest limitation is that the background tracking features must be visible at all times to produce a usable result. This means that the camera operator has to pay attention to maintaining contact

with the blue screen environment at all times. If the camera operator were to choose an extreme close up, thereby obscuring the background completely, the system not have enough information to continue, and would stall. Additionally, due to the large computational requirements associated with real-time image processing, the tracking system introduces a delay of three fames into the video feed. Though this is not excessively long, in live interaction settings (such as remote interviews), this delay can be problematic.

In the future as computational power becomes cheaper, and as television studios begin using higher resolutions, the accuracy of the system will increase and the cost will come down. At that time, this type of system would appear to be a near ideal solution as it has great accuracy and few other limitations. However, until that time comes most professional virtual set solutions will likely use this system along with another tracking device, working in tandem to produce a single, robust stream of data.

Matte Generation & Compositing

The matte generation, and corresponding compositing operation is the process where the different layers, the foreground video signal and the background CG environment, are finally brought together into a single image. (See Chapter 1) Though creating high quality mattes has traditionally been difficult, the devices of recent times have sufficiently improved from the bad compositing of the 1980s such that the issue of quality is now largely moot. However, a few recent technologies are notable enough to require inclusion in a survey of the area.

Figure 3.9: Modern matte generating devices produce very high quality results. Observe in the insets that the compositing process correctly accounts for the notoriously hard features of fine hair silhouettes and reflections. Without prior knowledge, one would be hard pressed to judge the lower image as a composite. Top: Original blue-screen footage from "The Saint" (1999), Bottom: Final Composite. Images From [Rickitt00].

Ultimatte

Ultimatte is a patented product, and it is the most common technique used to generate mattes in television as well as motion pictures. It works on principles similar to chromakeying, though it goes well beyond that from an algorithmic point of view. A traditional chromakey device functions by rendering specified colors transparent. The process is fundamentally binary and usually results in hard, unrealistic edges. The Ultimatte process however, designed by Petro Vlahos [Ultimatte02], addresses the observation that edge pixel colors are typically a combination of the foreground and background. Although recent versions of the Ultimatte hardware add additional features such as the ability to better deal with film grain and blue spill, the basic principles of the

Ultimatte device approximate those of a linear-keyer, the functionality of which will be described in a later chapter.

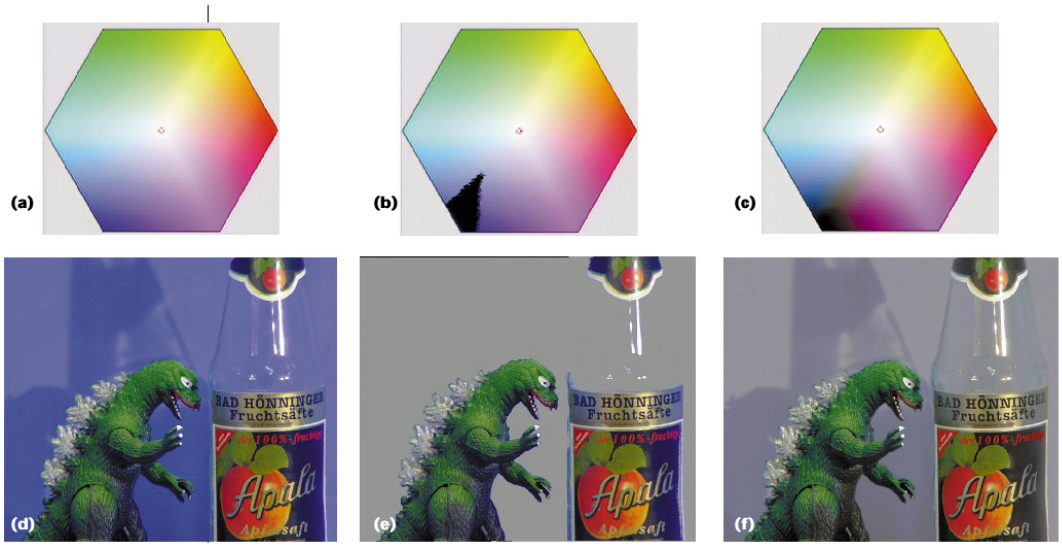


Figure 3.10: Example of Chromakey (binary keyer) vs. Ultimatte (a linear keyer) Left: Original image and color gamut. Center: Results of Chromakey. Note that the process has produced hard edges to the matte, resulting in undesirable artifacts. Right: The Ultimatte linear keying process correctly accounts for transparent objects, edge antialiasing, and blue spill artifacts. The linear properties of the algorithm are observable in the color wheel (as demonstrated by the smooth gradient). Images courtesy of [Gibbs98].

Though the results of modern Ultimatte hardware are impressive, there are a few downsides to the hardware. First, in order to acquire a high quality matte the green screen (or blue screen) must be lit in a highly consistent, accurate manner. This is often very difficult to achieve, considering that the studio is usually lit to artistically illuminate the characters. To properly light the green screen, sacrifices in the artistic aspects of character lighting may need to be made.

Another problem inherent in the Ultimatte process is that light from the background can spill onto the foreground characters. For instance, if reflective surfaces (such as glossy

tables or people's glasses) were to accidentally reflect the backing color to the camera, the objects would appear semi-transparent in the composite. This would no doubt destroy the illusion of the composite, as one would be able to see through the otherwise opaque table or human. Recent versions of the Ultimatte hardware address this issue as new algorithms have been incorporated that cancel blue spill, though the problem still causes trouble for lighting artists as it can change the hues of colors during the compositing operation.

LiteRing

The "Lite Ring" is a recent innovation from the BBC Research and Development team [Reflecmedia02] that attempts to address the difficulty of properly lighting a blue screen backdrop. Their innovative technique is to use LEDs in a ring around the camera lens to illuminate a gray mesh behind the actors. This mesh then takes on the color of the LEDs, and a traditional linear keyer is used to generate a matte. This technique has the advantage that the LEDs emit very low power so that practically no light from the screen is reflected back onto the set—eliminating the problem of blue spill. However, to use the technique one also must purchase a highly retro-reflective screen, so that almost all of the dim light aimed at the screen is reflected back into the camera lens.

Figure 3.11: The BBC's LiteRing system uses low-powered, colored LEDs to illuminate a mesh screen, which then feeds a traditional linear keying process. The integral part of the system is the gray mesh, which is made up of millions of very small glass beads. These beads make the screen highly retroreflective, allowing it to reflect almost all incident light back to its source, independent of the angle of incidence to the screen. This retroreflectivity cause the background to take on the color of the LEDs, observable only from the lens' location. Images courtesy of Reflecmedia.

This system also has other advantages. For example, as the retroreflective screen can overpower diffuse lighting, the screen lighting can be arbitrary, which further allows for the director of photography to focus on the task of lighting the actors. Additionally, switching from a green to a blue screen configuration is as simple as snapping on a new light ring, while with the traditional Ultimatte setups the process would be time consuming and expensive.

Depth Cameras

3DV's Zcam is a truly break-through technology, as it is the first camera that can simultaneously capture video information along with depth information in real-time in a robust manner. Based on LIDAR technology [Scott90], the ZCam uses nanosecond pulses of IR light to measure the depths to close objects (within 10m). Having the depth information, the Z-Buffer, for live video will allow for many new applications that go well beyond that of compositing, and enter the realm of special effects. For example, with depth information it is possible to reproject the camera's pixels from a new viewpoint,

allowing the viewers to interactively view a scene from angles different from what was recorded. Another possibility (as shown below) allows for virtual objects to light, or cast shadows on to real objects. This is not possible without accurate knowledge of the scene geometry, which the depth buffer provides. Should this technology be adopted and become less expensive, it will change the way people think about video footage, and vastly increase the functionality of virtual sets [Iddan02].

Figure 3.12: By acquiring the depth values for live scenes, the inserted virtual object can illuminate the woman's body and interact with her in a three dimensional environment. Image courtesy of 3dv.

However, as the technology is new, it has several limitations. First, compositing using only depth information to generate matte edges is fairly limited, as each pixel has a single, set distance. Therefore, edge pixels are be characterized as either foreground or background, but not combinations of both—as is required to produce smooth linear keys. Another limitation is that the depth values can only be recorded in the range of 1ft-30ft from the camera, with eight bits of precision. Though this will be increased in future versions, the limited range and precision makes the camera unacceptable for most outdoor use. Another limitation is that as it is emerging technology, each camera costs almost $500,000, making it prohibitively expensive for virtual set applications. However, with integrated chip technology and larger volume this technology should decrease in

price, as the only components are an infrared laser source, a video sensor, and control logic.

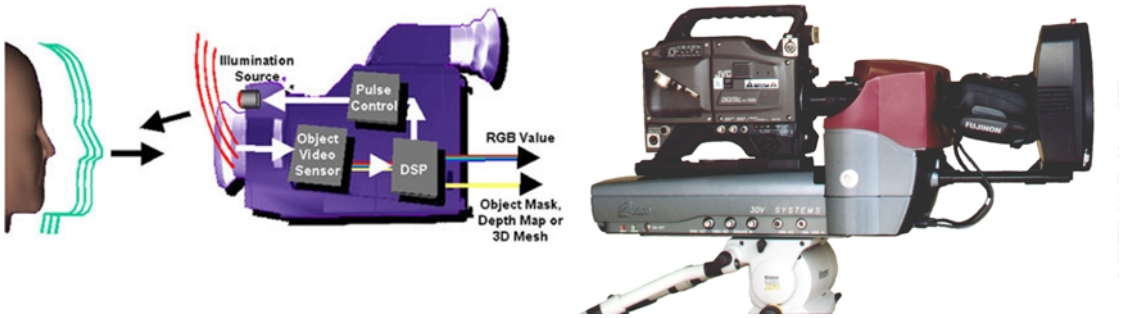


Figure 3.13: The ZCam, from 3DV Systems and JVC, uses pulsed infrared light detectors to capture distance information along the image plane, allowing the camera to transmit both a video signal and the corresponding depth buffer of a scene in real-time. Although this technology allows for the basic matting of objects without the need for a green screen, it also enables many more exciting avenues of research for virtual sets including image reprojection, and image relighting, and increased interactivity. Images courtesy of JVC.

As this technology is accepted in the future, virtual set compositing will be much more robust, and will possible eliminate the need for a green screen. Furthermore, the other opportunities this system offers-- relighting, interaction, and reprojection-- will provide much greater flexibility with the video data, allowing virtual set systems to generate realistic, coherent environments in ways not yet imagined.

Virtual Garbage Matte

The virtual garbage matte allows virtual sets and camera operators to film beyond the extents of the green screen, creating a true 360-degree world. This is accomplished by providing the geometric data for the actual studio to the rendering engine. This way, the rendering engine can automatically "key out" features that lie offstage.

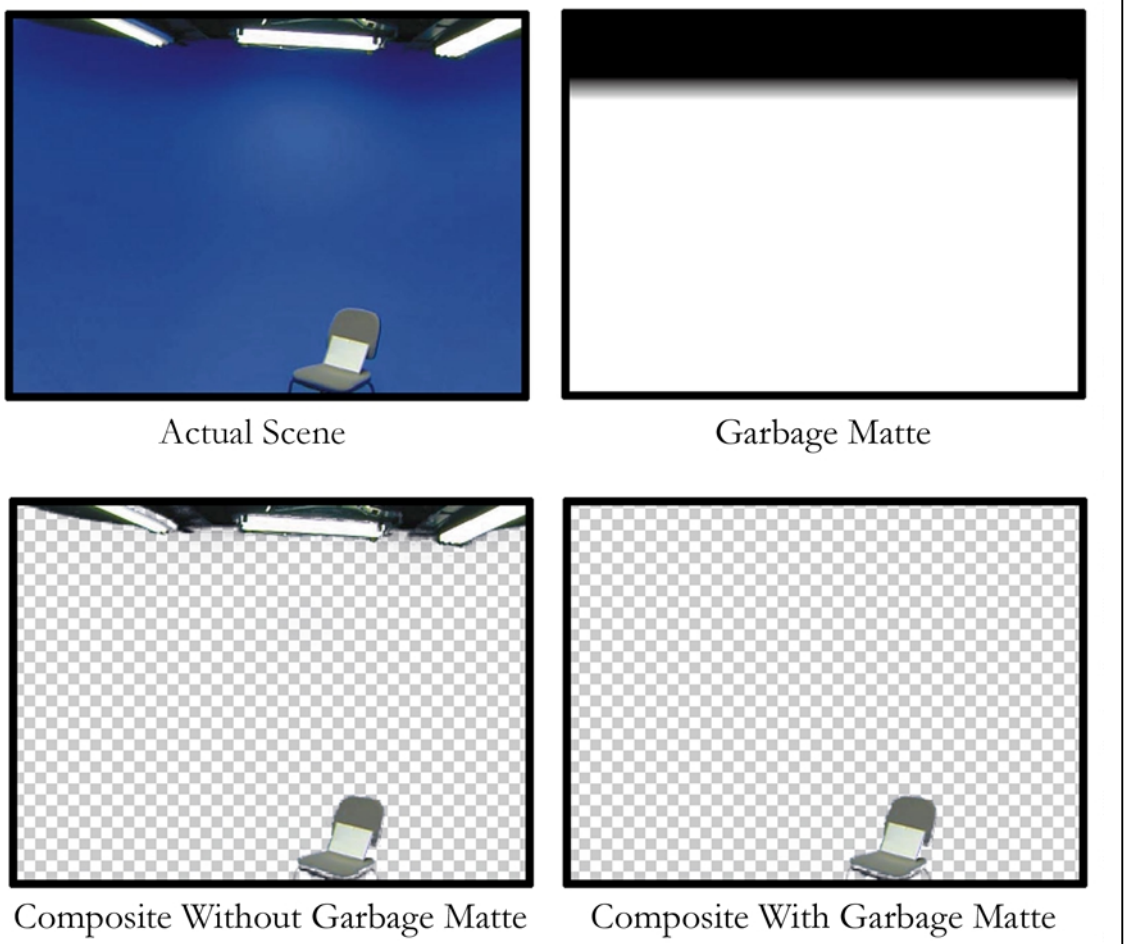


Figure 3.14: Note that the garbage mask (top right) removes the ceiling from being visible in the final precomposite (bottom right). Without the garbage mask (bottom left), the virtual set could not have been shot with this field of view, as the real set's boundaries would interfere with the matting process. Images courtesy of [Gibbs98].

This technique is very useful as it allows the camera operator to focus on composing a good shot, and not to worry about shooting off the edges of the green screen.

Real-Time Image Generation

As television cameras often have noticeable depth of field the rendering engine must be able to mimic these effects if the composite is to be realistic. However, accurately simulating depth of field, as described in [Potmesil81] is currently outside the realm of real-time graphics. Although modern graphics boards can often simulate depth of field effects using multi-pass rendering and an accumulation buffer, the results usually have noticeable artifacts unless a very large number of passes are used. This is not feasible for real-time applications. Therefore, a few methods have been designed to work around the current limitations of graphics hardware. For instance, BBC has introduced a hardware solution known as Dfocus, which performs a two-dimensional blur in real time based on the Z-distance information. Although their results are not physically accurate (one would need to convolve the high dynamic range image to produce the correct defocused image), the results are more realistic than without using the depth of field.

Figure 3.15: By changing the depth input, the DFocus system selectively blurs different parts of the visual field. In the left image the background is selectively blurred, while on the right the foreground is blurred.

Matched Lighting

As noted above, lighting virtual sets is an extremely difficult process, as it requires balancing the desire to evenly light the green screen with the necessity to recreate the lighting attributes of the target environment. However, achieving this goal and matching the lighting between the virtual set and the real set is crucial if one wants to achieve realistic results. As it is noted in the Ultimatte Guide to Lighting [Ultimatte02b],

> "Proper lighting is the key to realism in image compositing. Not only will it be difficult to make the Ultimatte function properly if the lighting is not right, but even a technically perfect composite will look phony with bad lighting."

However, most compositing research to date focuses on the technical aspects of compositing (getting rid of visual artifacts), with very little research addressing how closely the lighting must match the virtual environment to be considered "real". The next chapter addresses this issue.

Chapter 4

Perception of Lighting Errors in Image Compositing

When merging live action with synthetic imagery, one strives to create the illusion that the composited elements are part of a single, cohesive scene. However, errors in the compositing process, or inconsistencies in the source material can destroy the illusion, as shown in Figure 4.1.

Figure 4.1: Errors in the composite can reveal that the image is artificial. Left: Original photo. Center: A non-ideal composite, due to a "process" error; the colored silhouette around the object destroys the realism. Right: A non-ideal composite, due to a "source" error; gross lighting inconsistencies between the composite elements destroy the realism.

Although recent advances in compositing technology [Ultimatte02a] [Prim96] [Chuang02] have eliminated the majority of process errors, little attention has been paid to the characterization of source errors and their impact upon composite realism.

Unrealistic composites caused by conflicting sources are very common, and are most often found when the elements being combined have inherently inconsistent lighting characteristics. For example, if an object captured from a dimly lit indoor scene is composited into a bright, outdoor environment, it is unlikely that a believable result would ever be achieved. Although in the film and television industries there are rules of thumb for creating realistic results, there has not been a systematic study on the role that these errors play in compositing. A model of how source errors affect the realism of composite images would greatly facilitate the compositing process, helping to guide the realistic merging of live action with synthetic imagery.

In this chapter, we take the first steps towards such a model, based on the results of a series of psychophysical experiments. These experiments investigate the visual effects of four common lighting inconsistencies in image compositing: brightness errors, chromaticity errors, conflicting illumination directions and conflicting shadow directions. We studied these errors within two scenes representative of typical composites: a tabletop still life, and two people sitting at a desk. Our results show that the different types of lighting inconsistencies are not equally salient perceptually, and that the detectability of the errors depends in large part upon the subject matter of the scene.

Background

In the vision literature, there are psychophysical studies on the perception of objects and scene illumination [Blake00] [Beck59] [Gil77] [Mama98] [Pen82], but little of this work is directly applicable to the analysis of lighting errors in compositing. One exception is the work of Ostrovsky et al. [Ostrovsky01], which directly addresses the issue of directionality inconsistencies in images. Their experiment explored the task of searching for the inconsistently lit object in a cluttered scene. However, they failed to test natural environments and only focused on planar (as opposed to perspective) object distributions. Additionally, as they measured reaction times and not detection thresholds, their work is complementary to ours.

In the recent graphics literature, studies by Bolin [Bolin98], Myszkowski [Myszk01], Ramasubramanian [Rama99], and Watson [Wat01] address the use of perceptual metrics in rendering situations. Rademacher et al. [Rade01] address the novel goal of identifying the characteristics of images that cause viewers to identify scenes as being "realistic". Unfortunately, this work is not directly applicable to the problem of examining the errors in composite images, but merely serves as guides for the use of perceptual metrics in a graphics environment.

The most relevant recent work in matching illumination between virtual and real environments was presented by Debevec this year [Debevec02]. Although they sidestepped the perceptual issues of how closely the illumination of two environments needs to match to be considered consistent, they built a device that simulates arbitrary high-dynamic range illumination environments. By covering a sphere with 156 red, green, and blue LEDs, they were able to simulate the effect of variable, diffuse lighting in

arbitrary environments. Although they produced encouraging results, the work presented here somewhat discounts their assumptions, by suggesting that humans are not very good at detecting many properties of the illumination environment that their team worked so hard to reproduce. Specifically, although we agree that the precise color reproduction of environmental illumination is a crucial factor in effective compositing, we believe that only very rudimentary angular precision is necessary because of limitations inherent in the human visual system.

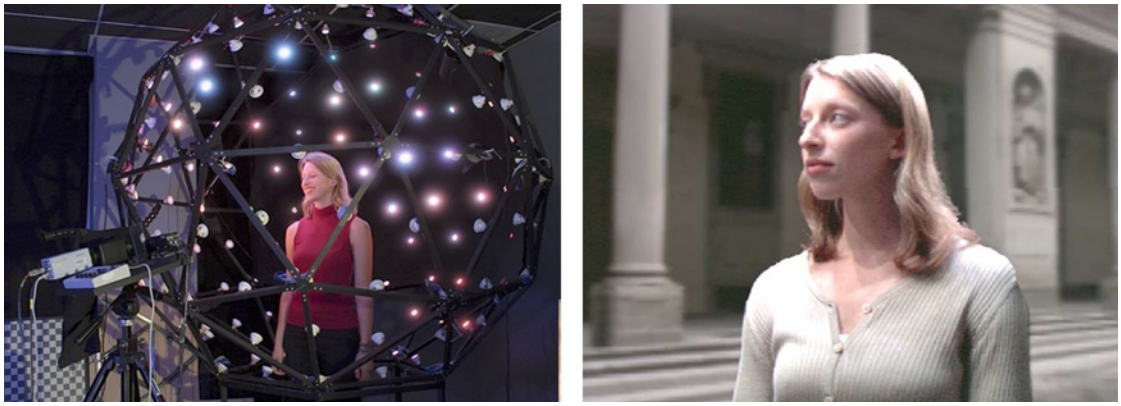


Figure 4.2: Debevec et al. have developed a light stage consisting of 156 RGB LEDs mounted on a sphere. The LEDs illuminate the subject in a manner that is consistent (both in color, intensity, and direction) with a pre-recorded high dynamic range environment. By accurately matching the lighting of an object to its target environment, highly realistic composites can be created. One of the pitfalls of the system is that specular objects reflect the 156 distinct point lights, often creating noticeable artifacts.

Recent advances in blue screen technology [Ultim02b] [Prim96] [Chuang02] have solved many of the problems traditionally associated with the technical process of compositing. For example, it has historically been difficult to generate high-quality mattes for objects that reflect part of the blue (or green) background. This problem is now solved automatically in many systems [Ultim02b], through a process known as spill suppression. In fact, a recent work by Zongker et al. [Zongker99] has even characterized the nature of this "blue spill", and can therefore accurately composite reflective and refractive objects.

Additionally, creating mattes for semi-transparent objects, such as smoke, has historically been difficult to accomplish. Once again, most current blue screen algorithms are able to do this flawlessly. One such algorithm, [Chuang02], is now even able to accomplish this using an arbitrary backing, by leveraging Bayesian estimation techniques.

Unfortunately, even as most matte generation techniques become more robust and accurate, most research is still ignoring the role that source factors have on the realism of the result. However, as compositing hardware gets better and better, the compositing hardware manufactures are beginning to realize that using coherent, realistic source material is quickly going to become the limiting factor in creating high quality composites. As such, the compositing literature has begun to identify some rules of thumb that tend to guide people towards realistic results:

1. Composite scenes that match exterior shots should be filmed outside. [Ultim02]
2. Discrepancies in skin tones between elements should be avoided. [Ultim02]
3. Matching the location of light sources is important for scenes with strongly cast shadows. [Brink99]
4. Color filters (gels) should be used to recreate the lighting color for scenes. [Brink99]

This chapter is a preliminary attempt to formalize these rules in a mathematical framework. Although we only tested two environments, further experiments would allow us to develop perceptual metrics to predict the saliency of different kinds of source errors.

Source Errors in Image Compositing

To create a believable composite, the merged elements must share common image characteristics, such that when brought together the result appears to have been created through a single process. Discrepancies in the original image characteristics, known as "source errors", can be categorized by their origin: the physical camera differences, incorrect scene interactions, or lighting differences.

Camera errors are typically associated with non-ideal image capture devices, and include film grain, color balance, motion blur, limited resolution, and lens distortions. These errors can be addressed in a straightforward manner, as most compositing software [Infero02] [Shake02] can automatically match these camera characteristics between image sources.

Scene interaction errors are harder to directly quantify, as they are associated with how a single object affects its environment. For example, real objects can cast shadows on their environments and can directly or indirectly illuminate their surroundings. To create a successful composite, the inserted element also must account for these interactions. Though currently an area of active research [Debevec98], these scene interaction errors are known to be of secondary importance to matching the primary scene illumination.

Inconsistencies in the lighting between composite elements can be placed into two categories: global pixel errors and cue errors. Lighting characteristics that can introduce global pixel errors are brightness and color differences between composite layers. With pixel errors, the image statistics of the composited element and the scene context are different, according to a uniform, histogram-modifying rule. Often it is possible to

correct for these discrepancies simply by changing the white point or exposure of the scene. Cue errors on the other hand, are caused by local inconsistencies of illumination between the elements. Sources of cue errors include local illumination / shading directionality differences, and cast shadow directionality differences. Cue errors preserve the image statistics of the composited region and the scene context, but the visual information about direction and qualities of illumination are contradictory. It is usually not possible to correct for these errors using common image manipulation techniques.

We chose to test the four light dimensions that we considered most likely to affect the realism of the result: brightness errors, chromaticity errors, shading directionality errors, and cast shadow directionality errors. These inconsistencies are representative of many situations encountered when merging live action with synthetic imagery. For example, in virtual set systems cast shadows are often generated as projected object silhouettes that do not correspond to physical light locations. By unlinking the shading and cast shadow directions, we can determine the perceptual relevance of this artifact, the results of which will guide the development of our virtual set algorithms.

Experiments

We conducted a series of experiments to measure the visual sensitivity to lighting errors in image compositing. Each experiment measured the participant's threshold for detecting one of four possible lighting error configurations. These measurements yielded a set of psychometric curves (one for each environment) that describe an average observer's ability to detect the compositing error.

Test Scenes

We measured the lighting error detection rate within two environments. The first scene was a tabletop still life consisting of two bowls of colorful vegetables over a kitchen backdrop. For the second scene we chose a standard television "talking heads" context (two people seated behind a desk), since many compositing applications involve placing humans into different environments. Samples of these environments are shown in Figure 4.3.

Figure 4.3: Tested composite environments. Left, kitchen still life. Right, library scene. Participants were asked to judge which object had been composited into the environment. The test objects were chosen so as to be similar enough to provide consistent perceptual cues in the environment, but dissimilar enough to prevent side-by-side pixel comparisons. In the library environment we initially considered using identical twins as subjects, but later decided against this approach, as participants were found to be skeptical of the trials—even in the unmodified control images.

In the experiments we asked the participants to judge which object had been composited into the environment. Thus, the implicit task was for the observer to compare both objects to the common environment, and to decide which object was a "better fit" for the scene. By modifying the lighting parameters for each bowl, we were able to measure the observers' sensitivity to the different categories of composite errors. However, we

also had to carefully control the environment to provide contextual cues, yet not appear overly "cluttered".

Stimuli

In producing the image sets used in the experiments, our goal was to vary the four error dimensions in such a way as to span the threshold from undetectable to detectable. We further chose to measure seven points (including the control location) on each axis.

Chromaticity: In modifying the light's color, we chose to hold the luminosity of the light constant, and to only modify the chromaticity. Though we could have chosen an arbitrary curve through the two-dimensional color space, we felt it appropriate to sample the color created by modifying the light's color temperature. In accordance with the laws of blackbody radiation, the seven points were sampled at chromaticity corresponding to: 2869K, 2972K, 3082K, 3200K (defined as white), 3328K, 3466K, and 3617K. Note that the control image exists in the center of the test space, with maximum errors located at both extremes. The full test space is illustrated in Figure 4.4.

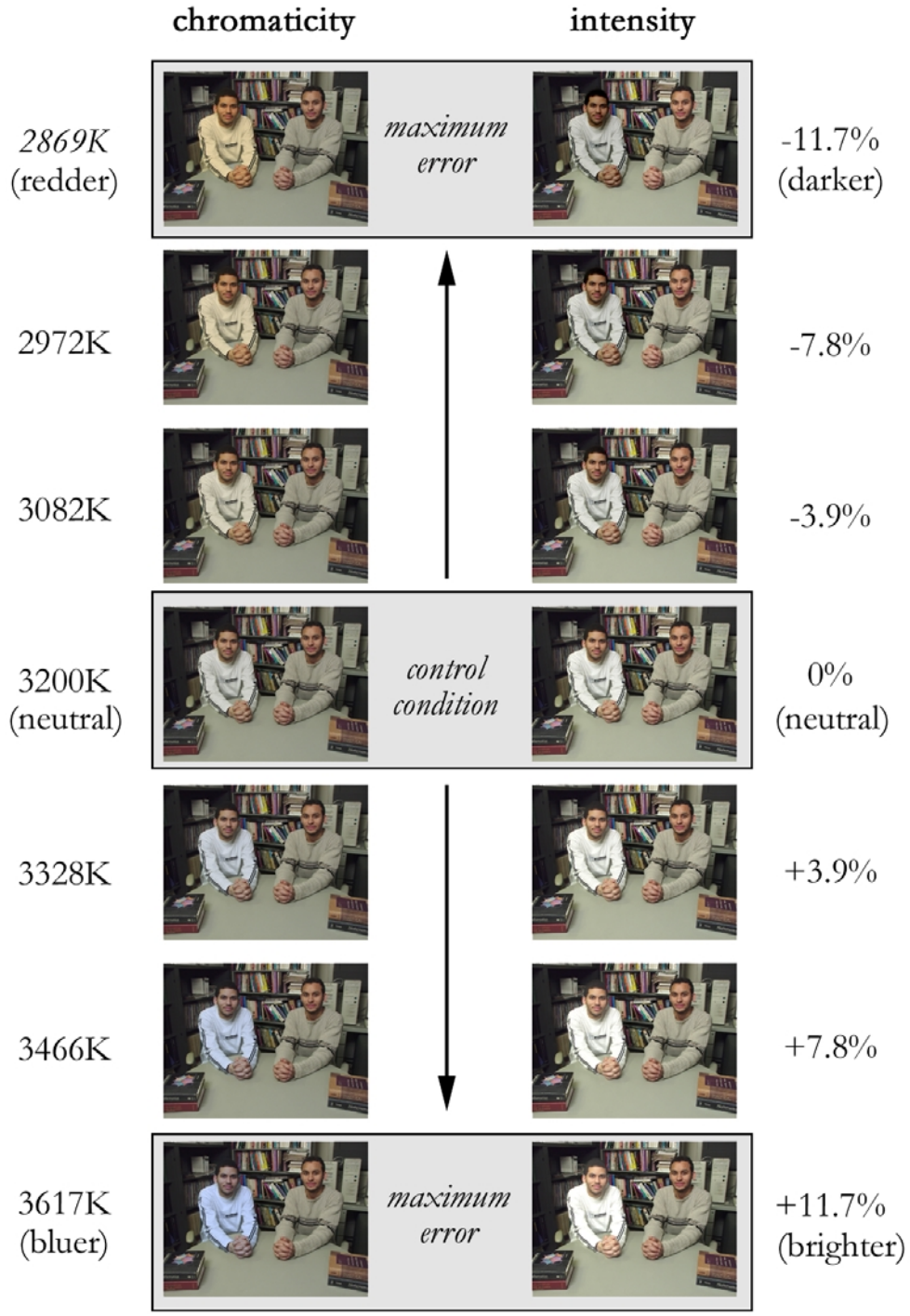


Figure 4.4: The above stimuli were created by perturbing the chromaticity and intensity of the illumination on the test subjects by increasing amounts. Although during the actual experiment the error condition was randomly associated with the either the right or left test subjects on a per trial basis, in this diagram the error is always shown on the left side. A parallel set of stimuli was also generated using identical error parameters within the kitchen environment. Note that the error gradient is double sided, with the central condition denoting the control case and the extremes denoting maximum error conditions.

Intensity: In modifying the composite's brightness, seven offsets of the black and white points were sampled, as shown above. The extremes were chosen to span the expected threshold values, with the complete measurements -11.7%, -7.8%, -3.9%, 0 (no change), 3.9%, 7.8%, 11.7%. Note that in the extreme darkening situation the black values were clamped, due to dynamic range limitations. No such clamping occurred with the white point as the original images had headroom. However, the clamping effect for the black point was not determined to be an experimental limitation, as this is a limitation of all eight-bit linear compositing software. Similar to the chromaticity modifications, note that the control exists in the center of the test space, with maximum errors located at the extremes.

Local shading direction: To modify the local illumination direction, we first captured the scene at the primary illumination direction (58 degrees left of the normal). We then replaced the test objects with angular variants in the illumination up to 115 degrees. Although we could have tested more oblique angles, we observed that beyond approximately 115 degrees the composite regions had excessive darkening, due to the glancing illumination. However, 115 degrees of error is a very large amount, likely to exceed even the largest angular mistakes.

Cast shadow direction: To create the cast shadow images, we used a process almost identical to that used in creating the local shading images. The only difference was that the regions falling within the cast shadows were replaced, instead of the regions within the local objects. The full test space is illustrated in Figure 4.5.

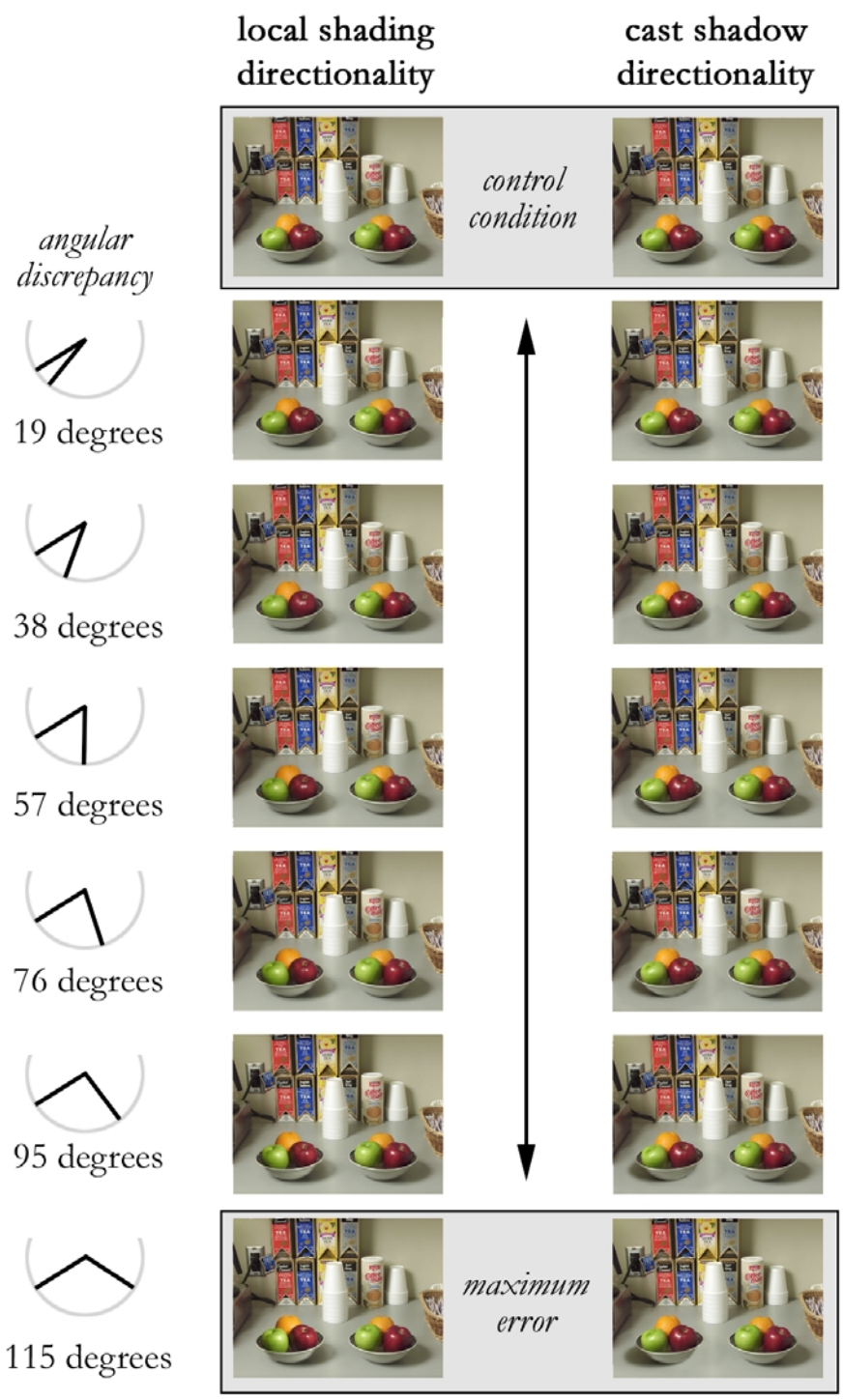


Figure 4.5: The seven variations within the local shading and cast shadow directionality error dimensions. Although during the actual experiment the error condition was randomly associated with the either the right or left test subjects on a per trial basis, in this diagram the error is always shown on the left side. A parallel set of stimuli was also generated using identical error parameters within the library environment. The inset arcs illustrate the angular discrepancy between the scene illumination and the error condition.

Eight sets of images were created, (two environments times four error dimensions), with each set of images containing both a left error and right error variant. With seven samples/dimensions, and an additional image (with the objects swapped) to serve as a left-right counterbalance, 128 images were made. All images were acquired using a Sony DSC-D770. The focus and exposure settings were held constant for the duration of the capture sessions. The images were transferred uncompressed at 1344 x 1024, and captured using an equivalent film sensitivity of 50 ISO. Our directional light was a Lowel Totalight tungsten halogen, rated at 3200K, with the optional bounce umbrella. Adobe Photoshop 6.0 was used for all histogram adjustments and image modifications.

Procedure

The experiments used a two-alternative forced-choice procedure (2AFC), with the two test objects next to each other in a common environment. For each image the viewer was asked to indicate whether the left or right object, appeared to be more realistic with respect to the scene context

Forty-six participants were tested, all college students, age 19-23. All were naïve to the purpose of the experiment, had no knowledge of compositing or psychology, and were generally non-technical majors. All had normal or corrected to normal vision.

All scenes were imaged on a Kodak XLS 8600 printer, a high-quality, thermal, continuous tone printer. The images were printed at 300 dpi on glossy paper. The participants sat at desks while taking the tests and freely viewed the images one at a time, at a comfortable viewing distance. The images were printed at 5.5 x 8.5 inches and subtended a horizontal viewing angle of approximately 16 degrees. Printer stimuli were tested (opposed to using

higher dynamic range monitor), as it was difficult to attract the requisite number of participants into a single test facility. Although a laptop could also have provided a common portable display, LCD technology currently does not produce as high quality imagery as printed glossy paper.

The presentation order was randomized between participants, across all error dimensions. However, the order was constrained such that the viewer never saw two stimuli from the same environment in a row. For each error condition, two test images were created, one with the error on the right, and one with the error on the left. Each participant was presented with a random set of right error/left error images, such that the average error was 50/50 left vs. right. This acted as a counterbalance towards preferentially choosing the right or the left sides. Finally, to counterbalance the possibility that the viewers may prefer a particular stimulus, control images were added to the experiment where the objects were shown in their reverse positions.

Each session lasted approximately 15 minutes, and afterwards the participants were debriefed and questioned about their general views about the experiment.

Analysis

The raw data was analyzed using Matlab 6. The left/right error conditions were merged when it was determined that there were no significant differences ($p<0.05$) between the sets. To test for significance, the Yates Chi-Square measure was used. The standard Chi-Square measure was not appropriate as the records for particular entries were low (<5) at times.

Results

The following summarize the results of the experiments. There are separate sections for each of the types of lighting error studied and within each section results are presented for tests of the kitchen and desk scenes. The data from each of the conditions of the experiments are illustrated in the graphs that follow. In each graph the abscissa indicates the magnitude of the lighting error, and the ordinate indicates the percent of trials on which the observers correctly detected the composited object. Detection rates range from 50% (pure chance in a two-alternative forced choice procedure) to 100% (perfect detection). The detection threshold is indicated by the upper dotted horizontal line and was determined using the Yates Chi-Square measure ($p<.05$). The lower dotted line denotes the pure chance (null hypothesis) detection rate.

Chromaticity Changes

Figure 4.6 shows the results for errors in the chromaticity of the composited object. On the left we see that in the library environment, observers were able to reliably detect the composited element when the color temperature was shifted by one stop toward the blue (temp>3328K), however the color temperature had to be shifted by three stops toward the red to be detectable. These effects were significant at the $p<.0001$, $\psi^2(1,230) = 14.12$, and $p<.01$ $\psi^2(1, 230) = 5.96$ levels respectively.

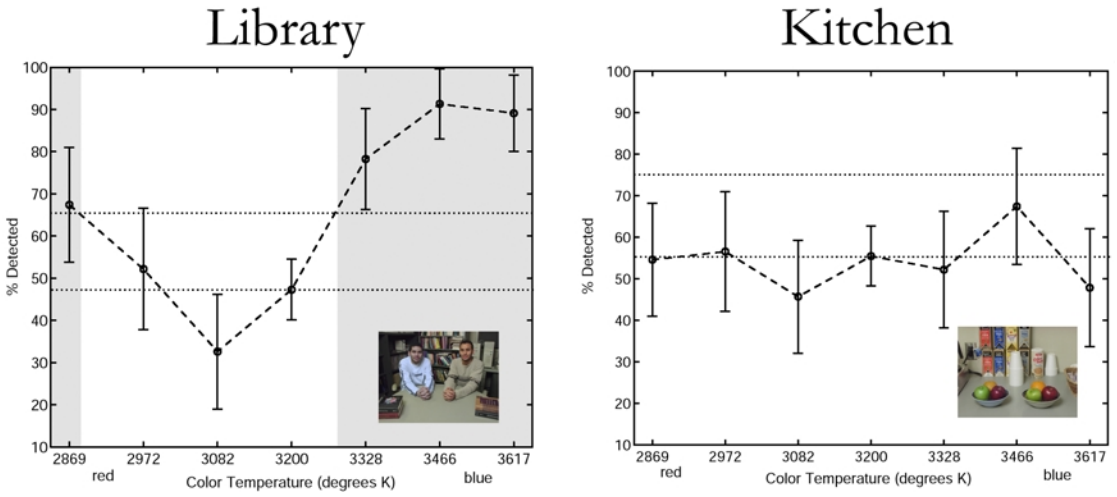


Figure 4.6: Gray regions denote detection of the chromaticity error in the environments. Note that the chromaticity errors were never reliably detected in the kitchen environment.

However, on the right in Figure 4.6 we see that that no such effect was found in the kitchen scene. In fact, over the full range of color shifts tested, participants were never reliably able to detect the composited object $p=.15$ $\psi^2(1,230) = 2.157$. This result was unexpected, as it occurred despite the fruits having highly saturated colors.

These results suggest that while observers are sensitive to chromaticity errors in lighting composites, the direction and degree of sensitivity depend upon the subject matter in the scene. Note that the luminance level of the test objects was the same for all stimuli in this category. It is our hypothesis that the bias toward lower detectability of red shifts in the library scene is probably due to the fact that shifts in this direction are within the acceptable range of human skin tones while large blue shifts are not. The lower overall sensitivity for color shifts in the kitchen scene probably also reflects a greater sensitivity for changes in skin tones over other object colors. However, we believe that color shifts of greater magnitude in the kitchen scene would eventually be detectable and that the data

in Figure 4.6 actually shows the central section of a shallowly sloped sigmoid psychometric function.

Intensity Changes

Figure 4.7 shows the results for errors in the intensity of the composited object. We see that in the library scene observers were able to reliably detect the composited element when the brightness (intensity) was increased by one stop (+3.9%), however the brightness had to be decreased by three stops to be detected (-11.7%). These effects were significant at the $p<<.001$, $\psi^2(1,230) = 14.97$, and $p < .01$ $\psi^2(1,230) = 6.45$ levels respectively.

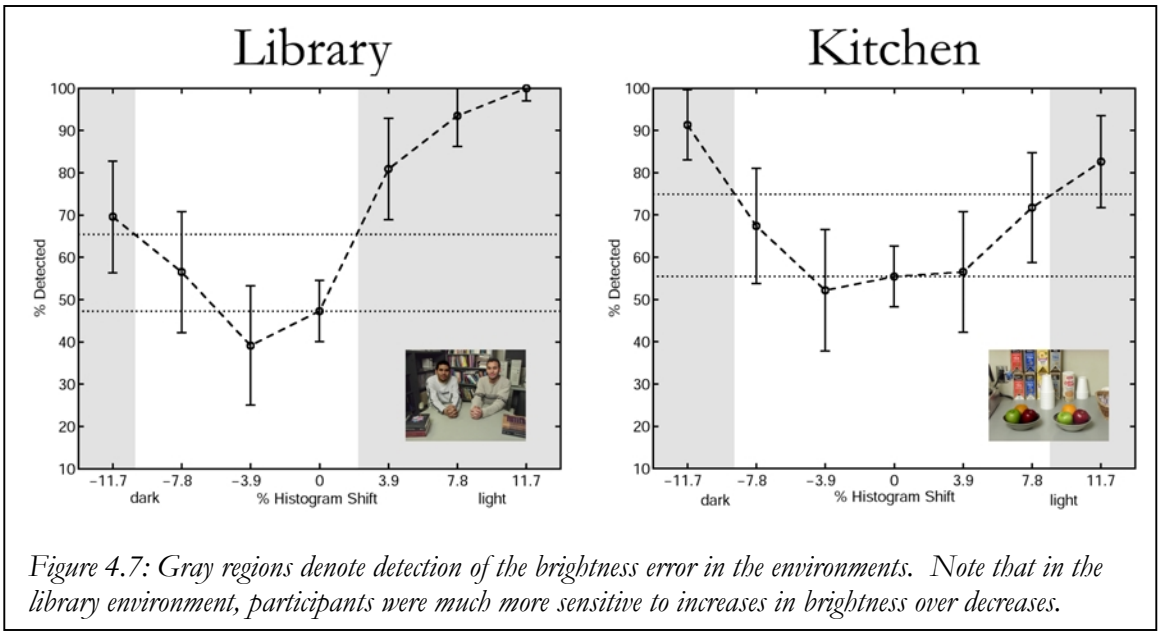


Figure 4.7: Gray regions denote detection of the brightness error in the environments. Note that in the library environment, participants were much more sensitive to increases in brightness over decreases.

Figure 4.7 shows that similar effects were found in the kitchen scene. Here, the observers were able to reliably detect the composited element when the intensity was increased by three stop (+11.7%), and also decreased by three stops (-11.7%). These effects were

significant at the $p<.001$, $\psi^2(1, 230) = 10.30$, and $p << .001$ $\psi^2(1, 230) = 18.72$ levels respectively.

These results indicate that intensity errors are detectable over the range of magnitudes we studied. Further, the asymmetry in sensitivity in the brighter and darker directions found in the library scene probably indicates a greater sensitivity for certain kinds of intensity errors in composites of human beings.

Illumination Direction Errors

Figure 4.8 shows the results for errors in the illumination direction of the composited object. On the left we see that for the library scene, observers were able to reliably detect the composited element when the illumination direction error was increased by 5 stops ($\theta>90^\circ$). This effect was significant at the $p<.01$, $\psi^2(1, 230) = 6.22$ level.

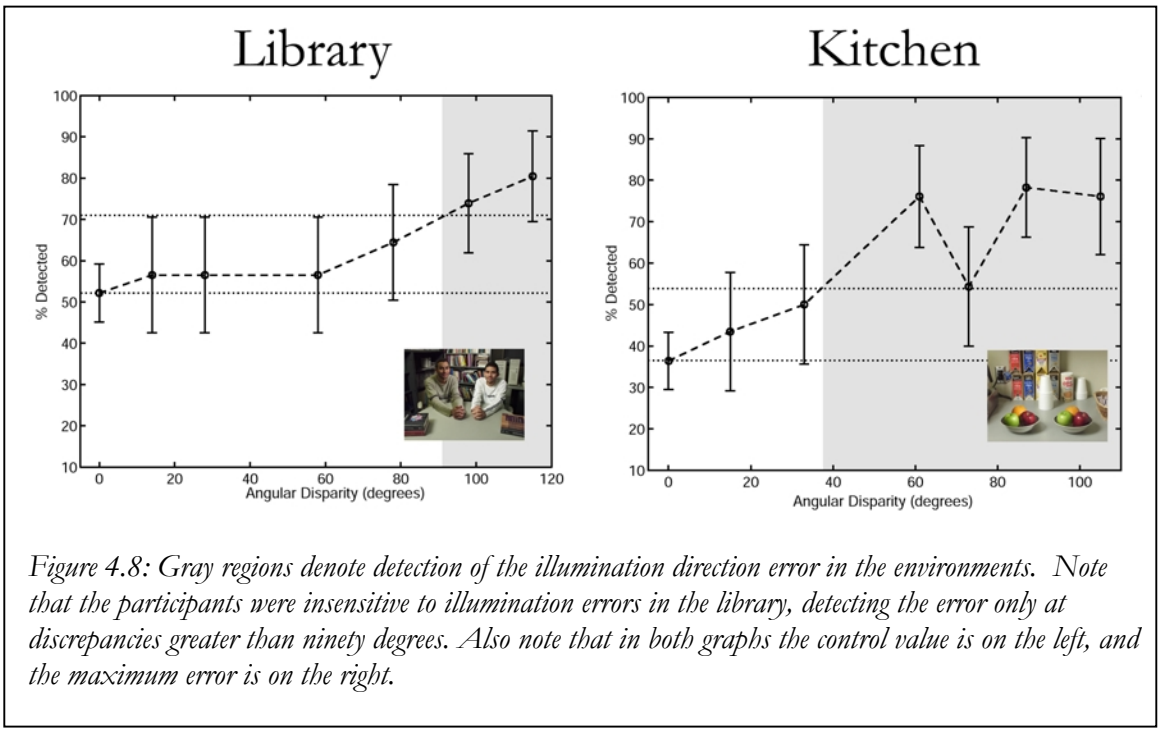


Figure 4.8: Gray regions denote detection of the illumination direction error in the environments. Note that the participants were insensitive to illumination errors in the library, detecting the error only at discrepancies greater than ninety degrees. Also note that in both graphs the control value is on the left, and the maximum error is on the right.

We can also see that similar effects were found in the kitchen scene. Here, the observers were able to reliably detect the composited element when the illumination direction error was increased by 3 stops ($\theta>61^{\circ}$). This effect was significant at the $p<<.001$, $\psi^2(1,230) = 21.89$ level.

These results suggest that observers are insensitive to errors in the direction of illumination in composited objects. This is in concert with previous studies of illumination perception [Otro01], and is an important result for the field of image compositing since it suggests that the careful angular matching of direct illumination is not always necessary. However further studies of this effect across a range of scenes and subject matter is essential before any hard conclusions can be drawn. Participants further reported that in the library scene, the more striking errors were more easily detectable because they made the composited person look "flat" relative to the rest of the scene.

It should be noted in Figure 4.8 a stimuli bias was discovered. This is observed in both graphs, where the control conditions differ from 50%. This is surprising from a visual inspection, as both bowls appear to be quite identical. Regardless, we are able to accurately remove this bias from our statistics by comparing the null condition to the counterbalance set.

Cast Shadow Directional Errors

Figure 4.9 shows the results for the shadow direction condition. Unlike the other errors studied, we observed significant differences in detection when the composited object was on the right and left sides and so we could not combine the data. However, the detection

rates should be expected to differ, as errors on the left side cause the cast shadows to diverge, while errors on the right cause the cast shadows to converge.

From Figure 4.9 we can observe that for the kitchen scene featuring converging shadows, errors were never reliably detected, although the rate of detection did increase along with the magnitude of the error. In contrast, we see the converging condition easily detected when the angular disparity exceeded 1 stop, ($\theta>15^{\circ}$). This effect was significant at the $p<.15$, $\psi^2(1, 208) = 2.13$ level. However, this greater sensitivity to the converging condition is readily explained, as diverging shadows are quite common in the real world, and are created from single relatively close light sources. Converging shadows, on the other hand, are quite unlikely in real situations and can only be created for closely spaced objects using multiple carefully balanced lights. Thus, the greater sensitivity that the participants showed in the converging condition is probably because the error is more salient, as the illumination conditions are less likely.

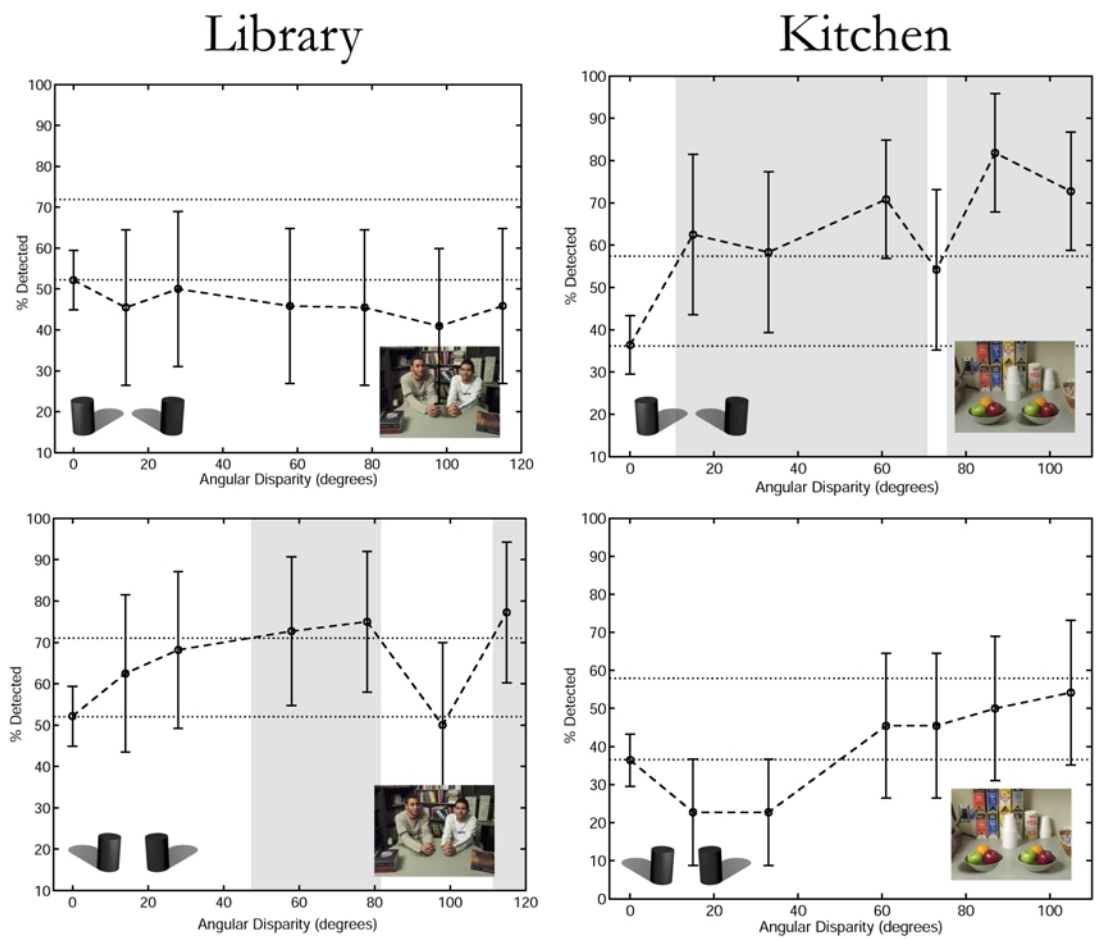


Figure 4.9: Gray regions denote detection of the cast shadow direction error. Note that the left and right error conditions, which correspond to converging and diverging shadows, could not be merged due to significant differences in their trends. Therefore, four graphs are presented to show all of the data. The inset images denote if the given graph corresponds to the converging or diverging case.

We also note that similar effects were observed in the library scene. Participants were able to reliably detect the diverging shadows when the error was increased by three stops ($\theta>58^{\circ}$), $p<.05$, $\psi^2(1,N=230)=3.59$ level. However, in the converging case the errors were never reliably detected. Though this contradicts our hypothesis, we believe it to be an experimental artifact as the people's shadows in the converging case are obscured by the table edge, and thus do not function as an effective cue. Note that in the kitchen environment the converging shadows presented very strong cues.

Conclusion

We conducted a series of psychophysical experiments to measure the visual sensitivity to four kinds of lighting errors that occur in image compositing. We present threshold measures for the detectability of the different classes of errors in two representative scenes. The results show that the human sensitivity to lighting errors exhibits a great amount of variability, and are highly dependent upon the subject matter of the scene. Furthermore, we conclude that observers are relatively sensitive to discrepancies in the chromaticity and intensity of illumination between environments, though surprisingly insensitive to cue errors (illumination/shadow direction).

These results are particularly relevant to virtual set design, as virtual environments often have decoupled local illumination and cast shadows directions. As the literature does not provide guidance as to the importance of this effect, this chapter is the first to suggest that in the design of compositing systems we should focus our attention on first matching the color balance and intensity characteristics of the illumination, and only afterwards match the local illumination and cast shadow directions. In the next chapter we focus on the implementation of a system that utilizes these principles.

There are many avenues of further exploration based on this work. First, we have only explored a subset of the error parameters that can be changed due to lighting. At least two other categories, the area of the light, and the key to fill ratio, will likely have similar impacts on the realism of a composite. Second, testing over a large number of scenes and objects will allow us to make informed generalizations. Finally, scene dynamics will likely play an important role in the realism of composite images. For example, do dynamic

sequences highlight or hide errors? The results of these explorations would be beneficial and necessary for the creation of additional metrics to those described above.

Chapter 5

Two Virtual Set Implementations

In this thesis we separate the realistic merging of live action with synthetic environments into three main components:

I. Utilizing the realistic synthetic imagery
II. Matching the live action and synthetic environment's image characteristics
III. Synthesizing the missing layer interactions (shadows / reflections) as a part of the compositing process

This chapter presents the design and implementation of two virtual set systems that adhere to these principles. In both of these systems we generate realistic synthetic imagery by utilizing techniques that mimic the visual complexity of the natural world. An image-based rendering algorithm and a software polygonal framework present alternative rendering methods to those traditionally employed in the virtual set industry.

We use a controlled studio environment to make the live imagery's visual characteristics (illumination, resolution and frame-rates) consistent with the target synthetic environment. This task is aided by our psychophysical studies from Chapter Four, which prioritize the illumination characteristics that have the greatest impact upon composite realism. We therefore make great efforts to precisely match the illumination's intensity

and chromaticity between the live and synthetic environments, but only roughly recreate orientation characteristics.

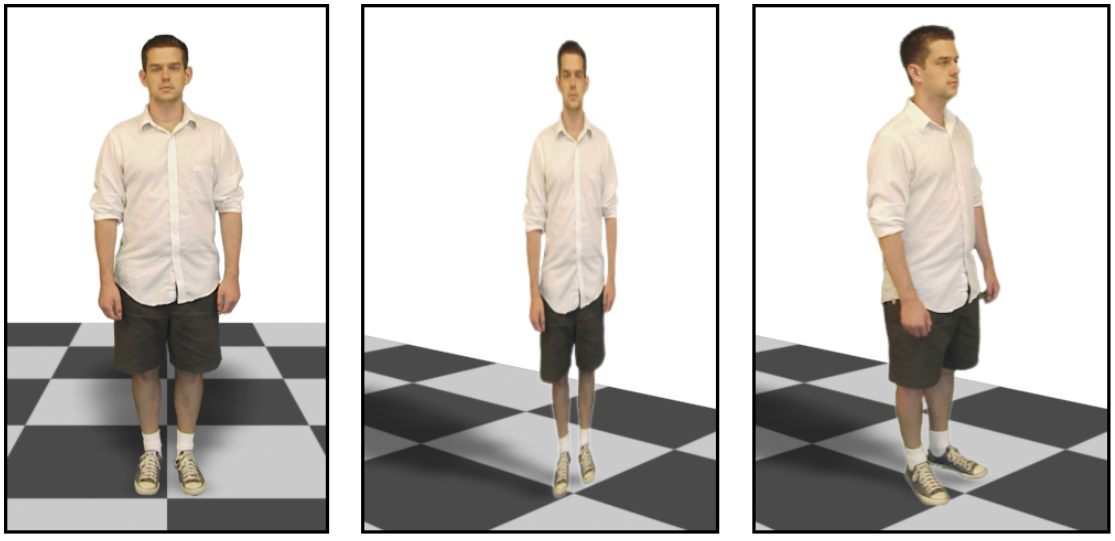
Finally, we synthesize the appropriate visual interactions between the live and synthetic elements. By accounting for the occlusion, shadowing, and reflections between the real and synthetic imagery we generate composites more realistic than those typical of the virtual set industry.

Single-Camera Virtual Set Limitations

Virtual sets that employ a single camera suffer from a common limitation: the rendering system does not have the information necessary to simulate the complete set of interactions between the live and virtual elements. Although multiple viewpoints are necessary to accurately render virtual reflections and shadows, the single camera only observes the scene from a single viewpoint.

Ideally, virtual set rendering systems would employ a real-time, three-dimensional textured representation of the live scene to integrate the live action with the synthetic imagery. This would permit the direct integration of the live and synthetic elements without modifications to common polygonal rendering algorithms, yielding physically correct shadows and reflections. Unfortunately, although incorporating multiple live cameras into the studio environment addresses these issues, current algorithms [Saito99] [Vedula02] are not sufficiently robust, and will likely remain unsuitable for commercial use in the immediate future.

In the current virtual set paradigm, the preferred real-time scene representation is a single stream of two-dimensional images. We are therefore forced to make assumptions of the three-dimensional nature of the video feed, and limit the visual interactions between the live and synthetic objects. For example, it is common for virtual set rendering algorithms to insert the video into the synthetic scene using a planar polygonal mapping. Although this generates accurate composites at perpendicular viewing angles, at non-normal orientations the approximation breaks down, (seen in Figure 5.1). Although cameras that capture per-pixel depth information (as discussed in Chapter Three) begin to address this problem, these 2 ½-dimensional approaches still suffer from occlusion limitations, yielding only an incremental improvement.



Original Viewpoint | Secondary Viewpoint (simulated perspective) | Secondary Viewpoint (camera tracked perspective)

Figure 5.1: In the original viewpoint (left) the physical camera corresponds to the virtual camera, capturing the correct perspective image. A unique perspective is created (center) using a planar mapping. Only a full three-dimensional scene representation can accurately generate the correct perspective image (right).

To accommodate our limited live scene representation, we thus place limitations on which virtual set designs are allowed. For example, although our virtual set system can simulate the virtual floor reflecting the front of a live object, it cannot simulate reflective surfaces behind the live object. This constraint is critical because floor reflections are generated by warping the existing (front view) video image. However, no warping allows us to generate the view from the back of the object, crucial for generating a rear reflection. It would undoubtedly be highly confusing to the viewer for a mirror behind the actor to reflect the image of their front!

Two Approaches to Realistic Environment Generation

Generating realistic virtual environments is the first step in the process of realistically merging live and virtual action. Though the motion-picture effects industry has developed numerous realistic image synthesis techniques (Chapter Two), the techniques are generally unsuitable for real-time use. For example, although procedural modeling and shading techniques generate highly realistic images, they typically requires hours of computation for each frame. Unfortunately, we also must look beyond the virtual set industry for real-time rendering solutions, as the current polygonal hardware methods typically employed do not achieve the requisite level of realism. We thus present two rendering methods that present attractive alternatives to those utilized in the television and movie industries: an image-based rendering (IBR) solution and a software, cluster-based polygonal renderer.

For the IBR approach we utilize a high-resolution rendering system [Levoy96] to synthesize realistic virtual set environments. Our implementation allows arbitrary

viewpoint generation (within a bounded region) in real-time at video resolutions. Furthermore, common to all image-based rendering solutions, the system's performance is independent of the output environment's illumination and geometric complexity.

Our second approach uses a software polygonal renderer to overcome many of the limitations inherent in IBR, such as allowing for arbitrary camera viewpoints and dynamically lit environments. By integrating our compositing system into the real-time global illumination (RTGI) project underway at Cornell University's Program of Computer Graphics [PCG02], we leverage a rendering platform that accounts for the complex materials, geometry, and illumination typical of natural environments. A cluster of PCs is employed to overcome software performance limitations.

An Image-Based Virtual Set

Image-based rendering algorithms typically use sets of images to represent the target object or environment. By utilizing scene representations that account for image characteristics such as light transport, geometry, and materials, IBR approaches avoid many of the complexities inherent in traditional polygonal rendering. As the set of images inherently capture the photorealism of the environment, the challenge of image-based rendering (IBR) is to accurately render a novel viewpoint from the fixed data set.

IBR algorithms utilize an abstraction known as the *plenoptic function* to efficiently characterize light rays in the natural world. The plenoptic function is a seven-dimensional function, capturing the radiance at all points in space, from all directions, at all wavelengths and times [Adelson91]. Simply sampling the function allows one to generate

any conceivable image within an environment. However, recording the high frequency, seven-dimensional function at interesting resolutions is not currently feasible[1]. Therefore, image-based rendering algorithms typically rely on recreating only a small subset of the full function, reducing the effective dimensionality of the problem to manageable levels.

As a first step, most IBR algorithms eliminate the variable of time, and use a three-component color representation-- compressing the plenoptic function to five dimensions. If one further considers the function at a single, fixed position in space, the data set can be effectively approximated using only two dimensions.

Apple's QuickTimeVR [Chen95] leverages this simplification, using the two-dimensional image to record the three-component plenoptic function at a fixed position and time. The great benefit of the QuickTimeVR technique is that immersive, photo-realistic environments can be navigated in real time (shown in Figure 5.2). Though the rendering process is independent of the scene's visual complexity, the user is constrained to viewing the scene from a single location. The process of capturing the two-dimensional plenoptic data set is also straightforward, requiring a single panoramic view of the scene.

---

[1] A seven-dimensional function sampled on a 128-point lattice would require $2x10^{15}$ bytes (two thousand terabytes) of storage.

Figure 5.2: The panorama (top) is the direct visualization of the two-dimension plenoptic function, constrained at a single focal point. Arbitrary images can be synthesized in real-time (bottom) by appropriately indexing the panoramic map. Although not shown, the techniques can also generate images directly above and below the viewer. Images from QuickTimeVR, [Chen95].

Two techniques, the Lightfield [Levoy96] and Lumigraph [Gortler96], extend the QuickTime VR rendering approach to allow the reconstruction of environments from both arbitrary viewpoints and arbitrary orientations. As both of these techniques are efficient enough for real-time rendering, they are ideal solutions for an image-based virtual set.

Both the Lightfield and the Lumigraph employ a common simplification, which allows the five-dimensional plenoptic variant (assuming the three-component color representation of static scenes) to be represented using only four dimensions. Their crucial observation is that light rays have the same color everywhere along their path,

barring intersections and assuming empty space[2]. Therefore, by restricting the viewpoint to locations that do not intersect captured objects, one can characterize the plenoptic function with four dimensions.

Both techniques parameterize the four-dimensional plenoptic function using the *two-slab approximation*, where rays are indexed by their intersections with two planes. This parameterization is particularly efficient, as the rendering algorithm for each pixel only depends on evaluating two ray-plane intersections (see Figure 5.3). The near plane is typically labeled the "uv plane", with the far plane is known as the "st plane".

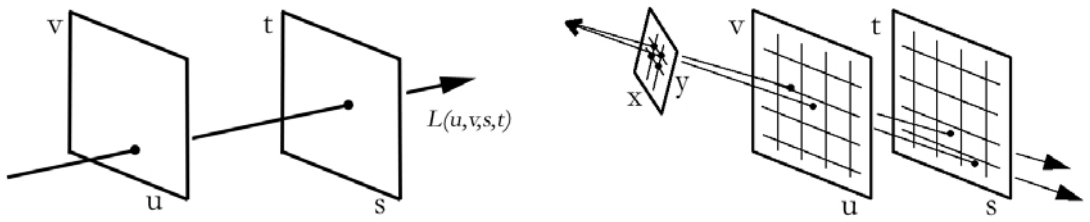


Figure 5.3: In lightfield rendering, the four-dimensional plenoptic function is sampled at the intersection of the view-rays through the two light slabs (left). This parameterization allows the plenoptic function to be stored as a two-dimensional array of images, with the uv plane defining the set of focal points, and the st plane serving as a common image plane. Novel images are synthesized (right) by sampling the uv and st planes along each viewing ray.

Although both approaches model the plenoptic function through similar parameterizations, the approaches differ in that the Lumigraph leverages depth information during the reconstruction process, while the Lightfield does not.

---

[2] The "empty space" assumption breaks down for environments with participating media (such as fog).

Although utilizing depth information allows the Lumigraph to typically generate fewer reconstruction artifacts, it imposes restrictions on the methods that can be employed to capture natural environments. Thus, we have chosen to solely utilize the Lightfield algorithm for reproducing synthetic virtual set environments.

Unfortunately, image-based rendering methods typically suffer a common restriction. Due to practical memory constraints, images reconstructions are typically low resolution. For example, an uncompressed 512x512 lightfield requires 275 gigabytes of storage. Images rendered from this data would be expected to have an even lower resolution, due to interpolation induced blurring. Though the data sets are largely redundant (allowing high compression rates), allowable compression algorithms must be very efficient to enable real-time compression and decompression. [Levoy96] utilizes a vector quantization (VQ) compression algorithm in conjunction with the Lempel-Ziv (LZW) encoding scheme [Ziv77] to achieve compression rates of 120:1. This allows the 275-gigabyte lightfield to be stored in two gigabytes. However, even with this reduction the 512x512 data set cannot fit within video-card memory—a necessity for real-time image synthesis. Thus, the [Levoy96] approach was limited to low-resolution (256x256) lightfields and display resolutions of only 192x192. This is unacceptable for rendering virtual set environments, as television has an effective 720x486 resolution.

We thus impose further constraints upon the plenoptic function to better allocate the limited memory to higher resolution imagery (albeit from a more limited set of viewpoints). We begin by observing that in typical studio applications the camera is mounted to a tripod at a fixed height. Although the tripod base can translate to arbitrary locations, and pivot to arbitrary rotations, it rarely assumes a new height. By locking this constraint we can store a greatly reduced subset of the four-dimensional plenoptic

function. With this reduction our 512x512 compressed lightfield data now consumes fifty-four megabytes— well within the storage capacity of modern rendering hardware. The only drawback is that we cannot render scenes from camera heights differing from the captured data.

The Lightfield-Based Virtual Set

We synthesize novel images through a four-step process (as seen in Figure 5.4). In the first stage environments are captured in a set of two-dimensional images, each containing a slice of the st plane (refer to Figure 5.3). In the subsequent stages, user interactions guide the resampling of this data set to generate novel views. Note that only views that lie within bounds of the data set can be accurately reconstructed.

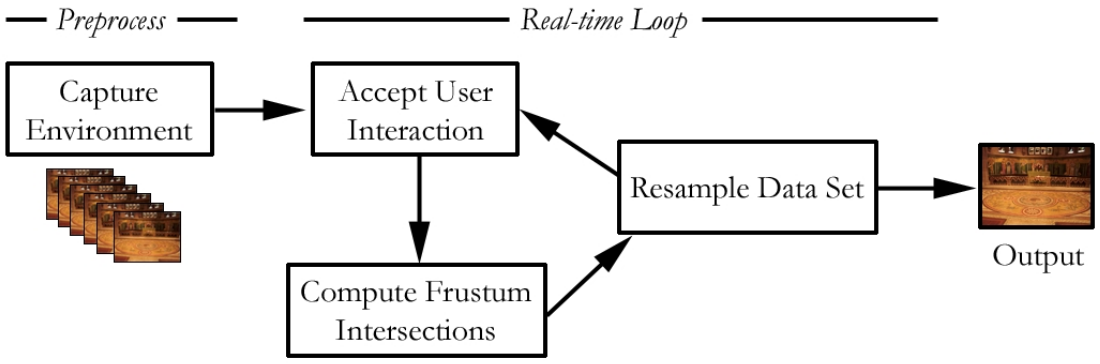


Figure 5.4: The image-based environment is synthesized though a four-step process. First, the environment is captured using a large set of still images. Next, the user specifies a unique camera position, one potentially not found in the original data set. The new view rays are intersected with the original data set, yielding a set of texture lookup coordinates. The original image set is then sampled using these coordinates, generating the novel output.

The process of recording the lightfield of an environment begins by acquiring a series of images along a known path, typically linear. By recording the location, orientation, and field-of-view for each image we generate a direct mapping between the captured data set and the physical light rays in the scene.

When capturing a real environment (vs. a computer generated one) the user must account for the analog inaccuracies of the recording process. Even with careful camera placements, small errors are likely to creep into the orientation and position measurements. It is thus necessary to quantify and correct for these errors during an intermediate stage. One also must insure that during the capture period all objects in the environment remain stationary, as motion in the initial data set causes ghosting artifacts during the reconstruction process. Figure 5.5 illustrates the capture of a physical environment.

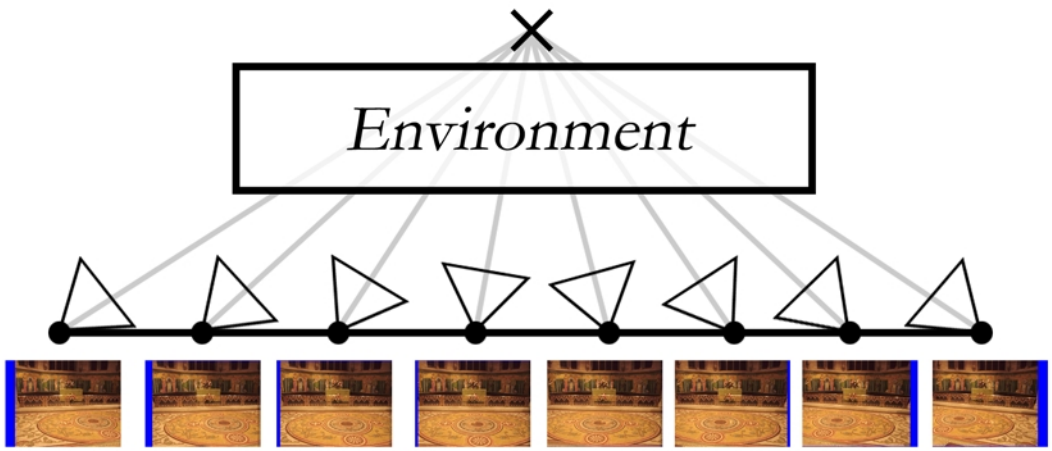


Figure 5.5: The original environment is sampled (photographed) along a specified route (typically a linear path). With careful measurements of the camera position, orientation, and field-of-view, the plenoptic function of the environment can be efficiently characterized. The vertical blue segments in the live frames are a result of correcting for orientation errors during the capture process. Although only eight images are shown, the full data set consists of thirty-two images.

Rendering synthetic environments using image-based rendering techniques yields incredibly high quality-imagery because the rendering process is ideal: the camera is distortion-free, the depth-of-field is infinite, the field-of-view is precise, and the rendered imagery is noise-free. Furthermore, the capture process is a simple task, as the computer can easily generate a sequence with the camera moving along a known path.

Once an environment has been captured, one renders new, novel viewpoints by selectively resampling the original data set. The process begins by having the user define a novel view frustum, which most likely does not correspond to any in the original view set.

To render the environment from the novel position we determine the color along each the view rays by following them until they intersect the path upon which the original viewpoints were recorded. Once the path intersections have been found, choosing the closest corresponding ray (both in angle and position) from the initial data set will fix the color for our novel ray. Note that this process relies heavily on our assumption that a given ray has the same color everywhere along its path.

Finding the closest ray in the original data set corresponding to our view ray is solved through a two-step process. First, the location of the view ray intersection is quantized to the nearest position corresponding to an original image location. Second, the angle of the quantized ray intersection defines the lookup coordinates within the particular image. This angle is transformed into the two-dimensional texture lookup coordinates by directly interpolating the view angle between the field-of-view for the original frustum. Figure 5.6 illustrates this process.

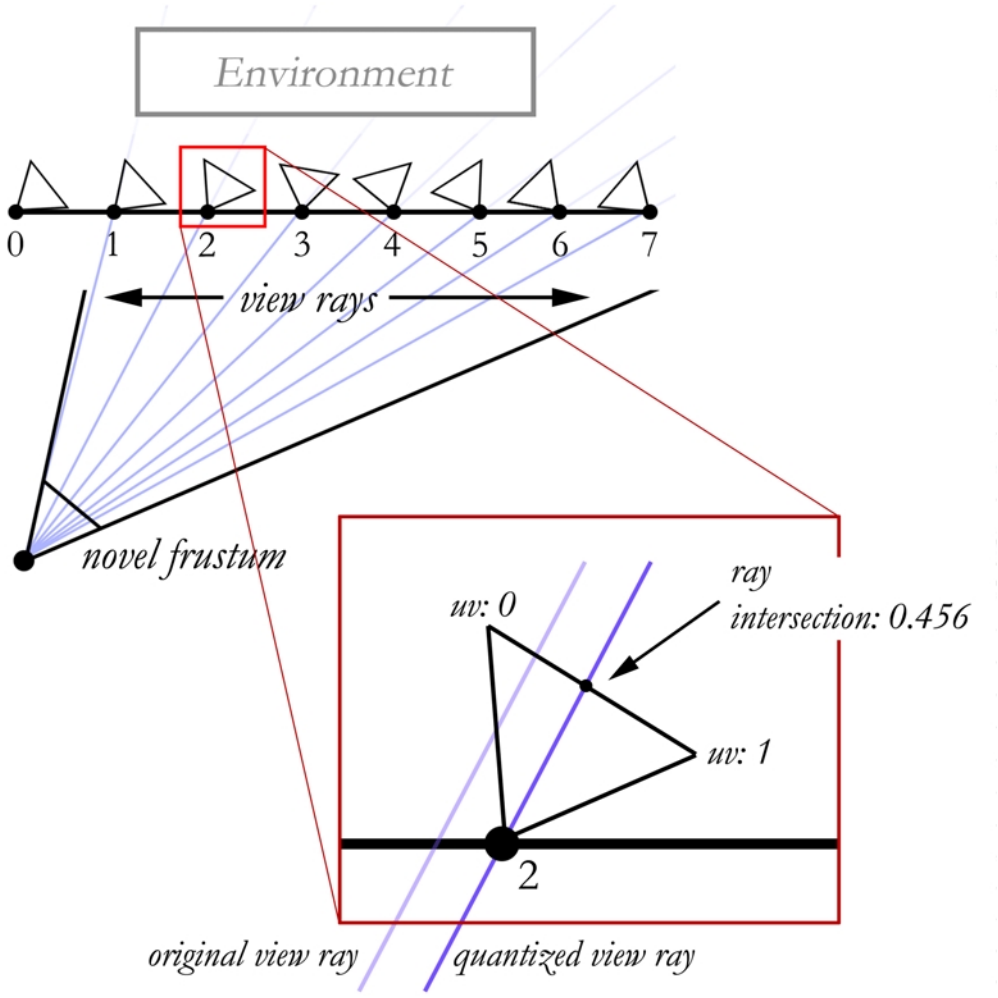


Figure 5.6: To generate a novel view, the intersections points of the viewing rays with the frustums is computed (top). Next, the ray intersection vectors are quantized to the predefined frustum locations (bottom). Finally, the ray' intersection angle defines the UV coordinate lookup. Thus, for each ray we compute a unique image index and UV coordinate, precisely specifying its texture lookup indicies.

The final image is rendered by resampling the initial data, using the computed UVs and image indices for each pixel. An example is shown in Figure 5.7.

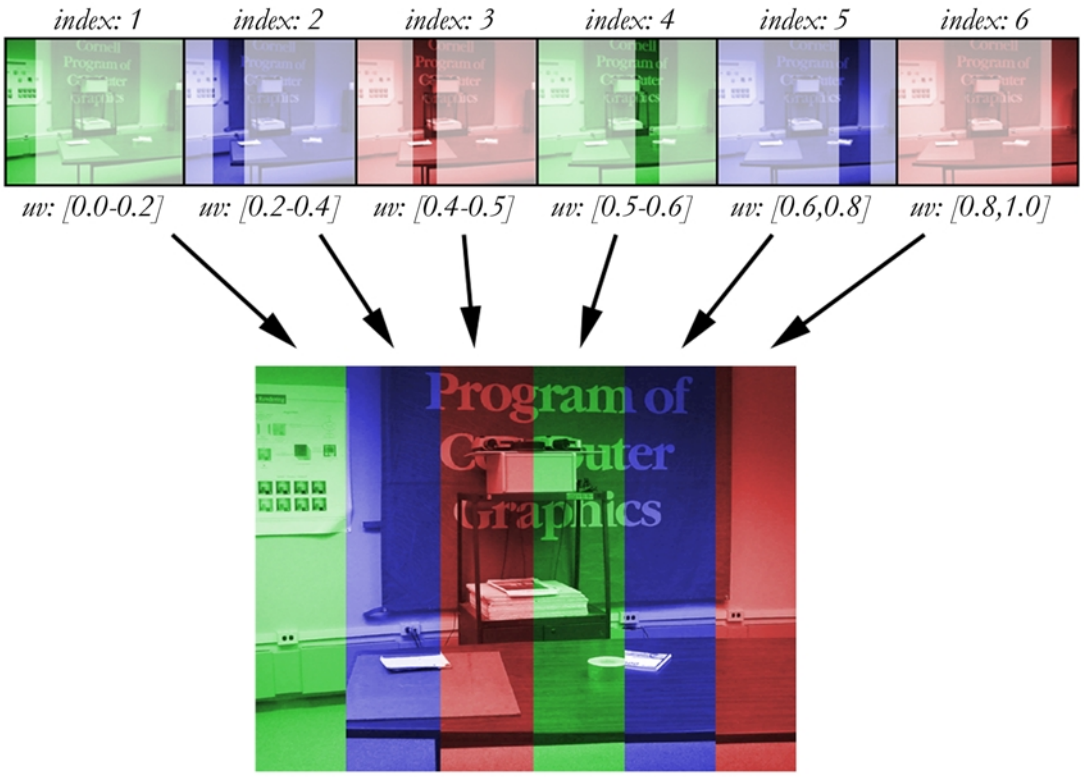


Figure 5.7: Novel images are constructed from the original data set using the computed image indices and UV's for each pixel. Although this scene is reconstructed from six base images, typical renderings use far more images to reduce boundary aliasing (usually more than twenty). The color scheme is for visualization purposes only.

Unfortunately, quantization of the intersection points generates hard boundaries between image slices (as shown above). This causes objects spanning a border to end abruptly, generating visual discontinuities. Fortunately, we eliminate these border artifacts by linearly interpolating the results of the lookups from the two nearest image indices. By weighting the image segments in proportion to the ray's distance to each index point, we can smoothly transition between the different regions.

The linear interpolation scheme also introduces ghosting artifacts of its own, (shown in Figure 5.8). The ghosting is caused by motion parallax in the scene, as near objects

translate greater distances across the image plane (between consecutive images) than far objects. Although pre-filtering the data set can eliminate these artifacts, it will also decrease the detail in the image. The best alternative is instead to maximize the number of images used in rendering the lightfield.

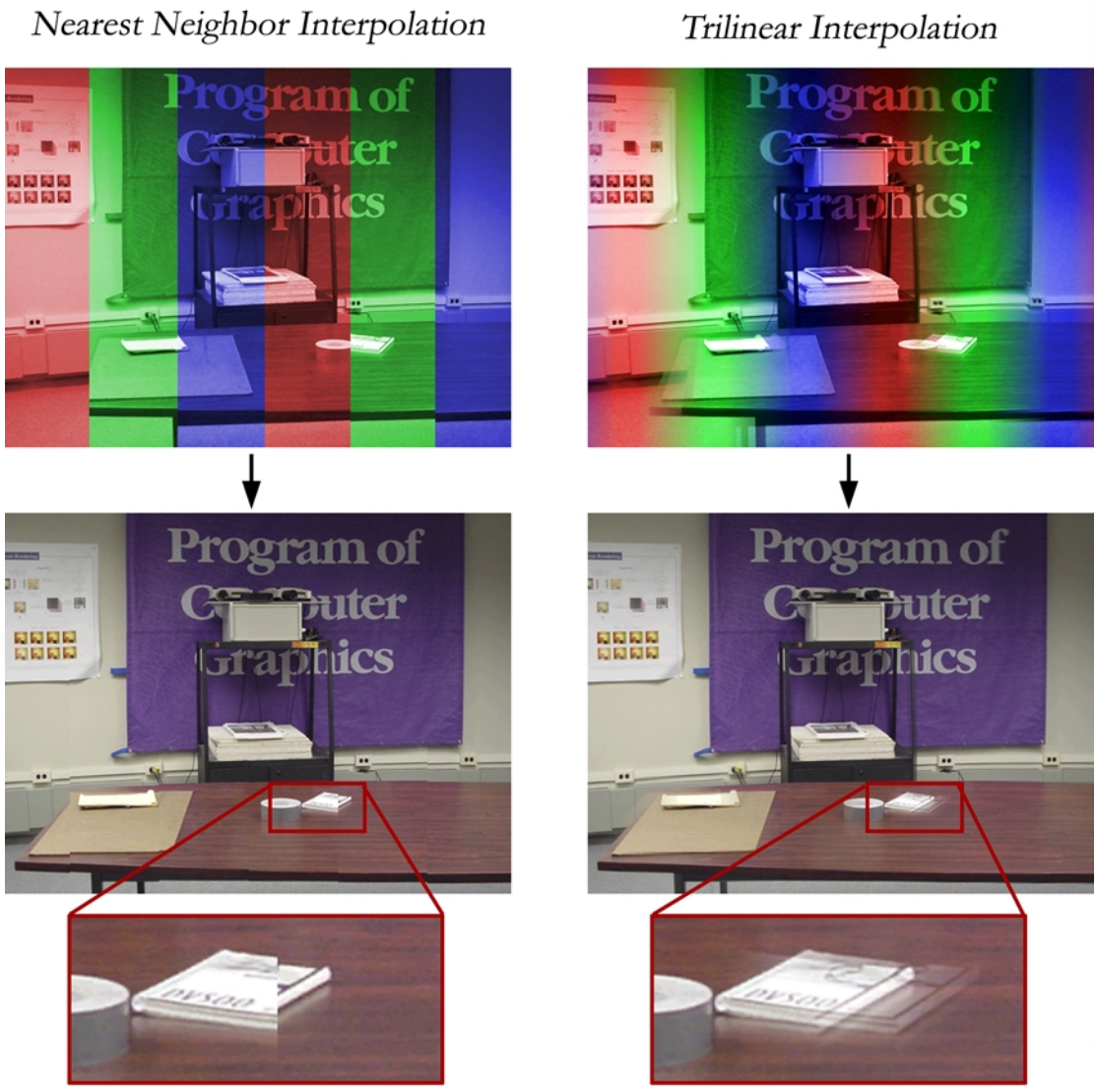


Figure 5.8: Two IBR sampling methods are compared. The nearest neighbor interpolation method generates seams in the final image at the boundaries where view rays switch to the next closest base image. This typically results in obvious artifacts (bottom left). Linearly interpolating between the image layers eliminates these seams in the final composite, though it introduces ghosting artifacts of its own (bottom right). Ghosting however, is typically less objectionable than hard edge errors-- especially during camera movements.

Caveats of Image Based Rendering

Although image-based rendering generates realistic results, it is somewhat limited in flexibility because it can only synthesize visual features found in the original data set. Thus, IBR is not well suited to rendering dynamic, interactive environments. Rendering environments that contain moving objects or dynamic lighting will quickly exceed even the largest storage systems, as each degree of freedom in the data set adds to the dimensionality of the storage requirements. Nonetheless, IBR remains an effective method for the real-time realistic rendering of static environments. In Figure 5.9 we show the Lightfield reconstruction of a synthetic environment.

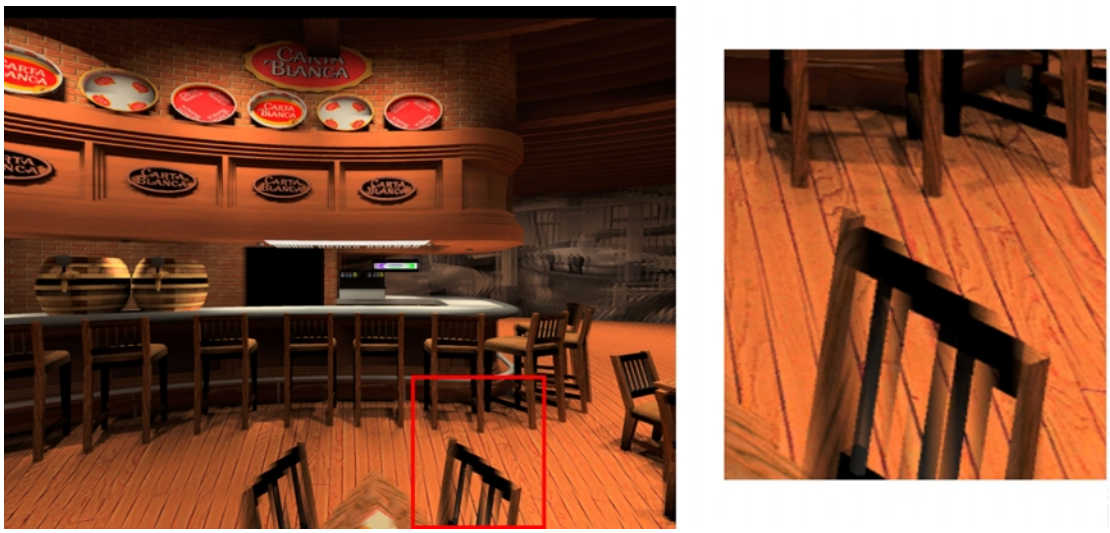


Figure 5.9: The Lightfield image reconstruction technique enables live walk-throughs of synthetic environments that are too complex to be rendered in real-time. The Bar Carta Blanca is originally composed of 2.1 million vertices, precluding its real-time rendering on traditional polygonal hardware. Our scene representation uses 64 1024x1024 synthetic images to generate novel images at thirty frames per second. Observe that objects very near the camera (such as the chair) exhibit ghosting artifacts (right). Using a higher density of input images would decrease these errors, though increase storage requirements.

A Software Polygonal Renderer

Our second image generation approach uses a software polygonal renderer to overcome many of the limitations inherent in image-based rendering, such as allowing for completely free camera viewpoints, and dynamically lit environments. By integrating our video compositing system into the real-time global illumination (RTGI) project underway at Cornell University's Program of Computer Graphics [PCG02], we surpass many aspects of commercial virtual set rendering hardware. Using a software approach to rendering allows algorithms that greatly increase the output realism (such as area-lights and global illumination). These approaches are not typically utilized because they do not easily map onto modern graphics hardware and take too long to compute. Furthermore, as hardware solutions tend to be inflexible, the software rendering approach allows a more direct integration of our compositing code.

The RTGI system is a distributed ray-tracer, consisting of two main components: a cluster of back-end point renderers and a front-end image generator. The point renderers answer requests for the radiance along individual view rays. To compute the radiance, the point renderers break the energy into three components: the direct, indirect, and directional diffuse lighting components. This segregation allows the use of algorithms that are highly optimized for specific illumination paths. For example, direct illumination in multi-light environments is accelerated using local illumination environments (LIE) [Fernandez02], and indirect illumination is computed using an irradiance-caching scheme, based on [Ward94]. Ray intersections are optimized using a KD Tree acceleration structure.

Individual point samples are shaded using physically captured BRDF measurements. Each processor in the server farm can handle approximately 100,000 ray-requests per second for simple scenes, though this number is highly scene dependent. Thus, on our current 128-processor configuration, the system can shade approximately thirteen million rays per second.

The image generator is responsible for handling user interaction, image synthesis, and image display. User interaction is offered through a mix of mouse and keyboard inputs. Two image generation routines are offered, presenting alternative approaches to real-time interactivity. When the point-servers are able to keep up with ray requests in real-time, a direct sampling approach of one ray for each pixel is used (known as Full Frame Mode). For complex scenes where the system is unable to render all of the pixels in real-time, the Render Cache point caching and reprojection technique [Walter02] is utilized to maintain interactivity.

The *Render Cache* algorithm exploits the scene's temporal coherence by keeping track of points (in object space) across multiple frames. On subsequent frames, the user's interactions guide the reprojection of the cached point cloud, maintaining the illusion of real-time interaction. The system uses motion prediction to request rays even before the synthesis algorithm requires them, further masking the system's latency. Finally, an image-plane algorithm fills small holes formed by the inevitable missing point data.

Before the final image is displayed (for both synthesis techniques), a tone-mapping operator transforms the high-dynamic range image into the eight-bit linear color space suitable for display on a CRT. Figure 5.10 shows a typical globally illuminated

environment, rendered using the Full Frame Mode, the Render Cache, and a Metropolis solution [Veach97] (for comparison).

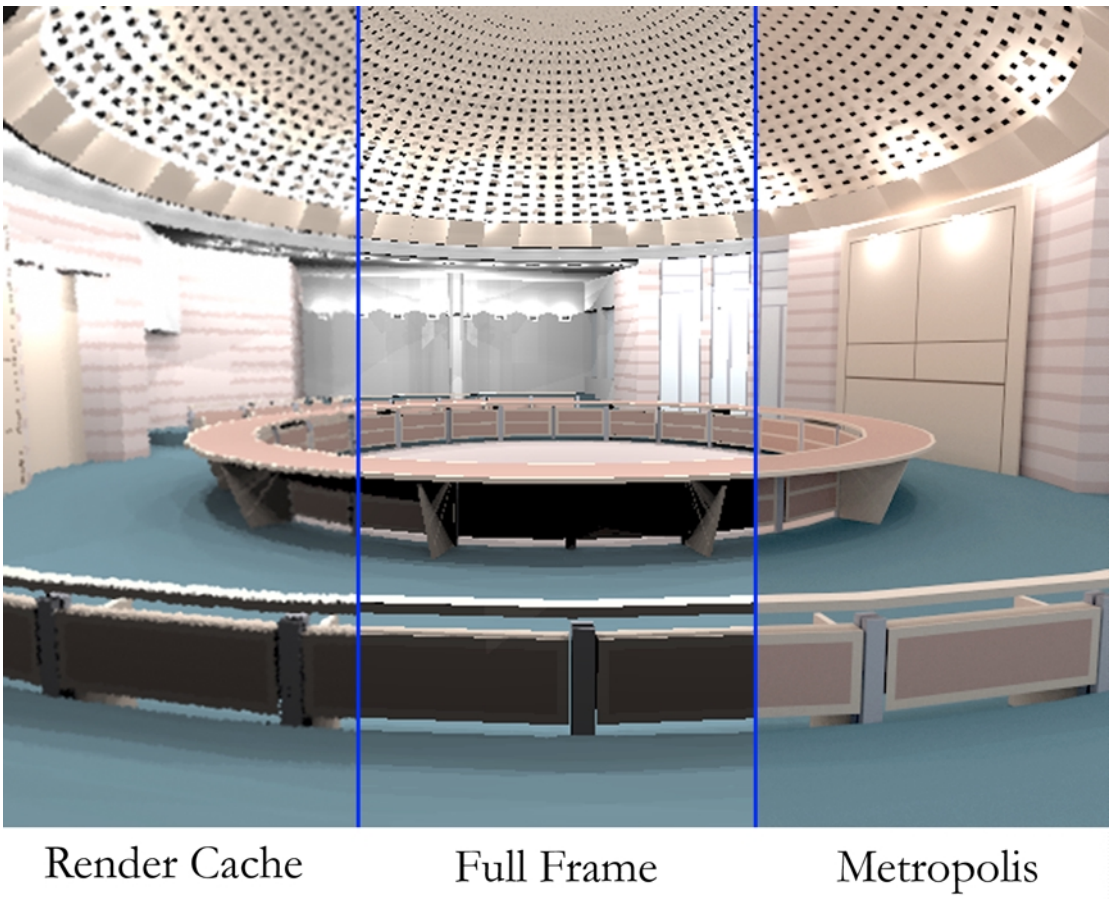


Figure 5.10: Examples of the image generation techniques available in Cornell University's real-time global illumination (RTGI) project. All images utilize the same geometry and back end point renderers. The center slice is computed using the Full Frame mode (one ray/pixel), while the image on the left is generated with the Render Cache point reprojection technique. Using a single node for both rendering and display, Render Cache can generate five frames per second, with the full frame mode requiring two minutes per frame. The Metropolis image synthesis (right) requires two hours of computation for a single frame on our 128-processer cluster.

Caveats of RTGI

Unfortunately, processing power has not reached the point where complex environments, simulating the full transport of light, can be rendered in real-time. For example, without the use of advanced caching technique one cannot interact with globally illuminated environments in real-time, even on 128-processor systems. However, as PC network bandwidth and processing power increases, the realism of these distributed global light-transport rendering algorithms will likely surpass the realism of hardware-based methods.

Controlling Live Action Image Characteristics

In Chapter Four, we quantified the effect that four types of composite lighting errors have upon the realism of composite images. Although the results were only applicable to our two tested environments, they still provide compelling evidence that the lighting and chromaticity characteristics of illumination are more important to reproduce than the cast shadow and local shading directionality effects. We thus focus on matching these two characteristics between the live and synthetic environments.

The most direct manner in which to manipulate the chromaticity and intensity of the live-action element is to manipulate the black and white points of the elements during the compositing process. For images that have balanced histograms[3], modifying the black

[3] It is reasonable to assume that the captured image has a balanced histogram (an equal distribution of luminance values across the recorded image). Although this requires precisely matched lights and cameras (along with appropriate white-balance and exposure settings), it is not prohibitively difficult to achieve in the controlled studio environment.

point can correct exposure and contrast discrepancies, while the chromaticity can be adjusted by modifying the white-point.

We match the illumination intensity and chromaticity between the real and synthetic environments by recording additional information during the capture of the physical environments. Similar to the visual effects industry, by capturing an image of diffuse black and white spheres, we can compute the black and white points for the scene. We then use this data to modulate the color characteristics of the live video roughly approximating the brightness and chromaticity characteristics of the target environment[4]. Figure 5.11 shows such an example.

A similar method is used to match the illumination characteristics with the RTGI virtual set. We estimate the white point for a scene by calculating the incident irradiance at the center of the video polygon, and mapping the result through the global tone-mapping operator. The black-point of the image is estimated as the ambient light present in the environment, also mapped through the tone-mapping operator. However, in most scenes there is no pure ambient component (indirect illumination is considered distinct from ambient), so the black point is typically defined to be zero. This technique also assumes that the live video is properly white balanced.

---

[4] Matching the black and white points from the scene will account for illumination characteristics that are constant across the video polygon. This method will not account for spatially varying environmental illumination.

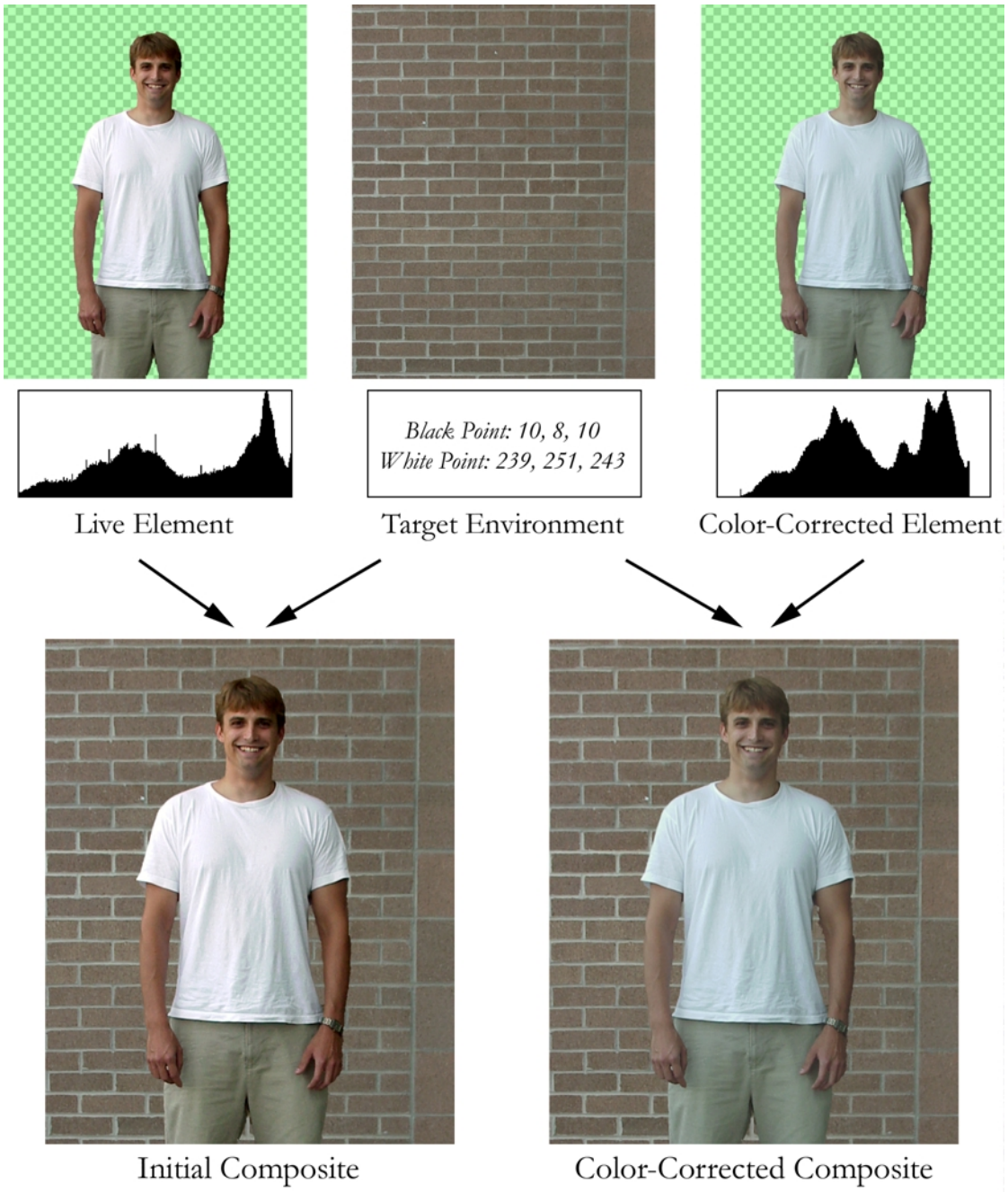


Figure 5.11: Accounting for the chromaticity and intensity discrepancies between the image elements enhances the realism of the composite image. We account for these characteristics by using the back and white points from the target environment to modulate the live element's histogram. Observe the composite (bottom left) formed using the raw, live element. By first applying a color and intensity modification, a more realistic composite is generated (bottom right). Note that although the color-corrected element's dynamic range has been compressed (observe the histograms), the resulting composite image is superior because the foreground element better "fits" into the target environment.

Compositing With Layer Interactions

Synthesizing the visual interactions between composite layers is an essential component to realistically merging live and synthetic imagery. The interactions, such as reflections and shadows, function as the "glue" between the live and synthetic imagery, helping to solidify the perception of the scene as a single, unified environment. Although the motion-picture visual-effects industry typically includes these interactions, the virtual-set industry does not usually do so. We therefore generate real-time shadows and reflections in our virtual environments using a unified matte generation and compositing system.

To simulate the live object casting shadows into the virtual environment, we use a technique similar to projective texture mapping [Segal92]. In projective texture mapping, one projects an image into the scene, akin to using a slide-projector to illuminate the environment. Shadows are cast efficiently using this approach by using a "negative light", which instead darkens objects that fall within the projective area. By treating our live video silhouette as a negative projective texture, we can darken the shadowed regions of the scene in real-time—closely approximating the shape of the cast shadow on the virtual environment. However, we adapt this method because there is no concept of geometry in IBR for which to project the textures onto. We thus make the modification of projecting the live matte directly into screen space. Note that this is unnecessary in our polygonal virtual set, as it contains full three-dimensional geometric scene representation.

To increase the realism of the projected shadow we extend the technique to include soft shadows. Soft shadows, caused naturally by the partial occlusion of area light sources, are not often simulated in real-time due to excessive computational requirements. However, they have been shown to greatly add of the realism of synthetic images [Rademacher01].

Although techniques such as [Fernando02] begin to address real-time soft shadowing, it has not reached the mainstream or been included in commercial virtual set renderers.

We generate soft shadows using a progressive, non-uniform filter kernel to blur the hard-edged silhouette. The approach is similar to [Soler98], but where they perform an (expensive) convolution using the explicit blocker shape, we assume a symmetrical Gaussian shadow distribution. This approximation is critical to simulating the soft shadow in real-time, as it allows us to simulate the convolution using the tri-linear texture hardware. The blurring is simulated in real-time by generating the set of Mipmap levels for the live silhouette, then using texture lookup bias[5] to selectively defocus the shadow in the appropriate locations. The size of the Gaussian convolution kernel is specified by user parameters, where a larger kernel creates larger amounts of blurring. Figure 5.12 illustrates the importance of adding shadows and reflections to compositing.

[5] The texture lookup bias (an OpenGL extension) adds a specified fractional offset to the Mipmap level during the trilinear lookup. Thus, one can access video textures that have been selectively defocused (or sharpened) at no additional computational cost.

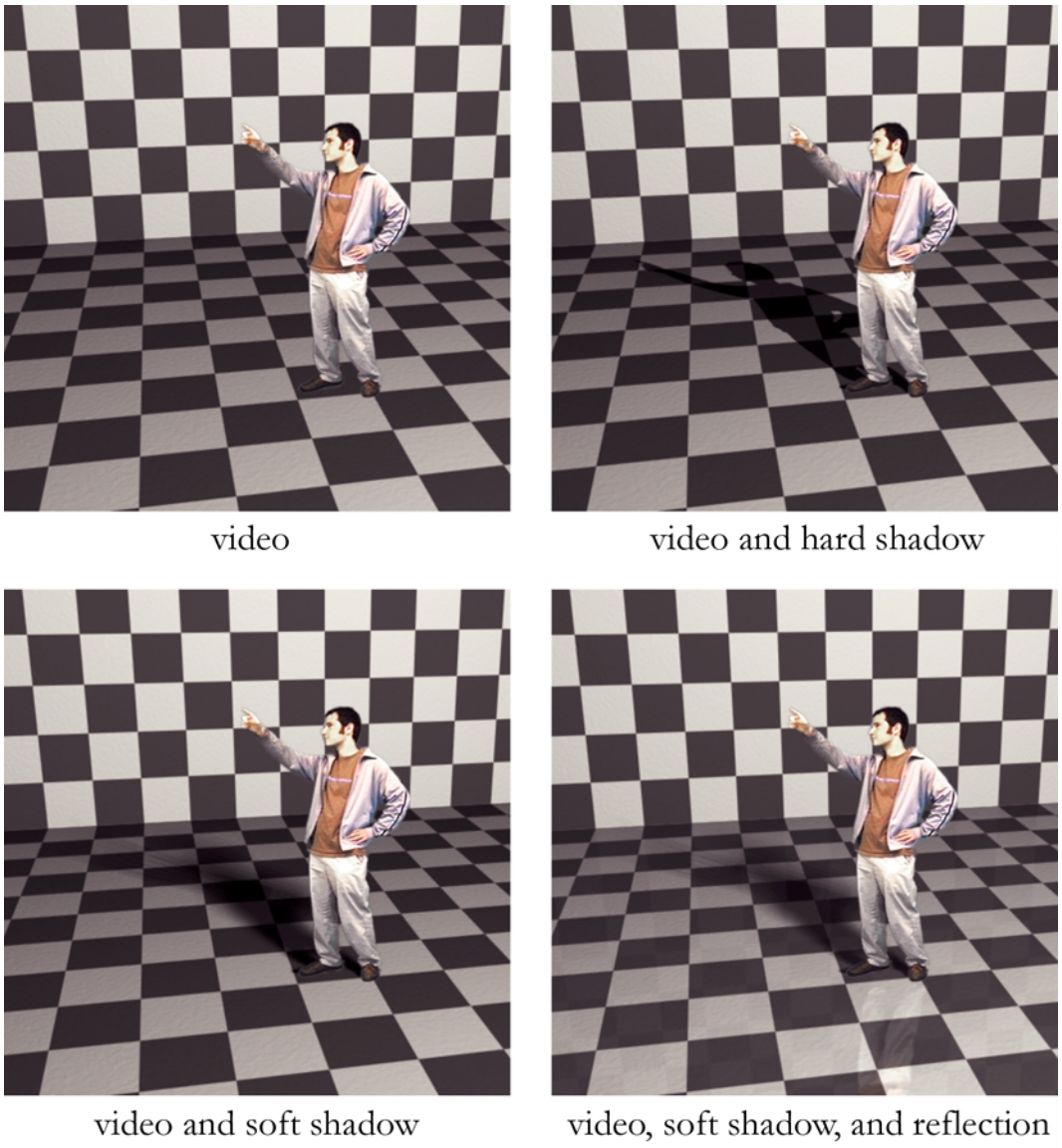


Figure 5.12: Without the synthetic shadow, the foreground is not well integrated into the scene, appearing to float at an imprecise depth (top left). Adding the shadow anchors the live video into the synthetic scene (top right). The hard shadow is generated by projecting the matte onto the environment, while soft shadows are synthesized by defocusing the hard shadow during texture lookup. Different parts of the shadow are defocused by different amounts, generating the hard to soft transition characteristic of real shadows. Adding the reflection (bottom right) further serves to anchor the live object into the synthetic environment.

Shadows are captured from live environments by placing a test object (a cylinder) at the location where the matte object will eventually be inserted into the physical scene. The test object casts shadows onto the environment, which guide the placement of the

shadow and reflection layers. One quantifies the intensity and direction of each shadow mode in the environment by comparing the scene with and without the test object. Using additional images featuring the test object at intermediate locations can aid in the capture of shadows modes cast on non-planar surfaces.

To generate reflections in our image-based environments we use a method similar to that used for shadows. However, instead of projecting the shadow matte onto the environment, we project the full four-component (RGBA) video image. By projecting the video stream at a partial opacity, one can create the convincing illusion of partially reflective virtual surfaces.

Projective Shadow and Reflection Caveats

The shadow projection technique suffers from the single camera virtual sets silhouette projection problem, as mentioned at the beginning of this chapter. Specifically, the virtual shadow has the wrong shape, as we are projecting the front silhouette of the object as opposed to the silhouette as viewed from the light's location. This is most noticeable when the virtual light is at oblique angles to the video polygon.

The projective reflection approach has two main limitations. First, for single camera virtual set systems we can only generate reflections on virtual objects that lie in front of the live action. This is because we simulate front reflections by warping our video feed. The second limitation is that the live object reflection does not properly occlude the reflection of the environment. This is because we are not computing the proper reflection for the entire virtual surface at once, but only adding a warped, transparent video image on top of the environment. Although storing scenes using multiple

components (one for the direct light, one for the reflection) would eliminate this artifact, we have chosen not to do so as the error is typically not noticeable in moderately reflective environments.

Implementation

Two video cameras were used for input: a consumer-grade, single-CCD Sony Hi-8 camera and a broadcast quality, three-CCD Sony DXC-537 camera. Images were captured using a consumer grade, Viewcast Osprey-100 capture board. Images were captured at 640x480, at 30 Hz. An NVIDIA Geforce2 Ultra was used for rendering, driving a Tview-Gold scan converter for output to a standard television monitor.

The image-based virtual set was implemented on a single PC, a dual 1.0 GHz Pentium-III system with 512MB of RAM. The RTGI-based virtual set used a dual 2.0Ghz Pentium-IV front-end display node, with a sixty-four node (128-processor) similarly configured backend.

Camera Tracking

Without the ability to track the camera parameters, it was necessary affix to the camera at a known position in order to accurately composite the video into the virtual environment. The physical and virtual cameras were allowed to diverge, as we wanted the freedom to freely navigate around the virtual environment. However, when viewing the environment from orientations not perpendicular to the video polygon, the system generates perspective distortion in the video composite (as previously seen in Figure 5.2). Virtual

pans and zooms do not generate artifacts however, as these motions do not generate perspective shifts in imaged objects.

Matte Generation

As demonstrated in Chapter Four, a robust matte generation algorithm is critical to the success of all compositing systems. After considering numerous algorithms including the linear keyer [Ultimatte02], Bayesian matting [Chuang02], rotoscoping, and the basic chromakey algorithm, we found that the linear keying algorithm was most appropriate. The other techniques were excluded on account of excessive CPU usage (Bayesian), excessive user interaction (rotoscoping), or insufficient quality (chromakeying). Though commercial hardware could have provided a real-time solution, they generally do not provide the integration flexibility that software solutions offer. An overview of the basic matte generation operation in shown in Figure 5.13.

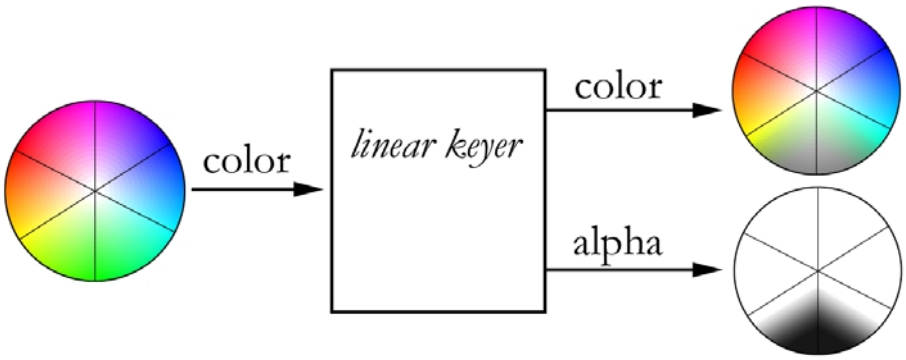


Figure 5.13: The linear keying algorithm maps the input color space to output alpha values and colors. This particular implmentation maps green hues to transparent alpha values, with all other colors rendered opaque. Colors with green tints are desaturated according to their transparency, aiding in the practical removal of edge artifacts.

The result of the matting process is that colors within a particular range are "keyed out" (rendered transparent) while all other colors pass untouched. Furthermore, partially transparent pixels are desaturated in proportion to their opacity. This operation is crucial, as it eliminates obvious matte artifacts, such as colored halos around objects.

As implemented, the linear keyer maps the 24-bit input color space to a 32-bit output space, consisting of a 24-bit color component and an 8-bit alpha component. The mapping can be adjusted in real-time, allowing the keyer to be tuned to the precise hue and saturation of the backing screen. The keying algorithm is a linear process, in that as more of the key color is detected in a pixel, the output alpha values decreased linearly. Mattes generated through linear processes can accurately recreate both semi-transparent objects and partially occluded edges. Note that using a linear keying process generates mattes that have alpha values covering the entire 8-bit output space, whereas nonlinear mattes often only have the values of zero or one.

The linear keying process evaluates each pixel independently of its neighbors. Unlike more sophisticated approaches such as Bayesian Matting, this approach makes an efficient, real-time implementation practical. However, as CPU power increases in the future, algorithms that take local pixel neighborhoods into account will no doubt be able to generate more robust results.

Through experimentation, we found that the green-screen (as opposed to the blue-screen) yields superior mattes. We attributed this difference in quality to three causes. First, green paint typically reflects more incident illumination than blue. This difference is relevant because brighter backings yield greater dynamic range in the recorded pixels, lowering matte noise. Another contributing factor is that CCD technology is inherently

noisier in the blue channel, once again allowing green backings to yield lower-noise mattes. Finally, commodity color cameras generally capture images from a single CCD, using the Bayer mosaic pattern to generate color information. As a consequence of the mask pattern, the green channel is sampled with twice the special frequency of the blue channel. Thus for single CCD cameras, masks derived from the green channel have a greater effective resolution than masks derived from blue. Figure 5.14 illustrates the matte generation process.

Figure 5.14: The linear keying algorithm usually generates high-quality grayscale mattes (center), even from roughly lit green screens (left). Note in the inset (right) that the process is not exact, and errors are introduced. For example, the pant leg partially reflects the green light off of the backing, rendering partially transparent in the output composite.

We also had difficultly consistently lighting the green screen, which is crucial for obtaining high quality mattes. Ideally, the green screen would be positioned a great distance behind the talent, allowing it to be evenly lit with its own diffuse light sources. However, due to our limited space constraints we were forced to position the green screen within a few feet of the talent, lit by a common light source. The close proximity of the green screen often resulted in the talent casting a shadow onto the backing.

Unfortunately, linear keying algorithms map shadowed regions as semi-transparent, black foreground objects, generating unwanted elements in the matte. We therefore often had to increase the non-linearity of the keyer to eliminate the shadows, which unfortunately also increased the hardness of the matte edges.

IBR System Overview

Whereas commercial virtual set systems typically require numerous pieces of expensive, dedicated hardware, our implementation used a single PC to perform the matte generation, rendering, and compositing processes. This convergence of function to a single platform is the major advantage of our system, allowing us to generate effects such as soft shadowing and reflections.

In order to store the large set of environment texture space, a few adaptations were used. Textures were stored using both video and system memory (RAM), leveraging the AGP 4x bus for fast data transfer. Thus, our system allowed for a texture storage space of 256 MB. Textures were stored use the lossy S3tc compression system, which both decreased the size of the data set, and increased the effective data transfer rate (the video accelerator performs decompression, thus using 1/8 of the typical bus bandwidth).

After generating the matte in software, the live video is represented internally as a four-component (RGBA) video texture. Compositing is implemented in hardware using the hardware alpha blending function, with multiple passes for each additional shadow and reflection component. Figure 5.15 outlines the data flow in the image-based virtual set.

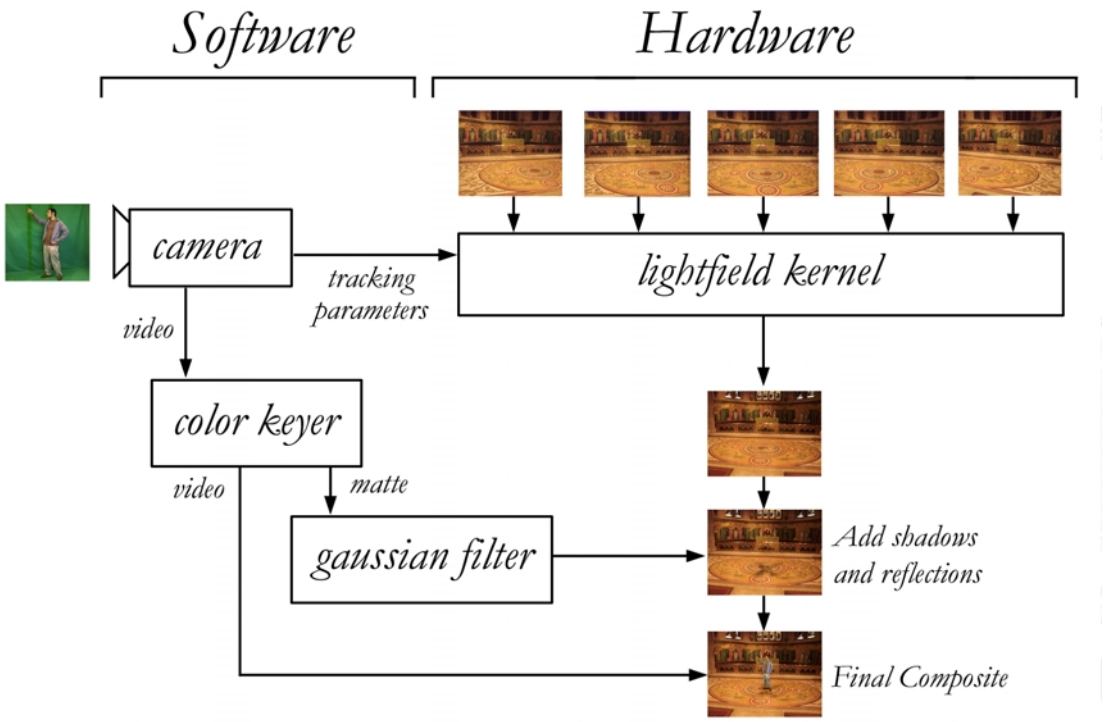


Figure 5.15: First, the live video is captured. Next, the color keyer generates the output alpha channel from the RGB stream. The alpha matte is blurred using a software Gaussian filter, then loaded into the video card's texture memory. Simultaneously, the lightfield kernel uses the live camera parameters to resample the original environmental data set, generating a novel viewpoint. The shadow modes are then added over the background image by reprojecting the matte with a subtractive composite operator. Reflections are then added. Finally, the direct video is added, generating the final composite.

Polygonal System Overview

Two methods are used to integrate the live action into the synthetic RTGI environment. Our first approach is to directly insert the video element as a textured polygon into the scene. Thus, the texture is updated at each rendering node for each frame; generating the illusion of movement. However, transmitting uncompressed, video resolution images at 30hz consumes 294Mbps, whereas our network only operates at 100Mbps. Thus, this approach is severely bandwidth limited! The major advantage of this integration approach is that secondary features such as shadows, reflections, and occlusions are handled seamlessly, as the video is completely integrated with the synthetic environment. Figure 5.16 illustrates the data flow for this method.

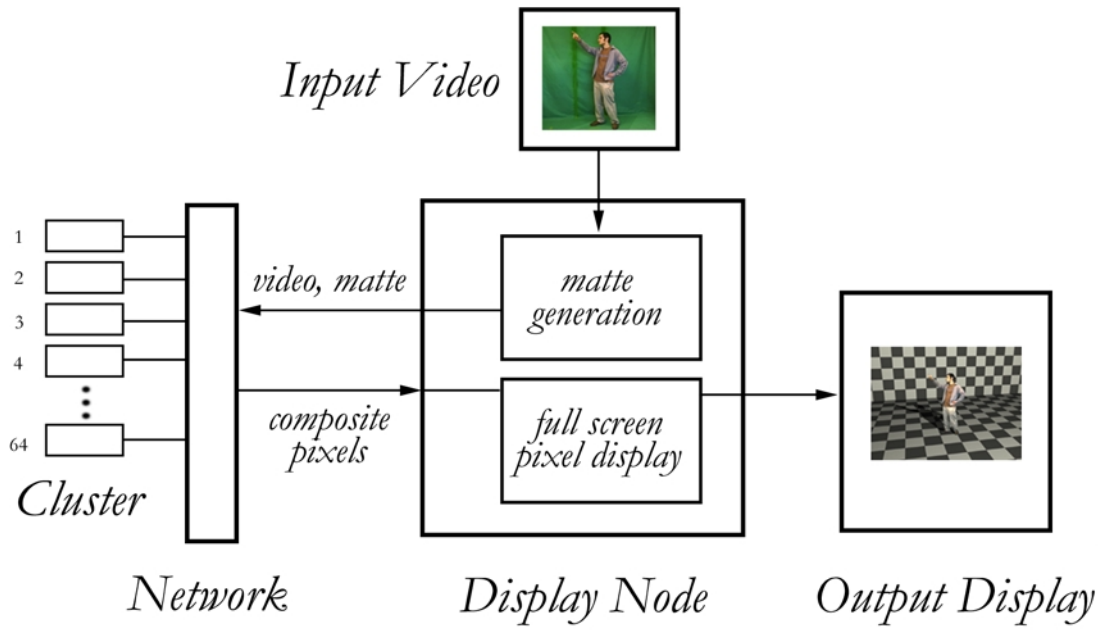


Figure 5.16: By directly inserting the video stream as a textured polygon into the environment, the system is able to correctly render virtual shadows and reflections. However, transmitting the video stream to each node across the network results in decidedly non-real time performance.

The second integration approach is to utilize the Render Cache image generation method along with a local compositing operation to achieve real-time integration. However, we cannot directly add the video stream to the Render Cache point cloud. As the Render Cache algorithm was not designed to selectively invalidate specific segments (our video polygon) from the global point cloud, adding the video would generate distracting temporal artifacts. Furthermore, even if the RenderCache system included such as routine it would not be efficient because treating the video stream as an unconnected point cloud ignores the spatial coherency of our data set. We thus use an alternative compositing approach, directly inserting the video as an overlay on top of the final RTGI output. The schematic for this approach is shown in Figure 5.17.

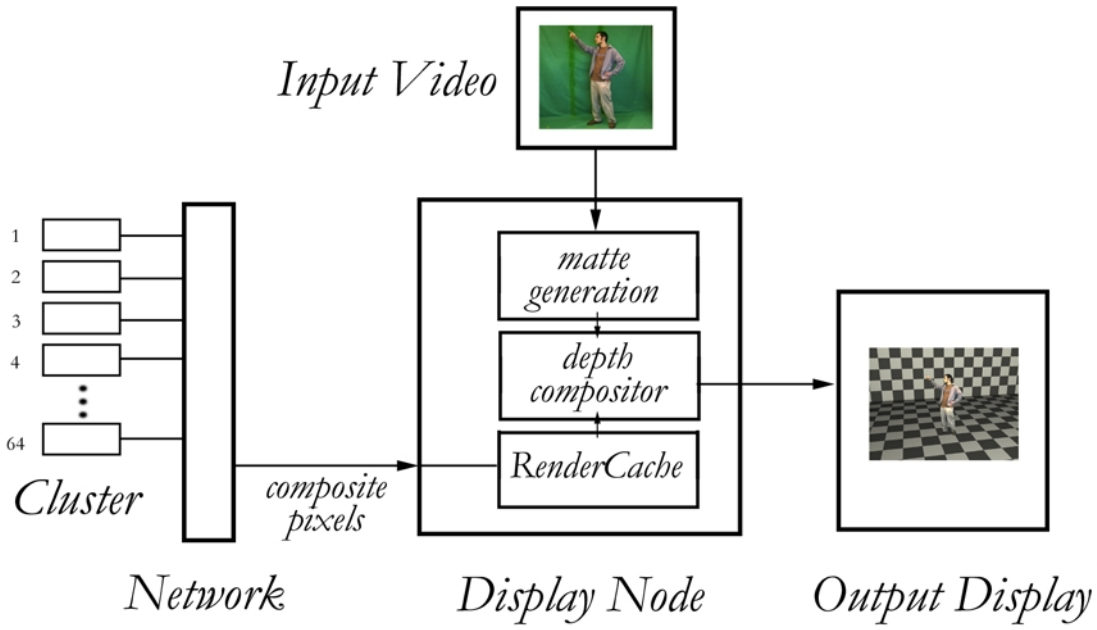


Figure 5.17: By directly compositing the video stream over the output of the Render Cache (eliminating the transmission of the video over the network), the system can operate in real-time. However, integration effects such as shadows and reflections are sacrificed. In order to properly account for occlusion, the system performs a depth comparison during the compositing process for each pixel.

This approach is very efficient at compositing the live and synthetic imagery because it does not rely upon sending a live video stream across the data network. The video signal is only present at the final output node, and can thus be composited directly into the scene using hardware acceleration.

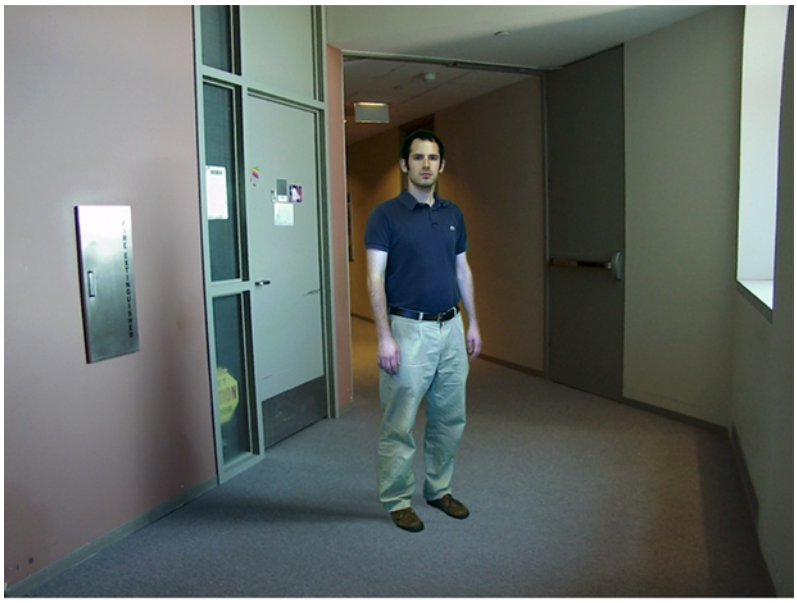
In order to composite the video directly over the Render Cache output, two corrections are needed. First, the video has to be warped from its orthographic footprint into the correct perspective view. This is accomplished by passing the camera parameters to the compositing kernel, rerendering the layer with the video in its perspective-correct location. Second, as the video is overlaid directly on top of the output of Render Cache, one must separately account for the occlusion of the video stream. We thus perform a depth comparison during the compositing process, comparing our rendered Z values to those provided by Render Cache.

The disadvantage of this integration approach is the decreased visual integration with the synthetic environment. As our composite kernel is not aware of the scene geometry, we cannot cast shadows or reflections. Methods for overcoming this limitation will be discussed in the conclusion of Chapter Six.

Image-Based Virtual Set Results

To determine the efficacy of our compositing system, we compare images of the true physical environments to our simulation. Figures 5.18 and 5.19 demonstrate the results of our image-based shadowing and reflection techniques. Note that for these examples the background was not rendered using the Lightfield technique, but simply generated as

a single background image. This is necessary in order to accurately observe the shadow and reflection characteristics, independent of image synthesis algorithm.



Composite Image

Actual Scene

Figure 5.18: A comparison of image-based soft shadowing. The soft shadows in the synthetic environment are generated in real-time using our silhouette reprojection method (top). The success of the approach is evident. Although there are differences between the images, there is a good overall agreement between the real and synthetic shadows. Note that motion in the video, reflection, and shadow layers further serve to enhance the illusion of a unified composite scene.

Composite Image

Actual Scene

Figure 5.19: A comparison of image-based shadowing and reflection. The shadows and reflections in the synthetic environment are generated in real-time using our silhouette reprojection method (top). The success of our approach is evident when compared to the real scene's shadows and reflections (bottom). Note that motion in the video, reflection, and shadow layers further serve to enhance the illusion of a unified composite scene.

We observe that in the two test environments the system was very effective at reproducing realistic shadows and reflections. In the hall environment, there is a good match between the real and synthetic environments. In the outdoor environment, the system was able to realistically account for both hard shadows and reflections. Furthermore, our system was able to render both scenes at frame rates of greater than 30hz.

In the hall environment we observe a few differences between the real and synthetic images. First, it is immediately apparent that the direct illumination on the subject differs in the real and synthetic environments. As discussed in Chapter Four, it is imperative to match the studio lighting with the synthetic environment. Although we matched the chromaticity and intensity as previously discussed (observe that the skin color is constant across images), we failed to account for the relative ratio of the key to fill lights between the environments. The other discrepancy is that the contact shadow (formed where the subject meets the floor) is not well matched. This is a direct result of the subject's pose, as the right leg being further back (higher up in the image plane) casts a weaker shadow. This is easily solved by standing the talent perpendicular to the camera.

In the outdoor environment we observe the two additional limitations of our silhouette reprojection approach. First, comparing the cast shadows between the real and synthetic environment we observe differences in shape. As mentioned earlier in this chapter, this is caused by projecting the front silhouette of the object as opposed to silhouette seen from the light's location. Second, we observe that the reflection of the live video does not properly occlude the reflection of the environment. This is a result of not having access to the occlusion information, common to image-based techniques. Figure 5.20 illustrates these differences between the real and synthetic shadows and reflections.

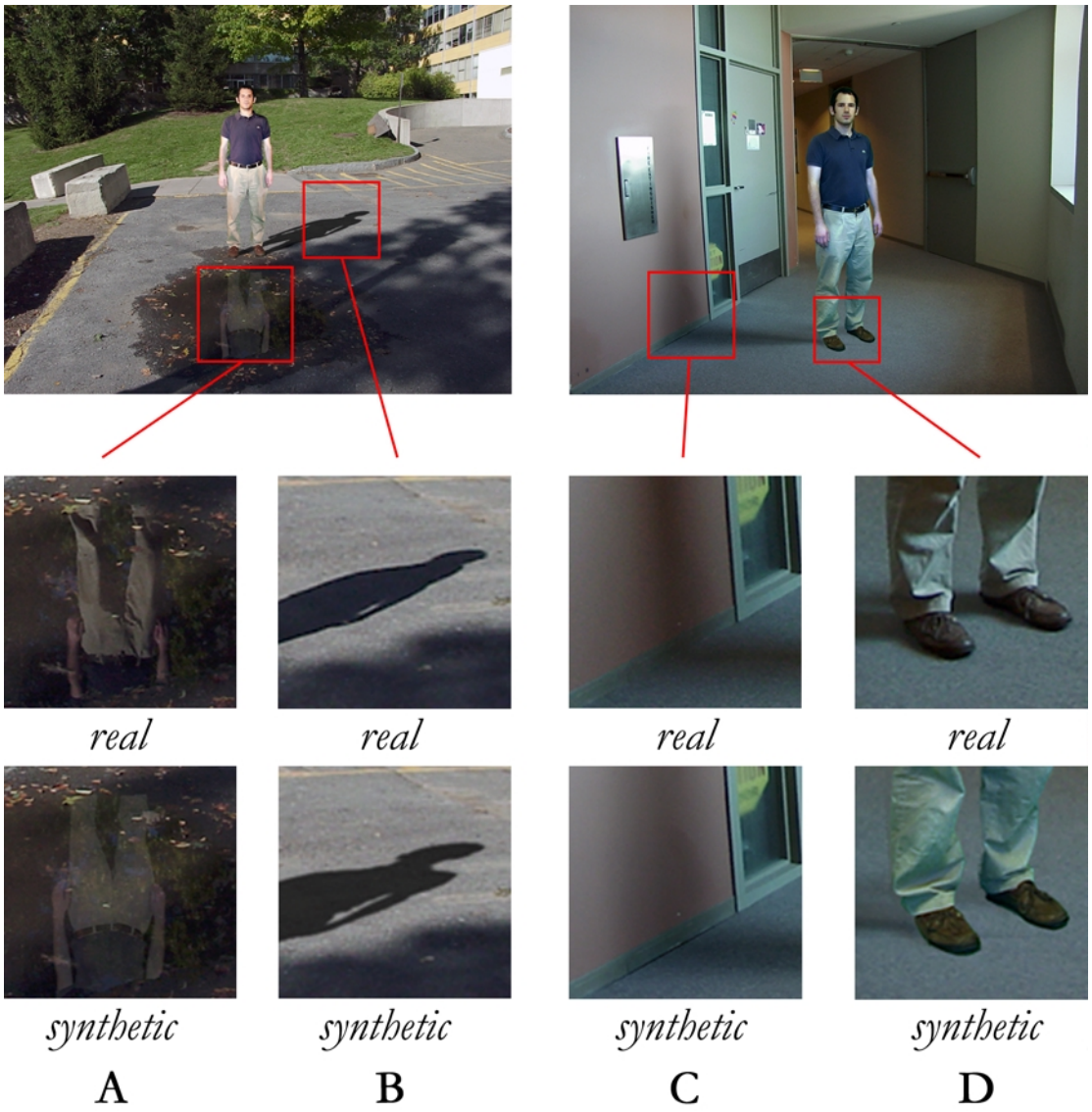


Figure 5.20: Analyzing the real and synthetic shadows and reflections shows numerous minor discrepancies. In A we observe the reflection occlusion limitation of our system. Note that real foreground reflections occlude the reflection of the environment, while our simulation allows the background to partially show through the reflection. In B, although the cast shadows are very similar, we observe the shape discrepancy caused by projecting the silhouette approximation. In C we observe that although the real and synthetic shadow modes are very similar, there are minor intensity discrepancies between the two. D illustrates the errors caused at the contact shadow formed between the video polygon and the floor.

In Figure 5.21 we observe the complete image-based virtual set system, combining shadows, reflections, and occlusion along with our lightfield environment synthesis technique. Once again, the system maintains a steady frame rate of 30hz.

Figure 5.21: These images combine all aspects of the image-based virtual set: soft shadows, reflections, occlusions, and white-point balancing. Observe in the top environment that the soft shadows and subtle reflection anchor the live action into the environment. Similarly, in the bottom environment although no shadows or reflections are present, the occlusion helps to anchor the live video at a precise depth. Both environments are rendered using the Lightfield synthesis technique.

Finally, we show that our image-based technique can be used to simulate synthetic environments using only a single base image. Even without the use of a test object, non-planar shadows can be reasonably approximated (Figure 5.22).

Figure 5.22: A virtual environment can be constructed from a single still image. Observe that the non-planar shadow on the desk has been reconstructed, with the darkening appropriately wrapping around the edge of the geometry.

Results, RTGI Virtual Set

Two compositing approaches are implemented, offering contrasting levels of performance and scene integration characteristics. We will first describe the results of the integration with Render Cache, followed by the direct integration of the video into the RTGI environment.

Observe that the keyer utilized in the RTGI compositing system is of decreased quality compared to the image-based virtual set keyer. This is because the Render Cache algorithm requires the bulk of the display node's computational resources. Thus, in these examples we use a binary chromakey operation that requires less processing power. The result of the chromakey method is that edges are binary valued, generating harder matte edges.

Although shadows and reflections are not simulated, the Render Cache integration method properly handles occlusion and white point matching, as shown in Figure 5.23.

Figure 5.23: Integrating video into the Render Cache environment allows for real-time interaction. Note that although the system accurately accounts for occlusion (right), it does not simulate secondary effects such as shadows or reflections (left).

By directly inserting the video object as a textured polygon (with transparency) into the environment, a complete integration (white-level, shadows, and reflections) is achieved. However, performance is decidedly not interactive, with typical frame rates ranging between one and two frames per second. Figure 5.24 illustrates the result of this approach.

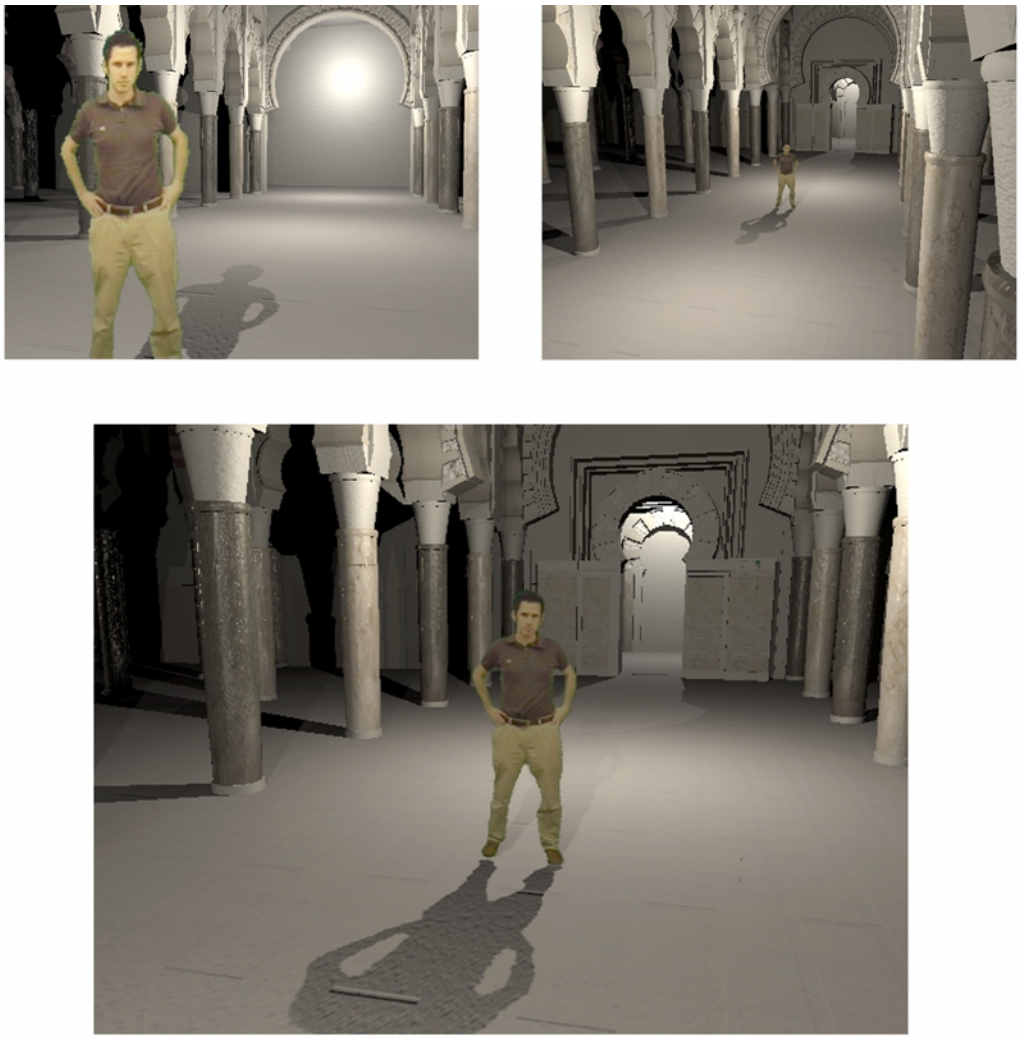


Figure 5.24: The video polygon is directly inserted into the Mosque at Cordoba. Shadows are generated by directly integrating the video polygon into the distributed RTGI environment. Note that the inclusion of the human figure serves to define an absolute size for the environment, which would not otherwise be provided.

Chapter 6

Conclusion

In this thesis we identify the three major components necessary to realistically merge live video with synthetic imagery in real-time: generating realistic synthetic imagery, acquiring coherent live action, and using a compositing system that accounts for the virtual object interactions. In generating realistic synthetic imagery, we demonstrate that high-resolution lightfields provide an attractive alternative to polygonal rendering in situations that do not require dynamic or interactive environments. For interactive environments we show that distributed ray-tracing that accounts for global illumination and physically based light interaction models generates compelling imagery. Our perceptual study supports the goal of acquiring coherent live action by identifying the environmental illumination factors that have the greatest impact upon composite realism. Finally, we present techniques for generating realistic soft-shadows, reflections, and occlusion in real-time.

We have shown that our real-time rendering system offers a promising mix of both realism and performance, especially when compared to the systems currently used in the motion picture visual effects and virtual set industries. Furthermore, as commercially available systems are very expensive, our use of a single PC for all aspects of image generation will likely make this approach very appealing in the future.

Specific contributions of this research are identified below. We have:

* Identified the components necessary to realistically merge live video with synthetic imagery.
* Initiated the task of quantifying the perceptual salience of illumination errors in image compositing.
* Validated the single-PC virtual set architecture for generating soft-shadows, reflections, and occlusions.
* Implemented a high-resolution, lightfield rendering system.
* Combined live video with the real-time global illumination (RTGI) project.

Future Research

There are many avenues for further research. Many optimizations can be made to our software virtual set implementations. For faster integration with the distributed video-rendering system, light paths can be handed in different manners. For example, the direct video could be directly composited into the synthetic environment at the display node, while the other light paths (such as shadowing and reflection) could be optimized by sending compressed, low-resolution images to the compute nodes. Alternatively, an additional display node could add shadows and reflections to the real-time composite.

In enhancing the image-based virtual set, additional system RAM would allow for more accurate lightfield reconstructions across a greater number of viewpoints. Additional memory could also be used to record dynamic, interactive, and high dynamic range environments. Combining our image-based rendering system with video texturing also

appears to be a promising line of research. Furthermore, as memory costs become cheaper, despite the large data storage requirements, these will likely become viable options.

Perhaps the most exciting avenue of future research is to use a more accurate representation of the live video. Multiple camera approaches hint at the realization of a full three-dimensional model, which will allow live image relighting and better integration with the synthetic world. Live image relighting would enable the system to intelligently correct for poorly lit physical environments, while also accounting for complex virtual set interactions, such as the accurate casting of virtual shadows onto the live object.

Finally, offloading computation from the central processor to the graphics card will enhance the performance of our compositing subsystem. Future programmable pixel mapping hardware will no doubt be able to efficiently perform many of the processes (i.e. matte generation), greatly reducing both CPU and bus utilization. This will allowing virtual set systems to leverage high-bandwidth, high-definition video elements.

Bibliography

[Adelson91] Adelson, E. and Bergen, J. *The plenoptic function and elements of early vision.* In Computational Models of Visual Processing, MIT Press, Cambridge, 1991.

[Alex01] John Alex, John Hughes, and Andy van Dam. *Realism in Computer Graphics. Introduction to Computer Graphics Course Notes, Brown University. 2001.*

[Apodaca99] Anthony A. Apodaca and Larry Gritz. *Advanced Renderman: creating CGI for motion pictures.* Morgan Kaufmann. New York. 1999.

[ATI01] ATI Technologies Inc. *ATI Technologies White Paper: Truform.* Ontario. 2001.

[Baraff90] David Baraff. *Curved Surfaces and Coherence for Non-penetrating Rigid Body Simulation.* Computer Graphics (Proceedings of SIGGRAPH 90). *24 (4)*, pp. 19-28, 1990.

[Barzel88] Ronen Barzel and Alan H. Barr. *A modeling system based on dynamic constraints.* Proceedings of ACM SIGGRAPH 88. *15 (3)*, pp. 179-188, 1988.

[Beck59] Beck, J. *Stimulus correlates of judged illumination of a surface.* Journal of Experimental Psychology, 58: 267-274, 1959.

[Berney00] Jim Berney, Jay Redd. *Stuart Little: A Tale of Fur, Costumes, Performance, and Integration: breathing real life into a digital character.* Course #14. SIGGRAPH 00. 2000.

[Best02] http://www.bestdigest.com/movies/bo/alltime_grossing.htm

[Birn00] Jeremy Birn. *Digital Lighting & Rendering.* New Riders Publishing. Indianapolis. 2000.

[Bishop01] Gary Bishop, Greg Welsh, and B. Danette Allen. *Tracking: beyond fifteen minutes of thought.* Course #11. SIGGRAPH 01. 2001.

[Blake00] Blake, A. and Bulthoff, H.H. *Does the brain know the physics of specular reflection?.* Nature. 343: 165-168, 1990.

[Blinn78] J. Blinn. *Simulation of Wrinkled Surfaces.* Computer Graphics (Proceedings of SIGGRAPH 78). 12(3), pp. 286-292. 1978.

[Brinkmann99] R. Brinkman. *The Art and Science of Digital Compositing*. Morgan Kaufmann, New York. 1999.

[Bruce98] V. Bruce, A. Young. *In the eye of the beholder: The science of face perception*. Oxford University Press, 1998.

[Catmull78] E Catmull and J Clark. *Recursively Generated B-spline Surfaces On Arbitrary Topological Meshes*. Computer-Aided Design, 10:350–355, 1978.

[Chen95] Shenchang Chen, *QuickTime VR: an image-based approach to virtual environment navigation*. ACM Computer Graphics (Proc. Of SIGGRAPH '02), 1995.

[Chuang02] Yung-Yu Chuang, Aseem Agarwala, Brian Curless , David H. Salesin, and Richard Szeliski. *Video Matting of Complex Scenes*. ACM Computer Graphics (Proc. Of SIGGRAPH '02), 2002.

[Clark76] Clark, J. H. *Hierarchical geometric models for visible surface algorithms*. Communications of the ACM, 19:547-554. 1976.

[Cook84] Robert L. Cook. *Shade Trees*. Proceedings of ACM SIGGRAPH 84. *18 (3)*, pp. 223-231, 1984.

[Debevec98] P. Debevec. *Rendering Synthetic Objects Into Real Scenes*. ACM Computer Graphics (Proc. of SIGGRAPH '98) , 1998.

[Debevec02] Paul Debevec, Andreas Wenger, Chris Tchou, Andrew Gardner, Jamie Waese, and Tim Hawkins. *A Lighting Reproduction Approach to Live-Action Compositing*. Computer Graphics (Proc. Of SIGGRAPH '02), 2002.

[Doo78] D Doo and M Sabin. *Behaviour Of Recursive Division Surfaces Near Extraordinary Points*. Computer-Aided Design, 10:356–360, 1978.

[Drosopoulos00] Drosopoulos, A.; Xirouhakis, Y.; Delopoulos, A. *Optical camera tracking in virtual studios: degenerate cases*. Pages: 1114- 1117. Proceedings. 15th International Conference on Pattern Recognition, Vol.3, 2000.

[Fernandez02] Sebastian Fernandez, Kavita Bala, Donald Greenberg. *Local Illumination Environments for Direct Lighting Acceleration*. Thirteenth Eurographics Workshop on Rendering, Pisa, Italy. June, 2002.

[Fernando02] Pemith Randima Fernando. Adaptive Techniques for Hardware Shadow Generation. Master's Thesis. Cornell Univeristy. 2002.

[Gibbs98] Gibbs, S.; Arapis, C.; Breiteneder, C.; Lalioti, V.; Mostafawy, S.; Speier, J. *Virtual Sudios: An Overview*. Multimedia, IEEE, Vol.5, Iss.1, Pages: 18- 35, 1998

[Gortler96] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. *The Lumigrap*. Computer Graphics (Proc. Of SIGGRAPH '96), 1996.

[Hayashi98] Hayashi, M. *Image Composition Based On Virtual Cameras*. Multimedia, IEEE, Pages: 36- 48. Vol.5, Iss.1, 1998.

[Iddan98] G.J. Iddan and G. Yahav. *3D Imaging in the Studio (and elsewhere…)*. SPIEVol 4298. pp. 48.

[Isaacs87] Paul M. Isaacs and Michael F. Cohen. *Controlling dynamic simulation with kinematic constraints*. Computer Graphics (Proc. Of SIGGRAPH '87), 1987.

[Joy99] Kenneth I. Joy. *On-Line Geometric Modeling Notes: Catmull-Clark Surfaces*. Department of Computer Science. University of California, Davis. 1999.

[Kogler98] Kogler, R.A. *Virtual Set Design*. Multimedia, IEEE, Pages: 92- 96, Vol.5, Iss.1, 1998.

[Lee00] Aaron Lee, Henry Moreton, Hugues Hoppe. *Displaced Subdivision Surfaces*. Proceedings of ACM SIGGRAPH 2000. pp. 85-94, 2000

[Levoy96] Marc Levoy , Pat Hanrahan, *Light Field Rendering,* Proceedings of ACM SIGGRAPH 96. 1996.

[Lindenmayer68] A. Lindenmayer. *Mathematical models for cellular interaction in development*. Journal of Theoretical Biology 18: 280-315 1968.

[Lowell99] Ross Lowell. *Matters of Light & Depth*. Lowel-Light Manufacturing, Inc. 1999.

[Orad97] Orad Hi Tec Systems Limited. *United States Patent: Method and apparatus for determining the position of a TV camera for use in a virtual studio*. US Patent # 6,304,298. 1997.

[PCG02] Program of Computer Graphics. Real-Time Global Illumination Project Web Site: http://www.graphics.cornell.edu/~bjw/. Cornell University. 2002.

[Pixar89] Pixar. RenderMan Interface Specification. 1989.

[Potmesil81] M. Potmesil and I. Chakravarty, "*A Lens and Aperture Camera Model for Synthetic Image Generation",* ACM Computer Graphics (Proc. Siggraph 81), ACM Press, New York, 1981, pp 298-306.

[Primatte96] Primatte S-100 Manual/Tutorial. www.photron.com. 1996.

[Prusinkiewicz96] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The Algorithmic Beauty of Plants*. Springer-Verlag. New York. 1996.

[Rademacher01] Rademacher,P, Lengyel,J., Cutrell, E. and Whitted, T. 2001. *Measuring the perception of Visual Realism in Images*. Rendering Techniques 2001, 235--248, 2001.

[Reactor01] Reactor Reference and Tutorial Manual. Autodesk, Inc. Pp. 61. 2001.

[Reeves83] W. T. Reeves, *Particle Systems - A Technique for Modeling a Class of Fuzzy Objects*, Computer Graphics, vol. 17, no. 3, pp 359-376, 1983.

[Reflecmedia02] Lite Ring Data Sheet. www.reflecmedia.com. 2002.

[Reynolds87] Craig W. Reynolds. *Flocks, herds and schools: A distributed behavioral model.* ACM Computer Graphics (Proc. Siggraph 87), ACM Press, New York, 1987.

[Rickitt00] Richard Rickitt. *Special Effects, the history and techniques*. Billboard Books, 2000.

[Scott90] Scott, M.W. *Range imaging laser radar*. US Patent # 4,935,616. 1990.

[Sederberg98] Thomas W. Sederberg, Jianmin Zheng, David Sewell, Malcolm A. Sabin. *Non-Uniform Recursive Subdivision Surfaces*. Proceedings of ACM SIGGRAPH 98. pp. 387-394, 1998.

[Segal92] Mark Segal, Carl Korobkin, Rolf van Widenfelt, Jim Foran and Paul Haeberli. Fast shadows and lighting effects using texture mapping. ACM SIGGRAPH 92. pp. 249-252. 1992.

[Soler98] Cyril Soler and François X. Sillion. *Fast calculation of soft shadow textures using convolution.* Proceedings of the 25th annual conference on Computer graphics and interactive techniques. ACM Press. 1998.

[Ultimatte02a] Ultimatte Overview Manual. www.ultimatte.com. 2002

[Ultimatte02b] Ultimatte Technical Bulletin #1, *Matching Foreground and Background in a Composite.* www.ultimatte.com. 2002.

[Veach97] Eric Veach and Leonidas J. Guibas. *Metropolis light transport.* Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press. 1997.

[Walter99] Bruce Walter, G. Drettakis, and S. Parker. *Interactive rendering using the render cache.* Eurographics Rendering Workshop. 1999.

[Ward94] Gregory J. Ward, *The RADIANCE lighting simulation and rendering system.* Proceedings of the 21st annual conference on Computer graphics and interactive techniques. ACM Press. 1994.

[Welsh96] Welch, Greg. *SCAAT: Incremental Tracking with Incomplete Information.* Ph.D. Dissertation , University of North Carolina at Chapel Hill, Department of Computer Science, TR 96-051. 1996.

[Welch99] Welch, Greg, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, D'nardo Colucci. *The HiBall Tracker: High-Performance Wide-Area Tracking for Virtual and Augmented Environments.* Proceedngs of the ACM Symposium on Virtual Reality Software and Technology. 1999.

[Ziv77] Ziv, J. Lempel, A. *A Universal Algorithm for Sequential Data Compression.* IEEE Transactions on Information Theory, IT-23:337-343. 1977.

[Zordan02] Victor B. Zordan and Jessica K. Hodgins. *Motion capture-driven simulations that hit and react.* Proceedings of ACM SIGGRAPH 02. 2002.