Image-based 3D Scanning System using Opacity Hulls

by

Wai Kit Addy Ngan

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2003

Author . . . .
Department of Electrical Engineering and Computer Science
May 11, 2003

Certified by . . . .
Leonard McMillan
Associate Professor
Thesis Supervisor

Accepted by . . . .
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Image-based 3D Scanning System using Opacity Hulls

by

Wai Kit Addy Ngan

Submitted to the Department of Electrical Engineering and Computer Science
on May 11, 2003, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science and Engineering

Abstract

We have built a system for acquiring and rendering high quality graphical models of objects that are typically difficult to scan with traditional 3D scanners. Our system employs an image-based representation and can handle fuzzy materials such as fur and feathers, and refractive materials like glass. The hardware setup of the system include two turntables, two plasmas displays, a fixed array of cameras and a rotating array of directional lights. Many viewpoints can be simulated by rotating the turntables. By an automatic process, we acquire opacity mattes using multi-background techniques. We introduce a new geometric representation based on the opacity mattes, called the *opacity hull*. It is an extension of the visual hull with view-dependent opacity, and it significantly improves the visual quality of the geometry and allows seamless blending of objects into new environments. Appearance models based on the surface reflectance fields are also captured by a hybrid sampling of the illumination environment. The opacity hull, coupled with the reflectance data, can then be used to render the object in novel lighting environments from arbitrary viewpoints photo-realistically. This system is the first to acquire and render surface reflectance fields under arbitrary illumination from arbitrary viewpoints.

Thesis Supervisor: Leonard McMillan
Title: Associate Professor

Acknowledgements

I would like to thank my advisor Professor Leonard McMillan. His mentoring and support are important to me during my first year here at MIT. I would like to thank Wojciech Matusik, who has been a great partner in the related projects the past year and has given me enormous help. His relentless effort in his research has inspired me to work (at least try to) as hard as him. Thanks also go to Hanspeter Pfister for his many ideas and advices for the project, and Remo Ziegler for the infinite amount of time he spent on rendering our objects. I would also like to thank the Computer Graphics Group at MIT, and colleagues at MERL for their support. Finally I want to thank my family and my girlfriend Joyce for always being there for me during the difficult times.

Contents

List of Figures

Chapter 1

Introduction

Creating faithful 3D models manually demands considerable time and expertise. This can be a bottleneck for many practical applications. For 3D models to become ubiquitous, one important milestone would be to make 3D content creation as easy as taking a picture with a digital camera. Currently, it is both difficult to model shapes with complex geometry and to recreate a complex object's appearance using standard parametric reflectance models. 3D scanning is the most straightforward way to create models of real objects, and these capturing techniques have been employed frequently in the gaming and movie industry. An ideal 3D scanning system would acquire any object automatically, and construct a detailed shape and appearance model sufficient to be rendered in an arbitrary environment with new illumination from arbitrary viewpoints. Scalability is also important as different applications require a vastly different level of details for the 3D model, both in terms of geometry and appearance.

Although there has been much recent work towards this goal, no system to date satisfies the requirements of every application. Many current acquisition systems require substantial manual involvement. Many methods, including most commercial systems, focus only on capturing accurate geometry. In cases when the reflectance properties of 3D objects are captured they are usually fitted to parametric reflectance models, which often fails to represent complex materials and does not model important non-local effects such as interreflections, self-shadowing, translucency, and subsurface scattering. These difficulties suggest that image-based method would be an viable alternative. Image-based methods do not rely on any physical models, they merely interpolate between observations from cameras. As the images are produced from real cameras, the aggregate effects of all light transport are captured, taking care of general reflectance modeling and all other nonlocal effects, and they are by definition photorealistic. However, previous image-based methods all have some limitations, such as lack of 3D geometry, only support static illumination, or allow rendering from only a few viewpoints.

We have developed a prototype of an image-based automatic 3D scanning system that strikes a unique balance between acquiring accurate geometry and accurate appearance. It is very robust and capable of fully capturing 3D objects that are difficult, if not impossible, to scan with existing scanners (see Figure 1). Coarse geometry based on the visual hull and sample-based appearance are captured at the same time. Objects with fine-detail geometry, such as fuzzy objects, are handled correctly with acquired pixel-level opacity. Transparent and refractive objects can also be handled. The system automatically creates object representations that produce high quality renderings from arbitrary viewpoints, either under fixed or novel illumination. The system is built from off-the-shelf components. It uses digital cameras, leveraging their rapid increase in quality and decrease in cost. It is easy to use, has simple set-up and calibration, and scans objects that fit within a one cubic foot volume.

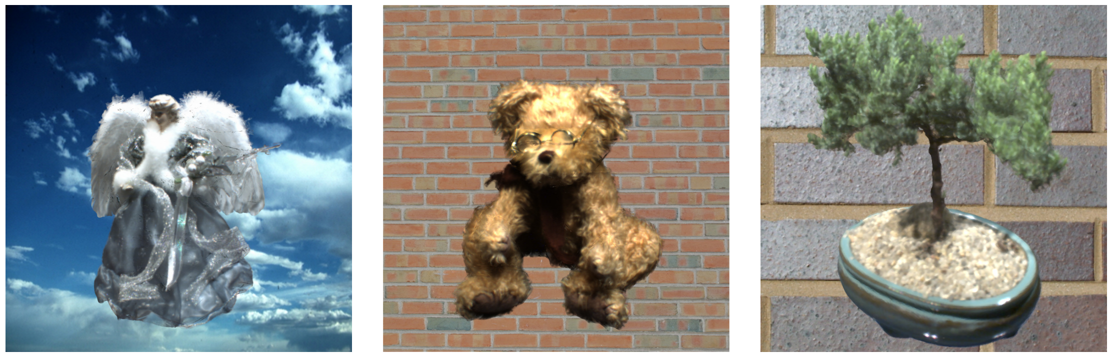


Figure 1-1: Renderings of acquired objects with a mixture of highly specular and fuzzy materials.

The main system has been described in a paper jointly authored with Matusik et al [34]. The environment matting extension to the system that facilitate acquisition of refractive/transparent objects is described in a follow-up paper [35]. In this thesis, a more thorough description and discussion of the system and the results will be given. The main contribution of this thesis includes:

- An automatic system that acquire surface light fields and surface reflectance fields.
- Opacity Hull, a novel extension on the visual hull, which enhances geometric detail by view-dependent opacity.
- Compression of the acquired surface reflectance data using principal components analysis(PCA).
- Separation of the illumination field into high- and low-resolution partitions for efficiently employing environment matting techniques for the capturing of refractive objects.
- Rendering of the acquired view-dependent data under novel illumination, which include smooth interpolation of environment mattes.

A review of background and previous work in the related fields will be given in Chapter 2. We will describe the acquisition process in Chapter 3, including the hardware setup, the opacity mattes computation using multi-backgrounds techniques, and the acquisition of appearance images. In Chapter 4 we will introduce the *opacity hull*, a new shape representation based on the visual hull, especially suited for objects with complex small-scale geometry. In Chapter 5 we will describe the appearance modeling based on the surface light fields and the surface reflectance fields. In Chapter 6 we will describe the rendering method of our point-based model and the scheme for interpolating between acquired viewpoints. Quantitative results with discussion will be presented in Chapter 7. Finally, in Chapter 8 we will conclude the thesis, and propose possible directions for future research.

Chapter 2

Background

In this chapter we will give an overview of the background and previous works relevant to the thesis, such as geometry digitization, reflectance modeling, image-based modeling and rendering, and image matting and compositing techniques.

In Section 2.1 we will briefly look at different methods of acquiring geometry including passive and active techniques, and the visual hull technique we employ. In Section 2.2 we will describe previous works in image-based modeling and rendering. Then in Section 2.3 we will give an overview of the acquisition and modeling of surface reflectance. In Section 2.4 we will describe image matting, where the foreground object with *partial transparency* is extracted from an image. In Section 2.5 we will describe environment matting, which generalize traditional image matting to incorporate effects like reflections and refractions from the environment.

2.1 Geometry Acquisition

There are many approaches for acquiring high quality 3D geometry from real-world objects, and they can be mainly classified into passive and active approaches. Passive approaches do not interact with the object, whereas active approaches make contact with the object or project light onto it. Passive methods attempt to extract shape, including shape-from-shading for single images, stereo triangulation for pairs of images, and optical flow methods for video streams. These methods typically do not yield dense and accurate digitizations, and they are often not robust in cases when the object being digitized does not have sufficient texture. Most passive methods assume that the BRDF is Lambertian or does not vary across the surface. They often fail in the presence of subsurface scattering, inter-reflections, or surface self-shadowing.

Active digitizing methods include contact digitizers and optical digitizers. Contact digitizers typically employ calibrated robot arms attached to a narrow pointer. The angles at the joints of the arm give an accurate location of the pointer at all times, and by making a contact on the surface with the pointer, it is possible to record 3D locations. This method is very flexible but slow and usually requires human effort.

Active digitizing based on optical reflection, such as laser range scanners, are very popular. The survey by Besl [4] gave a very comprehensive survey of optical rangefinding. Of the many available methods, the systems based on triangulation are the most popular. The object is typically illuminated by a laser sheet and imaged by a camera. As the laser source and the camera positions are both known, depth information can be calculated based on geometric intersection of the eye ray and the laser beam. The entire shape of the object can then be acquired by either translating or rotating the object, or by moving the laser beam. The depth values are recorded as *range images*, which can then be used to reconstruct the surface using various techniques. Digitizers based on triangulation techniques can be used with a large range of object sizes, and they have been employed to acquire very large models with fine resolution [29, 44].

However, all optical digitizers place restrictions on the types of materials that can be scanned, as discussed by Hawkins et al [21]. For example, fur is very hard to digitize since it does not have a well defined surface. The laser would scatter through the fur, failing to provide a clear depth value. Specular materials also present problems as the laser is dominantly reflected in the specular direction and cannot be observed by the range camera from other directions. Hence, specular objects often need to be painted with white diffuse material before scanning. Similarly, transparent objects, or objects that exhibit significant surface scattering, are very difficult for laser scanners. Moreover, these methods also require a registration step to align separately acquired scanned meshes [48, 13] or to align the scanned geometry with separately acquired texture images [3]. Filling gaps due to missing data is often necessary as well. Systems have been constructed where multiple lasers are used to acquire a surface color estimate along the line of sight of the imaging system. However, this is not useful for capturing objects in realistic illumination environments.

In order to support the broadest possible range of materials in our system, we have chosen to employ an image-based approach, and relax the requirement for accurate geometry. As our acquisition is always coupled with appearance data, the lack of accurate geometry can be compensated by view-dependent appearance. This relaxed requirement allows us to use a geometric representation based on the visual hull. The visual hull is constructed using silhouette information from a series of viewpoints, where the inside and outside of an object are identified. Each of the silhouette, together with the calibrated viewpoint, represents a cone-like shape in space that enclose the object. These cone-like shapes can then be intersected in space to produce a conservative shell that enclose the object. Matusik et al [33] described an image-based algorithm that render the visual hull in realtime based on multiple video streams, without explicitly constructing a representation of the actual geometry. Our system employ similar techniques to efficiently produce a point-sampled visual hull from the acquired opacity mattes.

It is clear that the fidelity of shape produced by the visual hull is worse than that of active digitization methods. However, the relatively simple and robust acquisition of silhouettes make the method much more general, handling all kind of materials including those that are fuzzy, specular or semi-transparent. Moreover, to enhance the visual quality of the geometry, we have devised a new representation called opacity hull, by extending the visual hull with view-dependent opacity information. The computation of the visual hull and the opacity hull will be described in more details in Chapter 4.

2.2 Image-based Modeling and Rendering

QuickTime VR [8] was one of the first image-based rendering system. A series of captured environment maps allow a user to look at all directions from a fixed viewpoint. Chen and Williams [9] investigated smooth interpolations between views by optical flow. McMillan and Bishop [36] proposed a framework of image-based methods based on the plenoptic function. The plenoptic function is a parameterized function that describes radiance from all direction from a given point. It is a 5D function parameterized by the location (x,y,z) in space and direction (θ,ϕ). Paraphrasing McMillan and Bishop, *given a set of discrete samples from the plenoptic function, the goal of image-based rendering is to generate a continuous representation of that function.* Light field methods [28, 20] observe that the plenoptic function can be described by a 4D function when the viewer move in unoccluded space. Lumigraph methods by Gortler et al. [20] improves on these methods by including a visual hull of the object for improved ray interpolation. However, all these methods assume static illumination and cannot be rendered in new environments.

The view-dependent texture mapping systems described by Pulli et al. [42] and Debevec et al. [17, 18] are hybrid image-based and model-based methods. These systems combine simple geometry and sparse texture data to accurately interpolate between the images. These methods are extremely effective despite their approximate 3D shapes, but they have some limitations for specular surfaces due to the relatively small number of textures. Also the approximate geometry they require can only be acquired semi-automatically.

Surface light fields [38, 52, 40, 22, 10] can be viewed as a more general and more efficient representation of view-dependent texture maps. On each of the surface vertex, the outgoing radiance is sampled from all directions, and the surface can then be rendered from arbitrary viewpoints by interpolating the sampled radiance data. Wood et al. [52] store light field data on accurate high-density geometry, whereas Nishino et al. [40] use a coarser triangular mesh for objects with low geometric complexity. As the radiance data are acquired from real photographs, surface light fields are capable of reproducing important global effects such as interreflections and self-shadowing. Our scanning system is capable of automatic surface light field acquisition and rendering (Section 5.1).

2.3 Surface Reflectance Acquisition and Modeling

Surface reflectance represents how light interacts at a surface and it relates to the radiance that eventually reaches the eye under different illumination. Reflectance is often described using the bidirectional reflectance distribution function (BRDF) [39], the ratio of the light incident at a surface point from each possible direction to the reflection in each possible direction. The BRDF is a 4D function, as direction in three-dimensional space can be specified with two parameters. Bidirection texture function (BTF) [14] is a 6D function that describes varying reflectance over a surface, and bidirectional surface scattering distribution function(BSSRDF) [39] is a 8D function that takes surface scattering into account, by allowing the outgoing location of a light ray to be different from the incident location.

Since the early days of computer graphics, numerous parameterized reflectance models of the BRDF have been proposed. There are phenomenological models which try to simulate natural phenomenon by devising a function with approximate behavior, of which the Phong model and its variants are the most well-known [5, 24]. Physically based models derive equations guided by the underlying physics, and the common ones include the Cook-Torrance model [12] and the Ashikhmin model [1]. All these models are designed with different goals in terms of efficiency and generality. Simpler models like the Phong model are quite limited in terms of the class of materials they can represent, but they are simple and, hence, often employed in realtime rendering. The more complex models are more general and are typically used during offline rendering. Nonetheless, even the most complicated parametric models do not encompass all the different kinds of BRDFs. Also, even when a model is general enough, recovering the parameters are generally non-trivial, even with human intervention.

Inverse rendering methods estimate the surface BRDF from images and geometry of the object. To achieve a compact BRDF representation, most methods fit parametric reflectance models to the image data with certain assumptions. Sato et al.[46] and Yu et al. [53] assume that the specular part of the BRDF is constant over large regions of the object, while the diffuse component varies more rapidly. Lensch et al. [27] partition the objects into patches and estimate a set of basis BRDFs per patch.

Simple parametric BRDFs, however, are incapable of representing the wide range of effects seen in real objects. Objects featuring glass, fur, hair, cloth, leaves, or feathers are very challenging or impossible to represent this way. As we will show in Chapter 7, reflectance functions for points in highly specular or self-shadowed areas are very complex and cannot easily be approximated using smooth basis functions.

An alternative is to use image-based, non-parametric representations for surface reflectance. Marschner et al. [32] use a tabular BRDF representation and measure the reflectance properties of convex objects using a digital camera. Georghiades et al. [19] apply image-based relighting to human faces by assuming that the surface reflectance is Lambertian.

More recent approaches [31, 15, 21, 23] use image databases to relight objects from a fixed viewpoint without acquiring a full BRDF. Debevec et al. [15] define the 8-dimensional *reflectance field* of an object as the radiant light from a surface under every possible incident field of illumination. To reduce the acquisition time and data size, they restrict the illumination to be non-local, or in other words only directional illumination is supported. This reduces the representation to a *non-local reflectance field*, which is six-dimensional. Debevec et al. further restricted the viewpoint to a single camera position to further reduce the dimensionality of the reflectance field to 4D. Human faces [15] and cultural artifacts [21] were captured and shown to be re-illuminated convincingly. However, these reflectance field approaches are limited to renderings from a single viewpoint. Our scanning system capture an appearance model based on the reflectance field from many viewpoints. We also extend the reflectance field capturing to sample part of the illumination environment at high-resolution to support transparent or refractive objects, using environment matting techniques. The details of our reflectance modeling will be described in Chapter 5.

2.4 Image Matting

The pixels of an image can be augmented in many ways to facilitate flexible image synthesis. Porter and Duff introduced an alpha channel at each pixel to allow images to be processed and rendered in layers and then combined together [41]. The first stage of our scanning involves acquiring such alpha channel, which we call the opacity mattes. The opacity matte describes the per-pixel opacity of the foreground object. An opacity value of 1 means the pixel is totally occupied by the foreground object, a value of 0 means the pixel does not coincide with the object, or equivalently the foreground is totally transparent. In-between values denote partial occupancy, either due to subpixel coverage or translucency.

Recovering an opacity matte from images is a classical problem well known as the *matting problem*. Vlahos [47] pioneered the algorithms to solve the problem assuming a constant colored background(typically blue), and that the foreground object is significantly different from that color. To formally define the problem, we follow the exposition of Smith and Blinn [47]:

For a pixel, given background color C_k and foreground color C_f, solve for α_o and C_o

$$C_f = C_o + (1 - \alpha_o) C_k \tag{2.1}$$

such that α_o and C_o gives the opacity and opacity-premultiplied color of the object respectively.

Here, the foreground image is a picture of the object and the background image is the same picture with the object removed. The same equation can also be used to composite the foreground onto another image, simply by replacing C_k with the new background. Smith and Blinn showed that the matting problem is under-specified with one set of background and foreground images. However, a general solution exist when two or more pairs of background/foregrounds are available. The solution to the matting problem with two pairs of background/foreground is as follows:

$$\alpha_o = 1 - \frac{\sum_{i=r,g,b}(C_{f1} - C_{f2})(C_{k1} - C_{k2})}{\sum_{i=r,g,b}(C_{k1} - C_{k2})^2}, \tag{2.2}$$

where C_{k1} and C_{k2} are the colors of the two different background pixels, and C_{f1} and C_{f2} are the corresponding foreground colors.

Our system use techniques based on Smith and Blinn's multi-background matting to acquire opacity mattes. Two plasma displays are placed on the opposite side of the cameras to provide a computer controlled background. Further details and discussion of our implementation will be given in Section 3.4.

2.5 Environment Matting and Compositing

Traditional image matting extract the foreground object with per-pixel opacity. While the opacity values accurately represent subpixel coverage for opaque objects, for refractive objects straight-through transparency is incorrect in general, as the direction of the light ray is altered due to refraction. Also, objects that do not exhibit perfectly specular transparency (e.g. stained glass) usually blur the background through refractions, and hence a general region of the environment can contribute to a single pixel. To deal with the general matting problems of this kind, Zongker et al. [55] developed the techniques of environment matting, a generalization of traditional matting techniques which also gives a description of how the foreground object refracts and reflects light from the scene. Chuang et al. [11] extended their techniques to produce more accurate results.

The environment matte describes how the illumination from the environment is attenuated and transported at each pixel, in addition to the traditional opacity and foreground/background colors. To capture such an environment matte, the object are surrounded with multiple monitors which can provide controlled color backgrounds. Many pictures of the object are then taken with different patterned backgrounds on the monitor. A naive approach would turn on one pixel of the background monitors at a time and record the response on the object by taking a picture. This would in theory give the most accurate result, but the capturing time and data size is clearly impractical. Hence both Chuang et al. and Zongker et al. employed coded background patterns to reduce the number of images they need to take to acquire the mattes. The main differences between the two methods lie in the patterns they used and the way they approximate the resultant mattes. Zongker et al. use square-wave stripe patterns of two colors in the horizontal and the vertical directions, according to one-dimensional Gray codes. Chuang et al. used sweeping white Gaussian stripes in the horizontal, vertical and diagonal directions. Zongker et al. assumed the contribution from the environment to a pixel can be modeled using a single axis-aligned rectangle, while Chuang et al. used a sum of several orientable Gaussian kernels, giving more accurate results. The remainder of the section will follow the exposition of Chuang et al., whose method is used in an extension of our system to capture a high-resolution reflectance field.

Following Blinn and Newell [6]'s exposition on environment mapping work, we can express the color of a pixel in terms of the environment, assuming the environment is at infinity:

$$C = \int W(\omega)E(\omega)d\omega \tag{2.3}$$

where $E(\omega)$ is the environment illumination in the direction ω, and $W(\omega)$ is a weighting function, and the integral is evaluated over all directions. W comprises all means of light transport through the pixel, which may include reflections, refractions, subsurface scattering, etc. However, the illumination model is directional, which means local illumination effects cannot be captured. For example, the occlusion pattern of a local light source is different from a directional one, so some of the self-shadowing effects cannot be modeled correctly.

Equation 2.3 can be rewritten as a spatial integral over some bounding surface, for example the monitors surrounding the object. As these monitors typically do not cover the entire sphere, we include a foreground color F, which describe color due to lighting outside the environment map, or emissivity. Equation 2.3 becomes

$$C = F + \int W(\mathbf{x})T(\mathbf{x})d\mathbf{x} \tag{2.4}$$

where $T(\mathbf{x})$ is the environment map parameterized on the pixel coordinates of the monitors. With this formulation, the problem of environment matting is then to recover the weighting function W. Zongker et al. represent W as an axis-aligned rectangle. Chuang et al. generalize it to a sum of Gaussians:

$$W(\mathbf{x}) = \sum_{i=1}^{n} R_i G_i(\mathbf{x}) \tag{2.5}$$

where R_i is an attenuation factor, and each G_i is an elliptical oriented 2D Gaussian:

$$G_i(\mathbf{x}) = G_{2D}(\mathbf{x}; \mathbf{c_i}, \sigma_\mathbf{i}, \theta_i) \tag{2.6}$$

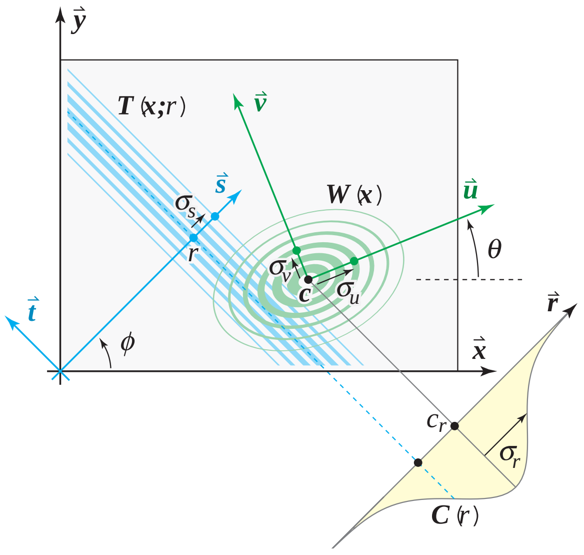


Figure 2-1: Illustration of the variables used in recovering an unknown elliptical, oriented Gaussian by sweeping out convolutions with known Gaussian stripes. As a titled stripe $T(x,r)$ of width σ_s and position r sweeps across the background, it passes under the elliptical Gaussian weighting function $W(x)$ associated with a single camera pixel. The camera records the integral, which describes a new observed function $C(r)$ as the stripe sweeps (Reprinted from [11] with author's permission).

where G_{2D} is defined as

$$G_{2D}(\mathbf{x};\mathbf{c},\sigma,\theta) = \frac{1}{2\pi\sigma_u\sigma_v} exp[-\frac{u^2}{2\sigma_u^2} - \frac{v^2}{2\sigma_v^2}] \tag{2.7}$$

with

$$u = (x-c_x)\cos\theta - (y-c_y)\sin\theta \ , v = (x-c_x)\sin\theta + (y-c_y)\cos\theta \tag{2.8}$$

where $\mathbf{x} = (x,y)$ are the pixel coordinates, $\mathbf{c} = (c_x,c_y)$ is the center of each Gaussian, $\sigma = (\sigma_u,\sigma_v)$ are the standard deviations in a local uv-coordinate system on the two axes, and θ is the orientation of the Gaussian.

To acquire the environment matte with this formulation, images of the object in front of a sequence of backdrops of swept Gaussian stripes are taken. The camera will, in theory, observe a evolving Gaussian, the result of the convolution of the weighting function W and the Gaussian backdrop (See Figure 2-1). The number of Gaussians, and the parameters for each Gaussian can then be solved using the optimization procedure described in Chuang

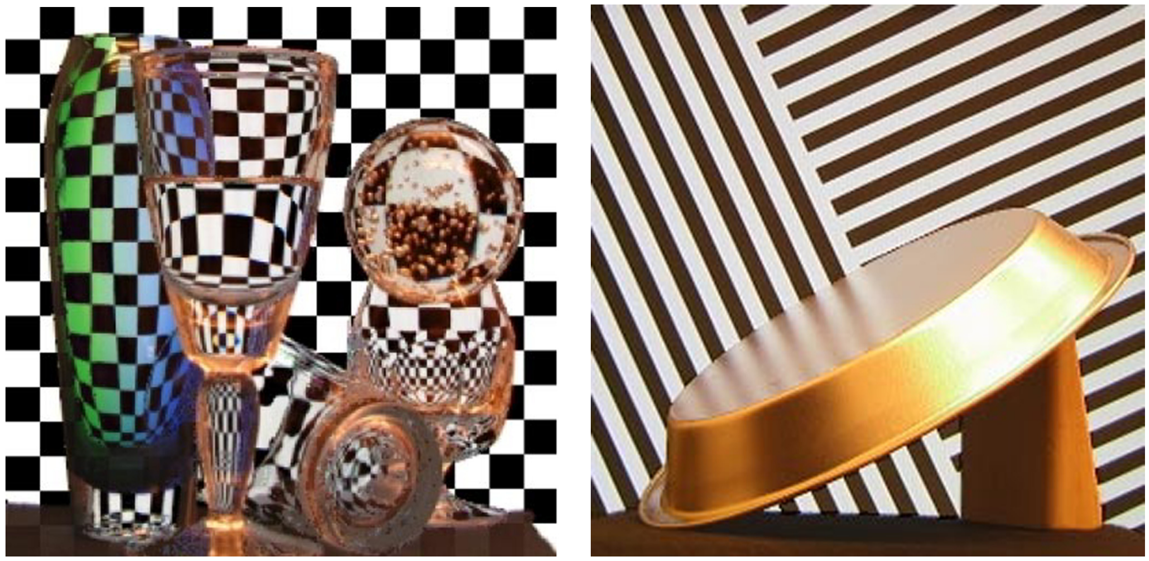


Figure 2-2: Some results from the high-accuracy method of Chuang et al. (Reprinted from [11] with author's permission).

et al. With each sweep we can observe the center and the spread of the Gaussian along that direction. By sweeping in three directions it is possible to locate multiple oriented Gaussians in the response. For further details of the optimization process the reader is referred to the original paper [11]. Figure 2-2 shows some results from [11], illustrating the accurate composite of some acquired reflective/refractive objects onto new backgrounds.

Our system optionally employs environment matting techniques for capturing of semi-transparent objects. Environment mattes are acquired from every viewpoint using the technique described in [11]. The incorporation of the environment mattes into our surface reflectance fields representation will be discussed in Section 5.3.

Chapter 3

Acquisition Process

In this chapter we describe the acquisition process of our scanning system. In Section 3.1 we describe our hardware setup, and an overview of the entire process will be given in Section 3.2. Finally we describe each pass of the acquisition, including: calibration (Sec 3.3), opacity(Sec 3.4), surface light fields(Sec 3.5), surface reflectance fields(Sec 3.6) and environment mattes(Sec 3.7).

3.1 Hardware Setup

Figure 3-1 shows a schematic diagram of our digitizer. The object to be scanned is placed on a plasma monitor that is mounted onto a rotating turntable. A second plasma monitor is placed vertically in a fixed position. The plasma monitors can provide controllable background patterns during scanning, which are used for both the opacity and environment matte acquisitions. There are six digital cameras fixed on a rig opposite to the vertical monitor, spaced roughly equally along the elevation angle of the upper hemisphere and pointing towards the object. By rotating the turntable at regular intervals, we can in effect take pictures of the object with arbitrary number of virtual cameras in the longitudinal dimension. Typically we use 5 or 10 degree interval to achieve a roughly uniform distribution of virtual cameras in both angular dimensions. An array of directional light sources is mounted on an overhead turntable and they are equally spaced along the elevation angle as well. Using this overhead turntable we can rotate the lights synchronously with the object to achieve

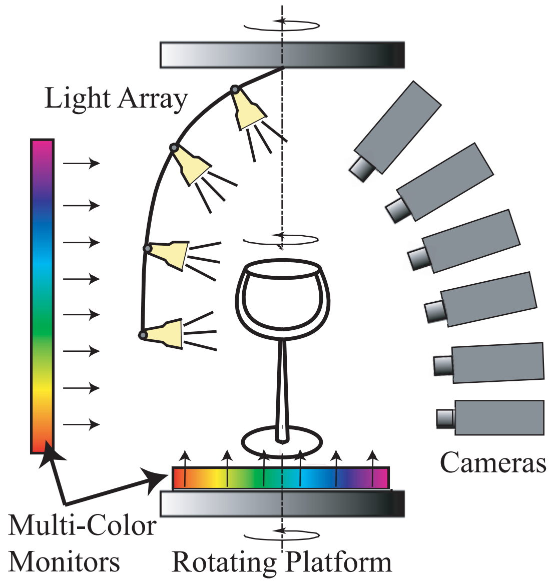


Figure 3-1: A schematic diagram of the system

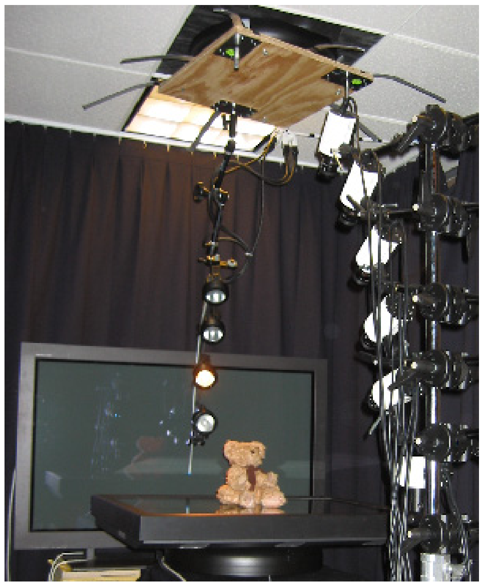


Figure 3-2: The actual 3D scanning system

fixed lighting with respect to the object, or we can rotate it for 360 degrees for each object position to capture the object under all possible lighting orientations.

Figure 3-2 shows a picture of the actual scanner. The two plasma monitors both have a resolution of 1024×768 pixels. We use six QImaging QICAM cameras with 1360×1036 pixel color CCD imaging sensors. The cameras are photometrically calibrated. They are connected via FireWire to a 2 GHz Pentium-4 PC with 1 GB of RAM. We alternatively use 15 mm or 8 mm C-mount lenses, depending on the size of the acquired object. The cameras are able to acquire full resolution RGB images at 11 frames per second.

The light array holds four light sources. Each light uses a 32 Watt HMI Halogen lamp and a parabolic reflector to approximate a directional light source at infinity. The lights are controlled by an electronic switch and individual dimmers through the computer. The dimmers are set once such that the image sensor do not exhibit blooming for views where the lights are directly visible.

In many ways, our system is similar to the enhanced light stage that has been proposed by Hawkins et al. [21] as future work. Hawkins et al. describe a *Evolved Light Stage*, which extend their single viewpoint acquisition of the reflectance field to many viewpoints. A key difference in our system is that we employ multi-background techniques for opacity matte extraction and we construct approximate geometry based on the opacity hull. Also, the number of viewpoints we acquire is less than that proposed by Hawkins et al. (16 cameras on the rig), due to limitation of acquisition time and cost. Nonetheless, the availability of approximate geometry and view-dependent opacity greatly extends the class of models that can be captured and improve the quality of viewpoint interpolation.

3.2 Overview of the process

During the acquisition process the following types of images are captured for each view, each of them in a different pass:

1. Image of a patterned calibration object. (1 per view)
2. Background images with the plasma monitors showing patterns. (several per view)
3. Foreground images of the object placed on the turntable, with the plasma monitors showing the same patterns as the background images. (same number as background images)
4. Radiance image showing the object under static lighting with respect to the object. (1 per view)
5. Reflectance images showing the object under all possible lightings. (number of lights per view)
6. Optional input images for computing environment mattes (300 per view, see Section 3.7)

As 1 and 2 are independent of the acquired object, we only need to capture them once if the cameras are not disturbed. We recalibrate whenever we change the lens of the camera, or inadvertently disturb any part of the system. One issue that arises during our scanning is that it is difficult to determine whether the calibration is still valid after some period of time, or even during one scan. If the digitizer is upset significantly, the subsequently computed visual hull will produce an inaccurate shape. However, we have no general method to detect these minor deviations.

Provided that we have the system calibrated and background images captured, we can start scanning. First we put the object onto the turntable, roughly centered at the origin. We capture the foreground images for all views by doing a full rotation of the bottom turntable. Then we repeat another full rotation of the bottom turntable to capture either the radiance or the reflectance images, depending on the type of data we want to acquire. Finally, if the object is transparent or refractive, we optionally apply another pass to capture input images for environment mattes from each viewpoint. It is clear that we require high precision and repeatability of the turntable to ensure registration between the subsequent passes. One way of validation is to composite the radiance/reflectance images, that are acquired during the last pass, using the opacity mattes that are constructed from the foreground images in the first pass. Misregistrations of more than a few pixels can be easily observed in the composite image, however we have no general way of detecting more minor discrepancies.

All radiance and reflectance images are acquired as high-dynamic images using techniques similar to that of Debevec et al [16]. Our cameras support raw data output so the relationship between exposure time and incident radiance measurements is linear over most of the operating range. We typically take pictures with four to eight exponentially increasing exposure times and calculate the radiance using a least-square linear fit to determine the camera's linear response. Our camera CCD has 10 bits of precision and hence has a range of 0 to 1023 for each pixel. Due to non-linear saturation effects at the extreme ends of the scale, we only use values in range of 5 to 1000 in our computation, while discarding the under- or over-exposed values. These computations are done on-the-fly during acquisition to reduce the storage for the acquired images. The high-dynamic images are stored in a RGBE format similar to Ward's representation [51]. A pixel is represented by four bytes (r, g, b, e), one each for the red, green, blue channels and one for an exponent. The actual radiance value stored in a pixel is then $(r, g, b) \times 2^{e-127}$. Our high-dynamic images are not compressed otherwise.

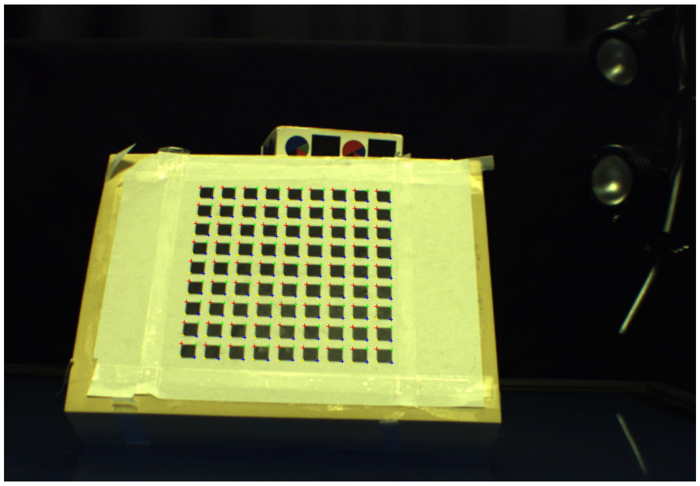


Figure 3-3: Image of the grid pattern for intrinsics , marked by the calibration software.

3.3 Calibration

To calibrate the cameras, we first acquire intrinsic camera parameters using Zhang's calibration method [54]. An object with a pattern of square grids is positioned in several different orientations and captured from each of the six cameras. Figure 3-3 shows one such image, with the corners of the squares tracked. With a list of these tracked points in the five images, the intrinsics can be computed accurately using the *Microsoft Easy Camera Calibration Tool* [37] which implemented Zhang's method. The intrinsics calibration is repeated each time that the lens or focus setting are changed.

To acquire the extrinsic parameters[1], we acquire images of a known calibration object, a multi-faceted patterned object in our case (Figure 3-4), from all possible views (216 views for 36 turntable positions with 6 cameras) . The patterns consist of multiple discs on each face, each uniquely colored. Using Beardsley's pattern recognition method [2], we can locate and identify the projections of each of the unique 3D points in all camera views from which they are visible.

The extrinsic parameters are then computed by an optimization procedure that uses the correspondences of the projections of the 3D points. Most of these points typically are visible from at least a third of all views and in each view about a hundred points can be observed. With a rough initialization of the camera extrinsics based on the rotational sym-

[1]These include the position of the center of projection, and the orientation of the camera.

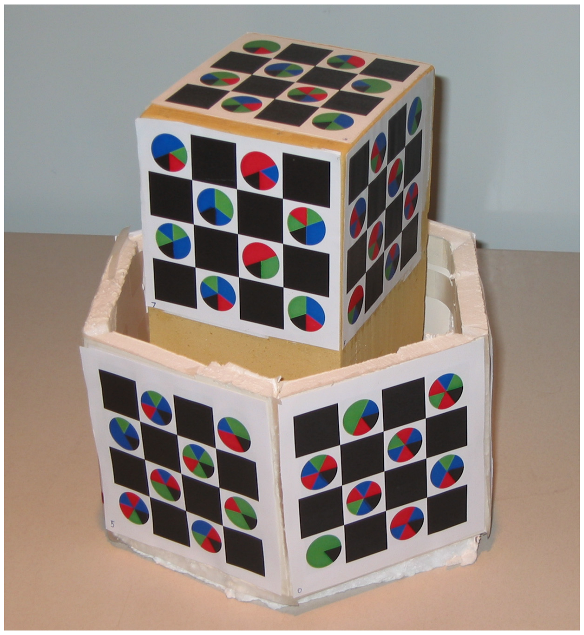


Figure 3-4: The calibration object with uniquely colored discs.

metry of our virtual cameras the optimization converges fairly quickly, and the reprojection error is consistently less than 1 pixel.

3.4 Opacity Acquisition

In order to reconstruct the visual hull geometry of the object, we need to extract silhouette mattes from all viewpoints, and distinguish between the object and the background. Backlighting is a common segmentation approach that is often used in commercial two-dimensional machine vision systems. The backlights saturate the image sensor in areas where they are visible. The silhouette images can be thresholded to establish a binary segmentation for the object.

However, binary thresholding is not accurate enough for objects with small silhouette features such as fur. It does not allow sub-pixel accurate compositing of the objects into new environments. To produce more precise result, we want to compute per-pixel opacity(commonly known as the alpha channel) as a continuous value between 0 and 1. This is commonly known as the matting problem. As noted by Smith and Blinn [47], the problem is under-determined for one pair of background and foreground pixels and thus do not have a general solution. They formulated the matting problem with multiple background/foreground pairs algebraically and deduced a general solution. For two pairs of background/foreground images,

$$\alpha_o = 1 - \frac{\sum_{i=r,g,b}(C_{f1} - C_{f2})(C_{k1} - C_{k2})}{\sum_{i=r,g,b}(C_{k1} - C_{k2})^2}, \tag{3.1}$$

where C_{k1} and C_{k2} are the colors of the two different background pixels, and C_{f1} and C_{f2} are the corresponding foreground colors. The only requirement for this formulation is that C_{k1} differs from C_{k2} so that the denominator is non-zero. If we measure the same color at a pixel both with and without the object for each background, Equation 3.1 equals zero. This corresponds to a transparent pixel that maps through to the background.

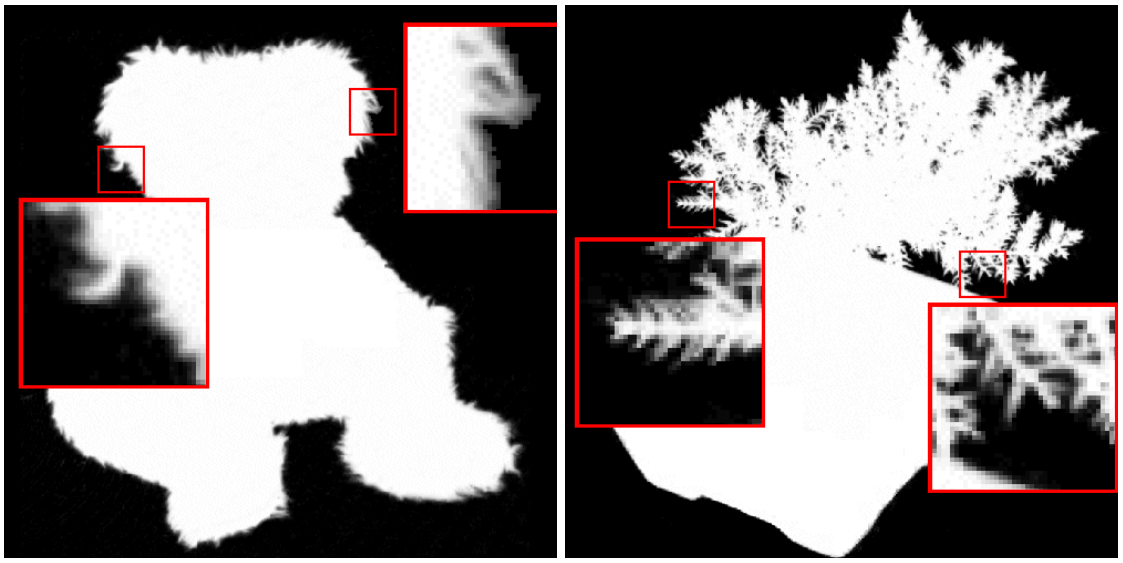


Figure 3-5: Alpha mattes acquired using multi-background techniques.

Based on Equation 3.1, we can compute opacity mattes using two pairs of background and foreground images. After the calibration images are acquired, we capture two more images from each viewpoint with the plasma monitors displaying different patterned backgrounds. Then the object is placed on the turntable, and a second pass capture the two sets of foreground images with the same patterns displayed on the plasma monitors as in the first pass. The per-pixel opacity value can then be computed for every viewpoint using Equation 3.1. Figure 3-5 shows two alpha mattes acquired with our method.

One common problem in matte extraction is color *spill* [47] (also known as blue spill), the reflection of the backlight on the foreground object, which confuses the matting algorithm between these spill reflections and the background. As a result, opaque region of the foreground object would be considered as semi-transparent, and in general opacity values will be lowered because of the color spill. Spill typically happens near object silhouettes because the Fresnel effect increases the specularity of materials near grazing angles. With a single color active backlight, spill is especially prominent for highly specular surfaces, such as metal or ceramics, as the reflection is typically indistinguishable from the background.

To reduce the color spill problem, we use a spatially varying color background pattern. The varying color reduces the chance that a pixel color observed due to spill matches the pixel color of the straight-through background. Furthermore, to make sure Equation 3.1 is well-specified, we choose the two backgrounds to be dissimilar in color-space everywhere such that the denominator remains above zero. Specifically, we use the following sinusoidal pattern:

$$C_i(x,y,n) = (1 + n\sin(\frac{2\pi(x+y)}{\lambda} + i\frac{\pi}{3})) \times 127. \tag{3.2}$$

where $C_i(x,y,n)$ is the intensity of color channel $i = 0,1,2$ at pixel location (x,y), and λ is the wavelength of the pattern. $n = -1$ or 1 gives the two different patterns. The user defines the period of the sinusoidal stripes with the parameter λ. Indeed, patterns defined this way yield a constant denominator to Equation 3.1 everywhere.

Nevertheless, with this varying pattern we still observe spill errors for highly specular objects and at silhouettes. If λ is too big, the pattern has a low frequency and spill reflections at near silhouettes will still produce pixels with very similar colors to the straight-through background, since reflections at grazing angles only minimally alter the ray directions. However, if λ is set to be too small our output are often polluted with noise, possibly due to quantization errors of the plasma displays and the camera sensors.

To reduce the spill errors we apply the same matting procedure multiple times, each time varying the wavelength λ of the backdrop patterns. Similar to Smith and Blinn's observations, we observe that in most cases the opacity of a pixel is under-estimated due to color spill. So for the final opacity matte we store the maximum α_o from all intermediate mattes. We found that acquiring three intermediate opacity mattes with relatively prime periods $\lambda = 27, 40$ and 53 is sufficient for the models we captured. The time overhead of taking the additional images is small as the cameras have a high frame rate and the plasma monitors can change the patterns instantaneously.

Spill problems are not completely eliminated, but our method is able to significantly reduce its effect. Also, while common matting algorithms acquire both the opacity(α_o) and the uncomposited foreground color(C_o) at the same time, we acquire only the opacity values. The color information of the foreground object is captured in another pass with different illuminations, where the surroundings of the object are covered with black felt. Thus, even though our opacity values could be inaccurate due to color spills, we avoid the foreground object color from being affected by the undesired spill reflections.

3.5 Radiance Images

After we acquire opacity mattes for all viewpoints, we cover the plasma monitors with black felt, with extra care not to disturb the object. This is mainly to avoid extra reflections of the lights on the plasma monitor surfaces, which could then act as undesired secondary light sources.

Multiple lights can be set up on the overhead turntable to achieve the desired illumination. We have three mounting positions on the overhead turntable with rigs attached and lights can be mounted on these rigs. We then acquire high dynamic range *radiance images* of the illuminated object at each viewpoint, by doing a full rotation of the turntables. The object turntable and the overhead turntable are rotated synchronously so the illumination is fixed with respect to the object. These images are stored in RGBE format and will be used to construct the surface light field, described in more details in Section 5.1.

3.6 Reflectance Images

Alternatively, we can acquire objects under variable illumination conditions from every viewpoint to construct a representation that can be re-illuminated under novel environment.

For each object turntable position, we rotate the light rig around the object. For each light rig position, each of the four lights are sequentially turned on and a high dynamic range image is captured from each of the six cameras. We call this set of images *reflectance* images as they describe the reflections[2] from the object surface as a function of the input illumination environment. All the reflectance images are stored in RGBE format, which will be later used to construct the *surface reflectance field*. The reflectance model based on these reflectance images will be described in more detail in Section 5.2.

3.7 Environment Mattes

In [35] we extended our system to also handle transparent and refractive objects by employing the environment mattes, which provide high-resolution sampling of our illumination environments.

We use the plasma monitors as high-resolution light sources that complement the light rig. Ideally, the entire hemisphere would be tiled with plasma monitors to produce a high-resolution lighting environment everywhere. However, this would require many more plasma monitors, or fixing the monitors on a rotating platform. Both methods are costly and mechanically challenging to build. Also, the size of the acquired data will be enormous. Thus, we make a simplifying assumption that most transparent objects refract rays from roughly behind the object, with respect to the viewpoint. With this assumption, we split the lighting hemisphere into high- and low-resolution area, Ω_h and Ω_l respectively (see Figure 5-2). Note that this separation is viewpoint dependent, Ω_h is always on the opposite side of the cameras. We measure the contribution of light from Ω_l by rotating the light rig to cover the area and capture the reflectance images as described in the previous section (Section 3.6). The contribution of light from Ω_h is measured using the environment matting techniques.

The acquisition process for environment matting involves taking multiple images of the foreground object in front of a backdrop with a 1D Gaussian profile that is swept over time

[2]We use the terms reflection and reflectance loosely. Indeed the reflectance images capture all light paths that subsequently reach the camera. The paths include refraction, transmission, subsurface scattering, etc.

in horizontal, vertical, and diagonal direction. We uses a standard deviation of 12 pixels for our Gaussian stripes, and the step size of the sweeping Gaussian is 10 pixels. As the aspect ratio of our monitors are 16:9, we use a different number of backdrop images for each direction: 125 in diagonal, 100 in horizontal and 75 in vertical direction. Hence, 300 images are captured in total for a single viewpoint. This process is repeated for all viewpoints in an optional final pass.

However, for some positions in the camera array, the frames of the plasma monitors are visible, which gives incomplete environment matte and those viewpoints are practically unusable. Consequently, we only use the lower and the two upper most cameras for the acquisition of environment mattes. The lower camera can only see the vertical monitor, while the two upper ones can only see the rotating monitor. Hence, for 36 turntable positions, we only acquire environment mattes from 3×36 viewpoints. All the images are acquired as high dynamic images. Using these images as input, we solve for the environment matte for each viewpoint. For each pixel we solve for at most two Gaussians to approximate the weighting function W we described in Section 2.5.

$$W(\mathbf{x}) = a_1 G_1(\mathbf{x};\mathbf{C_1},\boldsymbol{\sigma_1},\theta_1) + a_2 G_2(\mathbf{x};\mathbf{C_2},\boldsymbol{\sigma_2},\theta_2) \tag{3.3}$$

We perform a non-linear optimization using the same technique of Chuang et al. [11] to compute the parameters for the two Gaussians. These parameters are then saved as environment mattes used later for rendering.

Chapter 4

Geometry Modeling

Many previous methods of acquiring 3D models have focused on high accuracy geometry. Common methods include passive stereo depth extraction, contact digitizers, and active light system. In particular, there has been a lot of success in structured light technologies such as laser range scanners based on triangulations. They reproduce the geometry of a wide range of objects faithfully with fine resolution. They have been employed to acquire very large models, for example in the Digital Michelangelo Project [29] and another project that digitized Michelangelo's *Florentine Pieta*.

However, active light methods are not suitable for all kinds of models. For example, specular or transparent objects are very hard for these methods as the laser cannot be detected in non-specular directions. Materials like fur or fabric, which have very fine geometry, are also very difficult to handle. More discussions of limitations of these scanning methods can be found in Section 2.1.

At the same time, it is often unnecessary to acquire high-precision geometry in order to render a convincing visualization, where the subtle complexity of the geometry could be ignored. The goal of our system is to scan a wide class of objects as is (i.e. without painting or any other preprocessing), without imposing any restrictions on the type of material or the complexity of the geometry. We achieve this goal by relaxing the requirement for accurate geometry. We represent the geometry of the model using a conservative bounding volume based on the *visual hull*. While the visual hull only provides an approximate geometry, we compensate by applying view-dependent opacity to reproduce fine silhouette features. In

Section 4.1 we will discuss the construction of the visual hull from our opacity mattes. In Section 4.2 we will discuss the opacity hull, which extend the visual hull by employing view-dependent opacity.

4.1 Visual Hull

Using silhouette information from calibrated viewpoints, we can construct a generalized cone emanating from the center of projection that enclose the object. Laurentini [25] introduced the visual hull as the maximal volume that is consistent with a given set of silhouettes. This volume can be produced by a intersection of all the cones constructed from the silhouettes of each viewpoint, and the result is the visual hull, which always conservatively encloses the object. As the number of viewpoints approaches infinity, the visual hull converges to a shape that would be identical to the original geometry of an object without concavities. The visual hull is easy to acquire, can be computed robustly, and it provides a good base geometry for us to map view-dependent appearance and opacity onto. Also, in cases when fuzzy geometry is present, the visual hull could be the only choice as accurate acquisition of the very fine geometry is difficult.

Our system acquires an opacity matte from each viewpoint during acquisition, using multi-background techniques as discussed in Section 3.4. To construct the visual hull, first we apply binary thresholding on the opacity mattes to produce binary silhouette images. We denote the opacity value of a pixel by the variable α, and it ranges from 0 to 1. Theoretically, each pixel with $\alpha > 0$ (i.e., not transparent) belongs to the foreground object. We use a slightly higher threshold because of noise in the system and calibration inaccuracies. We found that a threshold of $\alpha > 0.05$ yields a segmentation that covers all of the object and parts of the background in most cases.

In theory, we can compute a visual hull by volume carving or by intersecting the generalized cones analytically in object space. However, such direct computations are very expensive and may suffer from a lack of robustness. Volume carving methods typically sample the space uniformly with voxels, and intersection is achieved by carving away those voxels outside the projected silhouette cones of each viewpoint. The result is quantized due to the grid sampling, and the computation is costly when a high resolution is required.

Our system computes the visual hull efficiently using Matusik et al.'s image-based visual hull algorithm [33]. It produces more accurate geometry than volumetric methods as it does not quantize the volume, and it is more efficient than naive 3D intersection methods by reducing the intersections to 1D.

To compute the visual hull, we first build the silhouette contours from each silhouette images, represented as a list of edges enclosing the silhouette's boundary pixels. The edges are generated using a 2D variant of the marching cubes approach [30]. We then sample the object along three orthogonal directions, sending a user-specified number of rays in each direction, typically 512×512. We intersect each of the 3D ray with the visual hull. Since the intersection with the visual hull is equivalent to intersection with each of the generalized cones, the 3D cone intersection can be reduced to a series of cheaper 1D ray intersections with a 2D silhouette image. The 3D ray is projected to an silhouette image and intersected with the contours in that image. The intersection produces intervals which are inside the object according to the contours. The intervals are then lifted back to 3D object space. We repeat the intersection with all viewpoints and at the end we have a set of intervals along the ray direction which are inside the visual hull of the object. We repeat this procedure for all rays and the results of all the intersections is combined to form a point-sampled model of the visual hull surface, which is stored in a layered depth cube(LDC) tree [56].

The visual hull algorithm removes improperly classified foreground regions as long as they are not consistent with all other images. Hence it is tolerant to some noises in the silhouette image as the background regions incorrectly classified as foreground would be removed in most cases during the intersection with other viewpoints.

Finally, to allow correct view-dependent texturing, we precompute visibility information for each sampled point, which describes the set of viewpoints from which the sampled point on the visual hull is visible. This is computed efficiently using the visibility method from [33], and the result is stored in a visibility bit-vector for each surface point.

4.2 Opacity Hull - View-dependent opacity

Each point on the visual hull surface can be reprojected onto the opacity mattes to estimate its opacity from a particular observed viewpoint. When coupled with this view-dependent opacity, we can visually improve the geometric appearance of the object during rendering. For example, very fine geometric features like fur, which are not captured with the visual hull, present the aggregate effect of partial coverage in terms of opacity. We call this new representation the opacity hull.

The opacity hull is similar to a surface light field (see Section 5.1), but instead of storing radiance it stores opacity values in each surface point. It is useful to introduce the notion of an *alphasphere* $\mathscr{A}$. If ω is an outgoing direction at the surface point p, then $\mathscr{A}(p,\omega)$ is the opacity value seen along direction ω.

Figure 4-1 shows the observed opacity values for three surface points on an object for all 6×36 viewpoints. The opacity values are presented as 6×36 images, the x- and y-axis representing the latitudinal and the longitudinal indices of the virtual cameras respectively.

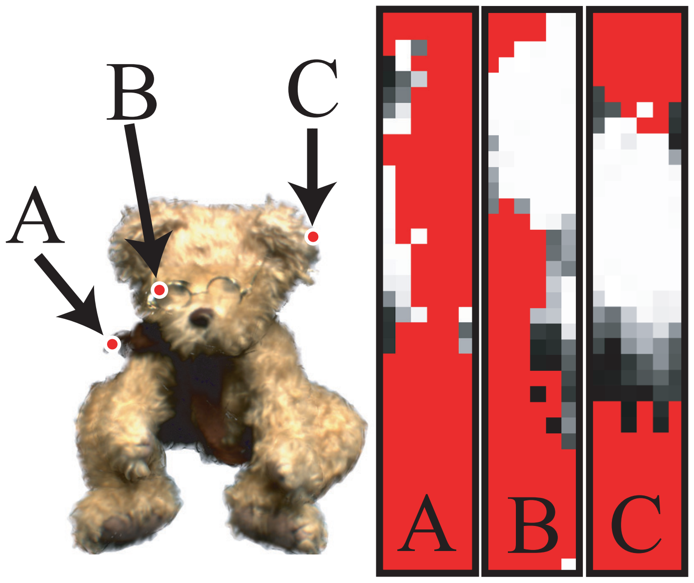


Figure 4-1: Observed alpha values for points on the opacity hull. Red color indicates invisible camera views.

Each pixel has been colored according to its opacity. Black corresponds to $\alpha = 0$, white corresponds to $\alpha = 1$, and grey corresponds to values in between. Red indicates camera views that are invisible from the surface point.

The function $\mathscr{A}$ is defined over the entire direction sphere. Any physical scanning system acquires only a sparse set of samples of this function. As is done for radiance samples of lumispheres in [52], one could estimate a parametric function for $\mathscr{A}$ and store it in each alphasphere. However, as shown in Figure 4-1, the view-dependent alpha is not smooth and not easily amenable to parametric function fitting. For example, Point A is on the scarf of the teddy bear, and in most views it is occluded. Point B is one of the surface point on the glasses and some views can see through the glasses directly to the background. These kinds of effects could lead to problems when trying to fit to a simple parametric model. Point C is a more typical point that lies on the fuzzy surface.

It is important to keep in mind that the opacity hull is a view-dependent representation. It captures view-dependent partial occupancy of a foreground object with respect to the background. The view-dependent aspect sets the opacity hull apart from voxel shells, which are frequently used in volume graphics [49]. Voxel shells are not able to accurately represent fine silhouette features, which is the main benefit of the opacity hull.

Also recognizing the importance of silhouettes, Sander et al. [45] use silhouette clipping to improve the visual appearance of coarse polygonal models. However, their method depends on accurate geometric silhouettes, which is impractical for complex silhouette geometry like fur, trees, or feathers. Opacity hulls are more similar to the concentric, semi-transparent textured shells that Lengyel et al. [26] used to render hair and furry objects. They augment their opacity model with a simple geometric proxy called textured fins to improve the appearance of object silhouettes. A single instance of the fin texture is used for all silhouettes of the object. In contrast, opacity hulls can be looked at as textures with view-dependent opacity for every surface point of the object. We can render silhouettes of high complexity using only visual hull geometry, as we will illustrate in Section 7.1.

In an attempt to investigate the viability of a parametric model, we studied the behavior of the view-dependent opacity on uniform materials. For example, we wrapped black felt on a cylinder and capture it with our system. We sampled the opacity densely at 1 degree intervals on the turntable, and thus have 360×6 views on the cylinder. We cannot achieve denser sampling on the latitudinal dimension as our physical setup limits us to only have six cameras. This would not cause a problem if we assume the opacity function is isotropic.

An analysis, with the isotropic assumption, failed to find a systematic model that fit the observed data well. This is clearly an avenue for future research. It is difficult to make conclusions based on our study, as our data could be polluted with noises. There are inaccuracies on the camera calibration and the turntable positions of each different pass. These errors are especially significant for the study of opacity as opacity is the aggregate result of tiny geometry and subpixel alignment in screen space. Also, as the visual hull is only a conservative bounding volume, the surface samples are generally not on the surface. Even on a model with uniform material and simple geometry, different samples would be in a range of different distances from the true surface. Possible future directions may consider a probabilistic study of the distribution of opacity functions on a surface.

Consequently, we do not try to fit our acquired opacity data to any parametric model, instead we store the acquired opacity mattes and use an interpolation scheme to render the opacity hull from arbitrary viewpoints (see Chapter 6).

Chapter 5

Reflectance Modeling

In this chapter we will discuss appearance acquisition of our system. Employing an image-based acquisition, our system is capable of capturing and representing a large class of objects regardless of the complexity of their geometry and appearance. Materials with anisotropic BRDFs, or global illumination effects like self-shadowing, inter-reflections and subsurface scattering are all handled implicitly through the use of real photographs of the object from many viewpoints.

In Section 5.1 we will describe the surface light field, which stores the view-dependent appearance of a surface captured under fixed illumination. Our system is capable of automatically acquiring surface light fields. In Section 5.2 we will discuss the modeling of the surface reflectance fields that incorporate variable illuminations to the appearance model. In Section 5.3 we will discuss the extension on our surface reflectance fields to handle transparent and refractive materials using environment matting techniques.

5.1 Surface Light Fields

A surface light field is a function that maps a surface point and an outgoing direction to a RGB color value [52, 38]. When constructed from observations of a 3D object, a surface light field can be used to render photorealistic images of the object from arbitrary viewpoints. They are well suited to represent material with complex view-dependent appearance which cannot be described easily with parametric BRDF models. Surface texture, rapid variation in specularity and non-local effects like inter-reflection, self-shadowing and subsurface scattering are all correctly handled.

Following the exposition of Wood et al [52], the surface light field can be defined as a 4D function on a parameterized surface

$$L : K_0 \times S^2 \to RGB \tag{5.1}$$

where K_0 is the domain of the parameterized surface, S^2 denotes the sphere of unit vectors in $\mathbb{R}^3$. Radiance is represented by points in $\mathbb{R}^3$ corresponding to RGB triples. Let ϕ be the parameterization of the surface. If $u \in K_0$ is a point on the base mesh and ω is an outward pointing direction at the surface point $\phi(u)$, then $L(u,\omega)$ is the RGB value of the light ray starting at $\phi(u)$ and traveling in direction ω.[1]

Using the opacity hull and the radiance images acquired by our system, we can produce a surface light field similar to that described by Wood et al [52]. They parameterized the surface light field on a high-precision geometry acquired from laser range scans. Instead, we use the opacity hull as an approximating shape, as described in Chapter 4. The opacity hull is a point-sampled model with view-dependent opacity associated with each point. During acquisition, we have acquired *radiance images* from every calibrated viewpoint under the desired fixed illumination, by rotating the overhead turntable synchronously with the captured object. Each of the surface point on the opacity hull can be projected to each of the radiance images where they are visible. Each of these projections then corresponds to a sample of the surface light field at the surface point, along the outgoing direction defined by the vector from the point to the camera.

In fact, we do not explicitly reparameterize our surface light field. Instead we keep the radiance images acquired from all the viewpoints, and they are used directly for lookup during rendering (see Chapter 6). To save storage, we discard regions of the images that do not cover the object. Each of the radiance images are partitioned into 8×8 pixel blocks. After the construction of the opacity hull, we project it back to each of the image and discard all the pixel blocks that do not coincide with the object. The blocks remained are

[1]Radiance is constant along a straight line in a non-participating medium.

then stored in an indexed table, allowing constant time retrieval of the data. We do not attempt to perform any further compression on the data.

However, compression of the surface light field data is clearly possible and indeed in some cases necessary to handle the large amount of data. Wood et al. demonstrate compression methods based on function quantization and/or principal components analysis. They achieve a compression ratio of about 70:1. Light field mapping techniques by Chen et al. [10] approximate the light field data by partitioning it over elementary surface primitives and factorizing each part into a product of two two-dimensional functions. Combining with hardware texture compression they are able to achieve up to 5000:1 compression. Our research has been more focused on the re-lightable surface reflectance fields, and the size of the uncompressed data for surface light fields are typically tractable(around 2 GB), as a result, compression methods were not applied. However, Vlasic et al. [50] have successfully employed light field mapping techniques to compress our data, and they were able to render our opacity-enhanced surface light fields in real time.

5.2 Surface Reflectance Fields

5.2.1 Modeling and Acquisition

Surface light fields can only represent models under the original illumination. To overcome this limitation, we acquire an appearance model that describes the appearance of the object as a function of variable illumination environment.

Debevec et al. [15] defined the *non-local reflectance field* as the radiance from a surface under every possible incident field of directional illumination. It is a six-dimensional function $R(P,\omega_i,\omega_r)$, where P is a surface point on a parametric surface, and ω_i and ω_r are the incident and outgoing directions respectively. It gives the radiance from P along the direction ω_r, when the object is subject to a directional illumination from ω_i. Notice that P can be in a shadowed region with respect to ω_i but still be lit through surface scattering. This is almost equivalent to the BSSRDF, except that the parametric surface is not assumed to coincide with the physical surface, and that the illumination is limited to be directional.

Hence the reflectance field cannot be rendered correctly under local light sources. However, generalizing to include local illumination effects will require two more dimensions and acquisition and storage will be highly impractical.

Debevec et al. use a light stage with a fixed camera positions and a rotating light to acquire such a reflectance field of human faces [15] and cultural artifacts [21]. Our work can be viewed as a realization of the enhanced light stage that has been proposed as future work in [21], where the reflectance field is sampled from many viewpoints. Also, as our system acquire geometry based on the visual hull and the reflectance field is parameterized on this surface, we call our representation the *surface reflectance field*.

During acquisition, we sample the surface reflectance field $R(P, \omega_i, \omega_r)$ from a set of viewpoints Ω_r and a set of light directions Ω_i. In previous approaches [15, 21, 23], the sampling of light directions is relatively dense (e.g., $|\Omega_i| = 64 \times 32$ in [15]), but only very few viewpoints are used. In our system, we sample the reflectance field from many view directions ($|\Omega_r| = 6 \times 36$). To limit the amount of data we acquire and store, our system uses a sparse sampling of light directions ($|\Omega_i| = 4 \times 15$).

For an observation of the surface reflectance field from an original viewpoint, it is useful to define a slice of the reflectance field called a *reflectance function* $R_{xy}(\omega_i)$ following the exposition of Debevec et al. [15]. P and ω_r are encoded by the pixel location (x,y) implicitly. It represents how much light is reflected to the camera through pixel (x,y) as a result of illumination on the object from direction ω_i. With this formulation, the light arriving at a camera pixel (x,y) can be described as:

$$C_{xy} = \int_{\Omega} R_{xy}(\omega_i) E(\omega_i) d\omega. \tag{5.2}$$

C_{xy} is the recorded color value at camera pixel (x,y), $E(\omega_i)$ is the environment illumination from direction ω_i and R_{xy} is the reflectance function, which comprises all means of transport from the environment through the object to the camera, contributing to the pixel (x,y). The integration is carried out over the entire sphere Ω and for each wavelength. It is important to integrate over all directions to account for non-local effects. We drop the wavelength dependency, assuming that all equations can be evaluated independently for the red, green and blue channels. Using this formulation, with known reflectance function R_{xy} we can reconstruct a pixel from the original viewpoint under novel illumination E' by evaluating Equation 5.2, substituting E by E'.

From here on we will drop the xy subscript, with the understanding that each pixel is acquired and modeled in the same manner. To measure R we discretely sampled the environment using directional lights and measure the response when each of the different light is turned on sequentially. It is a physical limitation of our system that we cannot acquire viewpoints or illuminations from below the turntable, so we only sample the upper hemisphere of the environment. In our setup, we have a light rig with 4 direction lights. By rotating the light rig around the object, we can achieve a variable number of lights on the longitudinal dimension. We typically use 15 positions for the light rigs, producing $4 \times 15 = 60$ discrete lights. The lights are photometrically calibrated and all have intensity L, and they all point towards the center of the object so that their subtended solid angles are equal. By taking a picture each time with a single light on from direction ω_i, we have an observation C_i:

$$C_i \approx R(\omega_i)L \tag{5.3}$$

The sampled value $R(\omega_i)$ is hence within a constant multiple of C_i. The image formed by C_i at each pixel is exactly what we capture as *reflectance images*, described in Section 3.6. For each viewpoint we have $4 \times 15 = 60$ reflectance images.

With the discrete sampling, we can approximate Equation 5.2 as

$$C = \sum_{i=1}^{n} R(\omega_i)E(\omega_i)dA(\omega_i) \tag{5.4}$$

As we have a very sparse sampling of the environment, our re-illumination is accurate only for relatively diffuse surfaces. For example, a focused highlight on a highly specular object will jump from one location to the next instead of moving smoothly under a moving light source. Based on Ramamoorthi and Hanrahan [43]'s formalization of inverse rendering under a signal-processing framework, BRDF recovery can be viewed as a deconvolution problem. Hence we can only recover the the high-frequency components of the BRDF when the lighting environment have similar frequencies. This applies similarly to the recovery of our reflectance field. With a sparse sampling of the environment, we can only acquire the illumination-dependent effects of relatively diffuse materials, and artifacts will be visible on very specular materials during rendering. To solve this problem, especially to handle transparent and refractive objects, we employ environment matting techniques to provide a much-denser sampling of part of the environment, which will be discussed in Section 5.3.

5.2.2 Compression

The raw reflectance image data would require about 76 GB of storage for 6×36 viewpoints. Storing only the pixel blocks within the object silhouette still would require between 20 and 30 GB, depending on the size of the object. To make this data more manageable, we use a simple compression scheme using principal component analysis (PCA). From each viewpoint, we divide the pixels into 8×8 blocks. For each block we have an observation under each of the 60 lights. Now we rearrange each observation under the ith light into a vector r_i of $8 \times 8 \times 3$ floats[2]. PCA compresses a set of vectors by finding the most significant vectors (called the principal components) that span the set with minimal distortion. Each element of the set can then be re-expressed as linear combinations of these principal components, and the coefficients are called the eigenvalues. As we add more principal components to our approximation, the error is monotonically decreasing. We apply PCA on the set of vectors $\{r_i\}$, and only store the first k principal components $(V_1, V_2, \ldots, V_k)$ such that the root-mean-squared error of the reconstruction is less than a threshold. We typically set the threshold to be 1% of the average radiance of all the reflectance images. After the compression, each block is represented as k principal components, and $60k$ eigenvalues (γ_{ij}) as we have k eigenvalues for each of the 60 lights. The block of the reflectance image under the ith light is approximated by:

$$C_i^{block} = \sum_{j=1}^{k} \gamma_{ij} V_j \tag{5.5}$$

[2]There are 8×8 floats for each of the R,G,B channels.

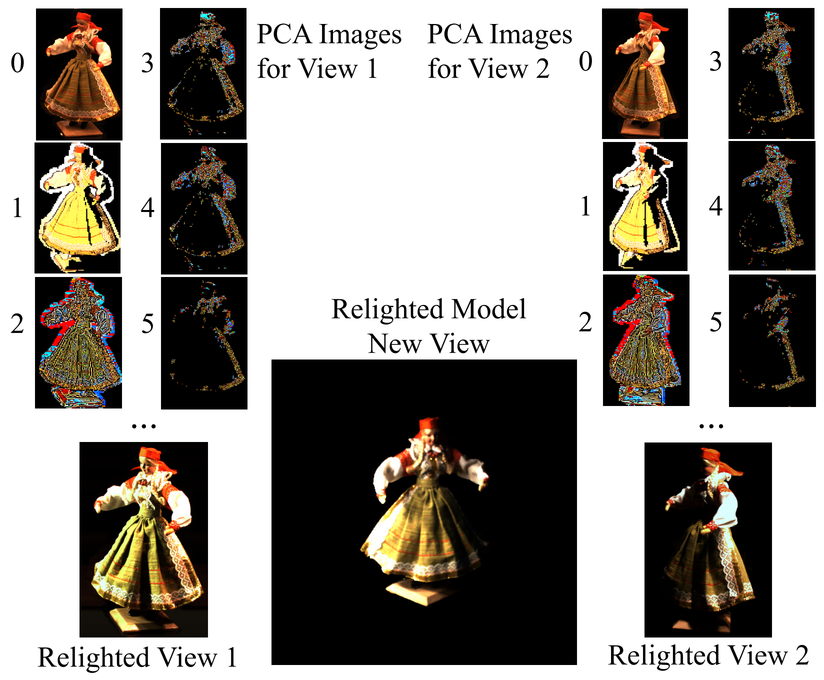


Figure 5-1: PCA decompositions of the reflectance functions from two viewpoints. The first six components are shwon from each viewpoint.

The average number of principal components for the models we captured is typically four to five per block and thus the size of the reflectance data is reduced by a factor of about 10. Figure 5-1 shows two views of a doll model, and their respective PCA decompositions. The components images (first six are shown, labeled 0 to 5) are falsely colored as the principal components contain negative values in general. Also, as each block has a variable number of principal components stored, some of the pixel blocks in the third or higher component images are empty. During rendering, we can recompute a radiance image of each view efficiently under a new illumination environment, and the final image from an arbitrary viewpoint can be interpolated from these radiance images. We will discuss the reconstruction and rendering methods in details in Chapter 6.

5.3 Surface Reflectance Fields with Environment Mattes

The surface reflectance field that we acquire allows us to put the scanned object in novel environment illuminated correctly. However, the very sparse sampling (4×15) of the illumination environment during acquisition limits the accuracy of the reconstruction. First,

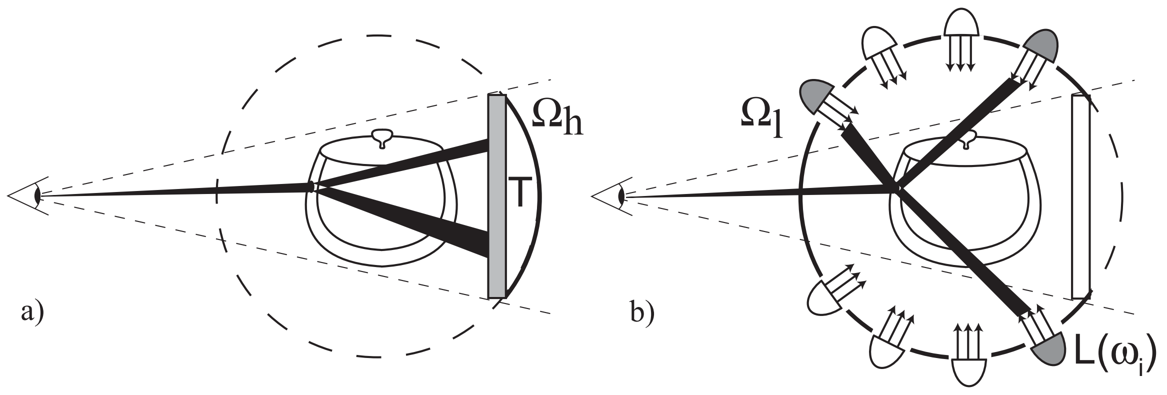


Figure 5-2: Illumination environment and light propagation model in our system. a) High-resolution sampling across Ω_h. b) Low-resolution sampling across Ω_l.

the object cannot be rendered accurately under high-frequency illumination environment[3]. Also we cannot handle very specular materials, as the specular highlight will be very focused and artifacts due to undersampling would be visible. One of the most difficult materials is glass, which both reflects and refracts light in a highly specular fashion. To capture such materials, we have to sample the illumination environment at a much higher frequency. As stated in Section 3.7, ideally we would use multiple plasma monitors to cover the entire hemisphere to achieve a high-resolution sampling everywhere. However, due to cost and efficiency constraints, we have chosen to employ a hybrid approach. We use the same plasma monitors used for multi-background matting to also act as high-resolution lights from behind and below the objects. The remaining regions of the hemisphere is covered by our light rig used in Section 5.2. Thus, the illumination hemisphere is partitioned into high- and low-resolution regions, Ω_h and Ω_l respectively (Figure 5-2). With this partitioning, we can rewrite Equation 5.2 as

$$C = \int_{\Omega_h} W(x)T(x)dx + \sum_{i=1}^{n'} R(\omega_i)E(\omega_i)dA(\omega_i). \tag{5.6}$$

The integral is across the high-resolution region Ω_h, while the region Ω_l is sampled the same way as in Section 5.2. We rename the reflectance function parameterized on T (the plasma monitors) to W for notational purpose. The discrete sum of Equation 5.4 now only sums over the region covered by Ω_l. The plasma monitor roughly cover 120 degrees of the

[3]In other words, the color and intensity in the illumination sphere have rapid variation spatially.

illumination hemisphere, and so we only employ 11 light rig positions roughly covering 240 degrees. Hence, $n' = 4 \times 11 = 44$.

Our plasma monitors both have a resolution of 1024×768, and in theory we can regard each pixel on the monitors as a single light and use the same capturing method described in Section 5.2. However, this is clearly very inefficient and the amount of data is intractable. Thus, we choose to employ environment matting techniques from Chuang et al [11], which allows us to capture the reflectance function efficiently and provide an implicit compression of the data.

Environment mattes describe for each pixel the origin of the contributing illumination. For example, glass object is typically lit by rays from behind the object through refraction. Following Chuang et al.'s exposition (Section 2.5, Equation 2.5), we modeled W as a sum of Gaussians. Also, to simplify our rendering, we assume W can be modeled with two Gaussians, one dominantly reflective and the other refractive. Hence,

$$W(\mathbf{x}) = a_1 G_1(\mathbf{x}; \mathbf{C_1}, \boldsymbol{\sigma_1}, \theta_1) + a_2 G_2(\mathbf{x}; \mathbf{C_2}, \boldsymbol{\sigma_2}, \theta_2) \tag{5.7}$$

G_1 and G_2 are elliptical, oriented 2D unit-Gaussians, and a_1 and a_2 are their amplitudes, respectively. $\mathbf{x} = (x, y)$ is the camera pixel position, $\mathbf{C_i}$ the center of each Gaussian, $\boldsymbol{\sigma_i}$ are their standard deviations on the two axes, and θ_i their orientations.

Now we can rewrite Equation 5.6:

$$C = \int_{\Omega_h} (a_1 G_1 T(x) + a_2 G_2 T(x))dx + \sum_{i=1}^{n'} R(\omega_i)E(\omega_i)dA(\omega_i). \tag{5.8}$$

With this formulation, we provided a hybrid sampling of the illumination environment, and we assume that most of the refracted and reflected rays arriving at a pixel originate from behind the object. This is true for most refractive objects, and for objects with a strong reflected components at grazing angles. This does not correctly reproduce specular reflections on the front of the object, or transparent objects with large-scale internal structure or surface facets, such as crystal glass.

The acquisition and processing of these data was described in Section 3.7. After acquisition, the environment mattes are produced using an optimization procedure, and at each pixel we have the parameters for the Gaussians G_1 and G_2. To save storage and computation time for the non-linear parameter estimation, we identify and remove areas outside the object silhouette like we did for the surface light field data. The environment matte is subdivided into 8×8 pixel blocks and only pixel blocks that contain the object are processed and stored.

With this hybrid sampling we are able to capture and render refractive objects convincingly. The rendering of surface reflectance fields with the environment matting extension will be discussed in Section 6.4.

Chapter 6

Rendering

In this chapter, we are going to describe the rendering of the geometric and appearance model acquired by our system. To recapitulate, the input to our rendering system includes (supposing we have acquired the surface reflectance field of the object):

- A point-sampled model of the visual hull, based on intersection of the opacity mattes.
- Opacity mattes from each viewpoint.
- Compressed reflectance images from each viewpoint.
- A new illumination environment.
- Environment mattes from each viewpoint. (Optionally acquired)

For rendering surface light fields, the last three items are replaced with a set of radiance images of the object under fixed illumination. In Section 6.1 we will describe the preprocessing of the surface reflectance fields under novel illuminations. In Section 6.2 we will discuss the surface splatting approach we employ to render the point-sampled model. Section 6.3 will be focused on the interpolation of radiance and opacity between different viewpoints. Section 6.4 describe the additional processing and special interpolation for the optionally acquired environment mattes.

6.1 Relighting under novel illuminations

Before we render the surface reflectance field of an object we precompute a set of radiance images as if the object was captured under the new given illumination. For now, we focus on one particular pixel from one particular viewpoint. Recall our lighting model (see Section 5.2) in terms of the *reflectance function*:

$$C = \sum_{i=1}^{n} R(\omega_i)E(\omega_i)dA(\omega_i) \tag{6.1}$$

For each pixel, the radiance value is equal to the sum of the product of environment illumination and the reflectance function, downsampled to n discrete illumination direction, multiplied by the subtended solid angle. For our reflectance field acquisition without environment mattes, the number of lights is equal to $4 \times 15 = 60$. To relight the model under novel illuminations, we first need to filter and downsample the new environment map E' to our light resolution. Also, similar to Debevec et al., we need to normalize the environment map based on solid angle, as our sampled illumination has more samples near the pole. We call the normalized, downsampled environment map E''

$$E''(\omega_i) = E'(\omega_i)\sin\phi \tag{6.2}$$

where ϕ is the azimuth angle of the light direction. Recall that our reflectance images captured during acquisition is in the form

$$C_i = R(\omega_i)L \tag{6.3}$$

where L is the intensity of our lights. From a particular viewpoint, for the ith light, we have a pixel C_i observed in the ith reflectance image. Now we sum over the product of C_i and $E''(\omega_i)$. We have

$$S = \sum_{i=1}^{n} C_i E''(\omega_i) = L\sum_{i=1}^{n} R(\omega_i)E'(\omega_i)dA(\omega_i) \tag{6.4}$$

The sum S is equal to our desired radiance value according to Equation 6.1, multiplied by L, the intensity of the light in our light rig. As we do not explicitly measure the actual value of L, we are only able to evaluate the re-illuminated pixel with respect to a scalar factor of L. This factor is left as a user parameter, and can be considered loosely as the *exposure time* setting of the rendering.

The description above illustrates how we compute a single re-illuminated pixel. To actually compute the desired radiance images, we do the computation in pixel blocks of 8×8. Recall that our reflectance images are compressed by principal components analysis. For a particular block in a viewpoint, the $n = 60$ captured blocks under the different lights are compressed into k principal component blocks V_j, and $n \times k$ eigenvalues γ_{ij}. The block of the reflectance image under the ith light can be reconstructed by:

$$C_i^{block} = \sum_{j=1}^{k} \gamma_{ij} V_j \tag{6.5}$$

Given the normalized environment map E'', the re-illuminated block S^{block} is then equal to

$$S^{block} = \sum_{i=1}^{n} C_i^{block} E''(\omega_i) = \sum_{i=1}^{n} (\sum_{j=1}^{k} \gamma_{ij} V_j) E''(\omega_i) \tag{6.6}$$

This evaluation is repeated for all blocks for all viewpoints. The result is then stored as a radiance image for each viewpoint. This allows us to render surface reflectance fields in the same way as surface light fields.

6.2 Surfel Rendering

The input to our rendering stage include the visual hull represented as a point sampled model, the opacity mattes, and the radiance images either acquired as a surface light field or computed from surface reflectance fields using the procedure described in Section 6.1.

For each surfel (surface element) on the visual hull, we can compute interpolated opacity and radiance values using the unstructured lumigraph interpolation method described in the next section. Once we have these interpolated values, the final task is to render these surfels to an image. Since the surfels are generally positioned irregularly, a naive rendering approach will most likely suffer from aliasing. Thus, we use the elliptical weighted average (EWA) surface splatting approach of Zwicker et al [56], a hierarchical forward-warping algorithm projects the surfels onto the screen. A screen space EWA filter reconstructs the image using the opacity, color, and normal stored per surfel. A modified A-buffer provides order-independent alpha blending and edge anti-aliasing. For further details of surface splatting, please refer to [56].

While the surfel rendering approach is a powerful tool for rendering our acquired objects in general, our acquisition system is in no way tied to a surfel representation. It is conceivable that we can construct a polygonal representation of our visual hull and render the model using polygons. The opacity mattes and reflectance images are both separate from the representation of the visual hull.

6.3 Viewpoint Interpolation

To interpolate the radiance images of the original viewpoints to arbitrary viewpoints, we employ the unstructured lumigraph interpolation of Buehler et al. [7].

Each surfel is associated with its position and a precomputed visibility bit-vector describing the set of cameras from which the surfel is visible. We first compute the normalized direction $r_c(i)$ from the surfel position to each visible camera i using the visibility bit vector and the global array of camera positions. We also compute the normalized viewing direction r_v from the surfel position to the center of projection of the current view. We then assign a penalty $p(i) = 1 - \cos\theta_i$ to each visible camera, where $\cos\theta_i = r_c \cdot r_v$. We consider only the $k = 4$ cameras with smallest penalty $p(i)$ when interpolating a value. All other cameras are assigned an interpolation weight $w(i)$ of zero. We take care that a particular camera's weight falls to zero as it leaves the set of the closest four cameras. We accomplish this by defining an adaptive threshold $\cos\theta_t = r_4 \cdot r_v$, where r_4 is the direction of the surfel to the fourth closest camera. The blending weight $w(i)$ for each camera is:

$$w(i) = \frac{\cos\theta_i - \cos\theta_t}{1 - \cos\theta_t} \tag{6.7}$$

This weight function has its maximum value of one for $\cos\theta_i = 1$, and it falls off to zero at $\cos\theta_i = \cos\theta_t$. To ensure epipole consistency, we multiply $w(i)$ by $1/p(i)$. Epipole consistency means that rendering the object from original camera viewpoints reproduces exactly the original images. We also normalize all $w(i)$ so that they sum up to one. Finally, we scaled the interpolated radiance value by the user-specified *exposure time parameter*. As we do not calibrate the intensity of the input lights of our system, the user is only able to control the exposure in terms of a scalar factor of the acquired illumination.

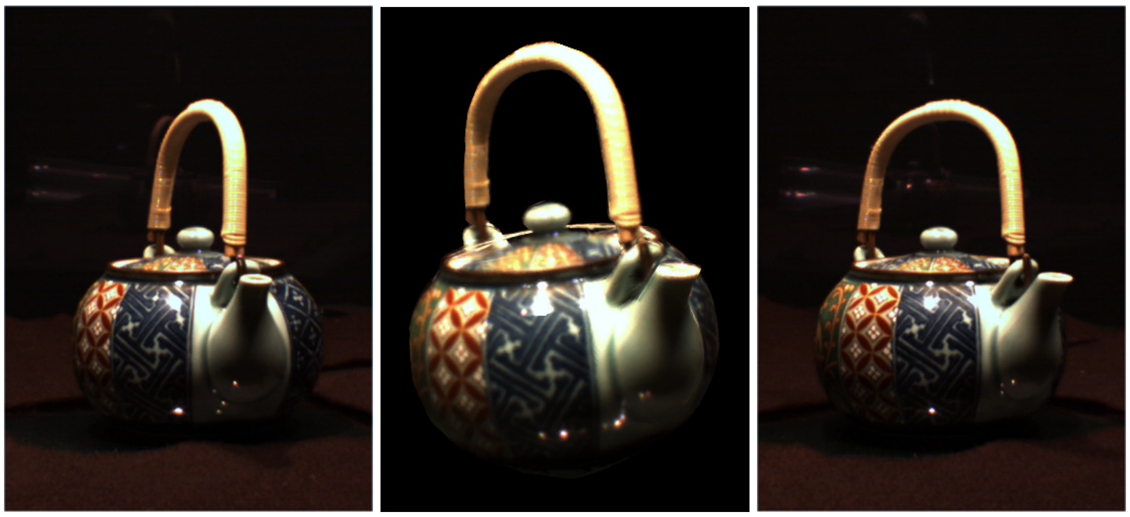


Figure 6-1: Rendering from arbitrary viewpoints. Left and right: Original images. Middle: Interpolated view.

Unstructured lumigraph interpolation for viewpoints other than those seen by reference cameras introduce small artifacts, most notably in specular or concave areas. Figure 6-1 shows acquired images of an object (Figures 6-1a and c). Figure 6-1b shows the object from an intermediate viewpoint.

Note that the figure shows only the two closest views, but we actually use the four closest views for interpolation. As can be seen in the figure, the artifacts are generally small. The smoothness of the interpolation can only be fully appreciated by a rendered video with a moving viewpoint. Please refer to our demonstration video on the website[1].

[1]URL: http://graphics.lcs.mit.edu/ãddy/video/sig2002.mpg

6.4 Rendering with environment mattes

To render surface reflectance fields with the environment mattes extension, we need to relight the pixel according to following equation (Equation 5.8):

$$C = \int (a_1 G_1 T(x) + a_2 G_2 T(x))dx + \sum_{i=1}^{n} R(\omega_i)E(\omega_i)dA(\omega_i). \tag{6.8}$$

The discrete sum in the second term can be precomputed as radiance images in the same way discussed in Section 6.1. The radiance images can then be interpolated using the method described in Section 6.3. To compute the first term, we interpolate the views before rendering. We first evaluate interpolated Gaussians from the observations of the closest viewpoints, and then use them to compute the integral.

Before we start we need to pre-process our environment mattes. The acquired environment mattes are parameterized on the plane T of the background monitor. However, for rendering they need to be parameterized on a global environment map $\tilde{T}$. Figure 6-2 shows a 2D drawing of the situation.

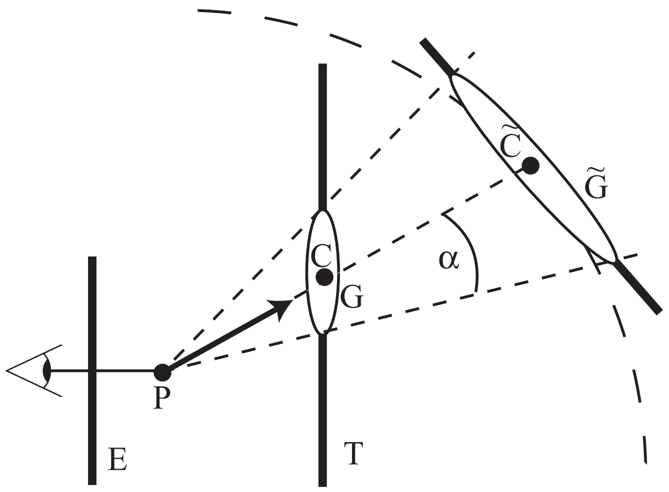


Figure 6-2: Reprojection of the environment matte Gaussian G from the monitor plane T into the environment map $\tilde{T}$.

During system calibration we determine the position of each monitor plane T with respect to each viewpoint. This information is globally stored per viewpoint. $\tilde{T}$ is the parameterization plane of the new environment map. The mapping from T to $\tilde{T}$ may be non-linear, for example, for spherical environment maps. A 3D surface point P on the object is projected onto a pixel of the environment matte E, which stores the parameters of the 2D Gaussian G. We compute the Gaussian $\tilde{G}$ that best approximates the projected Gaussian G on the parameterized surface $\tilde{T}$.

We represent the new Gaussian $\tilde{G}$ using the following parameters: a (the amplitude of G), $\tilde{C}$ (a 3D vector), (α, β) (the opening angles), and $\tilde{\theta}$ (the new rotation angle). This projection is performed for each change of the environment map.

As we have two reprojected Gaussians $(\tilde{G}_1, \tilde{G}_2)$ per pixel, we need to perform a matching of the Gaussians before interpolation. Figure 6-3 shows a simplified 1D drawing of the matching process. The two Gaussians per pixel are classified as *reflective* $(\tilde{G}_r)$ or *transmissive* $(\tilde{G}_t)$. We compute the angle ϕ of their center vectors $\tilde{C}_r$ and $\tilde{C}_t$ with the surface normal N. If $\phi > 90^o$, we classify the Gaussian as transmissive. If $\phi <= 90^o$, we classify it as reflective. If both Gaussians are reflective or refractive, we only store the one with the larger amplitude a.

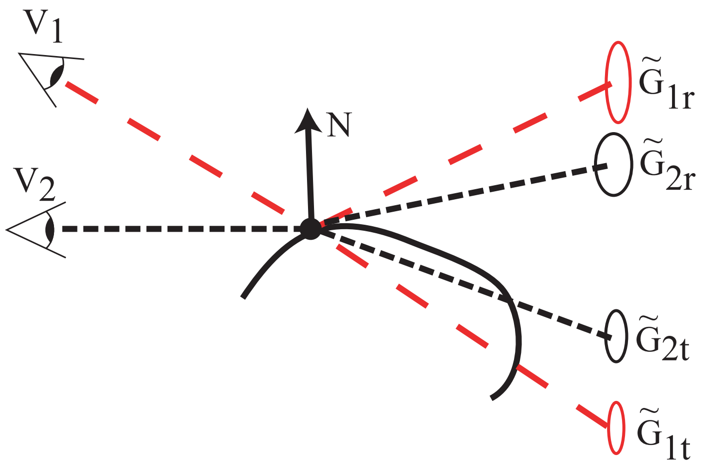


Figure 6-3: Matching of reflective and refractive Gaussians.

After the matching, for each of the two Gaussian $(\tilde{G})$ we interpolate the parameters of the 4 matching Gaussians $(\tilde{G}_i, i = 1,2,3,4)$ from the closest views. Using the weight w_i from unstructured lumigraph interpolation described in the previous section, we compute linear combinations for the amplitudes a_i and the directional vectors $\tilde{C}_i$. The angular parameters (α_i, β_i) and $\tilde{\theta}_i$ are blended using quaternion interpolation. The result is a new

Gaussian $\hat{G}$ that is an interpolated version of the Gaussians $\tilde{G}_i$, morphed to the new viewpoint.

To compute the color contribution C' from the interpolated environment mattes we compute Equation 6.8 under the reparameterized environment:

$$C = \int (a_r \hat{G}_r \tilde{T}(x) + a_t \hat{G}_t \tilde{T}(x))dx + \sum_{i=1}^{n} R(\omega_i)E(\omega_i)dA(\omega_i). \tag{6.9}$$

where the first and second term of the integral gives the contribution from the reflective and transmissive Gaussians respectively.

Note that by interpolating the environment matte before rendering we have achieved smoother interpolation, compared to interpolating the radiance values rendered from the original viewpoints. Suppose at a surface point p, we acquire the environment matte from two adjacent viewpoints, and for each viewpoint we have a Gaussian associated with p. Using the latter method, any in-between views will interpolate by cross-blending the two Gaussians convolved with the environment. The region in the environment between the two Gaussians does not contribute to these interpolated views. This introduces very noticeable artifact for refractive objects like glass as the refraction is nearly perfectly specular. By interpolating the Gaussians before convolution instead, we are able to produce a continuously moving Gaussian between the two observations, and no cross-fading artifacts would be observed.

Nonetheless, our interpolation of the Gaussian is only an approximation. In order for the interpolation to be valid, we have implicitly assumed that the direction of the Gaussian changes linearly between viewpoints, which is not true in general when there is more than one reflection/refraction present, and when the topology of the light transport paths change between viewpoints. However, to acquire and reproduce these complex light path is very difficult, and our approximate interpolation has produced smooth and reasonable rendering for the transparent and refractive objects we have acquired (see Section 7.5).

Chapter 7

Results

We have collected a wide range of objects with our scanning system. These include difficult surfaces of various genuses, with concavities, and with fine scale features. We have also captured different models with fuzzy, specular and refractive materials. In this chapter we will present and discuss results from the various stages of the acquisition and rendering processes. These include opacity mattes(Sec 7.1), visual hulls and opacity hulls(Sec 7.2), surface light fields(Sec 7.3), surface reflectance fields(Sec 7.4) and the extension with environment matting(Sec 7.5). We will conclude the chapter with quantitative measurement of the performance of the system (Sec 7.6).

7.1 Opacity Mattes

We capture opacity mattes by using multi-background techniques, with the plasma monitors serving as active backgrounds (See Section 3.4). Our method is robust and gives high-quality results on most materials. To reduce the color spill problem, we take several pairs of foreground/background images, and we take the maximum of the opacity values calculated from the different pairs.

Figure 7-1(a) shows a typical opacity matte we acquire. The opacity mattes are stored as a common greyscale image and thus the opacity values range from 0 to 254, while we reserve the value 255 to represent missing data. We classify opacity into three types: opaque($o \approx 254$), transparent($o \approx 0$), and partial(not opaque nor transparent). In this model

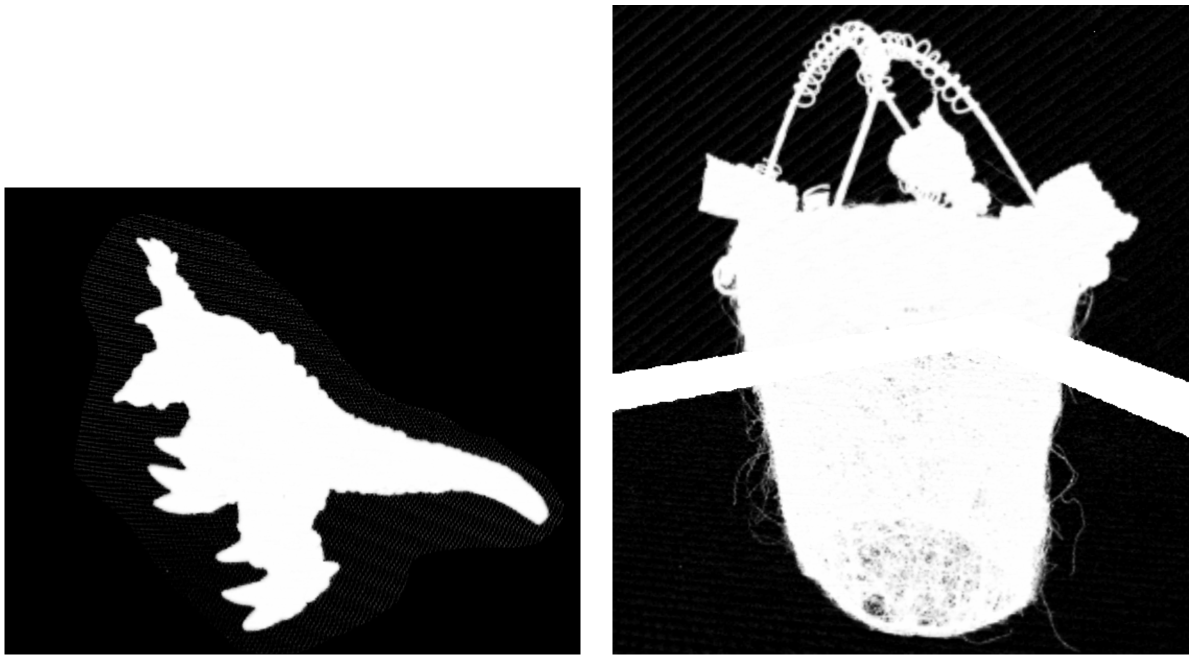


Figure 7-1: Opacity mattes of : (a) a decorative basket and (b) a toy dinosaur.

of a toy dinosaur, the dominant partial opacity values are at the silhouette due to partial occupancy of pixels. Within the silhouette the values are close to 254, and outside the silhouette they are close to 0. The typical maximum error without color spill problem is around 10.

Figure 7-1(b) shows delicate partial opacity of a decorative basket captured. Partial opacity values are observed almost everywhere within the basket due to the many holes in the weaving structure.

As the plasma monitors are framed and we cannot put the bottom monitor too close to the vertical one, there are regions in images from some viewpoints that lie outside the range of the display rectangles of the monitors. In these regions we cannot apply multi-background techniques and hence cannot measure opacity. For example, in Figure 7-1(b) the white bars represent missing data due to the frame of the bottom monitor. These opacity values are not used during interpolation.

Figure 7-2 shows an opacity matte of a bonsai tree with complex geometry. On the ceramic pot color spill is noticeable. The ceramic pot is highly specular and the reflection of the displayed pattern of the bottom monitor has confused the algorithm that the pixels

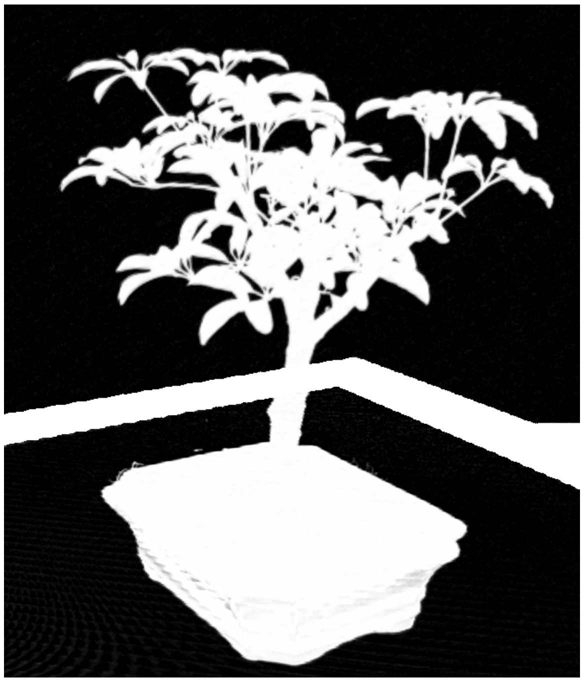


Figure 7-2: Opacity matte of a bonsai tree. Problems caused by color spill can be observed on the ceramic pot.

are partially transparent. On this matte the lowest opacity value on the opaque ceramic pot is 179, or 70%.

Nevertheless, our matting process yield high-quality results for most objects. In many objects with fine-detail structure, opacity mattes allow us to obtain the aggregate effect of geometric detail from a macroscopic point of view. This allow us to produce faithful rendering of the acquired objects without tackling the nearly impossible task of capturing the exact geometry.

7.2 Visual Hull and Opacity Hull

The visual hull is computed based on the opacity mattes using the algorithm described in Section 4.1. We conservatively assume the missing opacity values as opaque.[1] As the visual hull is an intersection process, conservative overestimation of the geometry and noise is eliminated, as long as they are not consistent with respect to all viewpoints.

[1]This is the reason we choose the special value 255 to represent missing opacity data.

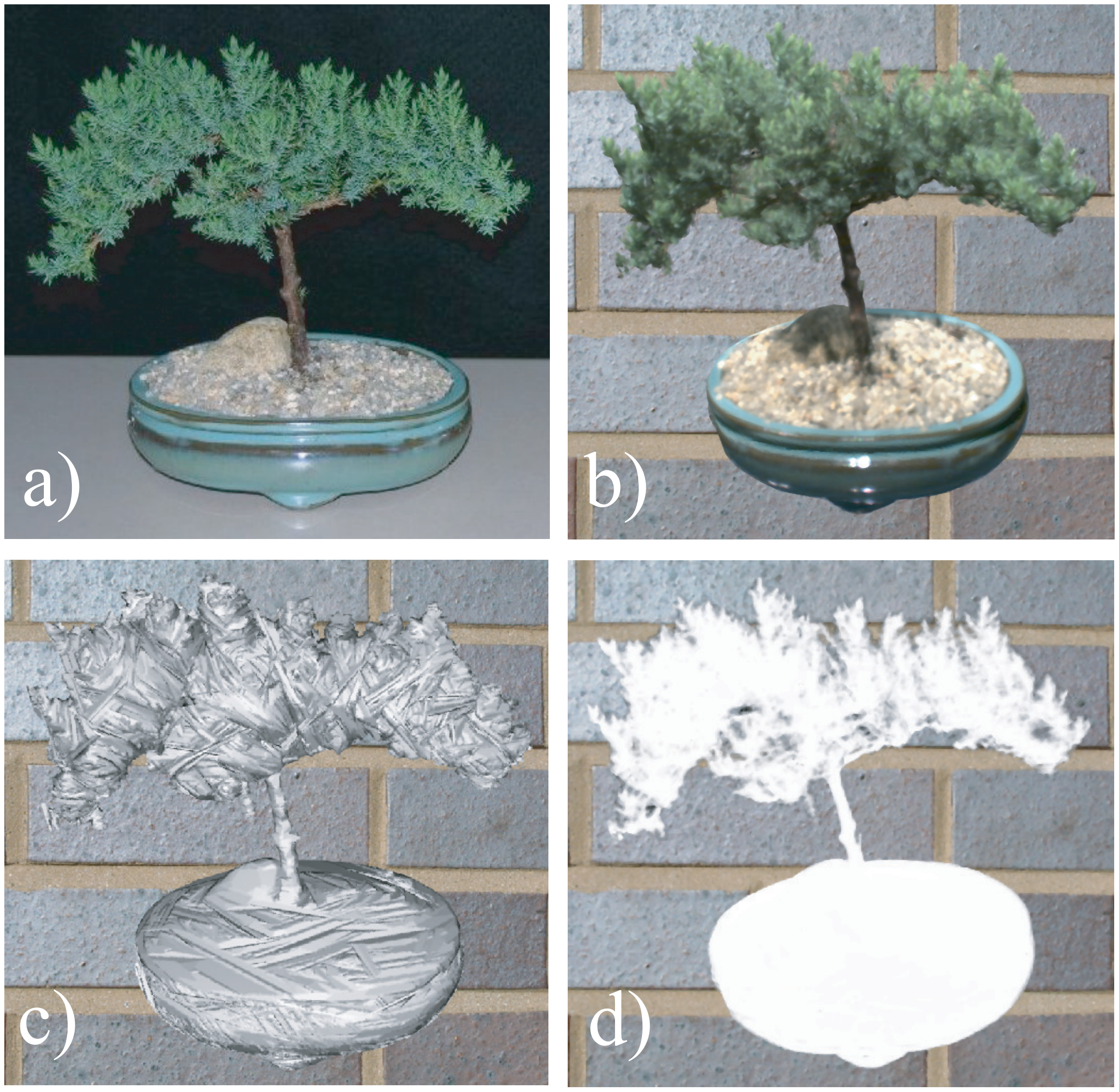


Figure 7-3: a) Photo of the object. b) Rendering using the opacity hull. c) Visual hull. d) Opacity hull.

Figure 7-3(c) shows the visual hull of a bonsai tree model(photograph showed in Figure 7-3(a)), using normal shading with surface splatting. The resulting geometry is a crude approximation of the true surface. However, it is the coarseness of the geometry that allow us to have a compact representation regardless of geometric complexity. It also allows the visual hull algorithm to be very robust. It worked on all models we acquired without any problems.

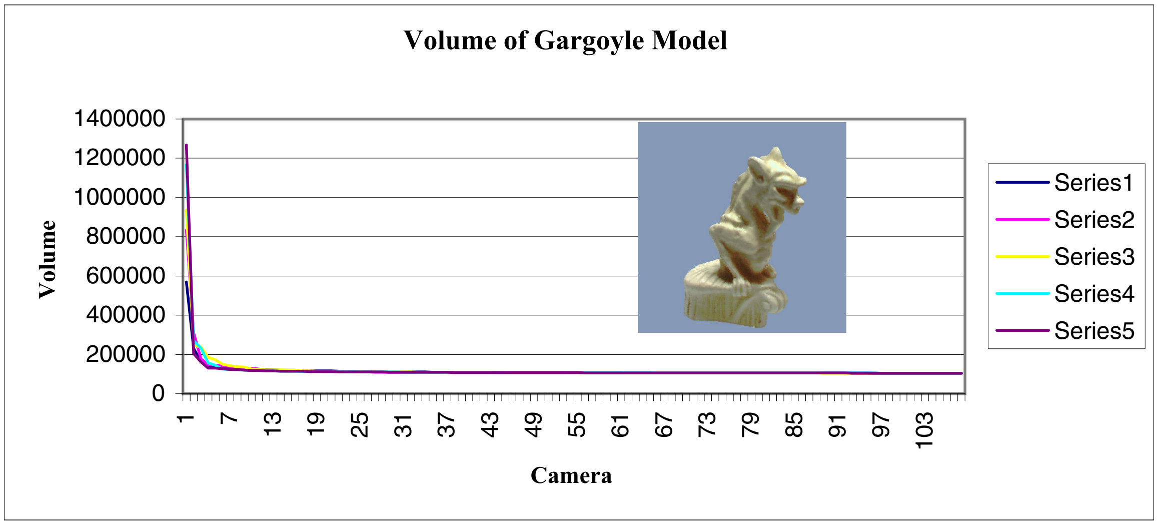
Figure 7-3(d) shows a rendering using the opacity instead of the radiance values to visualize the opacity hull. Notice the coarse shape of the visual hull and the much improved quality using the view-dependent opacity, despite the fact that their geometry is identical. Figure 7-3(b) shows the final rendering of the opacity hull with the surface light field.

Finally, to evaluate the number of images required to compute the visual hull, we instrumented our code to compute the change in volume of the visual hulls as each silhouette is processed. We then randomized the processing order of the images and repeated the visual hull calculation multiple times. The plots shown in Figure 7-4 illustrate the rather typical behavior.

Generally, the visual hull converges to within 5% of its final volume after processing around 20 images, and seldom is this plateau not reached by 30 images. Collecting data over the entire hemisphere ensures that this volume closely approximates the actual visual hull. This implies that the visual hull processing time can be dramatically reduced by considering fewer images to compute the hull model. However, dense alpha mattes are still important for representing view-dependent opacity. The view-dependent opacity and radiance measurements dramatically improve the final renderings.

7.3 Surface Light Fields

Acquiring surface light field is an easy and quick process. One of the challenge for some objects is the setup of the illumination. To simulate the effect that the illumination is fixed with respect to the object, in theory we should only use lights that are mounted on the upper turntable, and that the upper turntable rotate synchronously with the object. However, as we only have three mounting positions on the upper turntable, the light positions are very



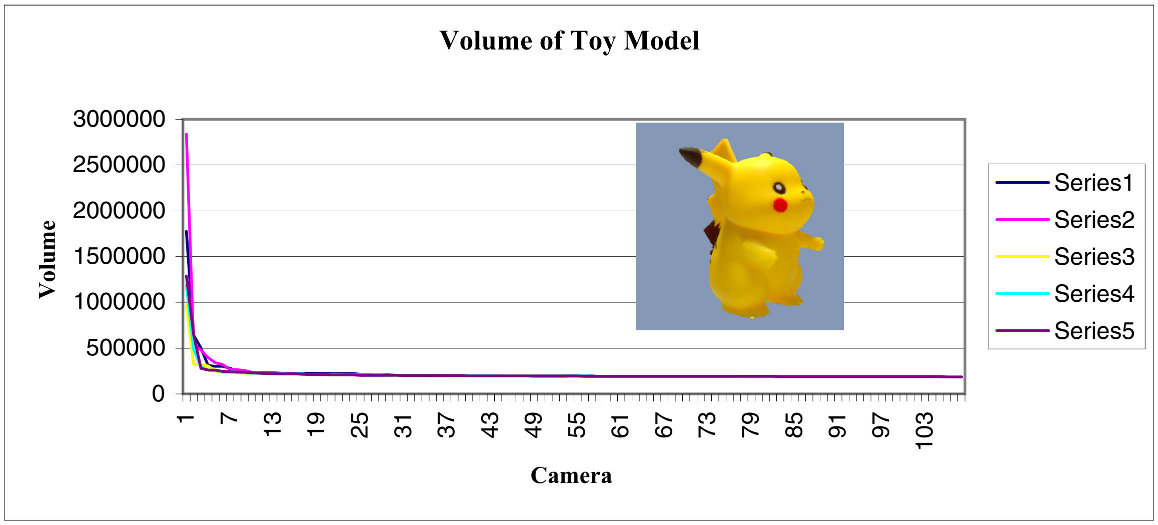


Figure 7-4: The volume of the visual hull as a function of the number of images used to construct the visual hull.

limited. Also, as we are only able to provide illumination in the upper hemisphere with the existing light rigs, sometimes the bottom part of the object is not well lit, especially when there is self-occlusion. To give a better distribution of the lights on the object, we use a standing lamp to provide ambient illumination. We cover the sides of the lamp to make sure the light does not directly illuminate the object. This provides an approximately rotation-invariant ambient lighting. Figure 7-5 show some renderings of acquired surface light fields from novel viewpoints, composited onto new backgrounds.

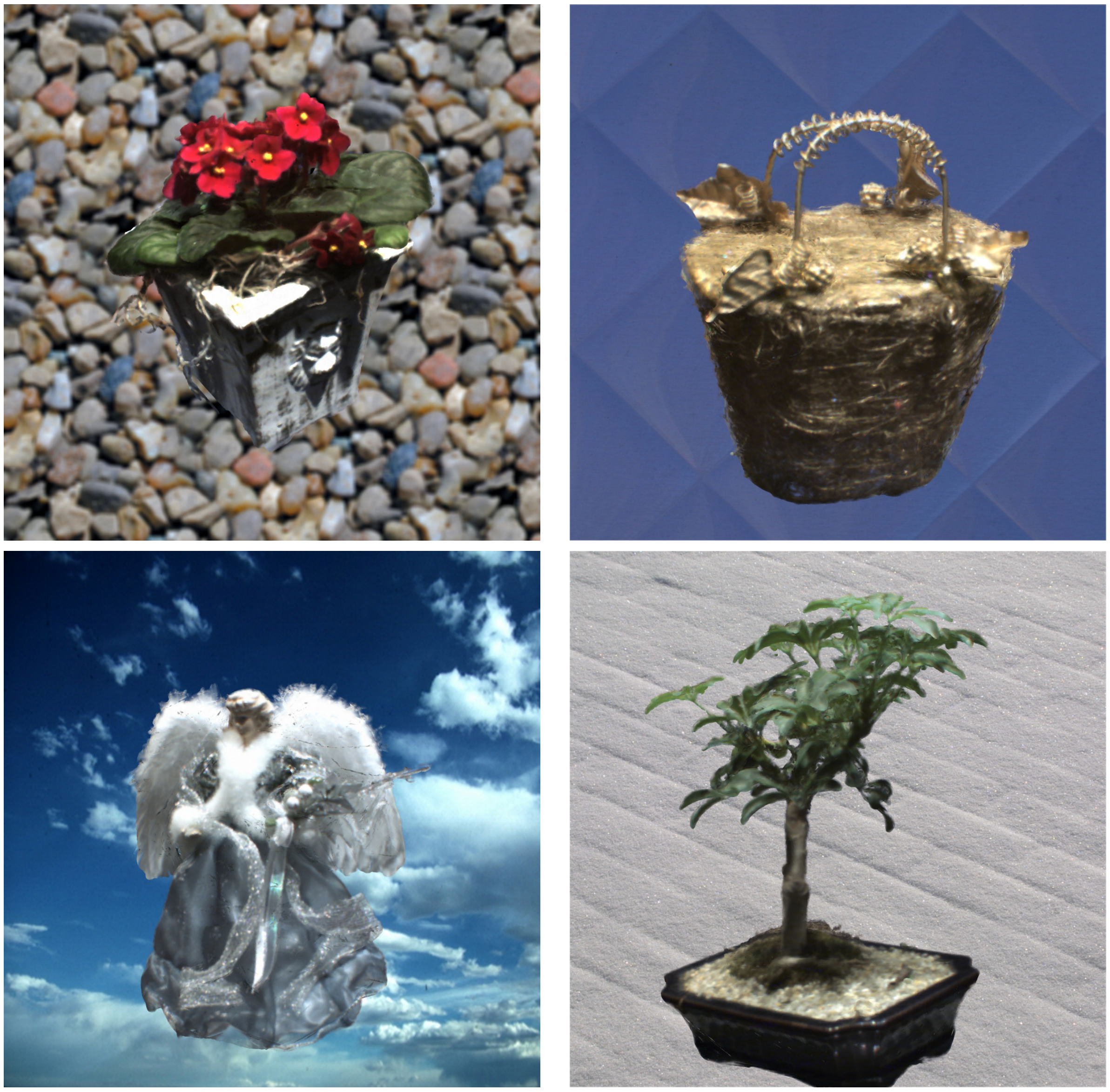


Figure 7-5: Surface light fields of several objects from new viewpoints.

7.4 Surface Reflectance Fields

We acquire surface reflectance fields by acquiring, under every viewpoint, all possible illuminations with our light rig. In this section we are going to discuss results from the low-resolution surface reflectance fields, and the environment matting extension will be discussed in Section 7.5.

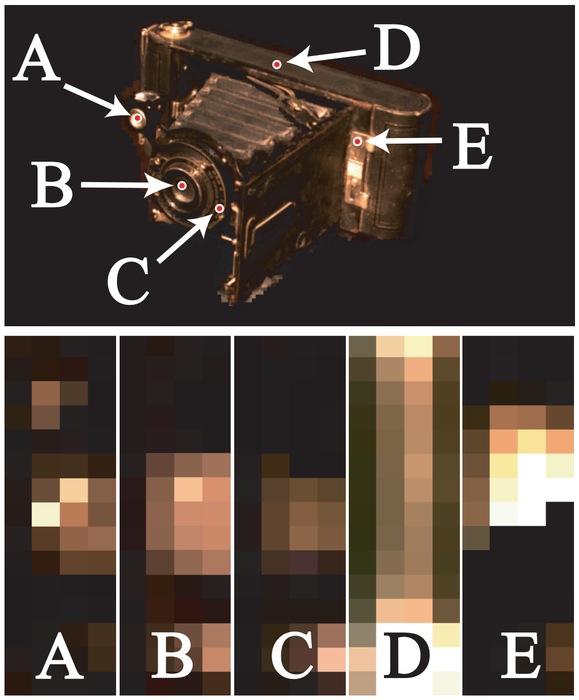


Figure 7-6: Measured reflectance function data for several surface points.

In the process of acquiring surface reflectance fields, we have made many interesting measurements and observations. Figure 7-6 shows plots of the measured reflectance field data for three surface points on an object. We chose the surface points to be in specular and self-shadowed areas of the object. The darker parts of the plots are attributable to self shadowing. The data lacks any characteristics that would make it a good fit to standard parametric BRDF models or function approximation techniques.

Figure 7-7 shows a visualization of the number of PCA components per 8 by 8 pixel block of the reflectance images from an original viewpoint. We set the global RMS reconstruction error to be within 1% of the average radiance values of all HDR reflectance images. Note that areas with high specularity or high frequency texture require more com-

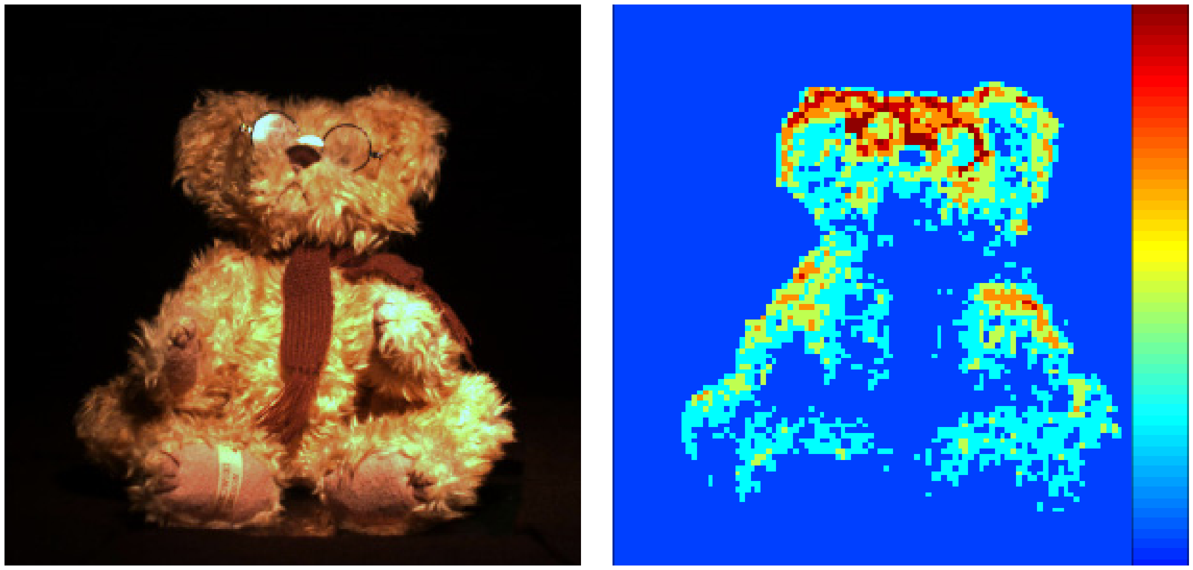


Figure 7-7: a) Original view. b) Visualization of number of PCA components per block (Max. = 15, Mean = 5).

ponents than areas of similar average color. The maximum number of components for this view is 10 while the average is 5. This is typical for all of our data.

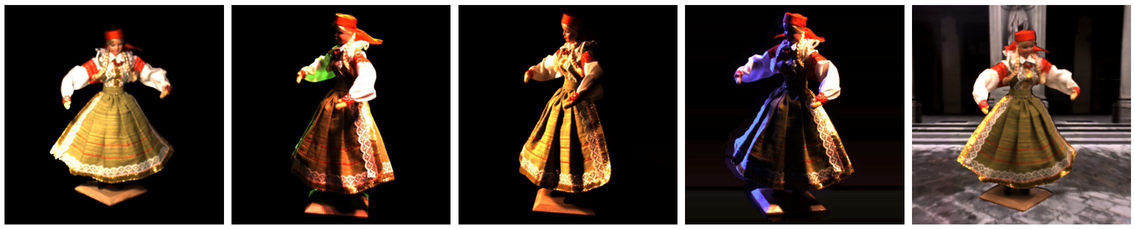


Figure 7-8: Re-lightable model under novel illumination.

With the captured surface reflectance fields, we can render our models in novel illuminations. Figure 7-8 shows a doll rendered under novel illuminations, the first four images by using several colored lights, and the final one using an environment map of the Uffizi gallery, courtesy of Paul Debevec.

To further illustrate the power of our surface reflectance fields techniques in seamlessly blending with the environments, we have taken a real video sequence from a library. We captured the illumination environments using light probes and techniques similar to Debevec et al [16]. We then scanned the surface reflectance fields of four object using our system, and render these objects into the video sequence captured. Figure 7-9 shows these

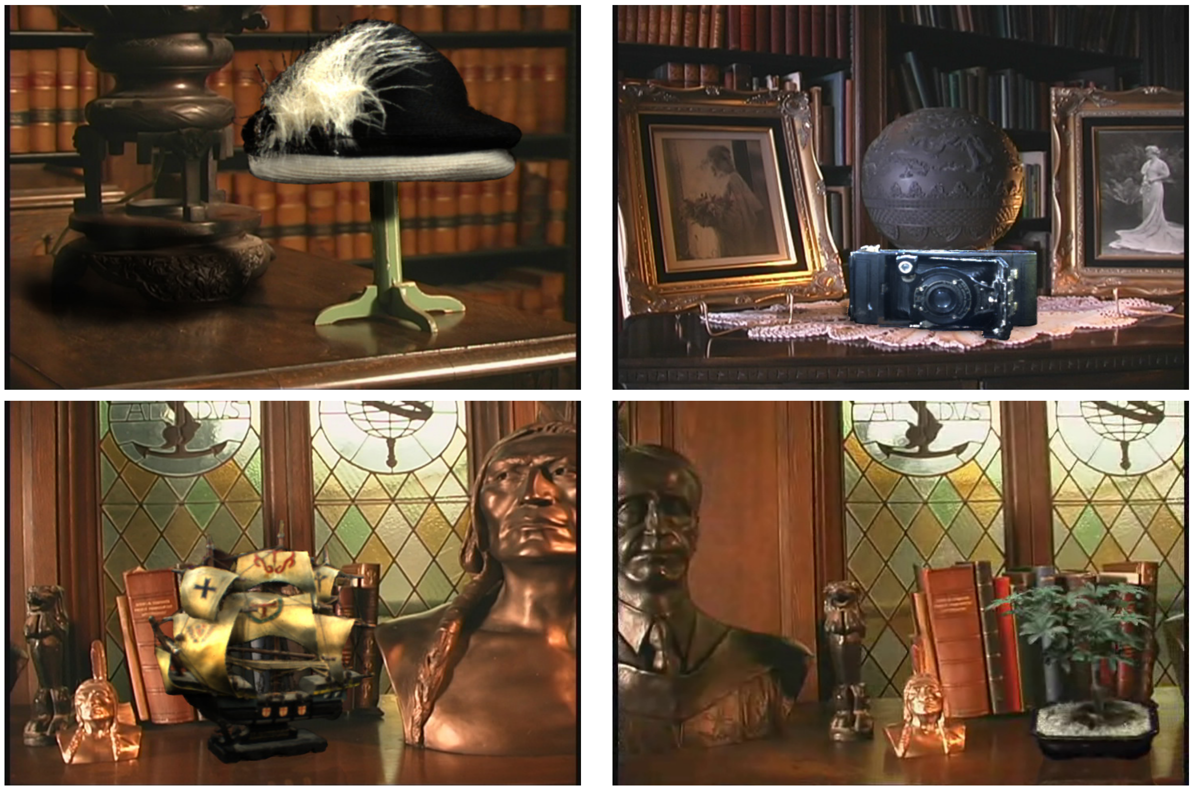


Figure 7-9: A combination of scanned and real objects in real environments. The scanned objects were illuminated using surface reflectance fields.

objects rendered into the scene, illuminated by light probes captured roughly at the same locations as the synthetic objects. We observe that the re-illumination is faithful, and the opacity mattes produced seamless blending into the background for objects with fine-detail geometry, e.g. the bonsai tree, the feather of the hat and the strings of the ship model.

7.5 Surface Reflectance Fields with Environment Matting

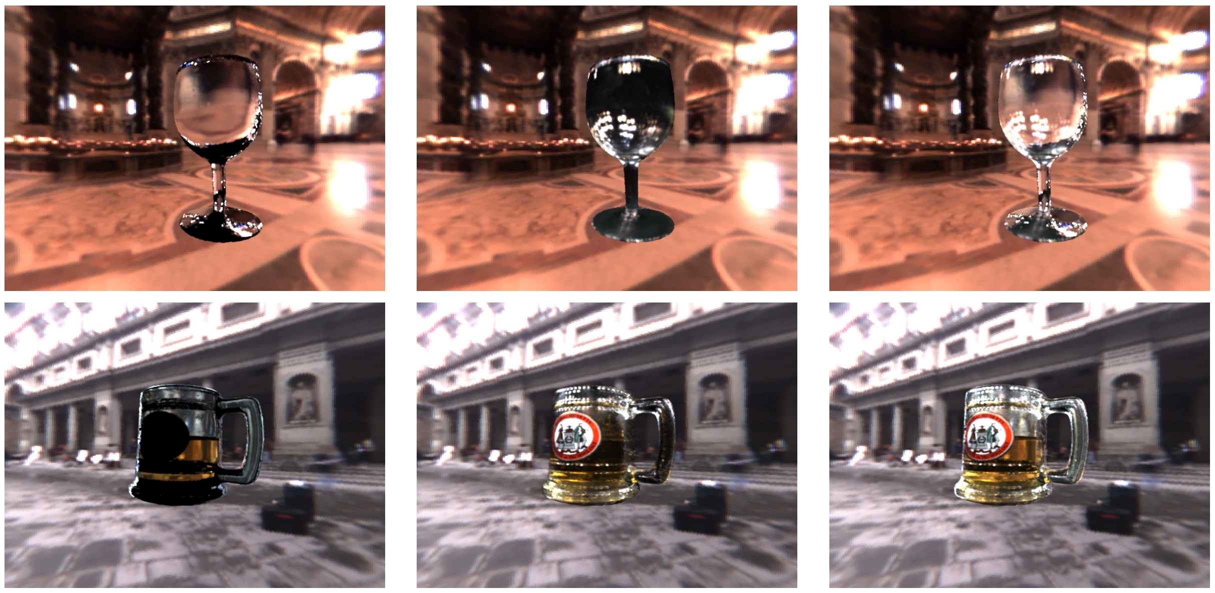


Figure 7-10: Left: High-resolution reflectance field from the environment mattes. Middle: Low-resolution reflectance field from the reflectance images. Right: Combined.

We have captured three refractive objects with our environment matting extension. Figure 7-10 shows renderings using only the environment mattes (left), using only the reflectance images (middle), and combining the two (right). Note that the environment mattes, storing the high-resolution reflectance field, mostly capture refractions, while the reflectance images, storing the low-resolution reflectance field, mostly capture reflections.

Figure 7-11 shows a few frames from an animation with a rotating viewpoint. Note how the specular highlights and refractive effects are accurately preserved by our interpolation procedure. The actual quality of the models and of our interpolation method can only be observed in an animated sequence, where the benefits of ray-interpolation rather than radiance interpolation would be obvious.

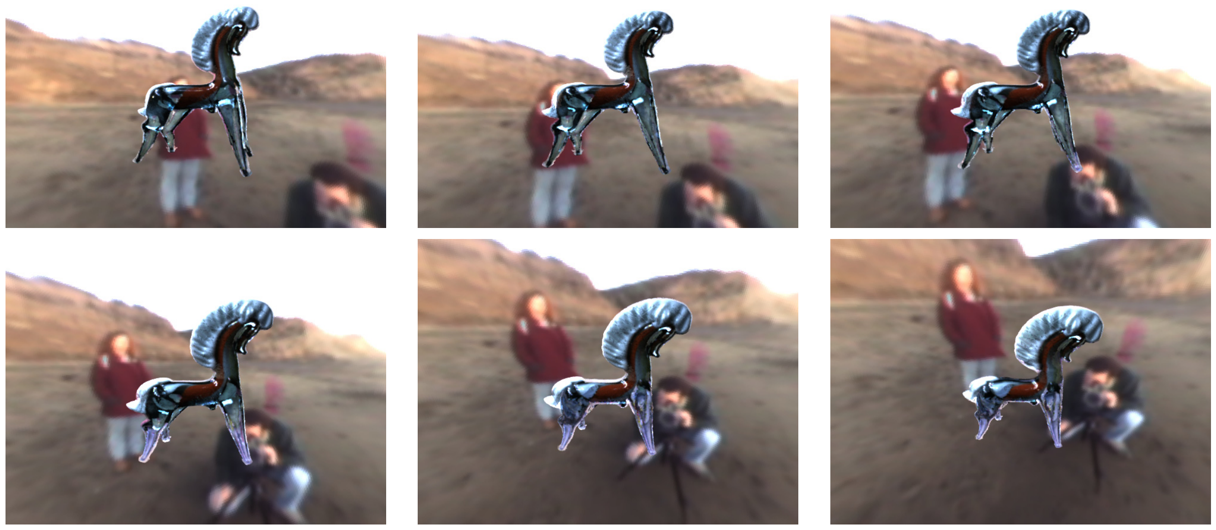


Figure 7-11: Frames from an animation with rotating viewpoint.

7.6 Performance

7.6.1 Acquisition

For all objects, we use six cameras and 36 or 72 turntable positions. For the performance study of this section we assume 36 turntable positions, unless otherwise stated. We use 4×15 light positions when only the low-resolution surface reflectance field is captured, and 4×11 light positions when environment matting techniques are employed.

The cameras have a resolution of 1360×1036. All the regular dynamic range images are stored as raw bitmaps, sized $1360 \times 1036 \times 3 \approx 4.1$MB. High-dynamic images are taken using 4 exponentially increasing exposure time, and the response line is computed on the fly to avoid excessive storage. The final high-dynamic images are stored using the RGBE[2] format and thus are sized $1360 \times 1036 \times 4 \approx 5.5$MB.

The data size and acquisition time of the different acquisition stages are as follows:

Calibration Images: We take one image per view of the calibration object. The acquisition process took about 22 minutes, and we store all frames uncompressed. The total size for $36 \times 6 = 216$ views is 890MB.

[2]See Section 3.2 for details of the RGBE format

Background/Foreground Images: We take three backgrounds and three foregrounds per view in different passes. The backgrounds can be reused if the system calibration does not change. The total size for each pass is $36 \times 6 \times 3 \times 4.1 \approx 2700$MB. The acquisition time for each pass is 54 minutes.

Radiance Images: We take one high-dynamic range image per view of the object under fixed illumination. The total size is 1217MB for 36×6 views and the capturing last 30 minutes.

Reflectance Images: Using a light resolution of 4×15, we capture 60 high dynamic range reflectance images per view. The total number of reflectance images is $36 \times 6 \times 60 = 12960$, and the total uncompressed size is about 70GB. The acquisition process takes approximately 18 hours.

Environment Mattes: If we employ environment matting techniques, we only need to capture 4×11 light positions for the reflectance images. The remaining 4 rig positions are covered by the plasma monitors. We acquire 300 images for environment matting for each viewpoint, but we only capture the mattes from 3 cameras (See Page 32). The total data size is more than 150GB for a full acquisition. Also, as the refractive objects are typically also very specular, we usually employ 72 turntable positions. Because the data is so huge, we have only acquired half or 1/4th of all views for the three objects we acquired. The acquisition time for a full acquisition with 36 turntable positions is estimated to be 18 hours.

We summarize the acquistion time and data size for each type of the images in the following table:

Image Type	**Time(minutes)**	**Size(MB)**
Calibration	22	890
Background	54	2673
Foreground	54	2673
Radiance(HDR)	30	1217
Reflectance(HDR,60 lights)	1178	73040
Environment mattes	540	150000

7.6.2 Processing

The results of this section are based on computation on a 2GHz Pentium 4 PC with 2GB of memory. We compute the opacity mattes from the set of background and foreground images using the algorithm described in Section 3.4. An opacity matte is produced from each viewpoint and stored as a greyscale image. The process for 36×6 viewpoints with three pairs of background/foreground took about 20 minutes, and the total size of the final opacity mattes is about 300MB.

The processing of the environment mattes take about 10 minutes per viewpoint, or 36 hours for 216 viewpoints. The size of the resulting mattes typically range from 1 GB to 5 GB, depending on the size of the object in the image.

With the opacity mattes we can compute the visual hull. We resampled all of our visual hull models to 512×512 resolution of the LDC tree. The processing time to compute the visual hull and build the point-based data structure is typically less than 10 minutes. The size of the result LDC tree is about 100MB.

We apply principal component analysis on the surface reflectance fields for compression. The PCA analysis of the surface reflectance field takes about 36 hours on a single PC using non-optimized Matlab code. To speed up the computation, we typically use a PC cluster to process different viewpoints in parallel. We achieve a compression ratio of about 10 to 1, and the total size of the result PCA images are about 2-4GB.

The rendering of the surface light field, or a surface reflectance fields preprocessed with novel illumination, takes between 60 to 200 seconds, depending on the number of point samples in the model. With environment matting, the rendering time goes up to about 300 seconds per frame.

Chapter 8

Conclusion and Future Work

We have demonstrated the techniques of image-based 3D scanning based on the opacity hulls with our system. It fulfilled our goal of an automatic scanning system which can handle objects made of general materials, and objects that have complex fine-scale geometry. The system acquires both an approximate geometry, based on the opacity hull, and the surface reflectance field which allows for reillumination under novel environments. High-quality renderings of the objects are produced by interpolation of the observed viewpoints, and the object blends seamlessly with the background with the aid of view-dependent opacity. View-dependent opacity also improve the fidelity of geometry visually on the visual hull, especially on the silhouettes. The application of environment mattes have extended our system to handle transparent and refractive objects, and the ray-interpolation scheme during rendering produces smooth transitions of the environment mattes between observed viewpoints.

However, the system and the methods employed are not ideal, and there are many avenues for future work in related topics. Our hardware setup, being a prototype system, is far from perfect. The main bottleneck of our scanning are the turntables, which take about 20 seconds to achieve a 10 degree rotation. Our cameras have fast frame rates and the scanning time can be improved significantly if we can rotate the turntables at a higher speed while maintaining stability. Also, while we can have an arbitrary dense set of cameras along the longitudinal dimension with the variable step size of the turntables, the resolution of our light field/reflectance field sampling is highly limited as we can only put six cameras on the latitudinal dimension. This limitation is due to the physical size of our cameras. We can improve the density if we use smaller cameras, or if we position the cameras at a further distance from the object. Also, our physical setup only allows us to capture views and illumination in the upper hemisphere. One solution would be to invert the object for a second scan, and combining the data from the two scan by aligning the two visual hull.

Our scanning hardware also limits the size of the acquired objects to be about 1 foot in each dimension. It does not allow scanning of faces, people, or dynamic objects. One could imagine extending our approach to hand-held or room size scanners. Major technical difficulties include accurate camera calibration, opacity mattes extraction, and controlled illumination. However, our approach is scalable and we believe there is a spectrum of possible digitizer implementations with varying quality and features based on our approach.

Due to the approximate visual hull shape, our techniques have problems in areas of concavities. The lack of accurate geometry can lead to jumping of features over a concavity with a pronounced texture. This could be addressed by improving the geometry using computer vision techniques. Another solution is to use adaptive sampling by taking more images in areas where the change of view-dependent radiance data per surface point is sufficiently non-smooth.

The opacity hull has produced nice results in terms of reproducing the aggregate effects of tiny geometry. While a general parameterization of opacity does not seem to be practical based on our pilot study, it is possible that we can parameterize it under certain constraints. Also, we have employed the unstructured lumigraph interpolation for our opacity interpolation. While this has the advantage of being consistent with the interpolation for the radiance values, it is not obvious that it would be the best way to interpolate opacity.

With environment matting, we are able to sample part of the illumination hemisphere at high-resolution (Ω_h), to correctly represent refractive objects with dominant contributions from the opposite side of the cameras. Refractions that originate from Ω_l possibly due to multiple bounces, and mirror reflections from the front are heavily undersampled. To represent general reflections/refractions, we would need to employ environment matting for the entire hemisphere, which would require more monitors or a rotating setup. Also we have assumed that reflectance at each surface point can be modeled by single refracted and reflected rays. This is clearly not general. Exploring the implications this has on factoring and interpolation is a subject of future work.

We have applied only minimal lossy compression to our data. The availability of alpha mattes for each image allows the application of the shape adaptive compression available in JPEG 2000 and MPEG-4. The temporal coherence of the acquired images should help in achieving high compression ratios. We also plan to use adaptive reconstruction errors for lossy compression of the reflectance field data.

Finally, the rendering speed of our objects are far from interactive. Indeed, recent work by Vlasic et al [50] has employed hardware acceleration to render our surface light fields with view-dependent opacity interactively. Further research could investigate interactive rendering of surface reflectance fields under dynamic illumination environment.

Bibliography

[1] Michael Ashikhmin, Simon Premŏze, and Peter Shirley. A Microfacet-Based BRDF generator. In Sheila Hoffmeyer, editor, *Proceedings of the Computer Graphics Conference 2000 (SIGGRAPH-00)*, pages 65–74, New York, July 23–28 2000. ACM-Press.

[2] P. Beardsley and J. Alon. Robust recovery of camera motion for a hand-held 3d scanner. Technical Report TR 2002-17, MERL, 2002.

[3] F. Bernardini, I. Martin, and H. Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE Trans. on Vis. and Comp. Graph.*, 7(4):318–332, Oct.-Dec. 2001.

[4] Paul J. Besl. Active Optical Range Imaging Sensors. In J. L. C. Sanz, editor, *Advances in Machine Vision: Architectures and Applications*. Springer-Verlag, 1988.

[5] James F. Blinn. Models of light reflection for computer synthesized pictures. *Computer Graphics*, 11(2):192–198, July 1977.

[6] James F. Blinn and Martin E. Newell. Texture and reflection in computer generated images. *Communications of the ACM*, 19(10):542–547, October 1976.

[7] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 425–432, Los Angeles, CA, 2001.

[8] S. E. Chen. Quicktime vr – an image-based approach to virtual environment navigation. In *Computer Graphics*, SIGGRAPH 95 Proceedings, pages 29–38, Los Angeles, CA, August 1995.

[9] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Computer Graphics*, SIGGRAPH 93 Proceedings, pages 279–288, Anaheim, CA, August 1993.

[10] W.-C. Chen, R. Grzeszczuk, and J.-Y. Bouguet. Light field mapping: Efficient representation and hardware rendering of surface light fields. In *Computer Graphics*, SIGGRAPH 2002 Proceedings, San Antonio, TX, July 2002.

[11] Y.-Y. Chuang, D. Zongker, J. Hindorff, B. Curless, D. Salesin, and R. Szeliski. Environment matting extensions: Towards higher accuracy and real-time capture. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 121–130, 2000.

[12] R. L. Cook and K. E. Torrance. A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1):7–24, January 1982.

[13] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 303–312, New Orleans, LA, August 1996.

[14] Kristin J. Dana, Bram van Ginneken, Shree K. Nayar, and Jan J. Koenderink. Reflectance and texture of real-world surfaces. In *ACM Transactions on Graphics*, volume 18 (1), pages 1–34, 1999.

[15] P. DeBevec, T. Hawkins, C. Tchou, H.-P. Duiker, W. Sarokin, and M. Sagar. Acquiring the reflectance field of a human face. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 145–156, July 2000.

[16] P. DeBevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Computer Graphics*, SIGGRAPH 97 Proceedings, pages 369–378, Los Angeles, CA, 1997.

[17] P. DeBevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 11–20, August 1996.

[18] P. DeBevec, Y. Yu, and G. Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Proceedings of the 9th Eurographics Workshop on Rendering*, pages 105–116, Vienna, Austria, June 1998.

[19] A. Georghiades, P. Belhumeur, and D. Kriegman. Illumination-based image synthesis: Creating novel images of human faces under differing pose and lighting. In *IEEE Workshop on Multi-View Modeling and Analysis of Visual Scenes*, pages 47–54, April 1999.

[20] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. Cohen. The lumigraph. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 43–54, New Orleans, LS, August 1996.

[21] T. Hawkins, J. Cohen, and P. DeBevec. A photometric approach to digitizing cultural artifacts. In *2nd International Symposium on Virtual Reality, Archaeology, and Cultural Heritage*, Glyfada, Greece, November 2001.

[22] K.Nishino, Y.Sato, and K.Ikeuchi. Eigen-texture method: Appearance compression based on 3d model. In *Proc. of Computer Vision and Pattern Recognition*, pages 618–624, June 1999.

[23] M. Koudelka, P. Belhumeur, S. Magda, and D. Kriegman. Image-based modeling and rendering of surfaces with arbitrary brdfs. In *Proc. of Computer Vision and Pattern Recognition*, page in press, Kauai, HI, December 2001.

[24] Eric P. F. Lafortune, Sing-Choong Foo, Kenneth E. Torrance, and Donald P. Greenberg. Non-linear approximation of reflectance functions. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 117–126. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[25] A. Laurentini. The visual hull concept for silhouette-based image understanding. *PAMI*, 16(2):150–162, February 1994.

[26] J. Lengyel, E. Praun, A¿ Finkelstein, and H. Hoppe. Real-time fur over arbitrary surfaces. In *Symposium on Interactive 3D Graphics*, pages 227–232, 2001.

[27] H. Lensch, J. Kautz, M. Goesele, W. Heidrich, and H.-P. Seidel. Image-based reconstruction of spatially varying materials. In *Proceedings of the 12th Eurographics Workshop on Rendering*, June 2001.

[28] M. Levoy and P. Hanrahan. Light field rendering. In *Computer Graphics*, SIGGRAPH 96 Proceedings, pages 31–42, New Orleans, LS, August 1996.

[29] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk. The digital michelangelo project: 3d scanning of large statues. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 131–144, Los Angeles, CA, July 2000.

[30] W. E. Lorensen and H. E. Cline. Marching cubes: a high resolution 3D surface construction algorithm. In M. C. Stone, editor, *SIGGRAPH '87 Conference Proceedings (Anaheim, CA, July 27–31, 1987)*, pages 163–170. Computer Graphics, Volume 21, Number 4, July 1987.

[31] T. Malzbender, D. Gelb, and H. Wolters. Polynomial texture maps. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 519–528, Los Angeles, CA, 2001.

[32] S. Marschner, S. Westin, E. Lafortune, K. Torrance, and D. Greenberg. Image-based brdf measurement including human skin. In *Proceedings of the 10th Eurographics Workshop on Rendering*, pages 139–152, Granada, Spain, June 1999.

[33] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan. Image-based visual hulls. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 369–374, Los Angeles, CA, July 2000.

[34] W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan. Image-based 3d photography using opacity hulls. In *Computer Graphics*, SIGGRAPH 2002 Proceedings, San Antonio, TX, July 2002.

[35] W. Matusik, H. Pfister, R. Ziegler, A. Ngan, and L. McMillan. Acquisition and rendering of transparent and refractive objects. In *Proceedings of the 13th Eurographics Workshop on Rendering*, Pisa, Italy, June 2002.

[36] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *Computer Graphics*, SIGGRAPH 95 Proceedings, pages 39–46, Los Angeles, CA, August 1995.

[37] Microsoft. Microsoft easy camera calibration tool. Online Software.

[38] G. Miller, S. Rubin, and D. Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *Proceedings of the 9th Eurographics Workshop on Rendering*, pages 281–292, Vienna, Austria, June 1998.

[39] F. E. Nicodemus, J. C. Richmond, J. J. Hsia, I. W. Ginsberg, and T. Limperis. Geometric Considerations and Nomenclature for Reflectance. Monograph 161, National Bureau of Standards (US), October 1977.

[40] K. Nishino, Y. Sato, and K. Ikeuchi. Appearance compression and synthesis based on 3d model for mixed reality. In *Proceedings of IEEE ICCV '99*, pages 38–45, September 1999.

[41] T. Porter and T. Duff. Compositing digital images. *Computer Graphics*, 18(3):253–259, July 1984.

[42] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle. View-based rendering: Visualizing real objects from scanned range and color data. In *Eurographics Rendering Workshop 1997*, pages 23–34, June 1997.

[43] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 117–128, 2001.

[44] H. Rushmeier, F. Bernardini, J. Mittleman, and G. Taubin. Acquiring input for rendering at appripriate levels of detail: Digitizing a pietà. In *Proceedings of the 9th Eurographics Workshop on Rendering*, pages 81–92, Vienna, Austria, June 1998.

[45] P. Sander, X. Gu, S. Gortler, H. Hoppe, and J. Snyder. Silhouette clipping. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 327–334, 2000.

[46] Y. Sato, M. D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. In *Computer Graphics*, SIGGRAPH 97 Proceedings, pages 379–387, 1997.

[47] A. R. Smith and J. F. Blinn. Blue screen matting. In *Computer Graphics*, volume 30 of *SIGGRAPH 96 Proceedings*, pages 259–268, 1996.

[48] G. Turk and M. Levoy. Zippered polygon meshes from range images. In *Computer Graphics,* SIGGRAPH 94 Proceedings, pages 311–318, July 1994.

[49] J.K. Udupa and D. Odhner. Shell rendering. *IEEE Computer Graphics & Applications*, 13(6):58–67, November 1993.

[50] D. Vlasic, H. Pfister, S. Molinov, R. Grzeszczuk, and W. Matusik. Opacity light fields: Interactive rendering of surface light fields with view-dependent opacity. In *Symposium on Interactive 3D Graphics*, 2003.

[51] Greg Ward. Real pixels. In *Graphics Gems II*, pages 80–83. Academic Press, Boston, 1991. ISBN 0-12-064481-9.

[52] D. Wood, D. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle. Surface light fields for 3d photography. In *Computer Graphics*, SIGGRAPH 2000 Proceedings, pages 287–296, Los Angeles, CA, July 2000.

[53] Y. Yu, P. DeBevec, J. Malik, and T. Hawkins. Inverse global illumination: Recovering reflectance models of real scenes from photographs. In *Computer Graphics*, SIGGRAPH 99 Proceedings, pages 215–224, August 1999.

[54] Z. Zhang. A flexible new technique for camera calibration. Tr98-71, MSR Redmond, 1998.

[55] D. Zongker, D. Werner, B. Curless, and D. Salesin. Environment matting and compositing. In *Computer Graphics*, SIGGRAPH 99 Proceedings, pages 205–214, August 1999.

[56] M. Zwicker, H. Pfister., J. Van Baar, and M. Gross. Surface splatting. In *Computer Graphics*, SIGGRAPH 2001 Proceedings, pages 371–378, Los Angeles, CA, July 2001.