Szymon Rusinkiewicz, Tim Weyrich, Wojciech Matusik, Filip Sadlo, Ronald Peikert, Hanspeter Pfister. "Acquisition." Chapter 3 in M. Gross and H. Pfister (eds.), *Point-Based Graphics*, Morgan Kaufmann / Elsevier, 2007, pp. 18–93. DOI 10.1016/B978-012370604-1/50004-3

CHAPTER 3

ACQUISITION

Szymon Rusinkiewicz, Princeton University, Department of Computer Science
Tim Weyrich, ETH Zürich, Computer Graphics Laboratory
Wojciech Matusik, MERL - Mitsubishi Electric Research Laboratories
Filip Sadlo, ETH Zürich, Computer Graphics Laboratory
Ronald Peikert, ETH Zürich, Computer Graphics Laboratory
Hanspeter Pfister, MERL - Mitsubishi Electric Research Laboratories

INTRODUCTION

This chapter provides a comprehensive overview of state-of-the-art 3D acquisition and scanning methods for point-sampled models. The chapter will focus both on geometry and appearance acquisition. The discussed methods and systems are intended to make the reader familiar with the essentials of scanning technology. Section 3.1 gives an overview of 3D geometry acquisition and introduces the basic concepts of 3D scanning. Section 3.2 demonstrates how the presented methods can be utilized in practice to design and build a low-cost 3D scanning system. The chapter concludes with the more advanced topic of 3D digital photography. Section 3.3 presents a method and system for appearance acquisition.

The field of 3D acquisition has emerged as an important source of both challenges and applications for point-based techniques. The high detail and ease of acquisition of 3D models of real-world objects have driven their acceptance in computer graphics productions, but the size and density of these datasets have required the development of efficient representations and processing algorithms. In this chapter, we present both the background of 3D scanning technology and related algorithms, as well as a complete example of a 3D model acquisition pipeline built around point-based

techniques. Note that in this chapter we deal exclusively with 3D *surface* scans and scanners, as opposed to the volumetric data (representing density as a function of position in space) produced by devices including computed tomography (CT) and magnetic resonance imaging (MRI) scanners.

Among the reasons for the natural affinity of 3D scanners and points are:

- *Size*: 3D scans can be large and detailed, with models in the range of 10^6 to 10^9 samples becoming commonplace. As argued throughout this book, point-based methods achieve greater efficiency in storage and certain kinds of geometric processing, and these advantages are put to greatest effect on scanned models.
- *Noise*: All scans have noise, and depending on the particular scanner design this noise can be on the order of the sample spacing. This makes it difficult to infer and preserve mesh connectivity, leading to a strong preference for techniques that defer or avoid inferring adjacency information and compute surface properties such as normals, when required, by considering variable-sized neighborhoods in space.
- *Sparseness*: Some 3D acquisition techniques, especially passive methods, can only return reliable geometry at a few locations on the surface (essentially, at "features" or "texture" resulting from color variation). This makes it error prone to estimate the connectivity of points within a scan, and mistakes can lead to difficulty in combining multiple scans of the same object. Even for those scanning technologies that return "dense" point clouds, establishing the connectivity of nearby samples must rely on a heuristic to avoid connecting across the depth discontinuities visible from a single view. Point-based techniques, by omitting or deferring the estimation of connectivity, can avoid such difficulties.

3.1

ACQUISITION OF POINT-SAMPLED GEOMETRY

Szymon Rusinkiewicz

This section presents an overview of 3D scanning hardware, focusing on the resolution, noise, and point density achievable by the different available technologies. In addition, it describes the algorithms for registration and scan merging necessary to go from raw 3D *data* to finished *models.* To see the need for such algorithms, we begin with an overview of the 3D model acquisition pipeline, some version of which appears in every research and commercial scanning system.

3.1.1 OVERVIEW OF THE 3D ACQUISITION PIPELINE

To begin, let us discuss the important distinction between *3D scanners* and *3D model acquisition systems.* The former refers simply to devices capable of returning the shape of (some portion of) surfaces visible from one viewpoint. These data are useful in a variety of contexts including robot navigation, metrology (e.g., for quality control), and cases in which the complete surface actually can be captured from one direction (e.g., terrain). In addition, single scans may be used together with color images for image-based modeling and rendering (IBMR) applications, such as view interpolation or foreground/background segmentation. For almost all objects, however, a single "scan" can only capture part of the surface. Thus, for most applications in graphics it is necessary to combine the information in 3D scans taken from multiple (usually overlapping) viewpoints, making necessary the development of a "pipeline" of algorithms that operate on multiple 3D scans and attempt to produce a single, integrated 3D model. We shall refer to this use of scanning as *3D model acquisition.*

A basic 3D model acquisition pipeline may include the following stages:

- *3D scanning*: A scan is taken at some viewpoint.
- *Registration*: The transformation of the newest scan relative to other scans is computed. This may include a combination of tracking the position of the scanner, user input, feature detection and matching, or iterative minimization of scan-to-scan distance. After all scans have been acquired, a *global registration* algorithm may be used to simultaneously minimize misalignment errors over all pairs of overlapping scans.
- *Merging*: The aligned scans frequently contain significant regions in which many scans overlap. Merging these logically separate scans into a single model both reduces storage and averages away some of the scanning noise (while, hopefully, keeping the "signal"). Many merging algorithms are possible, of which the most popular approaches include averaging of implicit functions [HDD+92, CL96, Kaz05]; stitching or "zippering" of the original range images [TL94]; moving least squares [Lev04, ABCO+03, AA04b, AK04, DS05]; and directly triangulating the union of the point clouds of all scans [EM94, ABK98, ACK01a, DG04]. Some of these methods are described in detail elsewhere in this book.

- *View planning*: A decision is made about where to position the scanner relative to the object in order to perform the next scan. In most systems, this decision is left to the user, though there has also been research on automated view planning systems [MB93, SA98]. In any case, view planning usually requires the availability of partial results from the above stages, in order to detect holes or undersampled regions in the data. In cases in which the view planning is performed by the user, these partial results must be in a form that can be efficiently rendered [RHHL02].
- *Postprocessing*: Depending on the final use for the model, the output of the merging step may undergo further processing [WPK+04], including noise or outlier reduction, automated filling of small remaining holes [DMGL02, SACO04, PR05], and curvature-adaptive resampling or decimation.

The form of the data that is passed among these stages can have a significant influence on the efficiency of the algorithms. Since it is the goal of this book to describe and advocate the use of point-based methods, we will confine ourselves to describing two possible representations for these points.

1. One possibility is to represent the data as *unorganized point clouds.* That is, the data consist of an unordered list of 3D point locations, together with any other per-point properties that are necessary or relevant.
2. A second possibility is to represent the data using *range images.* This is a data structure, analogous to a regular image, in which one stores not a color but a *depth* along each of a regularly spaced set of rays in space. Range images go by many names, and are also called range surfaces, depth images, depth maps, height fields, 2 ½D images, etc.

While unorganized point clouds are conceptually simple and are appropriate for all scanners and all stages of the pipeline, it is nevertheless the case that many 3D scanners naturally return a range image as their fundamental data type, and there are some advantages to maintaining the range image organization (at least until the merging stage of the pipeline). First, range images implicitly store information about empty space (i.e., the portion of each ray between the center of projection and the sample on that ray is known to be empty), which is used by some registration and merging algorithms. In addition, there is an implicit and easily calculated notion of adjacency or connectivity provided by a range image that, with some caveats, may be used to accelerate local computations such as determining normals. Of course, merely looking at neighbors in the range image does not provide connectivity directly, since one must be careful to avoid considering points lying across a depth discontinuity as being adjacent. In practice, this decision of whether or not to consider neighboring points as being adjacent is made on the basis of a heuristic, based either on a fixed depth difference threshold or a threshold on the ratio of depth difference (e.g., in z) to sample spacing (in x and y).

Because algorithms for the merging and processing stages of the 3D model acquisition pipeline are described elsewhere in this book, the remainder of this section will focus on 3D scanning technologies and registration algorithms. For the 3D scanners, our goal is not to provide a complete

description of how each possible technology is implemented (such details may be found in sources such as computer vision books [Fau93, TV98, FP03a]), but to describe enough of the process to be able to understand the characteristics of the resulting data.

3.1.2 TRIANGULATION-BASED 3D SCANNERS

We begin our look at 3D scanning technologies by considering the scanners based on the principle of triangulation. All of these scanners have the common element of possessing two (or more) viewpoints on the object being scanned, and determining *correspondences* between rays from those viewpoints. That is, they are able to find rays from the two viewpoints that intersect at a common 3D point on the surface of the object, and find the coordinates of that point by computing the intersection of the rays (Figure 3.1). This, of course, requires calibration: it is necessary to know the mapping of pixels from the two views to rays in space [Tsa86, HS97].

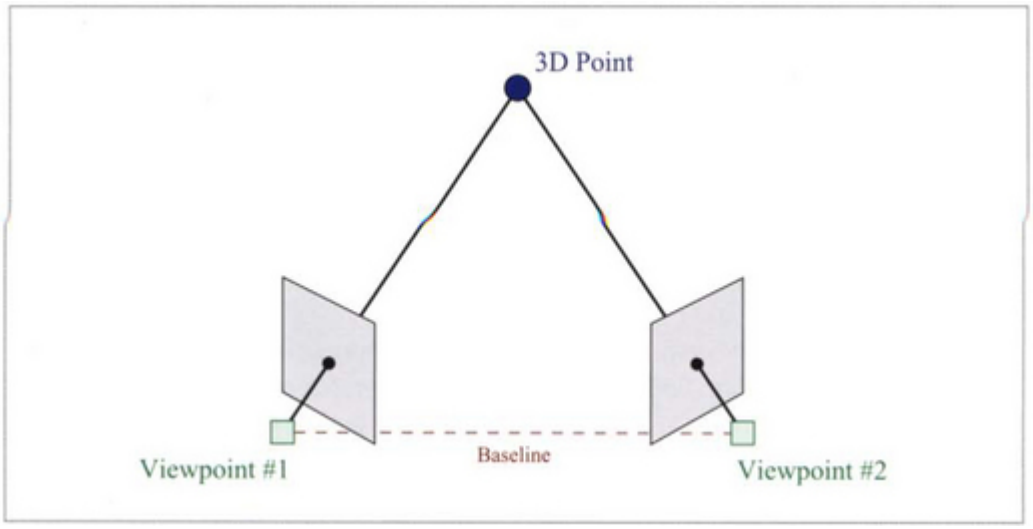

Figure 3.1 Triangulation-based 3D scanners find the positions of points on a surface by computing corresponding pixels from two viewpoints. The correspondence defines a pair of rays in space, and the intersection of the rays determines a 3D position.

The above description is deliberately vague about what exactly the two "viewpoints" represent. In fact, there are many possible variants of triangulation systems, differing largely in what type of device is present at the two viewpoints. Conceptually, the simplest setup is *passive stereo*, in which the viewpoints contain cameras and no controlled light is introduced into the scene. In this case, the correspondence problem requires finding pixels (or local neighborhoods of pixels) in one image that are as similar as possible to pixels or neighborhoods in the other. This problem proves to be difficult in many cases, so *active stereo* systems augment the setup with a spatially or temporally varying projected pattern designed to introduce features into the scene that make the correspondence problem easier to solve. This leads naturally to *structured-light* systems, in which one of the viewpoints for triangulation is replaced with a projector of known patterns of light. One important variant of structured light, namely *single-light-stripe* scanning, has been frequently implemented in commercial systems, largely because of its potentially low cost and the ability to optimize it for very

high precision. Finally, *multibaseline stereo* and *structure from motion* systems rely on more than two viewpoints on the scene to obtain advantages such as higher robustness or fewer calibration requirements.

As might be guessed from the above description, there are many qualitative differences in the kinds of data that may be obtained using each of the possible triangulation setups. Some, such as passive stereo, return data that are sparse and often noisy. In contrast, light-stripe scanners return dense, high-quality data with relatively few outliers, but do so at the cost of introducing light into the scene, as well as their long acquisition time (since the stripe must be swept across the scene). There are, however, certain advantages and disadvantages shared by all triangulation-based scanners. Among the advantages are

- *Flexible working volume*: The working volume (i.e., region of space in which the object being scanned must be contained) is a function of parameters such as the field of view of the camera(s), but most importantly is directly proportional to the baseline (i.e., the distance between the two viewpoints used for triangulation). Because of this, it is relatively straightforward to design triangulation-based scanners covering a wide range of working volumes, ranging from millimeters to tens of meters. At the low end, the limits are the same as those present in all scanners that rely on light, namely that they are inappropriate when the object being scanned begins to approach the wavelength of light (400–700 nm). The limits on the high end are discussed below.

- *Low complexity and cost*: Triangulation-based scanners require calibrated cameras and, depending on the design, possibly calibrated projectors, light sources, and/or motors. These are items that are readily available in both research and commercial settings, and are less expensive and easier to work with than some of the exotic devices necessary for designs based on principles such as time of flight. In particular, the rapid technological progress in electronic imager technology in the past several years has dramatically increased the resolution and decreased the cost of digital cameras, while the development of DLP "micromirror" chips has led to speed and precision gains for light projectors.

- *Precision scales with camera resolution*: Since triangulation-based scanners rely on ordinary cameras, and since the resolution of consumer-grade cameras had advanced considerably in the past decade, it is becoming easier than ever to design scanners that have both high precision and larger working volumes.

Some of the common disadvantages are

- *Two-line-of-sight problem*: In order to return the 3D position of a point, triangulation scanners must observe that point from two separate viewpoints. Depending on the specifics of the scanner, this may cause one of two different problems. For active systems, this merely implies that the patch of surface acquired from one scanner position will typically not cover one complete side of the surface: it will have missing data in regions shadowed from one view or the other. In extreme cases, such as a deep indentation in the surface, there may not be *any* position in which the scanner may be placed such that the deepest part of the indentation is unshadowed from both views simultaneously. This is then an ultimate limitation on scannable surface geometry that

cannot be overcome without changing the design (e.g., by reducing the baseline and hence the triangulation angle). For passive systems, the two-line-of-sight problem is potentially even more serious. In particular, the fact that there are regions of the surface visible from one camera but not the other can lead to difficulties in recognizing that the correspondence problem does not have a solution for those points. Special techniques, often relying on global regularization, are necessary to prevent the correspondence algorithm from returning outliers in such cases.

- *Sensitivity to shiny or translucent objects*: Essentially all triangulation systems make the implicit assumption that a diffuse, opaque surface is being scanned. Specular reflection and subsurface scattering violate this assumption and lead to errors in the returned depth estimates. For specular reflection, these errors can be arbitrarily large, as interreflection of light from multiple surfaces can lead to false correspondences between the reflections. These problems are exacerbated if the ratio of specularly reflected light to diffuse reflection is large, as is the case for dark objects. The effects of translucency on triangulation are less serious, and typically show up as a bias in the estimated depth (i.e., the scanned model is some distance *inside* the correct surface [GBR+01]). Of course, for completely transparent objects it is difficult to acquire depth using any optical means.
- *Difficulties for large scenes*: Although triangulation systems scale to different scene sizes better than many other designs, the practicality of scanning large scenes is limited by the difficulty of constructing and calibrating a system with a large baseline. Active triangulation systems face an additional difficulty in that the light they introduce must be detectable over any ambient light that is present. This leads to difficulties in using active systems for outdoor scanning during daylight.

We now describe the most popular triangulation scanner designs in more detail. Note that although we present these systems in something of a logical progression, emphasizing the similarities between them, this in fact has not been how these systems have traditionally been considered in the research literature. Indeed, there has been relatively little communication and exchange of ideas between, for example, the stereo and light-stripe scanner communities. This is beginning to change, however, with the development of some hybrid scanner designs that combine ideas originally explored in different research communities.

Passive Stereo

Constructing an effective 3D scanner based on stereo is essentially synonymous with solving the correspondence problem between images taken with two cameras. Passive stereo begins with the immediate recognition that it is hopeless to solve this problem by looking at individual pixels, hence it is necessary to match *regions* in one image against the other. Depending on the amount of texture in the scene, the optimal size of the patches may range from 3 × 3 pixels to, for example, 25 × 25 pixels or even larger.

In the simplest cases, the correspondence is computed merely by evaluating some *matching cost function* between a window of pixels in one image, known as the *reference image*, and windows of pixels in the other image at various *disparities*, or displacements in pixel location. The match having the lowest cost is taken as the correct one. The matching cost function ψ may be as simple as the sum of squared differences (SSD):

$$\psi_{SSD}(\delta) = \sum_i \left(I_1(i) - I_2(i+\delta)\right)^2, \tag{3.1}$$

where I_1 and I_2 are the two images, δ is the disparity, and the sum is overall pixels in the matching window.

Although the SSD metric corresponds strongly to the intuitive notion of similarity, several other functions are common. For example, the sum of absolute differences (SAD) function replaces the square with an absolute value, leading to less sensitivity to outliers. The normalized cross-correlation (NCC) incorporates terms that make the function sensitive only to brightness differences within the window, ignoring any overall difference in brightness between the two images:

$$\psi_{NNC}(\delta) = -\sum_i \frac{(I_1(i) - \bar{I}_1)(I_2(i+\delta) - \bar{I}_2)}{\sigma_1 \sigma_2} . \tag{3.2}$$

Here, $\bar{I}$ and σ are the mean and standard deviation of pixel values in the window, as evaluated in both the left and right images. Note that, for consistency with other matching functions, the sum is negated so that lower matching cost scores indicate higher similarity.

Although a naive implementation of stereo correspondence would require, for a window in the reference image, evaluating this similarity function at all possible windows in the other image, there is an important property of multiview or *epipolar* geometry that reduces the search space greatly. This is based on the observation that the correspondence to a point in one image must lie along the projection of the ray of that pixel into the other image. Thus, given any pixel in the reference image, it is possible to compute the *epipolar line* in the other image along which the match must lie.

In fact, in many stereo systems an additional shortcut is used to simplify this search even further. This is based on the observation that any plane through the baseline connecting the centers of projection of the two cameras intersects the two image planes in a pair of lines. Any points on one of the lines must have correspondences on the *conjugate epipolar line* in the other image plane (Figure 3.2). The process of *rectification* involves warping the two images such that conjugate epipolar lines lie, for example, along horizontal scanlines, and in fact may be done such that correspondences lie along the *same* scanline in both images [Fau93]. The warps necessary for rectification may be computed from the intrinsic and extrinsic calibration of the two cameras (i.e., the field of view and relative position and orientation), and performing the warps is typically relatively fast compared to the rest of the correspondence-finding process.

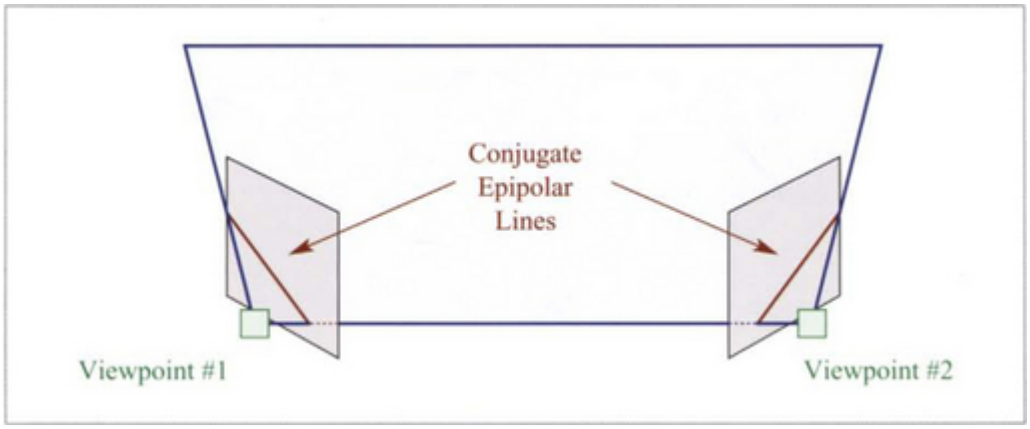

Figure 3.2 Each plane passing through the baseline intersects the two image planes in a pair of *conjugate epipolar lines*. All matches to points on the line in one image must lie on the conjugate line in the other image.

So, finding the correspondence between windows can be reduced to considering one window in the rectified reference image and evaluating the matching function for windows in the other rectified image, considering all possible horizontal disparities. The disparity yielding the highest similarity is chosen as the correct one (possibly with some threshold on the similarity value used to eliminate false matches), and rays corresponding to the two corresponding points are triangulated to find a 3D location. Many things can still go wrong, however. First, it is certainly the case that two image windows at the location of the correct match can look dissimilar. This can be because of the presence of nondiffuse materials, or because the surface is tilted a different amount relative to the two cameras, leading to a difference in the amount of foreshortening present in the two views. To address the first of these problems, one may use a matching metric such as NCC, which tends to be more robust to small differences in appearance, or a technique such as Helmholtz stereopsis (an active method), which relies on the reciprocity of Bidirectional Reflectance Distribution Functions (BRDFs) to compensate for non-Lambertian reflectance [ZBK02]. To address the problem of differential foreshortening, a common method is to use an iterative technique in which a first round of correspondence finding is used to estimate the shape of the surface, which in turn is used in a second round to warp the images of windows to account for the estimated foreshortening [LK81]. The process may be repeated a few times to obtain more precise correspondence estimates.

The above problems are, however, minor compared to the major weakness of passive stereo, which is the presence of ambiguous matches. Since the correspondence finding is at the mercy of the brightness variations that occur naturally in the scene, the presence of either regions of constant color or repeated texture can lead to equally low matching cost at multiple disparities. Of the many methods that have been developed to address this problem, here we will consider two.

One method for dealing with ambiguous matches involves adapting the window size, shape, and weighting to the data themselves. The basic idea is that larger windows can reduce ambiguity, but also lead to more blurring in the estimated shape. Thus, windows should be expanded locally, and only in regions in which this is necessary to provide unambiguous correspondence. Moreover, the pixels near the center of the window should be given more weight than pixels near the edge, in an effort to preserve local detail when this is possible. One system implementing this approach is

described by Scharstein and Szeliski [SS98]. They propose to start with the matching score for single-pixel windows and "diffuse" a fraction of the matching score from neighboring pixels in cases in which this reduces the matching certainty. The latter property is measured using metrics such as winner margin (i.e., what is the difference in the matching score between the best match and the next-best match) or entropy of the matching scores across all disparities.

An alternate method for reducing the errors due to ambiguous matches relies on regularization. That is, a scoring function is constructed that considers not only the matching score at each location, but also the extent to which the computed disparities are smooth across the scene. The latter is measured locally, as the difference between the disparity at each location and at its neighbors, but in practice it should be a nonlinear function of the disparity difference to prevent excessive blurring across true depth discontinuities. For example, a function such as the following is often used:

$$V(i,j) = \min(|\ \delta_i - \delta_j\ |, K), \tag{3.3}$$

where a penalty V is assigned to the disparities δ found at neighboring pixels i and j. The threshold K represents the disparity difference considered to be a true depth discontinuity, and no additional penalty is applied for greater jumps in disparity.

One may thus construct an energy function consisting of both matching and regularization terms, and attempt to compute a global assignment of disparities that minimizes the energy. Unfortunately, such an energy function invariably contains many local minima and minimizing it is provably NP-hard. Most practical algorithms use heuristics to find approximate minima of this function, using methods such as simulated annealing. Recent work by Boykov et al., for example, has demonstrated how to use graph min-cut algorithms to compute excellent approximations to the minimum of this function [BVZ01], while work of Meltzer et al. has shown that in some cases it is possible to compute the global minimum of the energy function in reasonable time, though of course there are no guarantees of fast running time [MYW05].

Ultimately, both the variable window-size and regularization approaches serve to return dense depth estimates by interpolating and extrapolating information from locations at which the correspondences are truly unambiguous. This motivates an alternative general approach to correspondence finding that first performs *feature detection* to locate regions in the image likely to have unambiguous matches, then simply tries to match those features between the two images. Although these methods return sparse correspondences, such matches are often of consistently high quality and, depending on the application, may be more useful than the dense but low-quality matches returned by correllation-based approaches. One such system is the Marr-Poggio method, which is based on the proposition that image edges, in conjunction with the epipolar constraint, make good features for matching [MP79]. The edges are detected and matched using a dynamic programming algorithm, which includes a regularization component that attempts to preserve the relative ordering of matches between the two images. The entire process is repeated in a coarse-to-fine manner, leading to coarse matching of the most important edges at early rounds followed by smaller-scale matches of the fine details in the later rounds (during which less smoothing is performed).

Characteristics

Passive methods differ from most of the other techniques considered in this section in that most of the characteristics of the datasets they produce, including density, accuracy, and presence of outliers, depend almost exclusively on the object or scene being acquired, rather than on the cameras and setup of the scanner itself. In the best case, namely scenes with significant texture, stereo can return accurate estimates of depth, and has the advantages of lowest equipment cost and the ability to return depth from a pair of frames taken at a single time instant. The depth maps in this case are also relatively dense, though the maximally accurate matches are still restricted to brightness or color edges in the scene. For this reason, in many applications the output of stereo is best suited to the unorganized point cloud representation. As scene contrast decreases, of course, so do data density and accuracy.

Active Stereo

In situations that permit additional light to be introduced into the scene, significant gains in robustness, accuracy, and data density may be obtained by moving to an active-stereo method. The typical assumption is that a pattern of light is projected into the scene, though the projector or other device used to introduce the light is not itself calibrated with respect to the cameras. The purpose of the projected light is, therefore, purely to reduce the ambiguity of stereo matching by introducing additional visible features in the scene.

The simplest way to incorporate an active element into stereo is to project a static pattern consisting of a high-frequency texture, such as an array of dots, a grid of lines, or even a random binary pattern. A traditional passive-stereo algorithm is run without modification on the resulting images, retaining the advantages described above while minimizing the presence of ambiguous matches. Alternatively, special algorithms that take advantage of knowledge of the projected pattern may be used. This is especially popular with patterns that are regular grids or arrays of dots [PGO96], and algorithms that take advantage of surface continuity to look for neighboring dots or corners are possible. Algorithms such as these are especially effective when acquiring relatively smooth objects with few depth discontinuities, and a popular application is scanning of faces. More complex spatial patterns are also possible, with some based, for example, on arrays of colored dots having specified uniqueness properties. We will look at such patterns in greater detail in the section on structured light.

An alternative way of extending passive stereo to include active lighting is to consider the time dimension. The idea behind temporal active stereo is to project a time-varying pattern into the scene, permitting the stereo matching functions to be evaluated over windows that have *temporal*, rather than spatial, extent [DC01, DNRR05]. As long as each pixel accumulates a unique pattern of lighting over time, the correspondences may be computed robustly using windows that have spatial extent of a single pixel. This leads to greater precision in the reconstruction, and typically reveals significantly more surface detail than methods using spatial windows greater than one pixel.

In fact, with good temporal illumination the ultimate limit on the accuracy of surface reconstruction relies heavily on the ability to perform good *subpixel estimation* of correspondence. A simple implementation of this is to find the best integer match, then consider the matching cost for the two

immediately neighboring pixels. By fitting a parabola to the three samples of the matching-cost-versus-position function, one may analytically find the minimum of the parabola and take its (noninteger) location as the best match. More complex schemes for subpixel matching are possible, and Nehab et al. [NRD05] show that it is necessary to simultaneously refine subpixel positions in both images to avoid bias and noise in the estimates.

While active stereo with static patterns operates on one frame at a time, hence allowing for moving objects, temporal stereo requires the object to remain still for many frames. Using matching with windows having both spatial and temporal extent, however, allows for systems that balance between the high precision of temporal stereo and the tolerance to movement of spatial stereo. Such *spacetime stereo* systems can be tuned for different speeds of motion, by adjusting the relative widths of the matching window in space and time [DNRR05], and, by estimating the motion that is present, can use diagonal windows through space-time to obtain the highest robustness and accuracy [ZCS03]. Recent work has demonstrated the ability of such systems to capture moving objects such as faces [ZSCS04].

Characteristics

The data returned by active stereo systems can be of high quality, and it is not uncommon to have noise in depth less than one-tenth of the sample spacing. The scanning also tends to be robust, with few outlier points (with some exceptions near depth discontinuities). The points returned are dense on the surface (though subject to the two-line-of-sight problem, or even a three-line-of-sight problem if the illumination source is shadowed), meaning that some algorithms can take advantage of the space carving and implicit connectivity provided by a range image representation. Relative to passive stereo, therefore, active stereo systems tend to return better data, at the cost of the requirement to introduce light into the scene and the inability to acquire the color of objects simultaneously with their shape.

Structured Light

Conceptually, structured-light systems may be thought of as active stereo systems in which the second camera is eliminated and the source of illumination is itself considered to be one of the viewpoints on the scene. This has a few disadvantages relative to active stereo, in that it imposes requirements on how precise the active light must be, and requires that it be calibrated (for both intrinsics and extrinsics). This precludes the use of space- or time-varying, yet uncontrolled illumination (which has been referred to as "unstructured light"), and requires full control over the light (and, for temporal systems, synchronization to the camera). On the other hand, structured-light systems have the advantage of reducing the amount of equipment that is required and, more importantly, allowing for algorithms that take advantage of the epipolar constraints between the camera and projector.

It is this latter characteristic that has driven the development of structured-light systems based on "light codes" [PA82, Bes88]. These are spatial or temporal (or, indeed, space-time) patterns of projected light that are carefully designed so that every point visible from the camera is guaranteed to correspond uniquely to one ray from the projector. For example, a spatial light code may consist of an array of colored dots [DN96] having the property that every local subset of $n \times n$ dots has a

different pattern of colors from every other. Alternatively, the code may consist of *stripes* perpendicular to the camera-projector baseline [BK87, CKS98], with the property that any n adjacent stripes have a unique color pattern. In this case, the epipolar constraint between the camera and projector is implicitly used: the triangulation proceeds by intersecting a *ray* from the camera with a *plane* from the projector (Figure 3.3). Because stripe codes are easier to design and easier to detect than codes of dots, most structured-light systems, whether spatial or temporal, use some type of stripe coding.

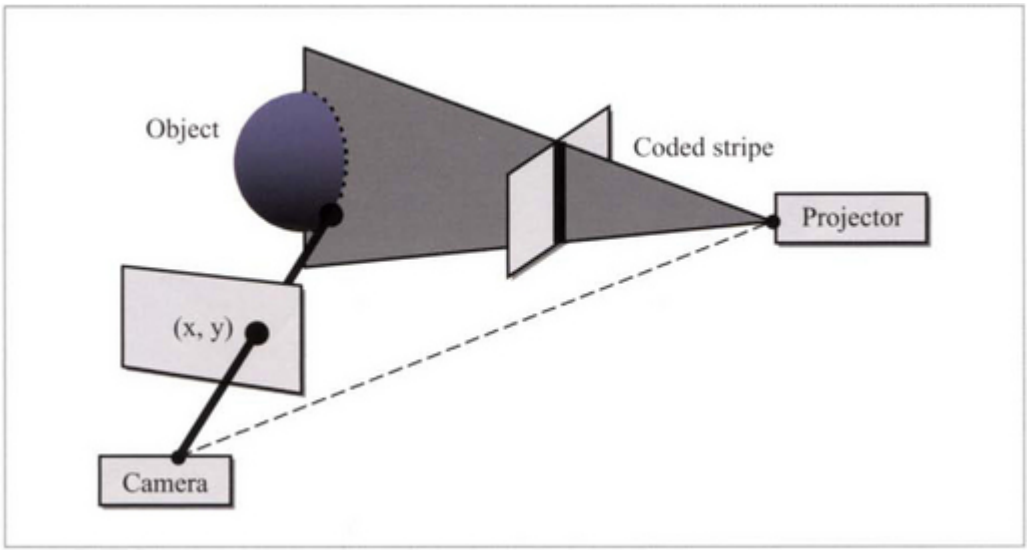

Figure 3.3 Structured-light systems frequently code only stripes of light from the projector, and perform ray-plane intersection to find 3D geometry.

Temporal structured-light systems provide perhaps the most "bang for the buck" in that they return high-quality 3D data for static objects while being easy and inexpensive to build. Indeed, homemade structured light scanners are frequently found in research labs, and they make excellent student projects. Section 3.2 describes a practical scanning system based on temporal structured light.

The most popular temporal structured-light scanners are based on binary stripe coding (i.e., each stripe is either on or off at each frame in time), and the most popular scheme for coding the stripes is based on the *binary Gray code* [BER76]. The latter is a way of arranging the 2^n binary numbers of n bits such that adjacent codewords differ only in one bit (this property leads to greater robustness in "decoding" the observed pattern). The reflected code, the most popular way of constructing a Gray code, arises from the following simple recipe:

- The reflected code for one bit is 0, 1.
- The reflected code for n bits consists of the code for $n-1$ bits, prefixing each element by a 0, followed by the *reverse* of the code for $n-1$ bits, prefixing each element by a 1.

Each codeword is assigned to a stripe, and at frame k, each stripe is colored black or white depending on the value of its k-th bit. The resulting pattern consists of a series of wide stripes on early frames, with the stripes becoming progressively narrower with each frame (Figure 3.4).

Additional all-white and all-black frames are frequently included to allow compensation for surface reflectance and ambient illumination. Even with these additional frames, scanning is usually relatively fast, with the number of required frames being logarithmic in the number of projector stripes.

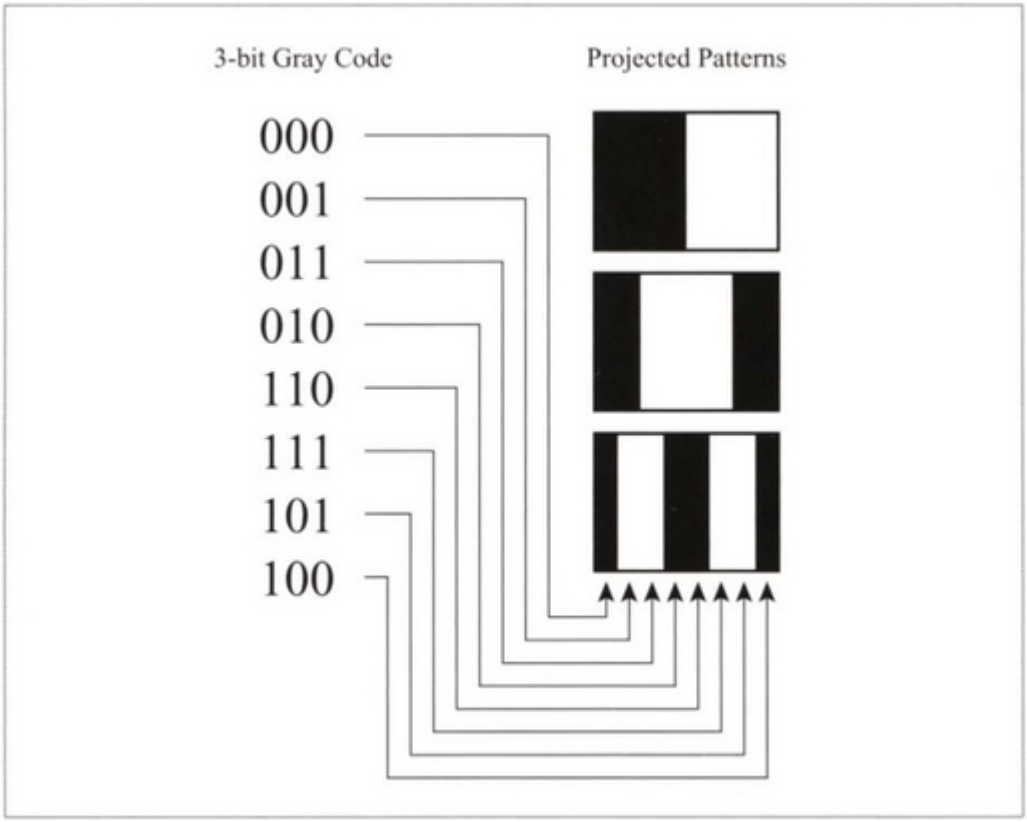

Figure 3.4 Example of a Gray code and the resulting frames that would be projected in a structured-light system.

Two other temporal structured-light codes are frequently encountered, and are worthy of mention. The *phase-shifting* code augments a low-resolution Gray code with a sequence of black-and-white stripes, each several projector pixels wide. The stripes are shifted by one pixel on successive frames, leading to greater robustness and better subpixel matching than is achievable with full-resolution Gray codes (which may include very narrow stripes, several of which can fall on the same camera pixel). The cost for this is the requirement for slightly more total frames in order to uniquely code a fixed number of projector stripes. Moving away from binary codes, the *gray wedge* technique uses only two frames: an all-white frame and a frame in which stripe intensity smoothly varies from black on one side to white on the other [CH85]. The identity of the stripe is found by dividing the intensities observed on the two frames.

Most structured-light systems are either purely spatial or purely temporal, but there has been some work on spatiotemporal codes as well. For example, Hall-Holt and Rusinkiewicz introduced a code in which each *pair* of adjacent stripes, as seen over n frames, is unique [HHR01]. As with spacetime stereo techniques, adjusting the relative spatial and temporal extent of the code-carrying windows provides for a natural trade-off between the degree of spatial continuity (i.e., surface smoothness)

and temporal continuity (i.e., motion) that is assumed. In this case, the method allows for slow motion of the object while assuming relatively little surface continuity. The stripe pairs (or, more specifically, the stripe boundaries) are tracked over time to provide robust and relatively dense 3D estimates at video rates. The scanner may be combined with real-time alignment, merging, and point-rendering methods to enable the entire 3D model acquisition pipeline to run in real time [RHHL02]. More recent research by Koninckx and van Gool has shown how structured-light codes may be adapted to the scene in real time, providing for both higher accuracy and greater robustness [KPDG05].

Characteristics

The data returned by structured-light systems qualitatively share many features with active stereo. Datasets tend to be accurate, dense, and large. Outliers are few, and are mostly caused by depth discontinuities or specular reflections. One interesting difference is the role played by surface texture (i.e., reflectance variation): as opposed to active stereo systems, which are often helped by texture, the data returned by structured-light systems are usually degraded by variations in reflectance. Phenomena such as "texture embossing," in which reflectance variations show up as systematic biases in estimated depth, are often observed [CL95]. In addition, structured-light systems often suffer from missing data in dark regions of the surface, where the projected light patterns cannot be detected.

Light Stripe

A simple way of thinking about light-stripe scanners is that they are structured-light scanners with a pattern that illuminates only one stripe per frame. The visual effect is that of a stripe being swept slowly across the surface, tracing out a single 2D "slice" of geometry at each point in time. This description should make it clear that light-stripe triangulation scanners are significantly slower than, for example, Gray code structured-light systems, but the single-stripe configuration allows for greater optimization of the physical scanner setup, leading to greater accuracy.

The first major optimization performed on light-stripe systems is to use a laser as the light source. This allows the stripe to be made bright and consistent, leading to lower sensitivity to ambient light (especially if a narrow-band filter is used in front of the camera). The use of a laser also allows the stripe to be focused to a very narrow width, leading to high resolution. Next, the laser-stripe generator and camera are often mounted to each other, with the scanning motion consisting of moving the laser and camera as a rigid assembly (Figure 3.5a). This configuration can simplify calibration, but more importantly it results in a working volume with a more useful shape and more uniform sample spacing than the intersection of two pyramids typical of structured-light systems (Figure 3.5b–c). There is no reason to restrict the motion to be purely translational, and configurations based on rotation of either the laser-camera scanhead or of the object itself are common. Finally, the optics of the camera in a laser-stripe system may be optimized to ensure that the image of the laser stripe is in focus, by tilting the sensor relative to the optical system (the "Scheimpflug principle"). In addition, cylindrical lenses may be used to effect the desired trade-off between depth of working volume and depth resolution. Manufacturers of high-quality laser scanners, such as Cyberware Corp., typically use all of these principles.

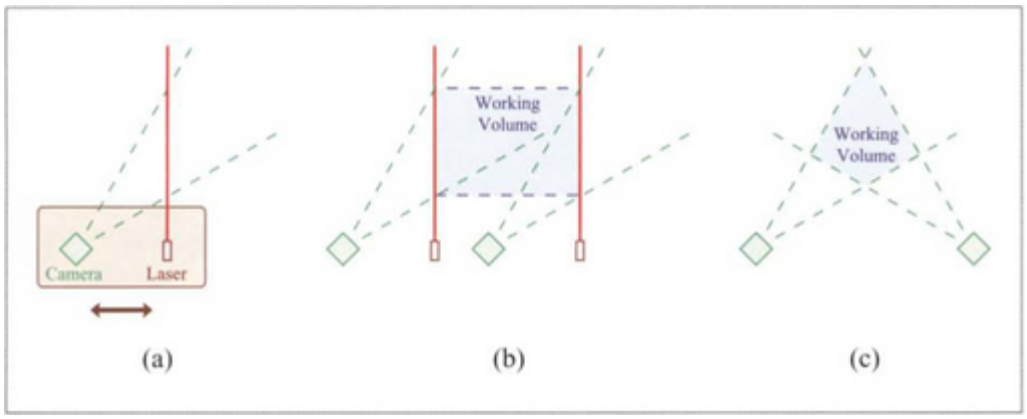

Figure 3.5 (*a*) Laser-stripe scanners are frequently designed with the laser and camera mounted rigidly on a translating assembly. This results in a working volume with a more useful shape and even sampling (*b*), as compared to the working volume of a structured-light scanner (*c*).

The processing to extract depth from a light-stripe scanner may broadly proceed along either spatial or temporal lines. The "spatial" approach to processing treats each frame in time separately, and finds the peak of the light stripe along each scan-line. The peakfinding can be based simply on maximum intensity, but it is more stable to use the mean of a Gaussian fit to the intensity profile along a scanline. Temporal processing, in contrast, treats each camera pixel independently, and finds the time instant at which it achieves the highest intensity. Temporal processing in general tends to result in fewer artifacts than spatial, but as shown by Curless and Levoy the optimal processing is in fact spatiotemporal, in which peaks are found along paths angled through space-time [CL95]. Despite this, many laser triangulation systems today use spatial processing because it is the simplest and least computationally expensive (data are naturally gathered and organized spatially, so temporal processing requires transposing the data and spacetime processing requires a more complex rotation).

Characteristics

The data produced by light-stripe scanners are among the highest in quality, and the use of laser light results in little missing data (other than due to occlusion of the camera or laser) and few outliers. Datasets are dense, and are most naturally organized as range images.

Multiview Triangulation and Structure from Motion

Any of the above techniques can be combined with additional calibrated cameras, leading to multiview techniques. Multiview stereo (both active and passive) has been explored in greatest detail, due to the ability of the additional cameras to reduce the ambiguity inherent in stereo matching [OK93]. This relies on the observation that an incorrect match from a pair of cameras, leading to an incorrect 3D position, will not in general appear to match when projected to a third viewpoint. Incorrect matches may appear consistent between pairs (or small subsets) of cameras, but only the true matches are likely to be consistent between all cameras. Passive multiview systems therefore suffer from significantly fewer false correspondences between regions that appear similar,

though of course the introduction of additional cameras does nothing to improve the situation in textureless regions. Single-pattern *active* multiview stereo systems are fairly popular, and can return relatively dense 3D estimates with high reliability.

In one sense, structure from motion (SfM) represents a limiting case of multiview stereo: there are many "cameras" placed densely together. In practice, the many views are acquired with a video camera that is moved throughout a scene. Features are detected in the video stream and are tracked over many frames, thus establishing correspondence. The major difference, however, between multiview stereo and SfM techniques is the portion of the algorithm that is considered to be difficult. For stereo, the difficulty is in finding correspondences; for SfM, the feature-tracking problem is relatively easy, and the challenge is performing the 3D reconstruction.

The interesting feature of most SfM techniques is that they do not require the camera to be calibrated (some methods require intrinsic calibration, however): they solve for the camera parameters simultaneously with solving for the 3D positions of the tracked features. The reason this is possible becomes clear if we consider the number of knowns and unknowns. Assume that we have p feature points that have been tracked for n frames. At each frame, we know the 2D position of each point in the image, leading to $2np$ knowns. In contrast, the unknowns for which we must solve are the 3D positions of the points, as well as the camera parameters (which, considering only the extrinsics, are the camera position and orientation at each frame). Thus, there are only $3p + 6n$ unknowns, which given enough points and enough frames can be fewer than the number of knowns. Of course, there are certain global unknowns that can never be solved for, such as the absolute scale of the scene. Methods for solving the SfM problem include a popular linear method [TK92] that assumes weak perspective (i.e., assumes that the scale of the scene is small relative to the distance to the camera), as well as full nonlinear bundle adjustment algorithms [PGV+04].

Characteristics

Both structure from motion and multiview stereo systems return sparse, though accurate, depth estimates. In addition, SfM, in contrast with most of the techniques considered here, can return points from all around an object, rather than just from one direction. Unorganized point clouds are therefore the appropriate representation.

3.1.3 OTHER 3D SCANNING TECHNOLOGIES

There exist several methods for 3D shape acquisition that are based on technologies fundamentally different from triangulation. While they are sometimes less accurate than triangulation scanners, they are also more appropriate in certain regimes such as very large or very small scenes. Here we consider some of the most popular alternative scanning technologies.

Pulsed and Modulated Time of Flight

Although SONAR and RADAR are both relatively well-known methods for obtaining 3D distances at large scales, there is a smaller-scale cousin, LIDAR (light detection and ranging), that uses the round-trip time of light between the scanner and the scene to estimate depth. One way to use the LIDAR principle involves firing a short (often on the order of picoseconds) pulse of light, and timing

how long it takes to return. A rotating mirror is used to change the direction of the pulses, and so a depth map is built up one point at a time. Pulsed time-of-flight scanners are usually expensive, require long scanning times, and have poor accuracy in absolute terms (many millimeters or larger, with slow progress on the technology to reduce this—after all, this requires picosecond-level timing). Nevertheless, they have some significant strengths. First, these scanners have large working volumes, often measured in tens or hundreds of meters. Thus, their *relative* depth accuracy can be quite significant. In addition, they require only a single line of sight to the surface point being scanned, since the light pulse returns along the same path along which it was projected. Therefore, these systems can both acquire geometry that may be impossible to scan using triangulation systems, and are significantly more practical for large scenes since they do not require a long, calibrated baseline. For this reason, pulsed time of flight is the dominant scanning technology used for large objects such as building interiors or exteriors.

The other main technology that makes use of time of flight relies on modulation. A continuous beam of laser light is projected, with its intensity modulated at a high frequency. The reflected light is then modulated at the same frequency, allowing the measurement of a signal that depends on the phase difference between the outgoing and reflected beams. Since this phase difference depends on the distance between the scanner and the scene, it may be converted to a depth measurement. Different variants of this scheme may measure either a single point or a complete 2D array of range pixels at a time. The choice of modulation frequency is important: too low a frequency will give poor depth accuracy, while too high a frequency will allow for only a shallow range of depths before the depths "wrap around" (because of the 2π ambiguity in phase). To improve on this, a series of sweeps using different modulation frequencies can be used to disambiguate the depths while retaining high accuracy.

In practice, most of these systems are optimized for working volumes of a few meters (i.e., room-sized) and are a natural choice for measuring people, furniture, etc.

Shape from Silhouettes and Other Space-carving Methods

The idea behind most space-carving methods is to maintain an estimate of which portions of space are occupied and which are not. This estimate is initialized to mark the whole working volume as occupied, and as information about empty space is acquired it is used to "carve off" regions. The effect is not unlike sculpting, in that material may only be removed, not added, to the shape estimate.

While some 3D scan-merging methods, such as VRIP [CL96], make use of space carving in addition to traditional depth estimates, there have been some 3D acquisition systems that rely purely on space carving. The most popular rely on the ability to distinguish the object's exterior silhouette (i.e., the boundary between the object and the background) in each of a set of calibrated views [MTSA97, MBR+00]. Space carving is performed based purely on this information, resulting in a final model that is a superset of the original object. The intermediate model representation used during the carving may be a binary voxel (occupancy) grid or a polyhedron, or for image-based applications, the 3D model might not be reconstructed directly (with the method relying on mapping candidate points into the original images and testing them against the silhouettes). In all cases, there is a

restriction on the types of objects whose shape may be recovered: there are some features such as spherical concavities that silhouette carving cannot recover. In general, with a sufficient number of views, the reconstruction converges to the *line hull* of the original shape. The strength of shape from silhouettes, therefore, is not in precise reconstruction but rather in obtaining a rough 3D model with little effort.

A related technique, sitting somewhere between space carving and multiview triangulation, is *voxel coloring* [SD99]. In this system, the shape being recovered is represented as a voxel grid and is carved from the outside in. Each voxel on the currently outermost layer is considered in turn, and is projected into all of the available views. A consistency metric is evaluated across all the views, testing whether the colors visible from the different cameras are the same. If the colors are inconsistent, the voxel is carved away. This algorithm results in reconstructions that have the character of multiview stereo in regions with significant texture, but that look more like space-carving results in regions of constant color.

One fact that should be noted about all carving methods is that they, by definition, produce a watertight surface as output. This property is important for many applications including rendering, simulation, and 3D hardcopy output. In contrast, most other 3D acquisition methods do not produce watertight surfaces automatically and require special support in the scan merging or holefilling stages of the pipeline.

Shape from Shading and Photometric Stereo

The brightness of a surface depends on its orientation relative to light sources, and there exist techniques that use this to estimate surface normals. Although this is different from obtaining 3D positions directly, the normals may be "integrated" to obtain shape, possibly with some additional stability provided by sparse or rough depth estimates obtained using a different method. Although passive shape from shading is of some interest, especially since it is clear that the human visual system uses such shading cues, most practical systems are active methods that rely on multiple calibrated, controlled light sources. These are known generally as *photometric stereo* [Woo80, RTG97]. Note that, despite the use of the word stereo, this is a single-view method and it is the lights of which there are two or more.

The most direct photometric stereo methods assume Lambertian reflectance and three light sources of known direction and radiance. The object is imaged under each light source in turn, and at each pixel we know that

$$I_i = L_i \rho(\mathbf{n} \cdot \mathbf{l}_i), \tag{3.4}$$

where I_i is the observed pixel intensity, L_i is the radiance of the light source, and $\mathbf{l}_i$ is the direction to the i-th light source. The unknowns are ρ, the diffuse albedo of the surface, and the surface normal **n.** Given three such measurements, we can write

$$\begin{pmatrix} l_{1,x} & l_{1,y} & l_{1,z} \\ l_{2,x} & l_{2,y} & l_{2,z} \\ l_{3,x} & l_{3,y} & l_{3,z} \end{pmatrix} \begin{pmatrix} \rho n_x \\ \rho n_y \\ \rho n_z \end{pmatrix} = \begin{pmatrix} I_1 / L_1 \\ I_2 / L_2 \\ I_3 / L_3 \end{pmatrix} \tag{3.5}$$

and solve for the vector ρ**n.** Since we know that the normal is unit length, we obtain both it and the albedo (in practice, the albedos are determined up to a constant, and finding them in absolute terms requires calibrating the gain of the camera, vignetting in the lens system, etc.). If there are more than three light sources, the resulting overconstrained system may be solved using least squares, or the most reliable three lights may be used, eliminating shadowed regions and specular highlights [RTG97]. In order to avoid singularities in the solution, the three lights plus the camera must be noncoplanar.

As we have mentioned, photometric stereo is very well suited for use in combination with other techniques that return noisy, inaccurate, or sparse depth estimates. When used in combination with scanners returning dense depths, such as those based on structured light, the normals may be associated one to one with the 3D points, for use in rendering or accurate surface reconstruction. The noise present in these normals is generally lower than in the surfaces obtained from triangulation, though photometric stereo is susceptible to systematic low-frequency bias resulting from non-Lambertian surfaces or imperfect estimation of light source radiance and position.

Nehab et al. have demonstrated that by combining the high-frequency information obtained using photometric stereo with the low-frequency information from triangulation, the benefits of both types of scanners may be obtained while avoiding their drawbacks [NRDR05].

A somewhat related technique to shape from shading is shape from texture, which relies on either the variation in density of texture elements or the foreshortening of individual elements to estimate surface orientation. Another similar technique is shape from specularity, which uses the presence of specular highlights to obtain (usually sparse) orientation measurements.

Shape from Focus and Defocus

It is possible to construct systems that determine depth based on the focus distance at which a lens must be set to observe objects with greatest clarity ("shape from focus") or based on *how* out-of-focus objects appear at a fixed lens setting ("shape from defocus"). Real-time systems have been demonstrated using active depth from defocus, which relies on a projected pattern of dots [NN94].

3.1.4 SCAN ALIGNMENT

Since most types of scanners return data from only one part of the object at a time, it is necessary to obtain multiple scans and align them together. The alignment stage is not necessary, of course, if the motion of the scanner and object relative to each other are calibrated, but in many instances this is not possible. However, provided the geometry or color of the object is sufficiently distinctive, and provided there is sufficient overlap in the scanned regions, the data themselves may be used to align

the different scans to each other. Alignment based on color is conceptually similar to image feature matching, and many of the same techniques that are used for stereo correspondence are applicable. Here we focus on alignment techniques that use geometric information, splitting them into methods that obtain an initial rough estimate for the alignment, methods that refine the alignment between a pair of scans, and methods that attempt to balance the pairwise registration errors across all pairs of overlapping scans.

Initial Alignment

The simplest method for obtaining an initial alignment, of course, is to ask the user. A typical user interface displays a pair of scans, then either provides the capability for directly moving the scans, using, for example, a virtual trackball interface, or allows the user to click on at least three pairs of corresponding points and solves for the optimal transformation. Despite the fact that this manual method does not sound particularly elegant or exciting, it should not be overlooked: automated methods for computing initial alignment can be slow, not robust, and tricky to implement correctly. Even for a large scanning session, manual initial alignment can be the fastest and most reliable technique.

In situations in which automated initial alignment is desirable, the general technique is as follows:

1. **Find distinctive feature points** on each scan.
2. **Compute local *shape descriptors*** characterizing the surface geometry around the features [Kaz04].
3. **Propose correspondences** between features by matching similar descriptors.
4. **Compute the rigid-body transform** that best aligns corresponding features.
5. **Verify** that the proposed alignment is reasonable, by checking the scan-to-scan distance at a large number of points. If the alignment does not seem plausible, repeat using a different set of proposed correspondences.

There are many possible variations on this broad outline, and to a large extent greater sophistication in one stage may compensate for simplicity in another. For example, if the feature points are particularly distinctive and are consistently found in similar locations on different scans, there may not be a need for a shape descriptor: all triplets of feature correspondences may simply be tried [CH99]. Conversely, if a sophisticated shape descriptor is used it may be sufficient to compute that descriptor at random locations on the surface [JH99, HH03]. Finally, in many situations it is not unreasonable to simply use a brute-force algorithm, testing a sparse sampling of possible rotations and translations.

Iterative Pairwise Registration Using ICP

The ICP (originally iterative closest point, though iterative corresponding point is perhaps a better expansion for the abbreviation) algorithm has become the dominant method for pairwise alignment of 3D models based purely on the geometry, and sometimes color, of the meshes. ICP starts with two meshes and an initial guess for their relative rigid-body transform, and iteratively refines the

transform by repeatedly generating pairs of corresponding points on the meshes and finding the transformation that minimizes an error metric, such as the sum of squared point-to-point distances. The original ICP algorithm computed the "correspondences" as being, for each point on one mesh, the *closest* point on the other mesh (Figure 3.6).

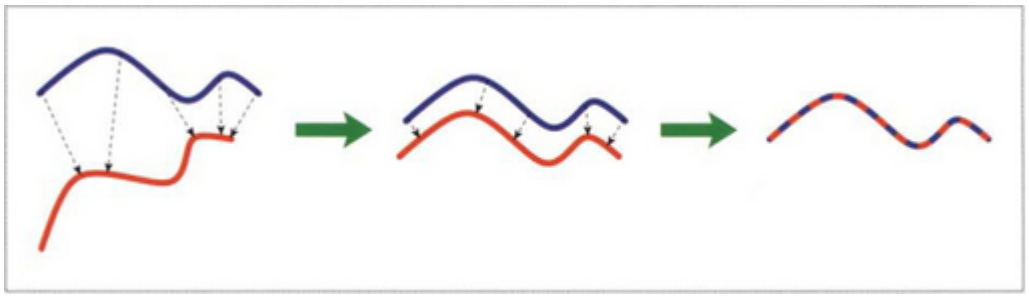

Figure 3.6 In the ICP algorithm, points on one scan (blue) are selected and matched to the *closest* points on the other scan (red). One of the scans is moved such as to minimize the distances between pairs of points and the process is iterated until convergence.

Since the introduction of ICP by Besl and McKay [BM92], many variants have been introduced on the basic ICP concept. We may classify these variants as affecting one of six stages of the algorithm:

1. **Selection** of some set of points in one or both meshes.
2. **Matching** these points to samples in the other mesh.
3. **Weighting** the corresponding pairs appropriately.
4. **Rejecting** certain pairs based on looking at each pair individually or considering the entire set of pairs.
5. Assigning an **error metric** to the current alignment, based on the point pairs.
6. **Minimizing** the error metric.

A survey of many of these variants is available [RL01a], so here we will restrict the description to variants generally regarded as current "best practice."

Point Selection

A reasonable method for selecting points at each iteration is simply random sampling of points on *both* scans, with a few hundred to a few thousand points usually sufficient for good stability. However, in difficult cases in which some components of the transform are restricted by features on a relatively small portion of the mesh, greater care must be taken to sample points that do a good job of constraining the transformation. A simple option is to estimate normals on the scan, then sample points having as uniform a distribution of normals as possible. A more theoretically well-grounded method, proposed by Gelfand et al., directly considers the error metric being minimized, performs an eigenanalysis on the linear system being solved, and uses the eigenvalues of the system to determine which eigenvectors are most underconstrained. Sampling according to these criteria ensures that all components of the transformation are well constrained [GIRL03].

Matching

Although the distinguishing feature of ICP-like algorithms is that they use the closest point (given the current best estimate of the transformation) as the approximation for the corresponding point at each iteration, greater stability and faster convergence may be achieved by instead matching to the *closest compatible* point, using some compatibility criterion. Criteria based on color, normals, curvature, etc. have been proposed, and all improve performance in certain cases [GRB94, Pul99, SLW02]. A simple condition, such as that normals should be within 45° of each other, can improve convergence at minimal cost. In all cases, a spatial data structure such as a K-d-tree may be used to accelerate the closest-point computations.

Weighting

Although many weighting schemes based on uncertainty, compatibility, point-to-point distance, and other criteria have been proposed, in most cases the effect of weighting on convergence rate tends to be small and highly data dependent.

Rejection

Since ICP is fundamentally a least squares approach, it is necessary to perform outlier rejection to avoid contamination of the results by incorrect matches. A good, all-purpose approach is to reject pairs whose point-to-point distance is larger than some small multiple of the median distance. In cases in which ICP is being performed on meshes or range images, such that it is possible to determine which points are on mesh boundaries, we additionally recommend rejecting all pairs containing such edge points (Figure 3.7). This strategy avoids erroneous pairings (that cause a systematic bias in the estimated transform) in cases when the overlap between scans is not complete [TL94].

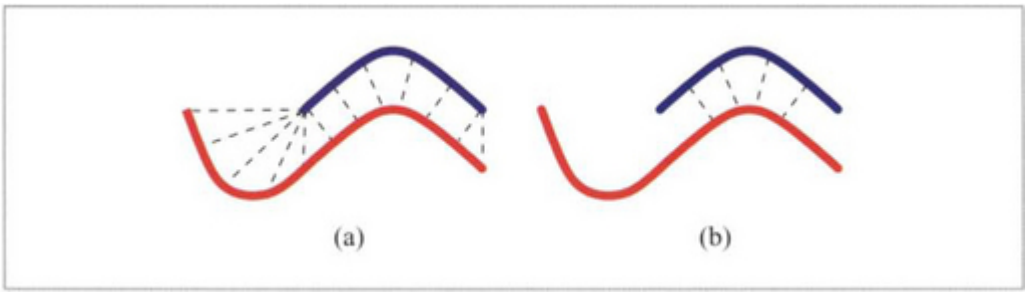

Figure 3.7 (***a***) When two meshes to be aligned do not overlap completely (as is the case for most real-world data), allowing correspondences involving points on mesh boundaries can introduce a systematic bias into the alignment. (***b***) Disallowing such pairs eliminates many of these incorrect correspondences.

Error Metric

The original ICP algorithm minimized the sum of squared point-to-point distances among corresponding pairs of points, but a variant by Chen and Medioni proposed using point-to-plane distances instead [CM92] (Figure 3.8). That is, the algorithm minimizes the distance from one point

to the plane containing the other point and perpendicular to its normal. This has the effect of making it easy to move the scans "along" the surface, leading to dramatically (order of magnitude) faster convergence in many cases.

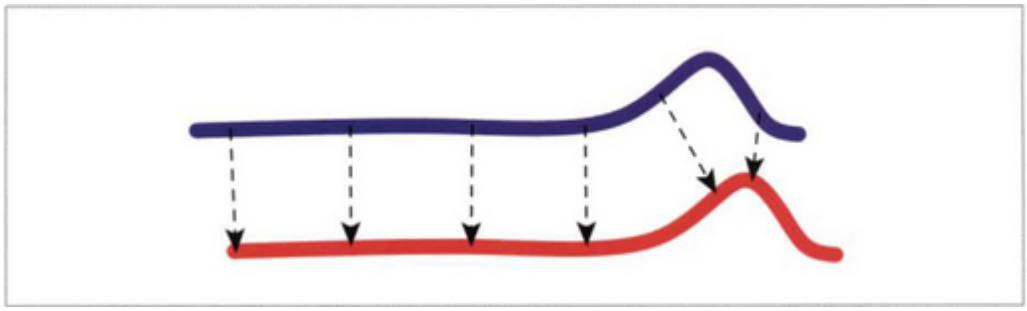

Figure 3.8 The point-to-plane metric for ICP captures the notion that sliding two planes along each other does not affect the distance between them. In this example, the two scans are mostly planar, with only the small bump at right constraining the left-to-right component of the translation. Using the original point-to-point minimization, however, the point pairs in the flat region will prevent the scans from "sliding along each other" to reach the correct transformation (it is useful to think of them as small springs that are attempting to contract). In this situation, using the point-to-plane metric will lead to significantly faster convergence.

Mitra et al. have also proposed precomputing a piecewise-quadric approximation for the function representing the distance to one of the scans and evaluating the function at locations on the other scan [MGPG04]. A special data structure is used to store this approximation, and the parameters of the quadric are used directly during a Newton's method-style minimization. This provides the stability of point-to-point minimization far away from the correct minimum while retaining the high convergence rate of point-to-plane minimization once the transformation is close to correct.

Minimization

While there exist closed-form solutions for the rotation and translation that minimize point-to-point distance [ELF97], there are no analogous methods for the recommended point-to-plane minimization. Instead, the recommended method is as follows.

Assume we have a collection of points ($\mathbf{p}_i$, $\mathbf{q}_i$) with normals $\mathbf{n}_i$. We want to determine the optimal rotation and translation to be applied to the first collection of points (i.e., the $\mathbf{p}_i$) to bring them into alignment with the second (i.e., the $\mathbf{q}_i$). Thus, we want to minimize the alignment error

$$\varepsilon = \sum_i \left[\left(\mathrm{R}\mathbf{p}_i + \mathrm{t} - \mathbf{q}_i \right) \cdot \mathbf{n}_i \right]^2 \tag{3.6}$$

with respect to the rotation **R** and translation **t.**

The rotation is a nonlinear function, incorporating sines and cosines of the rotation angles. If, however, we assume that incremental rotations will be small, it is possible to linearize the rotations, approximating cos θ by 1 and sin θ by θ. For example, in the case of rotation in x,

$$\mathrm{R}_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos r_x & -\sin r_x \\ 0 & \sin r_x & \cos r_x \end{pmatrix} \approx \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & -r_x \\ 0 & r_x & 1 \end{pmatrix}. \tag{3.7}$$

Thus, the full rotation may be approximated as

$$\mathrm{R} \approx \begin{pmatrix} 1 & -r_z & r_y \\ \mathrm{r}_z & 1 & -r_x \\ -r_{y+} & r_x & 1 \end{pmatrix}, \tag{3.8}$$

for rotations r_x, r_y, and r_z around the x, y, and z axes, respectively.

Defining

$$\mathrm{c} = \mathrm{p} \times \mathrm{n}, \quad \mathrm{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \tag{3.9}$$

and substituting Equation (3.8) into (3.6), we obtain

$$\varepsilon = \sum_i \left[(\mathrm{p}_i - \mathrm{q}_i) \cdot \mathrm{n}_i + \mathrm{t} \cdot \mathrm{n}_i + \mathrm{r} \cdot \mathrm{c}_i \right] 2 . \tag{3.10}$$

We now minimize $\boldsymbol{\varepsilon}$ with respect to r_x, r_y, r_z, t_x, t_y, and t_z, leading to the following linear system:

$$\left[\sum_i \begin{pmatrix} \mathrm{c}_i \\ \mathrm{n}_i \end{pmatrix} (\mathrm{c}_i \ \mathrm{n}_i) \right] \begin{pmatrix} \mathrm{r} \\ \mathrm{t} \end{pmatrix} = -\sum_i \left((\mathrm{p}_i - \mathrm{q}_i) \cdot \mathrm{n}_i \right) \begin{pmatrix} \mathrm{c}_i \\ \mathrm{n}_i \end{pmatrix}. \tag{3.11}$$

This is a linear matrix equation of the form **Cx** = **b**, where **C** is the 6 × 6 "covariance matrix" accumulated from the $\mathbf{c}_i$ and $\mathbf{n}_i$, **x** is a 6 × 1 vector of unknowns, and **b** is a 6 × 1 vector that also depends on the data points. The equation may be solved using standard methods (**C** is symmetric, so Cholesky decomposition is the preferred algorithm), yielding the optimal incremental rotation and translation.

As mentioned above, this 6 × 6 covariance matrix plays a critical role in the stability sampling, since it encodes the increase in the alignment error when the transformation is moved away from its optimum:

$$d\varepsilon = (dr \ d\mathrm{t})(C) \begin{pmatrix} dr \\ d\mathrm{t} \end{pmatrix}. \tag{3.12}$$

The larger this increase the greater the stability of ICP, since the error landscape will have a deep, well-defined minimum. On the other hand, if there are incremental transformations that cause only a small increase in alignment error, ICP will be relatively unstable with respect to these degrees of freedom.

By expanding **C** in terms of its eigenvectors we may see directly the effect of various incremental transformations. If all eigenvalues of **C** are large, any transformation away from the minimum will result in a large increase in alignment error. If, on the other hand, one or more eigenvalues are small, the corresponding eigenvectors are transformations that do not increase error much, and therefore represent directions in transformation space along which the error landscape is shallow.

Global Registration

Although it is sometimes possible to build complete 3D models just by aligning each scan to one other, this can be unstable. Consider the situation in which there is a chain of scans stretching all the way around an object. Simply aligning each scan to the one immediately before it can lead to an accumulation of any pairwise alignment errors that might have been caused by noise, miscalibration, or some other cause. As a result, the final scan can end up quite a distance away from the first. As a result, some method of "spreading out" the alignment error is necessary, so it is not concentrated in just one pair of scans. This is the goal of so-called "global registration" algorithms.

One global registration approach that is sometimes used is to take a special scan that covers a large part of the surface. This might be a "cylindrical scan" that goes all around the object, and some commercial laser-triangulation scanners have the capability of taking such scans in addition to the more-common linear scans. If such a scan is available, it certainly makes sense to take advantage of it by aligning all the other scans to it, rather than to each other [TL94]. In many cases, however, it is impossible or impractical to obtain such a scan.

The simplest "true" global registration technique is the brute-force solution: bring all scans into the ICP iteration loop. That is, at each ICP iteration we compute correspondences from points on one scan to the closest point on *all* overlapping scans, then find the transformation for that scan that minimizes all the pairwise transformations [BSGL96]. While this approach works, it requires holding all the scans in memory at once and can be computationally impractical.

A simple way of speeding up the previous idea is to precompute pairwise alignments between all overlapping scans. The final global registration step then just has to solve for a set of transformations that are as consistent as possible with all the pairwise ICPs. The advantage of this is that the global registration step does not need all the scans themselves in memory, just some data computed during the pairwise registrations. This makes it much more practical for models with many scans.

Pulli's algorithm is a combination of the two previous ideas: the result of the pairwise ICPs is a small set of precomputed point correspondences between the scans, never to be computed again, and these are the only thing that is used during the global registration. The final global step itself involves repeatedly considering each scan and finding the transformation that minimizes its precomputed point correspondences with respect to all other scans [Pul99]. Faster convergence is achieved by considering scans in decreasing order of the number of other scans that they overlap. Sharp has

proposed a different variant on the same theme, which focuses specifically on "spreading out" the mutual inconsistency between pairwise alignments [SLW05]. Specifically, the algorithm looks for cycles within the graph of scan overlaps. For each cycle, the net alignment error is spread out equally among all pairs of scans within that cycle. Scans belonging to multiple cycles receive the average transformation.

Finally, Krishnan et al. have proposed an SVD-based method that can solve for the optimal transformations directly, given precomputed pairwise transformations for a set of scans [KLMV05]. The method is an extension of one of the closed-form solutions for pairwise point-to-point alignment, and is exact in the absence of noise. Otherwise, the SVD provides an initial estimate, and the global registration proceeds via a Newton iteration.

3.1.5 CONCLUSION

Scanning hardware, based on technologies such as triangulation and time of flight, has emerged as an important source of detailed 3D models and has partially driven the development of point-based methods. Depending on the method, data are produced as either unorganized point clouds or range images and are processed by further stages in a 3D model acquisition pipeline, including registration (described in this section) and merging (described later in this book). While the scanning technologies are mature and capable of producing high-quality models for most objects, research continues to improve the speed and accuracy of scanning hardware, the ability to scan specular and translucent objects, and the ease and automation of creating complete 3D models. The following section describes a case study based on the principles surveyed in this chapter: a complete system for acquiring 3D models with color using a structured-light scanner.

3.2

A PRACTICAL LOW-COST SCANNER FOR GEOMETRY AND APPEARANCE

Filip Sadlo, Tim Weyrich and Ronald Peikert

3.2.1 OVERVIEW

The previous section gave an overview of state-of-the-art acquisition techniques. This section describes in detail an implementation of a temporal structured-light system based on binary Gray codes. It allows robust and accurate acquisition of objects with arbitrary geometry and a wide range of materials. A benefit of this method is that geometry and texture can be acquired with the same camera, resulting in texture that is consistent with the geometry. As stated in Section 3.1.2, another benefit is the low cost compared to other systems, since video projectors are often available and because only a single camera is needed. During the whole acquisition stage, the data are represented and processed in point-based format, resulting in an unorganized point-cloud model.

The object is rotated in small known steps by a turntable and for each position the view is reconstructed using structured light. Unlike many approaches, the demanding and error-prone task of mutually registering the single reconstructions is avoided. Instead, precise calibration of the projector, the camera, and the axis of the turntable is assured. This allows them to produce consistent multiview reconstructions (rings of overlapping views). Then some methods for the removal of artifacts are applied. After that, an efficient method for merging the overlapping reconstructions into a single-layered surfel representation is applied. For the reconstruction of texture, photometric calibration is added to the already computed geometric calibration of the projector. This way, a calibrated light source is obtained that is used for the per-surfel reconstruction of either Lambertian texture or texture according to the Phong reflectance model. At the end of this section a method for analyzing the geometric accuracy of the system is described.

3.2.2 SYSTEM OVERVIEW

The acquisition system (Figure 3.9) consists of custom hardware, namely a video projector that projects the structured-light patterns, a turntable that rotates the object, a camera that takes images of the projections, and a computer that controls the hardware and does the reconstruction.

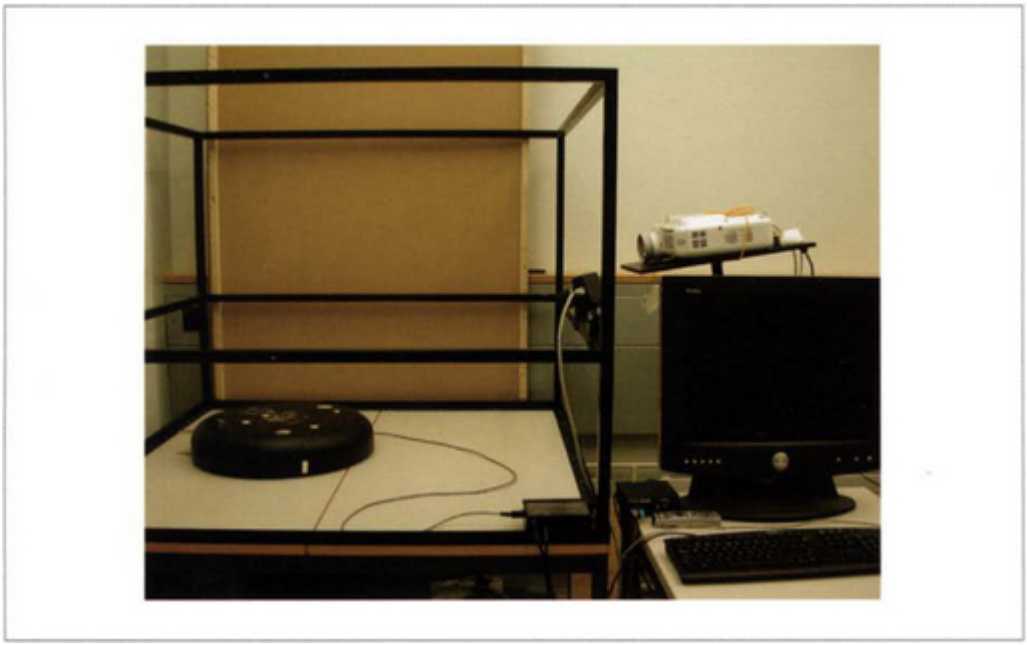

Figure 3.9 System overview: turntable, camera, projector, and control computer (from ***left to right***).

In our setup, the system contains an analog-input 1, 024 × 768 DLP video projector. It is beneficial if the projector allows for disabling features such as automatic synchronization and image size adaption. This way the calibration is not lost when the projector is turned off between scans. Another aspect is the projector's minimal image diagonal, or in other words, the minimal focal distance. This becomes important if small objects have to be acquired. Currently, an IEEE-1394 video camera with a resolution of uncompressed 1, 024 × 768 pixels is used to acquire the images. It is preferred to use monochrome cameras and to illuminate the object (or calibration pattern) with red, green, and blue projector light in order to acquire colors. The reason is that cameras using a Bayer pattern for color acquisition (each pixel has a red, green, or blue filter attached) exhibit increased blur and artifacts due to undersampling and interpolation. This complicates calibration, smoothes the reconstruction, and can produce artifacts comparable to those in Figure 3.14 shown later. For similar reasons, it is also avoided to rectify the images and instead the lens distortion is modeled.

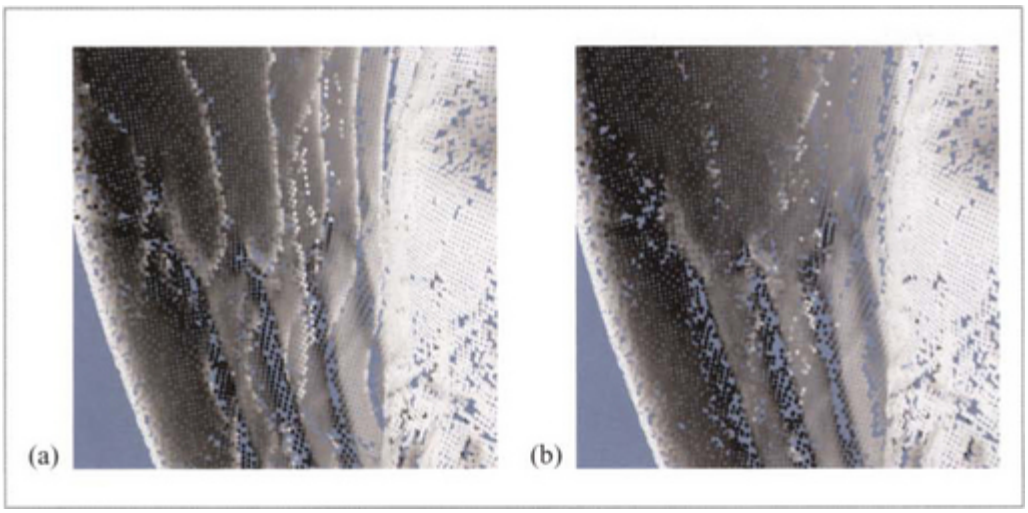

Figure 3.14 Removal of shadows (seam artifacts). (***a***) No removal. (***b***) Local range = 1 and minimal distance = 2 mm (18,452 surfels removed).

Often several rings of reconstructions are needed in order to get a complete reconstruction. This is accomplished by either tilting the object, or by moving projector and camera to a new position between the scans. The optimal solution would be to perform these transformations again around a calibrated axis.

But since this tends to be mechanically demanding, the object is tilted by hand and each time a ring of reconstructions is acquired by rotation of the turntable. Therefore, the resulting rings have to be mutually registered, treating each ring as a separate rigid model. Some methods for mutual registration of reconstructions are geometry based, such as iterated closest points (ICP) described in Section 3.1.4, and others are image based, such as that by Bernardini et al. [BMR01]. The current system uses ICP for registration. Some challenges of registration, such as error accumulation, play a minor role in this case because the rings are already consistent at high accuracy, leading to a simpler registration problem. After the rings have been merged to a single-layered surfel representation, reflectance samples for each surfel are collected into *lumitexels* [LGK+01] and used for texture reconstruction according to the Phong reflectance model, which is then rendered and edited using Pointshop3D [ZPKG02] as described in Section 5.2.

3.2.3 CALIBRATION

Although often underestimated, precise calibration of the structured-light system is one of the main prerequisites for a successful and accurate reconstruction. This is especially true for multiview reconstructions, where a given surface point is reconstructed several times and the corresponding reconstructions have to match. In order to increase accuracy and to ease calibration, separate intrinsic calibration of camera and projector is performed and then an extrinsic calibration step is applied.

For intrinsic and extrinsic calibration of the camera and projector, the Zhang calibration method [Zha00] is used as implemented by OpenCV's cvCalibrateCamera. This calibration model consists of focal length with respect to pixel widths and heights, the principal point, and a radial and a tangential lens distortion modeled by two parameters each. As proposed by OpenCV, a chessboard pattern is used for calibration. However, the pattern detection algorithm of OpenCV fails under difficult lighting or oblique viewing conditions. This conflicts with the experience that strong perspective views at oblique angles increase the accuracy of intrinsic and especially extrinsic calibration. It also conflicts with the need to detect a printed pattern and a (distorted) chessboard projection in the same view (for intrinsic projector calibration). Therefore, an alternative detection procedure based on projective geometry is applied.

Calibration Patterns

OpenCV's detection of the calibration pattern inside the images is done in two steps. First, approximative guesses of the chessboard corners are computed. Then, the corners are detected at subpixel accuracy from the guesses. The second step works well, but the first step fails under difficult light or viewing conditions.

To ease the pattern detection, identical color marks are attached onto the four corners of the chessboard pattern and an additional one for defining the origin. It has been chosen to use colors instead of spatial marks because of the invariance of color with respect to projective transforms. A detailed description of the detection procedure and the subsequent calibration steps is given in Sadlo et al. [SWPG05].

Intrinsic Camera Calibration

The camera is calibrated using different views of a printed chessboard pattern as described above. The pattern is illuminated by the projector in order to create similar lighting conditions as in the case of projector calibration. This allows the use of identical camera settings and HSV segmentation ranges for detection in all calibration steps.

Intrinsic Projector Calibration

The projector is calibrated as an inverse camera. This means that instead of taking pictures of a chessboard with known geometry and detecting the corners inside the images, a chessboard pattern with known geometry is projected to different orientations and positions of a plane and the projections are measured with the calibrated camera.

Extrinsic Calibration

The orientation of projector and camera relative to each other is determined in a similar way as in the projector calibration, but with a single camera image. This time both projector and camera are calibrated extrinsically. The reference coordinate frame is also determined in this step.

Turntable Axis Calibration

Calibration of turntables is approached in different ways, such as using markers permanently attached to the turntable [SPMS04], or by fitting an axis to rotated reconstructions. A color-marked chessboard pattern is put horizontally on the turntable and a full rotation is done in a given number of steps, typically 12. For each step, the position and orientation of the camera relative to the pattern is computed by extrinsic calibration. Then a circle is fitted to the resulting ring of virtual camera positions. The rotational axis of the circle represents the axis of the turntable.

Luminous Projector Calibration

Because the object is illuminated by the projector for texture acquisition, the irradiance from the projector has to be known at a given point in space. For each new projector, the luminous intensity I of the projector pixels is to be initially calibrated. This could be done using a calibrated reflection target, for example, Spectralon.

As our application does not require absolute physical quantities, a gray cardboard is used instead and I is scaled to map color intensities into a useful range. Two calibration modes have been implemented. I is either assumed to be identical for all pixels, or I is determined on a per-pixel basis, capturing spatial intensity variations at the cost of more noise. The irradiance E at a given surface point is then computed as follows: $E = I/d^2$, with d its distance to the projector's center of projection.

3.2.4 GEOMETRY RECONSTRUCTION

Geometry is reconstructed using structured light according to the Gray code and phase-shifting method described below. Normals are computed from the weighted positions of neighboring samples by plane fitting. They are computed separately for each view before merging the reconstructions. This avoids influence of registration errors.

Gray Code and Phase Shifting

Structured-light methods make use of a projection device to determine the z-depth of every illuminated camera pixel. This is done by optical triangulation of the camera ray with the corresponding projector ray that illuminated the surface element. There are many possibilities to structure light in order to allow the identification of a projector pixel by its light. It has been chosen to do time-multiplexing of gray level codes. This allows a wide range of materials but limits the system to static objects.

In the standard Gray code algorithm (see Section 3.1.2), the ray defined by a camera pixel is intersected with the plane defined by the corresponding projector column. However, this assumes no lens distortion inside the projector because otherwise the plane would be distorted. To make the reconstruction process more robust against lens distortion and decoding errors, both projector columns and rows are encoded. The plane-ray intersection problem becomes an overdetermined ray-ray intersection problem that also allows for the removal of artifacts by the ray skew criterion as described below.

Since the projected Gray codes are binary, the achievable precision is limited to integer projector coordinates. Therefore, the detected pixel relations are subsequently improved using a variant of the phase-shifting technique (Section 3.1.2) based on a grayscale sine pattern as shown in Figure 3.10. This also reduces decoding errors in the least significant bits of the Gray code. In addition, phase-shift reconstruction even allows to determine the projector coordinates at subpixel accuracy. It has also been experimented with line-shifting [GBBF00], which achieves subpixel accuracy in camera coordinates rather than in the projector domain. However, in our setup it generally produced inferior results.

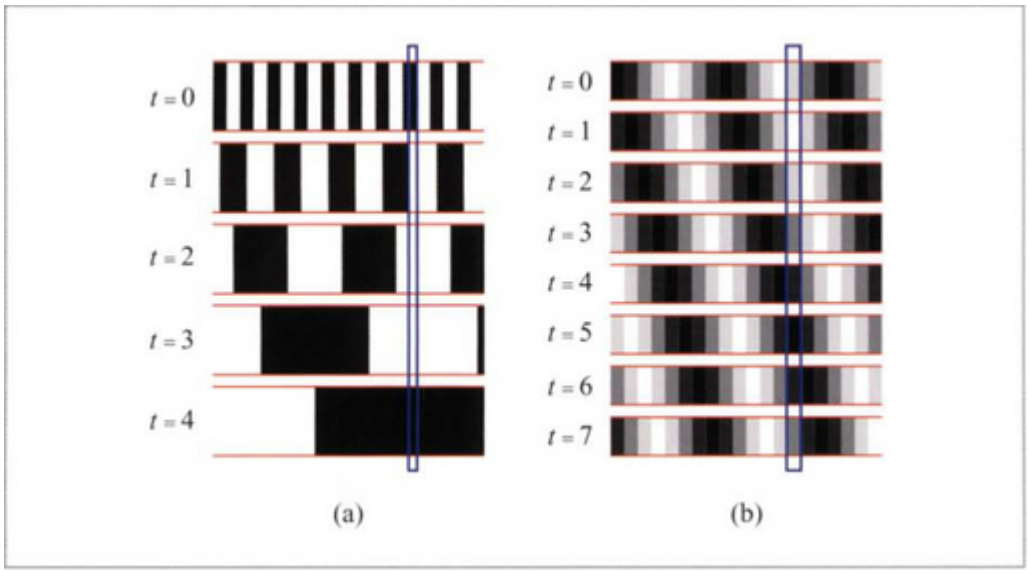

Figure 3.10 (*a*) Gray-coded structured light. The temporal signal (blue box) at a given object point represents the column of the corresponding projector pixel. The procedure is repeated with horizontal stripes to also encode the projector rows. (*b*) Phase shifting. Shifted sinusoidal stripe patterns are projected to the object, producing a temporal sine signal on a given object point. The phase of this signal represents the projector column. The procedure is repeated with horizontal stripes to also encode the projector rows.

Due to errors in calibration and decoding, the two corresponding rays usually do not intersect. The method presented by Guehring [Gue01] addresses the problem by nonlinear least squares and analysis of the residual, while Hartley and Sturm [HS94] give an overview and introduce a polynomial method. In our system the point of intersection is computed as the point on the camera ray that is closest to the projector ray. The solution is constrained to the camera ray because its calibration can be assumed more accurate than that of the projector rays. This way the projector only contributes depth information, which meets the original intention. The distance between the intersecting rays (ray skew) is used for the removal of artifacts as described below. The ray skew is also visualized by color coding the reconstructions for visual verification of the quality of calibration. Figure 3.11 shows an example.

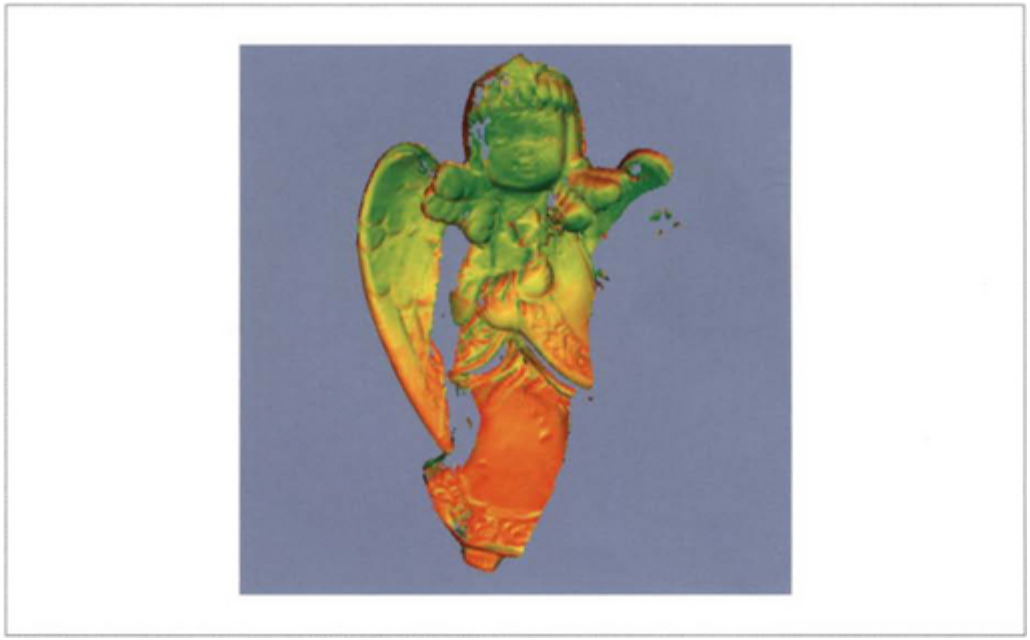

Figure 3.11 Single-ring reconstruction colorized by the ray skew for verification of calibration quality. Red means small skew; surfels with ray skew larger than 0.6 mm (outliers) have already been removed.

Artifact Removal

Here some of the implemented methods for the elimination of geometric artifacts are described. They are all applied to the single-view reconstructions before they are merged.

Signal Strength

A black reference image with all projector pixels set to black and a white reference image with all projector pixels set to white are taken for each view. They are used for elimination of camera pixels that receive no or too weak signals and they are also used for normalization of the structured-light signal. A camera pixel is eliminated if the white reference differs from the black reference by less than a user-defined threshold. The threshold is chosen in order to reject mainly background and part of the shadows.

Ray Skew

This method detects artifacts caused by decoding errors as well as artifacts that are produced by reflected or scattered codes. Assuming accurate calibration, it is unlikely that the projector ray corresponding to the falsely decoded code intersects with the camera ray as well as the correct projector ray would do. In other words, the ray skew tends to increase on decoding errors. A threshold for the minimal distance between intersecting rays is used and the reconstruction is rejected if the threshold is exceeded (Figure 3.12).

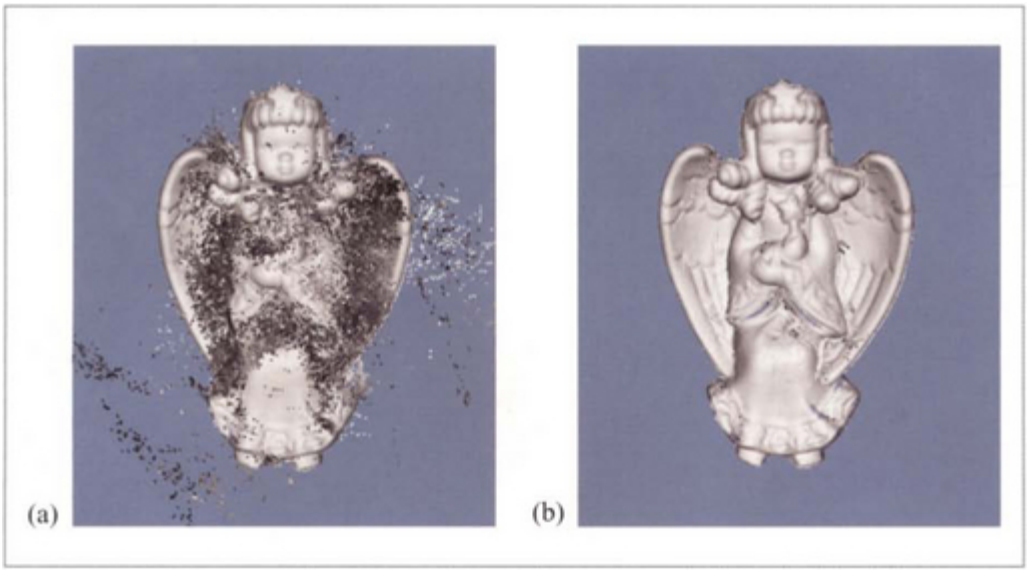

Figure 3.12 Removal of artifacts based on ray skew and component size. (***a***) Ray skew and component size are unlimited (2,527,362 surfels). (***b***) Ray skew is limited to 0.6 mm and connected component size to six surfels (2,053,543 surfels). Only the last few outliers were eliminated by component size.

Subpixel Variance

This method addresses artifacts that originate during phase shifting from object regions with varying reflectance and curved (or discontinuous) surface as described by Curless and Levoy [CL95]. The phase-shift signal is spatially integrated over the area of each camera pixel during acquisition. Assuming that the left half of the camera pixel looks, say, at black material, while the right half observes a bright material, the signal integrated over the pixel will contain only codes from the right half, leading to wrong projector rays usually visible as depth errors. As overlapping views have been acquired, it is possible to address this problem by eliminating surfels that lie on edges in image space. The pixel corresponding to the surfel is resampled at subpixel resolution using Lanczos interpolation, and its variance is computed. It is rejected if its variance exceeds a threshold. The method also removes sharp shadow boundaries.

Outliers are also removed based on other geometric criteria, such as small connected component size of surfels, and photometric criteria, such as saturated pixels.

View Merging

After the rings of reconstructions have been registered, we want to merge the overlapping reconstructions into a single-layered surfel representation in order to reduce storage and visualization cost. Doing so, care has to be taken that texture quality and geometric accuracy remain as high as possible.

Blending of the overlapping textures and averaging the overlapping surfel positions would require that the reconstructions fit together at subpixel accuracy, otherwise the resulting reconstruction would get blurred or doubled. Therefore, combining original patches of the overlapping

reconstructions, as described later in this section, is tried to preserve detail. This method relates to Turk and Levoy [TL94] and it also has a thinning effect because the patches have original resolution.

Bounded Projective Nearest Neighbors

The system does a nearest-neighbor search to get the overlapping surfels of a given surface element of the object. Instead of performing a standard nearest-neighbor search, it is more efficient to perform the search in the M camera images. Because every point is reconstructed by a corresponding camera pixel, one can make use of the calibrated setup. To find for a given point $\mathbf{p}$ and a given search radius r the nearest (or all) point reconstructions, $\mathbf{p}$ is projected to all camera images, yielding a $\mathbf{p}'_i$ for each view i. Then all point reconstructions that have been reconstructed by camera pixels within a search radius i'_i around $\mathbf{p}'_i$ in image space are tested for whether they lie within distance r to $\mathbf{p}$ in world space. The search radius i'_i is computed from r by projection.

The complexity of the search is $O(M \cdot r'^2)$, where M is the number of camera images (r'^2 is limited by the number of pixels per image). In the current application, the search radius is small and constant. Therefore r' has a small upper bound, resulting in an algorithmic complexity of $O(M)$.

Best Surfels

A simple greedy approach is used to select patches from overlapping reconstructions in order to get a single layer of points and for each object region the best patch regarding the quality of geometry and texture. The current approach is purely surfel based, hence it does not maintain a volumetric representation such as by Pulli et al. [PDH+97].

The algorithm defines and selects the patches implicitly in two steps. First, the homologous surfels representing the same point on the surface are determined using the nearest-neighbor search just presented. Then the best of these candidates is chosen for the resulting reconstruction.

The method is based on a simple idea: for each object point, take the surfel that has been reconstructed most orthogonally by its camera (regarding the surface normal computed from its neighbors). This addresses the fact that texture and geometry usually have the highest resolution when acquired by perpendicular view. At the same time, this simple criterion generates the patches. Figure 3.13 shows an example result. There are holes due to the greedy nature of the algorithm. However, the holes are small enough to disappear when the surfels are rendered.

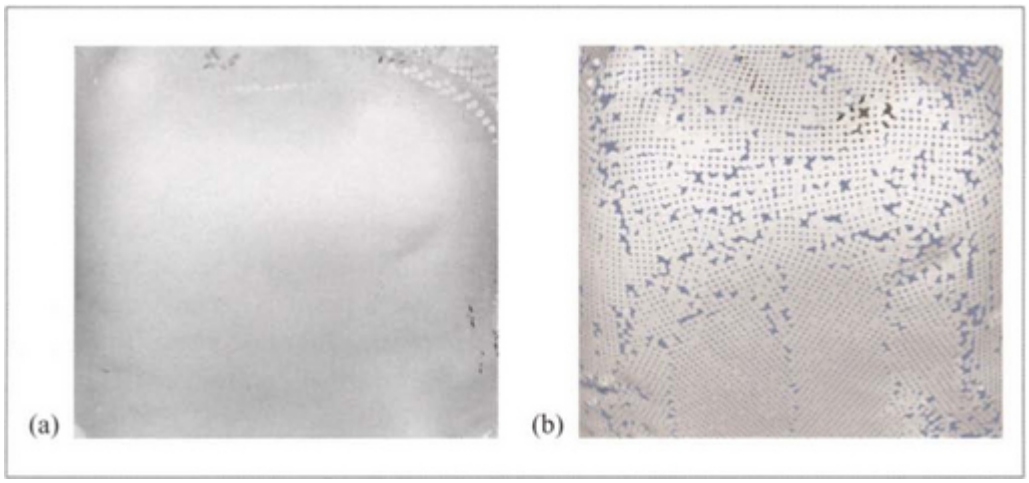

Figure 3.13 Best surfels. (***a***) Before merging (2,053,543 surfels). (***b***) After merging (226,664 surfels). Search radius in world space = 1.2 mm. Search radius in image space = 1 pixel in order to remove outliers.

A modification is applied to the described algorithm in order to remove outliers (additionally to the methods described above). This is achieved by modification of the nearest-neighbor search. Instead of computing the search range in image space r'_i using projection, it is set by the user to a smaller range. This way, the region where best surfel candidates are sampled is not spherical any more (as would result from a true 3D distance test); instead, the search range is restricted in directions perpendicular to the views. This eliminates outliers because it allows us to choose a larger search radius that captures the outliers without thinning out too many neighboring surfels. Figure 3.13 shows an example result.

3.2.5 TEXTURE RECONSTRUCTION

Most materials exhibit a certain amount of specular reflection. In the case of strong specular reflection, the effects are confined to a small region on the object for a given view and illumination by the projector. Since many overlapping reconstructions are acquired, one could remove the specularly reflecting parts in all of them before view merging. Theoretically this would lead to consistent diffuse reconstructions of specular objects. However, the specular information would be lost. Additionally, many materials have moderate specular reflection that leads to specular effects that cover large parts of the object. There it would not be possible to remove them without producing holes. Consequently, a specular reflectance model has to be fitted in these cases.

The diffuse reflectance model can however be used to generate high-resolution reconstructions of objects that do not possess too strong of a specular reflection. The surfel reconstruction preserves the subpixel information discretized by the camera if the surfels are located on the viewing rays of the camera pixels and if each surfel gets its uniform color only from the corresponding pixel. This can be achieved for the diffuse reflectance model, because it can be robustly fitted to a single reflectance sample.

Shadow Removal

The acquisition system can generally not acquire shadowed object regions, because the structured light does not reach these parts. However, the reconstruction still may yield points in shadows if they are indirectly illuminated by interreflection, subsurface scattering, and other effects. It is important to detect and remove shadows when computing texture. Furthermore, structured-light reconstruction of indirectly illuminated regions and at shadow boundaries leads to geometric artifacts (see Figure 3.14a). These artifacts include texture embossing (see Section 3.1.2) and additional depth error from indirect illumination. The texture embossing phenomena can be reduced by avoiding Bayer tiling in the camera and by the above subpixel variance method.

The shadows are detected using a depth test relative to the projector. For each position of the turntable a reconstruction with corresponding viewpoint and projector position is obtained. After view merging, the remaining surfels are projected into each virtual view of the projector. Projector pixels that get more than one surfel projection are tested for shadow. If the surfel producing the projection belongs to the same depth map as the virtual view and if it is not nearest to the camera, it is removed. In order to preserve surfels at silhouettes, the surfel is only removed if it is farther away from the nearest surfel than a user-defined small threshold.

The generated holes are filled with other surfels by repeating the view-merging procedure.

Although some shadow surfels are eliminated by the described method, there are usually shadow surfels remaining due to calibration errors, reconstruction errors, and the fact that the sampled surface elements have an extent. Therefore, instead of projecting each surfel to the projector and storing it only in the corresponding projector pixel, it is stored to a user-defined range around that projector pixel. This way the sensitivity for depth discontinuities is increased, and at the same time, surfels that have been reconstructed at grazing angles are removed.

Reflectance Sampling

In order to reconstruct the texture, it is necessary to collect for each surface point all available and reliable reflectance samples from the acquired texture views. Reflectance is computed from the image intensities using the calibrated illumination model described in Section 3.2.3. According to Lensch et al. [LGK+01], a surface point together with the corresponding samples is called a *lumitexel.* Not all views contribute a sample for a given surface point. There might be effects like occlusion and insufficient illumination or shadows that invalidate a sample in a given view. Insufficient illumination is detected using a user-defined threshold that rejects samples illuminated at grazing angles, in addition to the signal strength threshold described above.

The shadow and occlusion tests are based on the surfel representation. Only surfels that belong to the final merged reconstruction represent surface points and hence lumitexels. Therefore, the points that are removed by the merging procedure are only marked as such, since they are needed for sample selection.

Occlusion/Shadow Test

The structured-light method already performs a kind of implicit occlusion test in the sense that it can only acquire surface points that are visible by the camera. The same holds for the light source, so theoretically no shadows can be acquired. Therefore, it can be decided if a point $\mathbf{p}$ gets occluded in a given view v_i by projecting $\mathbf{p}$ to that view (onto the camera pixel $\mathbf{p}'_i$) and by examining the surfel $\mathbf{s}_{\mathbf{p}'i}$ that has been reconstructed by the pixel $\mathbf{p}'_i$ of view v_i. Possible cases are:

- There has been no $\mathbf{s}_{\mathbf{p}'i}$ reconstructed. This means that reconstruction failed for that pixel or that it has been removed by one of the presented methods. Another possibility is that it has been removed by the shadow removal process. In any case, the corresponding sample is rejected.
- $\mathbf{s}_{\mathbf{p}'i}$ has been reconstructed (and either selected as best surfel or not). This means either that v_i has an unobstructed view to the point $\mathbf{p}$ or that $\mathbf{s}_{\mathbf{p}'i}$ represents a surface point different to $\mathbf{p}$. To test if $\mathbf{s}_{\mathbf{p}'i}$ represents a different point, the distance between $\mathbf{p}$ and $\mathbf{p}_{vi}$ ($\mathbf{p}_{vi}$ is the camera position of v_i) is measured as well as the distance between $\mathbf{s}_{\mathbf{p}'i}$. and $\mathbf{p}_{vi}$. If the difference $\left| \| \mathbf{p} - \mathbf{p}_{v_i} \| - \| \mathbf{s}_{\mathbf{p}'_i} - \mathbf{p}_{v_i} \| \right|$ exceeds a user-defined threshold, the points are assumed to be distinct and the sample is rejected due to occlusion.

Lambertian Texture

Diffuse texture is computed from a single reflectance sample according to the Lambertian reflectance model:

$$\rho_d = \frac{L}{\mathbf{n}^T \mathbf{l} E}, \tag{3.13}$$

with, for a given object point, ρ_d the diffuse albedo, L the radiance observed by the camera, E the irradiance from the projector at that point, $\mathbf{n}$ its normalized surface normal, and $\mathbf{l}$ the normalized vector toward the light source. The surface normal is actually computed from the neighbors of the reconstructed point. The direction $\mathbf{l}$ is computed from the geometric projector calibration and E from the luminous projector calibration. Figures 3.15 and 3.16a show respective results.

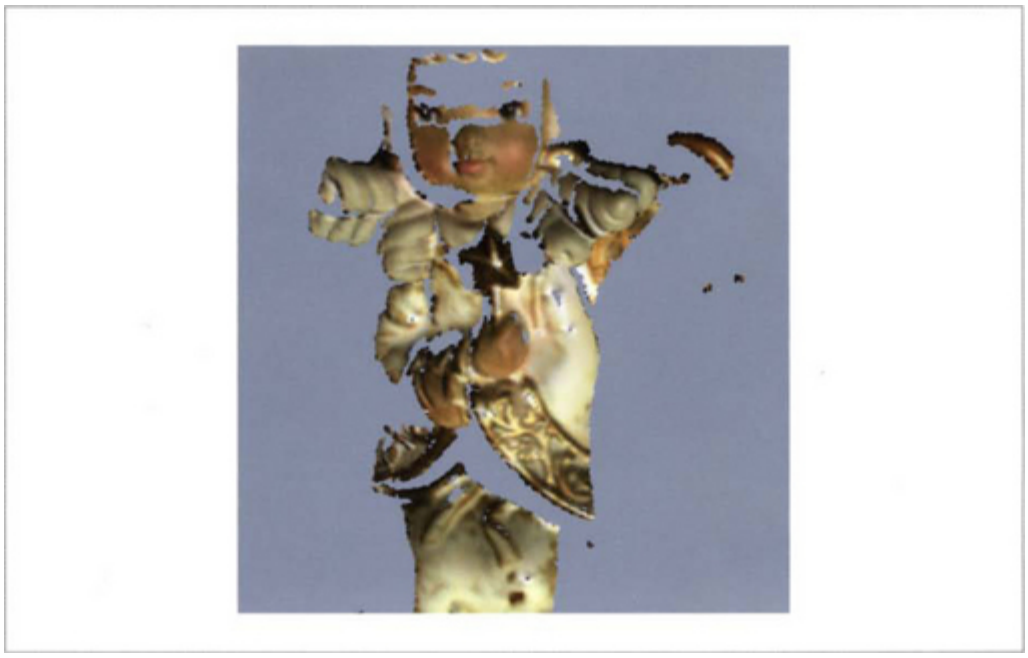

Figure 3.15 Single view reconstruction, Lambertian texture.

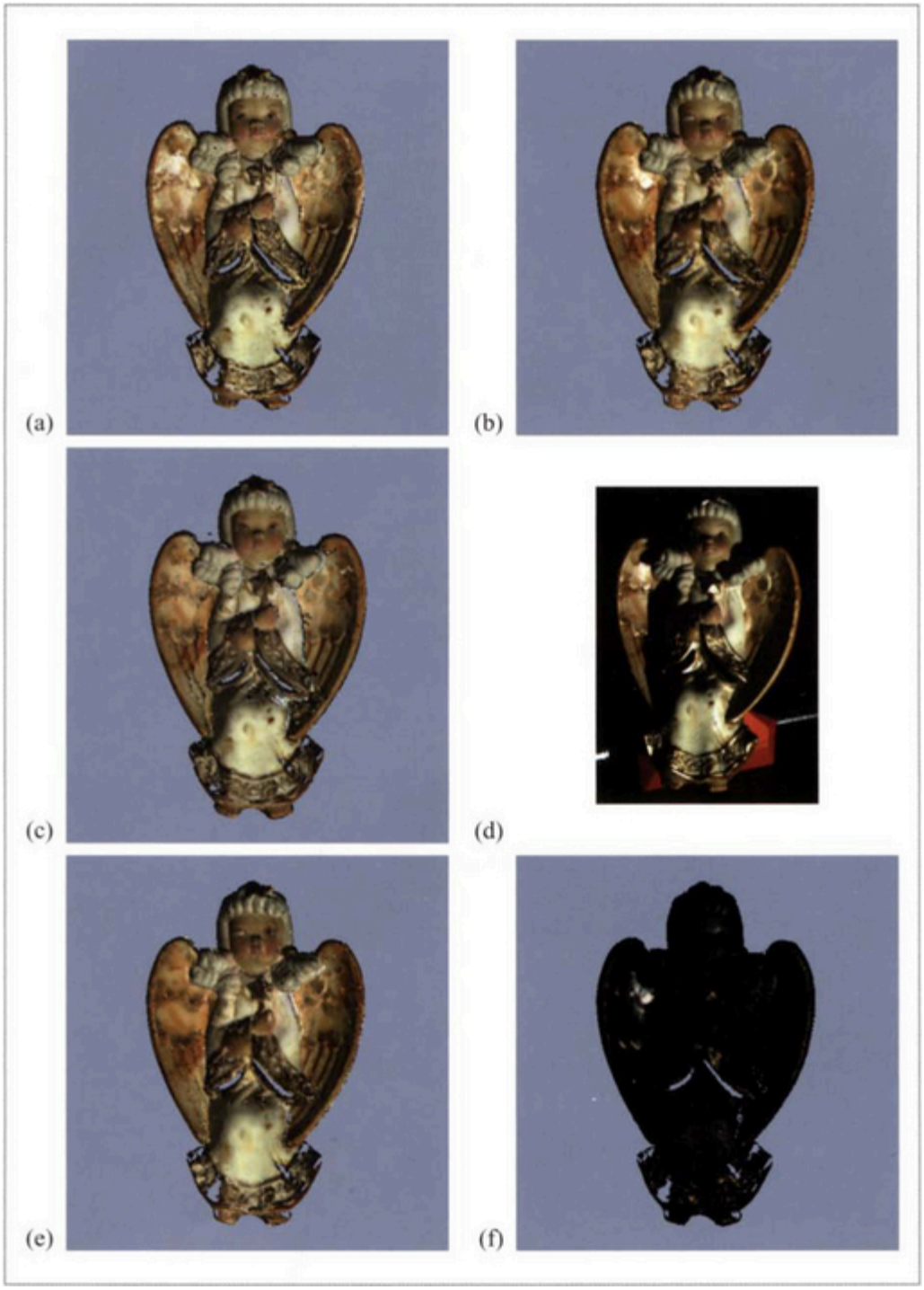

Figure 3.16 Reconstructed Lambertian texture (*a*), Phong texture (*b*), blended overlapping reconstructions (*c*), region of original camera view (*d*), diffuse component of Phong texture (*e*), and specular component (*f*).

Phong Texture

Alternatively, the diffuse term is used together with a specular Phong lobe as the reflectance model:

$$\frac{L}{E} = \rho_d \mathbf{n}^T l + \rho_s (l^T (2\mathbf{n}\mathbf{n}^T - I)v)^n, \tag{3.14}$$

with ρ_s the maximum specular albedo, **v** the normalized vector toward the camera, and n the specular exponent. Nonlinear least squares fitting of Equation (3.14) to the reflectance samples is done simultaneously for red, green, and blue ρ_d and ρ_s, but with a single exponent n using the Levenberg-Marquardt method as suggested by Lafortune et al. [LFTG97]. An initial estimate is computed by linear fitting. This is achieved by fixing n for optimization and by solving the resulting linear least squares problem. This is repeated for exponentially increasing n and each time the residual is computed. The fit leading to the smallest error is chosen as a result of the linearized fitting. Linear fitting can also be used as fall back to nonlinear fitting.

Figure 3.16 shows a result of the Phong fit. The reconstruction still shows some artifacts in regions of sparse sampling and at points with erroneous normals due to outliers. However, the diffuse part of the specular fit provides a significantly improved texture compared to the Lambertian fit, as the Lambertian fit overestimates brightness in specular regions.

Further improving the Phong fit is difficult due to the limited number of samples per lumitexel. One way to come around this would be to apply material clustering as presented by Lensch et al. [LGK+01]. Another possibility is to improve the accuracy of the normals by including them into the texture-fitting process, but this also requires a well-distributed reflectance sampling.

3.2.6 RESULTS

Figure 3.17 shows a single-ring reconstruction of a clay pot acquired from only 15 views. Figure 3.18 shows a telephone also reconstructed from 15 rotated views using the same setup. It can be seen that the clay pot produces better quality regarding texture and geometry because its material is mostly diffuse, in contrast to plastic material of the telephone.

Figure 3.17 Clay pot reconstructed from 15 rotary views (384,492 surfels). Lambertian texture (*a*) and synthetic Phong texture (*b*) for showing geometric detail. The Lambertian texture exhibits stripe artifacts due to specular reflection of the clay surface.

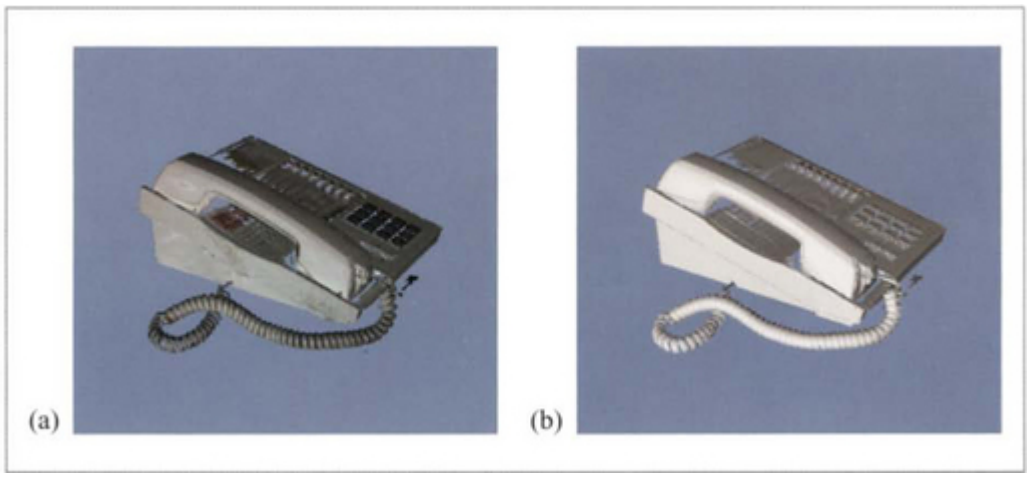

Figure 3.18 Telephone reconstructed from 15 rotary views (317,681 surfels). Lambertian texture (*a*) and no texture (*b*).

Because the chosen objects contain both diffuse and specular materials and relatively much occlusion, the achieved results still contain artifacts such as outliers, false normals, and holes. These are addressed by the postprocessing stage as described by Weyrich et al. [WPK+04], although in our case, most of the holes originate from suboptimal setup and could be addressed by interactive control of the turntable and more appropriate tilting of the object in order to generate optimal views.

Accuracy

The accuracy of the system is evaluated using a steel sphere of 150 mm in diameter that had been manufactured with a tolerance of 0.1 mm and painted in white color. A single ring of 10 partial reconstructions at a uniform angular step was taken and merged according to Section 3.2.4. For the merged reconstruction, a center was computed by fitting a least squares sphere to the surfels. The mean distance to the center was 74.947 mm and the maximal errors were less than 0.5 mm, which is roughly the surfel spacing (red histogram in Figure 3.19a). For further analysis of these errors, a sphere of the same radius was fitted to each partial reconstruction. The resulting centers (Figure 3.19b) reveal a small systematic error in the turntable calibration, plus a mechanical effect happening between the first two scans. The blue histograms show the higher accuracy of each partial reconstruction. The green histogram is obtained by translating the partial reconstructions to a common center. This step cannot be done for general objects, it just illustrates that there is some potential left in the merging process. However, registration of surfel objects based on geometry or texture to extreme subsurfel precision is difficult.

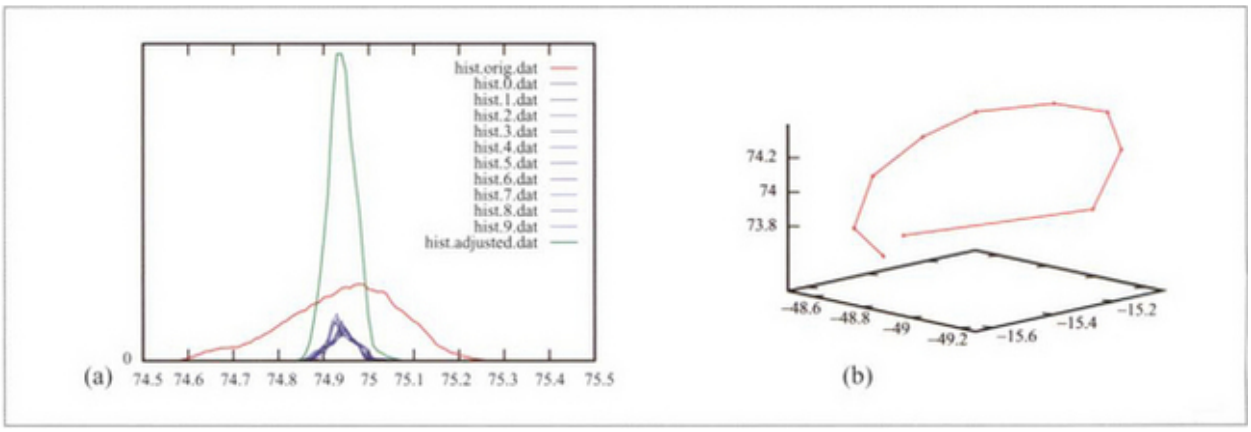

Figure 3.19 (*a*) Histograms of surfel distances to sphere center (red: full reconstruction; blue: partial reconstruction; green: partial reconstructions merged by optimal translations). (*b*) Centers of least squares fits for partial reconstructions. All units given in millimeters.

Performance

The angel figurine of 180 mm height was reconstructed from 3 × 30 views. Each view contributed in average 28,082 surfels, leading to a raw reconstruction consisting of 2,527,362 surfels. From these, 473,819 erroneous surfels were removed by the ray skew criterion or because they formed too-small connected components (outliers). Another 1,826,879 surfels were discarded during merging. Finally, 18,452 surfels got discarded by the shadow removal process. Removing the shadows is the computationally most expensive step—it took 38 minutes. Texture reconstruction took 16 minutes and computation of the normals another 6 minutes. The best surfels merging procedure took again 6 minutes and the cost of the remaining operations is negligible.

3.2.7 CONCLUSION

This section described a completely point-based acquisition system for geometry and texture. The system consists of a projector as a calibrated light source, a single grayscale camera, and a turntable. Methods for robust reconstruction and removal of artifacts have been presented as well as methods for merging multiple point-based reconstructions into a single-layered representation. Additionally, to the Lambertian reflectance model, the Phong model was fitted to the reflectance samples, utilizing the underlying structured-light approach.

The described system is oriented for low cost, simple operation, and arbitrary object geometry, and can handle cavities due to the structured-light approach. The next section presents a high-end system based on the visual hull method, aiming at the acquisition of objects with difficult surface properties, such as fuzzy and highly specular objects.

3.3

POINT-BASED 3D PHOTOGRAPHY

Wojciech Matusik and Hanspeter Pfister

3.3.1 OVERVIEW

The point-based acquisition systems described in the previous two sections, including most commercial systems, cannot scan very complex geometries and appearances, such as glass, fur, hair, cloth, leaves, or feathers. In this section we describe a robust, completely automated point-based scanning system that can capture objects with arbitrary geometric complexity and appearance. It 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 setup and calibration, and scans objects that fit within one cubic foot volume. The acquired objects can be accurately composited into synthetic scenes.

3.3.2 PREVIOUS WORK

There are many approaches for acquiring high-quality 3D shape from real-world objects, including contact digitizers, passive stereo depth-extraction, and active light-imaging systems. The previous two sections discuss passive and active scanning systems. To acquire objects with arbitrary materials, we use an image-based modeling and rendering approach. Image-based representations have the advantage of capturing and representing an object regardless of the complexity of its geometry and appearance.

Early image-based methods [MB95, CW93] allowed for navigation within a scene using correspondence information. Light field methods [LH96, GGSC96] achieve similar results without geometric information, but with an increased number of images. Gortler et al. [GGSC96] combine the best of these methods by including a visual hull of the object for improved ray interpolation. These methods assume static illumination and, therefore, cannot accurately render objects into new environments.

An intermediate between purely model-based and purely image-based methods is the view-dependent texture mapping systems described by Pulli et al. [PCD+97] and Debevec et al. [DYB98, DTM96]. 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 highly specular surfaces due to the relatively small number of textures.

As noted in Debevec et al. [DYB98], surface light fields [GMP98, WAA+00, NSI99a, CBCG02] can be viewed as a more general and more efficient representation of view-dependent texture maps. Wood et al. [WAA+00] store light field data on accurate high-density geometry, whereas Nishino et al. [NSI99a] use a coarser triangular mesh for objects with low geometric complexity. Chen et al.

[CBCG02] use a decomposition of surface light fields that can be efficiently rendered on modern graphics hardware. Surface light fields are capable of reproducing important global effects such as interreflections and self-shadowing. Our system is capable of surface light field acquisition and rendering.

Images generated from a surface light field always show the object under a fixed lighting condition. To overcome this limitation, inverse rendering methods estimate the surface BRDF from images and geometry of the object. To achieve a compact BRDF representation, most methods fit a parametric reflection model to the image data [SWI97, YDMH99, LGK+01]. Sato et al. [SWI97] and Yu et al. [YDMH99] 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. [LGK+01] 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 scenes. As observed in Hawkins et al. [HCD01], objects featuring glass, fur, hair, cloth, leaves, or feathers are very challenging or impossible to represent this way. Reflectance functions for points in highly specular or self-shadowed areas are very complex and cannot easily be approximated using smooth basis functions. In our work we make no assumptions about the reflection property of the material we are scanning.

An alternative is to use image-based, nonparametric representations for object reflectance. Marschner et al. [SHWC99] use a tabular BRDF representation and measure the reflectance properties of convex objects using a digital camera. Their method is restricted to objects with a uniform BRDF, and they incur problems with geometric errors introduced by 3D range scanners. Georghiades et al. [GBK99] apply image-based relighting to human faces by assuming that the surface reflectance is Lambertian.

More recent approaches [MGW01, DHT+00, HCD01, BKMK01] use image databases to relight objects from a fixed viewpoint without acquiring a full BRDF. Debevec et al. [DHT+00] define the *reflectance field* of an object as the radiant light from a surface under every possible incident field of illumination. They use a light stage with few fixed camera positions and a rotating light to acquire the reflectance field of a human face [DHT+00] or of cultural artifacts [HCD01]. The polynomial texture map system described in [MGW01] uses a similar technique for objects with approximately planar geometry and diffuse reflectance properties. Belhumeur et al. [BKMK] use essentially the same method as Debevec et al. [DHT+00] to render objects with arbitrary appearance. These reflectance field approaches are limited to renderings from a single viewpoint.

3.3.3 SYSTEM OVERVIEW

Our system uses a point-sampled visual hull [MBR+00] as the underlying geometric model. The visual hull can be computed robustly using active backlighting. We augment the point-sampled visual hull with view-dependent opacity to accurately represent complex silhouette geometry, such as hair. We call this new shape representation the *opacity hull.* To construct the opacity hull we use the multibackground matting techniques similar to Smith and Blinn [SB96].

Our system can acquire a surface light field of the object. It can also acquire reflectance fields of the object from multiple viewpoints. We call this representation a *surface reflectance field*, because the data are parameterized on the surface of the visual hull of the object. Surface reflectance fields can be rendered from any viewpoint under new illumination. We use images from the same viewpoints to compute the opacity hull and the surface reflectance field. This avoids any registration inaccuracies and has proven to be extremely robust.

Laurentini [Lau94] introduced the visual hull as the maximal volume that is consistent with a given set of silhouettes. The visual hull cannot represent surface concavities. Yet, due to its hull property, it provides a conservative estimate of an object's structure. The opacity hull and surface reflectance field extend the utility of visual hull considerably by faithfully representing complex silhouettes and materials.

Instead of relying on accurate geometry, our representation relies heavily upon acquired radiance information to produce accurate renderings of the object. We can adaptively acquire more images for objects with concavities or high specularity, and fewer images for objects with simple geometry and mostly diffuse surfaces. Naturally, this approach is not useful for applications where geometric fidelity is required. In this Chapter we demonstrate that the combination of opacity hull geometry and the image-based surface-reflectance field leads to an effective representation for rendering applications. Our system is capable of acquiring and rendering objects that are fuzzy, highly specular, or that contain any mixture of materials.

3.3.4 HARDWARE SETUP

Figure 3.20 (left) shows an overview of our hardware setup. Objects are placed on a plasma monitor that is mounted onto a rotating turntable. An array of light sources is mounted on an overhead turntable. The lights are spaced roughly equally along the elevation angle of the hemisphere.

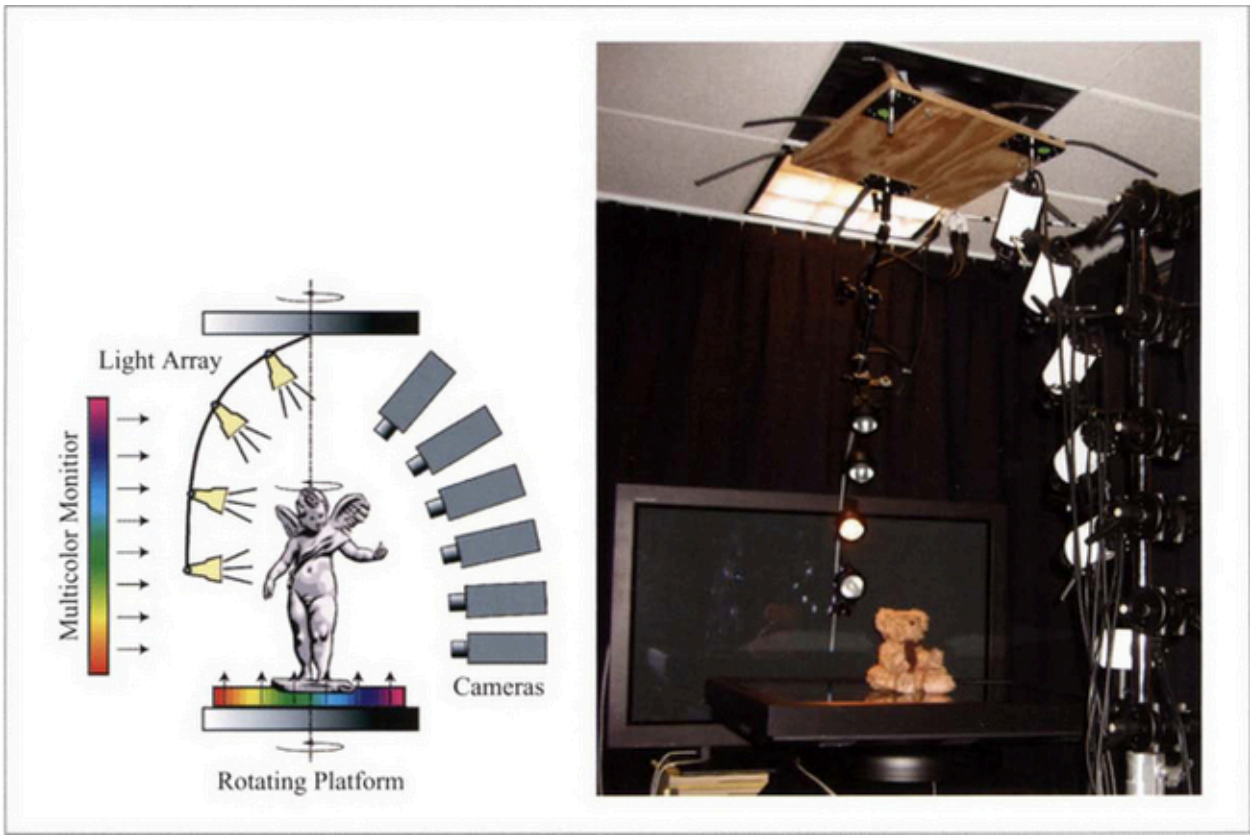

Figure 3.20 Our 3D digitizing system combines both active and passive imaging methods. Objects are rotated on a turntable while images are acquired. Plasma monitors are used to extract high-quality alpha mattes. An overhead array of light sources can be rotated to acquire surface reflectance fields. A system diagram is shown on the left and the photograph is shown on the right.

During object scanning, the lights can be fixed, rotate around the object for a fixed point of view, or made to rotate with the object. Six video cameras are pointed at the object from various angles. To facilitate consistent backlighting we mount the cameras roughly in the same vertical plane. A second plasma monitor is placed directly opposite of the cameras. Figure 3.20 (right) shows a picture of our third-generation scanner. The two plasma monitors have a resolution of 1,024 × 768 pixels.

We use six QImaging QICAM cameras with 1,360 × 1,036 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 to six directional 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. The dimmers are set once such that the image sensor is not oversaturated for viewpoints where the lights are directly visible.

In many ways, our setup is similar to the enhanced light stage that has been proposed as future work in Hawkins et al. [HCD01]. A key difference is that our system uses multicolor backlights for alpha matte extraction and construction of the opacity hull. As we will show, the availability of approximate geometry and view-dependent alpha greatly extends the class of models that can be captured.

3.3.5 DATA ACQUISITION PROCESS

Calibration

The scanning sequence starts by placing the object onto the turntable and, if necessary, adjusting the position and aperture of the cameras. If any camera adjustments are required, we must first acquire images of a known calibration object, a patterned cube in our case. An image of the calibration target is taken from each of the viewpoints. Intrinsic and extrinsic camera parameters are computed using a special calibration procedure for turntable systems with multiple cameras [Bea02]. Calibration can be computed reliably given the fixed rotation axis and the large numbers of images.

Reference Images

Next, the plasma monitors are turned on and we acquire images of patterned backdrops used for multibackground matting. For each viewpoint, each patterned backdrop is photographed alone without the foreground object. As in Zongker et al. [ZWCS99], we call these images the *reference images*. Reference images only have to be acquired once after calibration. They are stored and used for subsequent object scans.

Object Images

The object is then put on the turntable and a sequence of images is automatically acquired. The number of turntable positions is user specified and depends on the object. During this first rotation, both plasma monitors illuminate the object from below and behind with the patterned backdrops. As in Zongker et al. [ZWCS99], we call the images of the foreground object in front of the backdrops *object images*. The object images and reference images are used to compute alpha mattes and the opacity hull. We depend on good repeatability of the turntables to ensure that the reference images and the object images are well registered.

Radiance Images

We then switch off the plasma monitors and turn on one or more directional lights of the array. We found that we get the best results when using additional fill light to avoid dark shadows and high contrast in the images. We avoid specular reflections from the monitors by covering the vast majority of the display surface with black felt without upsetting the object position. We acquire a set of *radiance images* of the illuminated object during the second rotation of the turntable. The radiance images are used for surface light-field rendering. The directional lights can be fixed or made to rotate with the object. The coupled rotation case leads to greater coherence of radiance samples in each surface point.

Reflectance Images

If we want to relight the acquired object, we acquire an additional set of images used to construct the surface reflectance field. The array of lights is rotated around the object. For each rotation position, each light in the light array is sequentially turned on and an image is captured with each camera. We use four lights and typically increment the rotation angle by 24° for a total of 4 × 15 images for each camera position. This procedure is repeated for all viewpoints. We call the set of all images the *reflectance images.* They are used to construct the surface reflectance field.

Environment Mattes

If we would like to relight highly specular or refractive objects, we additionally acquire environment mattes from multiple viewpoints. Environment mattes are a compact representation for high-resolution surface reflectance. The acquisition process involves taking multiple images of the scanned object in front of a backdrop with a 1D Gaussian profile that is swept over time in horizontal, vertical, and diagonal directions. We capture environment mattes using three of the six cameras, roughly spaced 10° apart in elevation and 5° in angular turntable increments. For a full 360° view of the object this corresponds to 3 × 72 = 432 viewing positions.

HDR Images

All radiance and reflectance images are captured using a high dynamic range technique similar to that of Debevec and Malik's [DM97]. Since raw output from the CCD array of the cameras is available, the relationship between exposure time and radiance values is linear over most of the operating range. For each viewpoint, we take four pictures with exponentially increasing exposure times and use a least squares linear fit to determine the response line. Our imager has 10 bits of precision. Due to nonlinear saturation effects at the extreme ends of the scale we only use values in the range of 5 to 1,000 in our least squares computation. We can ignore the direct current (DC) offset of this calculation and store only the slope of the response line as one floating point number per pixel. This image representation allows for the specification of a desired exposure interval at viewing time.

Alpha Mattes

To construct the image-based visual hull on which we parameterize the opacity hull, we extract silhouette images from various viewpoints. Earlier versions of our system use fluorescent lights to acquire silhouette views. Backlighting is a common segmentation approach that is often used in commercial 2D machine vision systems. The backlights saturate the image sensor in areas where they are visible. We then threshold the silhouette images to establish a binary segmentation for the object.

However, binary thresholding is not accurate enough for objects with small silhouette features, such as hair. It also does not permit subpixel accurate compositing of the objects into new environments. An additional problem is color *spill* [SB96], the reflection of backlight on the foreground object. 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 particularly prominent for highly specular surfaces, such as metal or ceramics.

We use a variant of the multibackground matting technique of Smith and Blinn [SB96] to solve these problems. We acquire alpha mattes of the object from each viewpoint. An alpha matte of a foreground object can be extracted by imaging the object against two background images with different colors. We display the following sinusoidal background patterns on the plasma monitors:

$$I_j(x, y, n) = \left(1 + n \sin\left(\frac{2\pi(x+y)}{\varphi} + j\frac{\pi}{3}\right)\right) \times 127. \tag{3.15}$$

$I_j(x, y, n)$ is the intensity of color channel $j = r, g, b$ at pixel location (x, y). To maximize the per-pixel difference between the two backdrops, the patterns are phase shifted by 180° ($n = -1$ or 1). The user defines the period of the sinusoidal stripes with the parameter ϕ.

Using the multibackground matting equation from Smith and Blinn [SB96], the per-pixel object alpha α is computed as

$$\alpha = 1 - \frac{\sum |O_{n=1} - O_{n=-1}|}{\sum |R_{n=1} - R_{n=-1}|} \tag{3.16}$$

where $R_{n=1}$ and $R_{n=-1}$ are per-pixel background colors of the reference images, and $O_{n=1}$ and $O_{n=-1}$ are per-pixel foreground colors for $n = \pm 1$, respectively.

If we measure the same color at a pixel both with and without the object for each background, Equation (3.16) equals zero. This corresponds to a pixel that maps straight through from the background to the sensor. The phase shifts in the color channels of Equation (3.15) assure that the denominator of Equation (3.16) is never zero. The sinusoidal pattern reduces the chance that a pixel color observed due to spill matches the pixel color of the reference image. Nevertheless, we still observed spill errors for highly specular objects, such as the teapot or the bonsai pot.

To reduce these errors we apply the same procedure multiple times, each time varying the parameter ϕ of the backdrop patterns. For the final alpha matte we store the maximum alpha from all intermediate mattes. We found that acquiring three intermediate alpha mattes with relatively prime periods $\phi = 27$, 40, and 53 is sufficient. The overhead of taking the additional images is small, and we need to store only the final alpha matte. Figure 3.21 shows two alpha mattes acquired with our method. We found that in practice this method works very well for a wide variety of objects, including specular and fuzzy materials.

Figure 3.21 Alpha mattes acquired using our backdrops.

Opacity Hull Construction

Using the alpha mattes of the object from various viewpoints, we construct the opacity hull. First, we use binary thresholding on the alpha mattes to get binary silhouette images. 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.

The binary silhouettes are then used to construct the Image-Based Visual Hull (IBVH) [MBR+00]. The IBVH algorithm can be counted on to remove improperly classified foreground regions as long as they are not consistent with all other images. We resample the IBVH into a dense set of surface points as described in Section 3.3.6. Each point on the visual hull surface is projected onto the alpha mattes to determine its opacity from a particular observed viewpoint.

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

Figure 3.22 shows the observed alpha values for three surface points on an object for all 6×36 viewpoints. Each pixel has been colored according to its opacity. Black color corresponds to $\alpha = 0$, white color corresponds to $\alpha = 1$, and gray color corresponds to values in between. Red color indicates camera views that are invisible from the surface point.

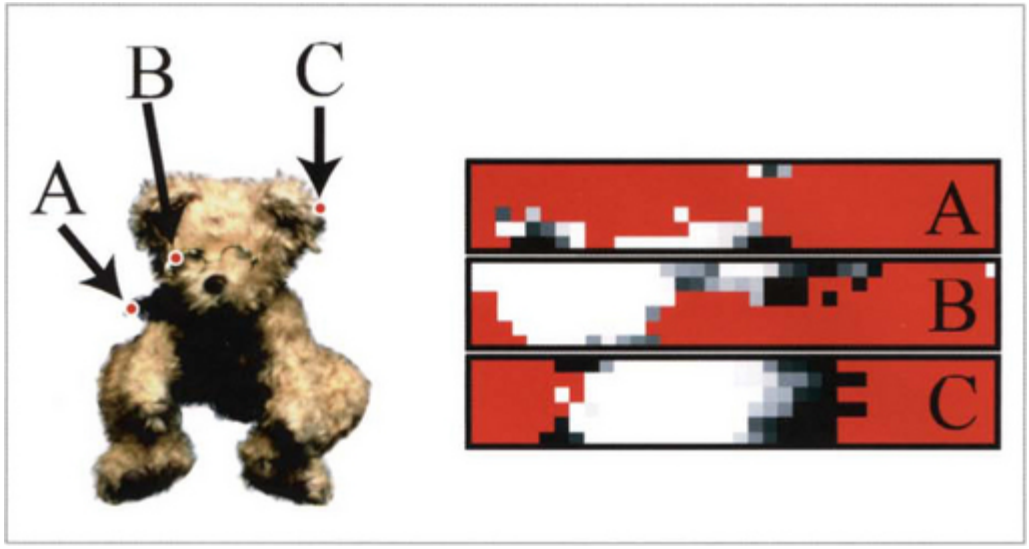

Figure 3.22 Observed alpha values for points on the opacity hull. Red color indicates invisible camera views.

The function 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 Wood et al. [WAA+00], one could estimate a parametric function for A and store it in each alpha sphere. However, as shown in Figure 3.22, the view-dependent alpha is not smooth and not easily amenable to parametric function fitting. Consequently, we store the acquired alpha mattes and interpolate between them to render the opacity hull from arbitrary viewpoints.

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 [UO93]. Voxel shells are not able to accurately represent fine silhouette features, which is the main benefit of the opacity hull.

Recognizing the importance of silhouettes, Sander et al. [SGG+00] 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 somewhat similar to the concentric, semitransparent textured shells that Lengyel et al. [LPFH01] used to render hair and furry objects. They use geometry—called *textured fins*—to improve the appearance of object silhouettes. A single instance of the fin texture is used on all edges of the object. In contrast, opacity hulls can be looked at as textures with view-dependent alphas for every surface point of the object. They accurately render silhouettes of high complexity using only visual hull geometry.

Surface Light Fields and Reflectance Fields

Similar to constructing the opacity hull, we reparameterize the acquired radiance images into rays emitted from surface points on the visual hull. This representation is a surface light field as described by Miller et al. [RMP98] and Wood et al. [WAA+00]. However, our surface light fields are created on the surface of the visual hull rather than on the surface of the object.

Surface light fields can only represent models under the original illumination. To address this limitation we acquire surface reflectance fields from multiple viewing positions around the object. Debevec et al. [DHT+00] define the reflectance field under directional illumination as a six-dimensional function $R(\mathbf{p}, \omega_i, \omega_r)$. For each surface point $\mathbf{p}$, it maps incoming light directions ω_i to reflected color values along direction ω_r. Thus, for each point $\mathbf{p}$ we have a four-dimensional function $R_{\mathbf{p}}(\omega_i, \omega_r)$.

During acquisition, we sample the four-dimensional function $R_{\mathrm{p}}(\omega_i, \omega_r)$ from a set of viewpoints Ω_r and a set of light directions Ω_i. In previous reflectance field approaches [DHT+00, HCD01, BKMK01], the sampling of light directions is dense (e.g., $|\Omega_i| = 64 \times 32$ in [DHT+00]), but only a single viewpoint is used. In our system, we sample the reflectance field from many 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 ($|\Omega_i| = 4 \times 15$). Thus, our illumination environment has to be filtered down substantially, and our re-illumination is accurate only for relatively diffuse surfaces [RH01a].

Reconstruction of an image from a new viewing direction under a new lighting configuration is a two-pass process. First, we reconstruct the images from the original viewpoints under novel illumination. Once we have computed these images, we interpolate the image data to new viewpoints. For a particular image from the original viewpoint, it is useful to define a slice of the reflectance field called a *reflectance function* $R(x_o, \omega_i)$ [DHT+00]. It represents how much light is reflected toward the camera by pixel x_o as a result of illumination from direction ω_i. We can reconstruct the image $L(x_o)$ from the original viewpoint under novel illumination as a weighted linear combination of the light sources as we will show in Section 3.3.8.

Environment Mattes

We observe that using a high-resolution environment matte in a particular viewing direction is superior to using only the light array to provide incident illumination. For example, Debevec et al. [DHT+00] use 2,048 light positions, which corresponds to a 32 × 64 pixel environment map. Using only our light array effectively constrains us to illumination from a 4 × 15 pixel environment map. These resolutions are not nearly enough to accurately capture and represent transmissive and refractive effects. For example, looking straight through a glass window shows the background in its full resolution. On the other hand, using a high-resolution illumination environment is only feasible with environment matting. The alternative would be to store a very large number of reflectance images for each viewpoint, which is impractical. Environment mattes are in essence a very compact representation for high-resolution surface reflectance fields.

To acquire environment mattes, we are using the high-quality procedure by Chuang et al. [CZH+00]. The acquisition process involves taking multiple images of the foreground object in front of a backdrop with a 1D Gaussian profile that is swept over time in horizontal, vertical, and diagonal directions. Using the nonlinear optimization procedure described by Chuang et al. [CZH+00], we then solve for a and the parameters of the 2D Gaussian's G.

To save storage and computation time for the nonlinear parameter estimation, we identify and remove areas outside the object silhouette. The environment matte is subdivided into 8 × 8 pixel blocks. Each surface point on the opacity hull that is visible from this view is projected into the

image. Only those blocks that contain at least one back-projected surface point are stored and processed.

For certain positions in the camera array, the rim of the plasma monitors is visible through a transparent object, which makes much of the field of view unusable. Consequently, we only use the lower and the two uppermost cameras for acquisition of environment mattes. The lower camera is positioned horizontally, directly in front of the background monitor. The two upper cameras are positioned above the monitor on the turntable. Using our environment matte interpolation (see Section 3.3.8), we can render plausible results for any viewpoint.

3.3.6 POINT-SAMPLED DATA STRUCTURE

We use an extended point representation based on the layered depth cube (LDC)-tree [PZvBG00] as our shape model on which we parameterize the view-dependent appearance data. In a preprocess, we compute the octree-based LDC-tree from the IBVH. The creation of the LDC-tree starts with the sampling of the visual hull from three orthogonal directions. The sampling density depends on the model complexity and is user specified. The layered depth images are then merged into a single octree model. Since our visual hulls are generated from virtual orthographic viewpoints, their registration is exact. This merging also insures that the model is uniformly sampled.

Point samples have several benefits for 3D scanning applications. From a modeling point of view, the point-cloud representation eliminates the need to establish topology or connectivity. This facilitates the fusion of data from multiple sources, as pointed out by Levoy and Whitted [LW85]. They also avoid the difficult task of computing a consistent parameterization of the surface for texture mapping. We found that point models are able to represent complex organic shapes, such as a bonsai tree or a feather, more easily than polygonal meshes. In particular, it would be hard to represent the view-dependent opacity values at each point of the opacity hull using polygonal models and texture mapping.

Each surfel (surface element) in the LDC-tree stores depth, normal, and a camera-visibility bit vector. The visibility vector stores a value of one for each camera position from which the surfel was visible and zero if it was not visible. It can be quickly computed during IBVH construction using the visibility algorithm described in Matusik et al. [MBR+00]. Our representation stores all of the acquired radiance and reflectance images with irrelevant information removed. This is accomplished by dividing each source image into 8 × 8 blocks and removing those blocks that lie outside the object's silhouette. For each image, we compute a simple mask by back-projecting all surfels from which this view is visible. Only the 8 × 8 pixel blocks that contain at least one back-projected surfel are stored. This simple scheme typically reduces the total amount of image data by a factor of 5 to 10, depending on the geometry of the model.

A relightable model requires more than 20 GB of raw image data. In order to make these data more manageable, we have implemented a simple compression scheme for reflectance images. For each original viewpoint, we apply principal component analysis (PCA) to corresponding 8 × 8 image blocks across the varying 4 × 15 illumination directions taken from a common viewpoint. We set a global threshold for the root mean square (RMS) reconstruction error and store a variable number of

principal components per block. As shown in Section 3.3.8, the average number of components per block is typically four to five. PCA compression typically reduces the amount of reflectance data by a factor of 10.

Figure 3.23 shows a depiction of our data structure for surface reflectance fields, simplified for clarity. The figure shows the first six PCA images for two original views. These images are combined into new radiance images from the same viewpoints under new illumination using the method described in Section 3.3.8. During rendering, points on the opacity hull of the object are projected into the radiance images based on their visibility. Each surfel's color is determined using interpolation among the four closest views. Note that the figure shows the two closest views.

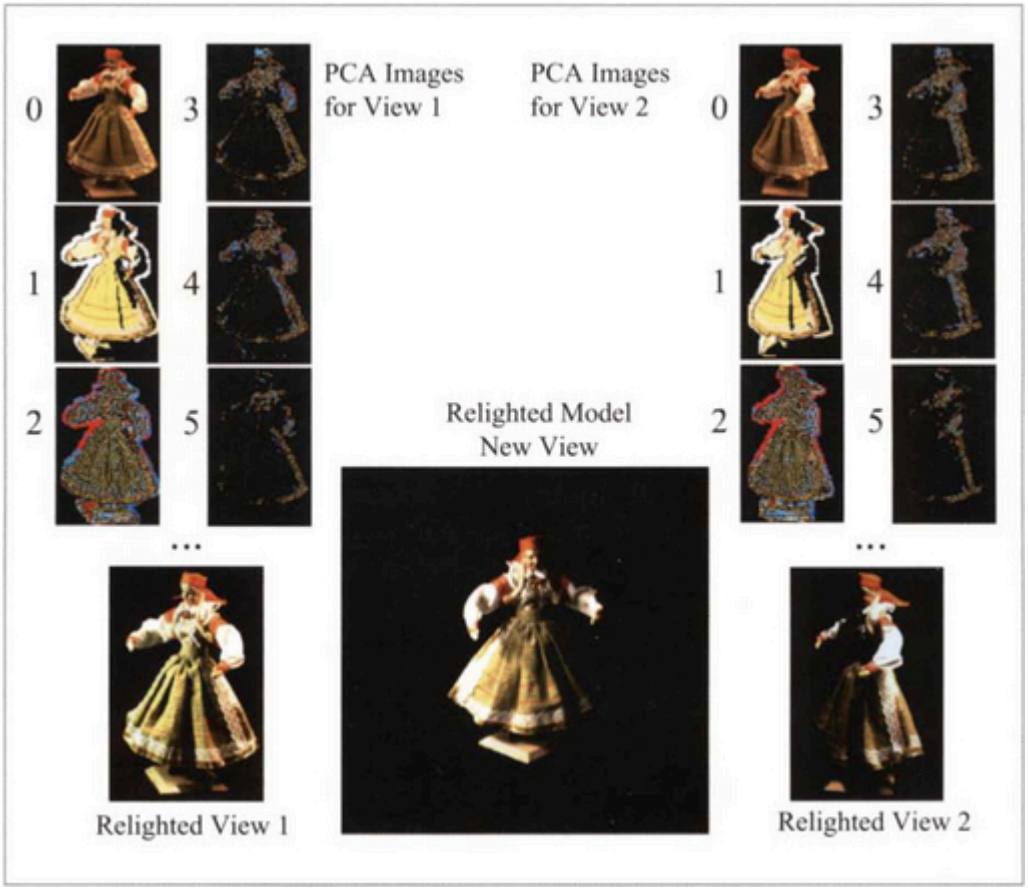

Figure 3.23 Data structure for surface reflectance fields.

3.3.7 SURFACE LIGHT FIELDS

Surface Light Field Rendering

To render our point-sampled models we use the elliptical weighted average (EWA) surface-splatting approach of Zwicker et al. [ZPvG01] (see Section 6.1). First, the opacity and color of each surfel are interpolated from the radiance images as discussed below. 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 antialiasing.

To interpolate the radiance images from the original viewpoints to arbitrary viewpoints, we use the unstructured lumigraph interpolation of Buehler et al. [BBM+01]. For each surfel, we use k-nearest neighbor interpolation to reconstruct view-dependent alpha and radiance values. This assures continuous transitions between camera views.

For each frame, we compute the normalized direction $r_c(i)$ from each surfel position to each visible camera i using the visibility bit vector and a 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

$$w(i) = \frac{\cos\theta_i - \cos\theta_t}{1 - \cos\theta_t} . \tag{3.17}$$

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 $1/p(i)$. This ensures 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.

Surface Light Field Examples

We have collected a wide range of objects and surface types with our system. We have acquired many difficult surfaces including those of various genuses, with concavities, and with fine scale features. We have also captured a wide range of materials, including fuzzy and highly specular materials. A variety of different models are shown in Figure 3.24.

Figure 3.24 Surface light fields of several objects from new viewpoints. Note the alpha compositing with the textured backgrounds.

For all objects, we use 6 cameras and 36 turntable positions. We acquire 6 object images for alpha matting from each viewpoint (over three ϕ values with $n = \pm 1$). For surface light fields, we capture 1 radiance image from each viewpoint for a total of $6 \times 36 \times (4 \times 1 + 6) = 2{,}160$ images. The entire digitizing process takes about one hour for a surface light field. The whole process is fully automated without any user intervention. All of our models are created from a single scan.

We resampled all of our visual hull models to 512×512 resolution of the LDC-tree. The processing time to segment the images, compute the opacity hull, and build the point-based data structure is less than 10 minutes.

Figure 3.25 shows the visual hull, opacity hull, and final composite rendering of a bonsai tree. Notice the coarse shape of the visual hull and the much improved rendition using the opacity hull, despite the fact that their geometry is identical. The opacity hull also allows high-quality compositing over complex backgrounds without edge aliasing.

Figure 3.25 (*a*) Photo of the object. (*b*) Rendering using the opacity hull. (*c*) Visual hull. (*d*) Opacity hull.

Unstructured lumigraph interpolation for viewpoints other than those seen by reference cameras introduces small artifacts, most notably for specular or concave areas. Figure 3.26 shows acquired images of an object (Figures 3.26a and c). Figure 3.26b shows the object from an intermediate viewpoint. Note that the figure shows only the two closest views, although we use the four closest views for interpolation. As can be seen in the figure, the artifacts are generally small. The animations show that the *k*-nearest neighbor interpolation leads to nice and smooth transitions.

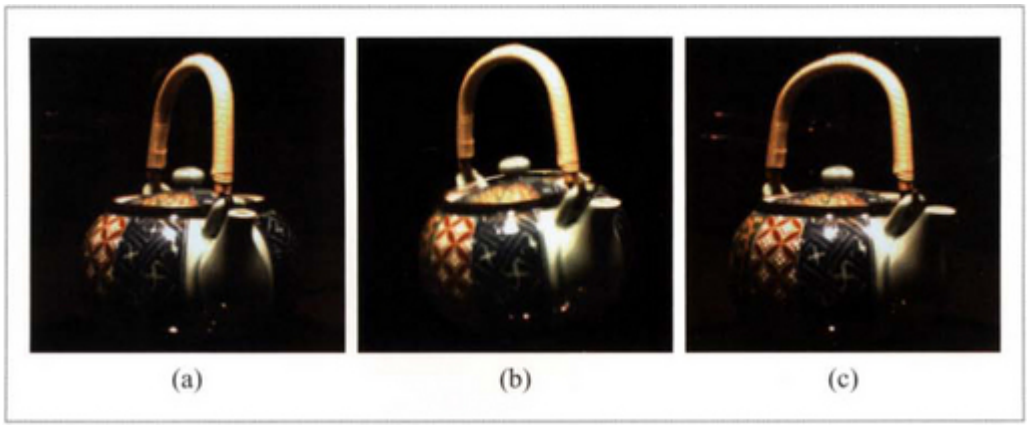

Figure 3.26 Rendering from arbitrary viewpoints. (***a and c***) Original images. (*b*) Interpolated view.

To evaluate the number of images required to compute the visual hull, we instrumented our code to compute the change in volume of orthographic visual hulls as each silhouette is processed. We then randomized the processing order of the images and repeated the IBVH calculation multiple times. The plots shown in Figure 3.27 illustrate the rather typical behavior.

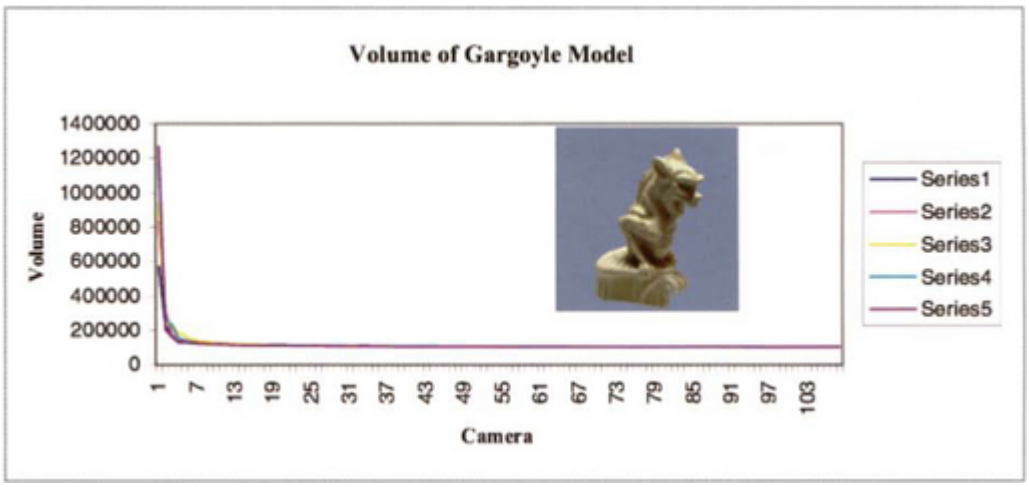

Figure 3.27 The volume of the visual hull as a function of the number of images used to construct the visual hull.

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. These view-dependent opacities and radiance measurements dramatically improve the final renderings.

3.3.8 RELIGHTING

Light Transport Model

We first develop our model for how light scatters at the object surface following the exposition of Chuang et al. [CZH+00] and Debevec et al. [DHT+00]. Assuming that the incident radiation originates infinitely far away from the object surface, the light arriving at a camera pixel can be described as

$$L(x_0) = \int_{\Omega} R(x_o, \omega_i) L(\omega_i) d\omega_i \,. \tag{3.18}$$

$L(x_o)$ is the recorded radiance value at each camera pixel, and $L(\omega_i)$ is the environment illumination from direction ω_i. $R(x_o, \omega_i)$ is a weighting function that comprises all means of light transport from the environment through the foreground object to the camera. Debevec et al. [DHT+00] call it the reflectance function. The integration is carried out over the entire hemisphere Ω and for each wavelength. We will drop the wavelength dependency in the rest of this Chapter, assuming that all equations are evaluated separately for r, g, and b.

Given a measured radiance $L(x_o)$ at a pixel and an environment $L(\omega_i)$, we want to estimate the function $R(x_o, \omega_i)$ for the point x_o on the object surface corresponding to the ray through that pixel. Our scanning system provides two different illumination fields for the environment: illumination

from a high-resolution 2D texture map behind the object (displayed on the plasma monitors), and illumination by the overhead light array from a sparse set of directions on the remaining hemisphere. Figure 3.28 shows the basic setup.

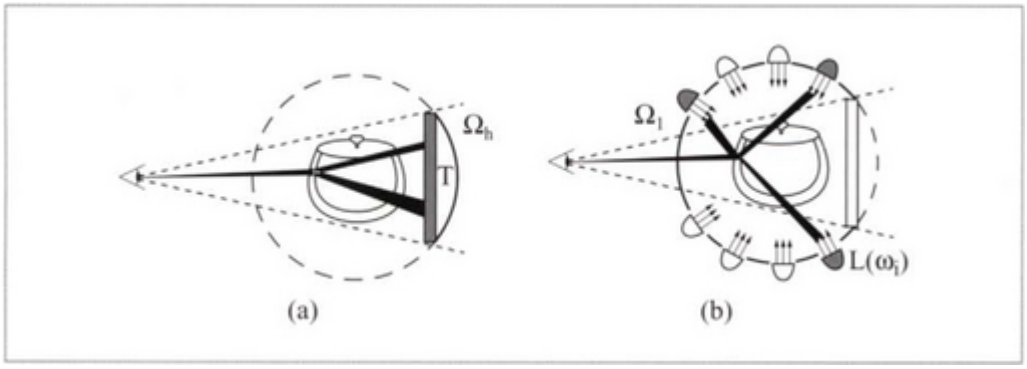

Figure 3.28 Illumination environment and light propagation model in our system. (***a***) High-resolution sampling across Ω_h. (*b*) Low-resolution sampling across Ω_l.

We call the sector of the environment hemisphere covered by the high-resolution texture map Ω_h, and the remaining sector covered by the light array Ω_l. Furthermore, we are making the simplifying assumption that light transport in Ω_h can be described by two components (see Figure 3.28a). As shown in the figure, we are approximating the (potentially complicated) paths of light through the object by two straight light bundles from the ray-surface intersection to the background monitor. On the other hand, light from one or more directional light sources $L(\omega_i)$ in Ω_l is refracted and reflected by the object before arriving at pixel x_o (see Figure 3.28b). Here we assume that the incident light field in Ω_l can be sampled at substantially lower resolution than light in Ω_h coming directly from behind the object. Thus, Equation (3.18) becomes

$$L(x_o) = \int_{\Omega_h} R_h(x_o, \omega_i) L(\omega_i) d\omega_i + \int_{\Omega_I} R_l(x_o, \omega_i) L(\omega_i) d\omega_i \,. \quad (3.19)$$

As proposed by Chuang et al. [CZH+00], we use a sum of 2D Gaussians to describe $R_h(x_o, \omega_i)$. We restrict ourselves to a maximum of two Gaussians per surface point *xo.* Thus,

$$R_h(x_o, \omega_i) = \sum_{j=1}^{2} a_j(x_o) G_j(\omega_i, x_o) \,. \quad (3.20)$$

G_j are elliptical, oriented 2D unit-Gaussians, and a_j are attenuation factors. Each Gaussian G_j is parameterized by the center C_j, its standard deviation σ_j, and its orientation θ_j.

Since we are sampling Ω_l with a discrete set of n light positions L_j, we can rewrite Equation (3.19) as

$$L(x_o) = \int_{\Omega_h} \sum_{j=1}^{2} a_j(x_o) G_j(\omega_i, x_o) L(\omega_i) d\omega_i + \sum_{j=1}^{n} R_j L_j \,. \quad (3.21)$$

Using the environment matting and reflectance field procedures outlined in Section 3.3.5, we estimate the parameters for a_j, G_j, and R_j. For each viewpoint, the estimated parameters a_j and G_j are stored in an environment matte T and R_j is stored in n reflectance images.

Discussion

Equation (3.21) is a compromise between high-quality environment matting [CZH+00] and the practical limitations of our 3D acquisition system. Ideally, one would surround the object with high-resolution monitors and acquire the parameters of an arbitrary number of weighting functions W distributed over multiple monitors. Instead, we assume that most of the refracted and reflected rays arriving at a pixel originate from the incident light field behind the object. This is true for most objects with a strong reflected component (mostly at grazing angles) and a strong refracted component. It is not necessarily correct for transparent objects with large-scale internal structure or surface facets, such as a crystal glass. However, in practice we found this approach to work reasonably well.

It is important to note that, despite the term surface *reflectance* field, we are capturing a much wider array of effects, including refraction, dispersion, subsurface scattering, and nonuniform material variations. These effects, which are typically costly or impossible to simulate, can be rendered from our model in a reasonable amount of time. As noted by Debevec et al. [DHT+00], the surface reflectance field is almost equivalent to the Bidirectional Surface-Scattering Distribution Function (BSSRDF). The main differences are that we do not know the exact physical location of a ray-surface intersection, and that the incoming direction of light is the same for any point on the surface. The first problem could potentially be addressed by improving the visual hull geometry using methods of stereopsis. Solving the second problem would require illuminating the object with a dense set of laser-point light sources.

Equation (3.21) differs from Equation (3.12) in Chuang et al. [CZH+00] by restricting the number of incoming ray bundles from the monitors to two, and by replacing the foreground color F with a sum over surface reflectance functions R_i. The first assumption is valid if reflection and refraction at an object causes view rays to split into two distinct ray bundles that strike the background (see Figure 3.28a). The second assumption results in a more accurate estimation of how illumination from Ω_l affects the object's foreground color. Chuang et al. [CZH+00] make up for this by capturing additional environment mattes using monitors on each side of the object.

Surface Reflectance Field Rendering

We start with a new environment map, for example, a spherical high-dynamic range light probe image of a natural scene. We first reconstruct the low-resolution reflectance function for the original viewpoints using this new illumination environment. Next we interpolate these relit images to the new viewpoint. The high-resolution reflectance function is reconstructed by first interpolating the parameters of the Gaussians for a new viewpoint. Then we integrate the Gaussians with the environment map to compute the outgoing radiance. (Note the difference between environment "mattes" and "maps.") Finally, we sum the low- and high-resolution components. We now discuss these steps in more detail.

Low-resolution Reflectance Function

During rendering, we compute the k-nearest ($k = 4$) visible original viewpoints used for each surface point. As mentioned in Section 3.3.5, visibility is determined during opacity hull construction and stored in the visibility vector. We compute the interpolation weights w_i for the four closest

viewpoints according to unstructured lumigraph interpolation [BBM+01]. The weights ensure continuous transitions between camera views and epipole consistency; in other words, rendering the object from original camera viewpoints exactly reproduces the original images. Using the global camera parameters, each surface point is then projected into its k-nearest alpha mattes. We use the interpolation weights w_i to interpolate the view-dependent alpha from the alpha mattes.

Now we describe how we compute images that show the object under the new environment map illumination for each of the k-nearest original viewpoints. The environment map must be down-sampled to match the light positions used during acquisition. In our case it contains only 4 × 15 positions. The original reflectance images for a viewpoint are compressed using PCA as described in Section 3.3.6. This allows us to express the reflectance function R_j of a specific incoming light direction for the low-resolution lighting environment as

$$R_j \approx \tilde{R}_j = \sum_{i=1}^{m} \gamma_{ij} e_i, \tag{3.22}$$

where e_i are the eigenvectors corresponding to the the largest m eigenvalues. Each e_i is a 64-element vector corresponding to an 8 × 8 pixel block. Given a new set of n directional lights $\tilde{L}_j$, we can compute the outgoing radiance $L_{\Omega l}$ for the relit image as

$$\begin{aligned} L_{\Omega_l} &= \sum_{j=1}^{n} \tilde{R}_j \tilde{L}_j = \sum_{j=1}^{n} \left(\sum_{i=1}^{m} \gamma_{ij} e_i\right) \tilde{L}_j \\ &= \sum_{i=1}^{m} \left(\sum_{j=1}^{n} \gamma_{ij} \tilde{L}_j\right) e_i \,. \end{aligned} \tag{3.23}$$

The inner summation over n is the same for each of the 8 × 8 pixels and can, therefore, be computed only once. Finally, we use the interpolation weights w_i to interpolate the k relit images to obtain the image from the novel viewpoint.

High-resolution Reflectance Function

Our acquired environment mattes are parameterized on the plane T of the background monitor. However, for rendering they need to be parameterized on the global environment map Ω. Figure 3.29 shows a 2D drawing of the situation.

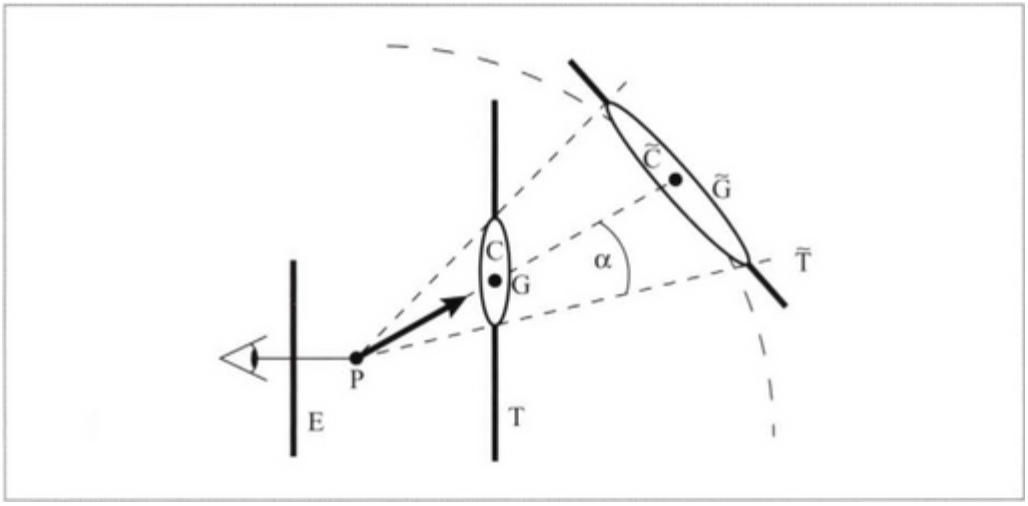

Figure 3.29 Reprojection of the environment matte Gaussian *G* from the monitor plane *T* into the environment map Ω.

During system calibration we determine the position of each monitor plane *T* with respect to each viewpoint. This information is globally stored per viewpoint. Ω is the parameterization plane of the new environment map. The mapping from *T* to Ω may be nonlinear, 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.

To compute the radiance contribution from the environment mattes involves two steps: interpolating the new Gaussian $\hat{G}$, and convolving it with the environment map to compute the resulting colors.

We first interpolate the parameters of $k = 4$ reprojected Gaussian $\tilde{G}_i$. Using w_i, we compute linear combinations for the amplitudes a_i and the directional vectors $\tilde{G}_i$. The angular parameters (α_i, β_i) and $\overline{\theta}_i$ are blended using quaternion interpolation. The result is a new Gaussian $\hat{G}$ that is an interpolated version of the Gaussian $\tilde{G}_i$, morphed to the new viewpoint.

Note that this interpolation needs to be performed on matching Gaussians from the environment mattes. Figure 3.30 shows a simplified 1D drawing of the matching process. We are only storing two Gaussians G_i per environment matte pixel, where each pixel corresponds to a viewpoint ray V_i in the figure. The two Gaussians per pixel are classified as *reflective* ($\tilde{G}_{ir}$) or *transmissive* ($\tilde{G}_{it}$). We compute the angle ϕ of their center vectors $\tilde{G}_{ir}$ and $\tilde{G}_{it}$ with the surface normal *N*. If $\phi > 90°$, we

classify the Gaussian as transmissive. If $\phi <= 90°$, we classify it as reflective. If both Gaussians are reflective or refractive, we only store the one with the larger amplitude a. This computation has to be performed for each change of the environment map, after computing the reprojected Gaussian $\tilde{G}$.

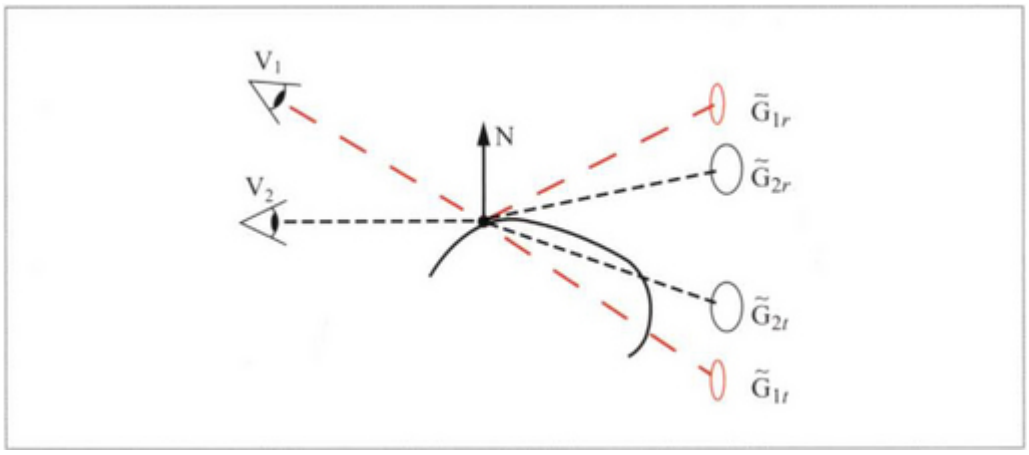

Figure 3.30 Matching of reflective and refractive Gaussians.

During interpolation, we match up refractive and reflective Gaussians. In other words, new Gaussians $\hat{G}_r$ and $\hat{G}_t$ are interpolated from $\tilde{G}_{ir}$ and $\tilde{G}_{it}$, respectively. Note that this matching would be much more difficult if we had stored more than two Gaussians per environment matte pixel, as proposed by Chuang et al. [CZH+00].

To compute the outgoing radiance $L_{\Omega h}$ for a new viewpoint we integrate the interpolated Gaussian $\hat{G}_{r,t}$ with the novel high-resolution illumination $L(\omega_i)$:

$$L_{\Omega_h} = \int_{\Omega_h} \sum_{j=r,t} a_j(x_o)\hat{G}_j(\omega_i, x_o)L(\omega_i)d\omega_i \,. \tag{3.24}$$

The final pixel color C according to Equation (3.21) is the sum of the low-resolution reflectance field (Equation 3.23) and the high-resolution reflectance field (Equation 3.24).

Surface Reflectance Field Examples

Low-resolution Illumination

First, we present the results and analysis of relighting using only low-resolution reflectance fields. For low-resolution surface reflectance fields, we acquire reflectance images using 4 × 15 light directions from each viewpoint for a total of 6 × 36 × (4 × (4 × 15)) + 6) = 53,136 images. The entire digitizing process takes about 14 hours for a surface reflectance field.

Figure 3.31 shows a model under new illumination. Figure 3.32 shows several scanned objects composited into real environments. We acquired spherical light probe images [DM97] at the respective locations to capture the illumination. All objects shown in this paper are rendered from novel viewpoints that are not part of the acquired image sequence.

Figure 3.31 Relightable model under novel illumination.

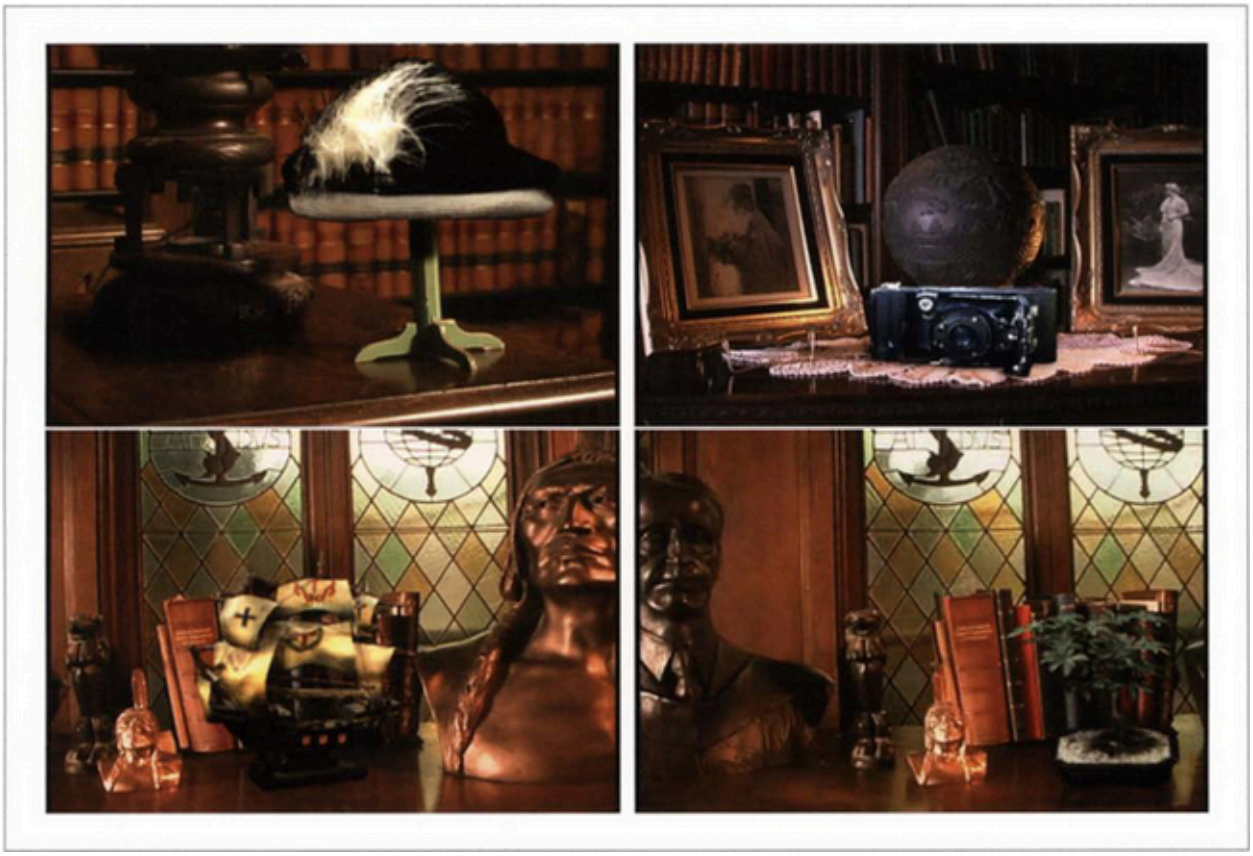

Figure 3.32 A combination of scanned and real objects in real environments. The scanned objects were illuminated using surface reflectance fields.

In the process of acquiring models, we have made many interesting measurements and observations. Figure 3.33 shows plots of the measured low-resolution reflectance field data for three surface points on an object. We chose the surfels to be in specular and self-shadowed areas of the object. The dark parts of the plots are attributable to self-shadowing. The data lack any characteristics that would make them a good fit to standard parametric BRDF models or function approximation techniques. This is typical for the data we observed.

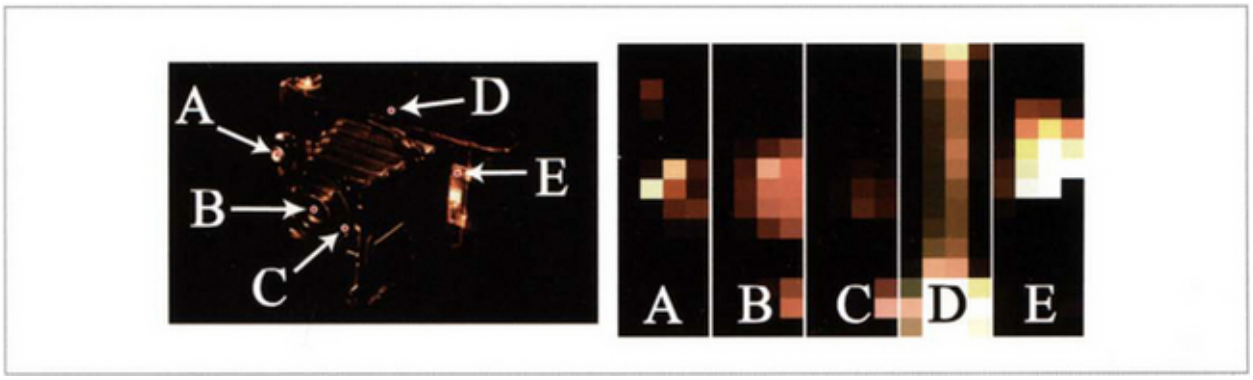

Figure 3.33 Measured reflectance function data for several surface points.

Figure 3.34 shows a visualization of the number of PCA components per 8 × 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 high dynamic range (HDR) reflectance images. Note that areas with high texture frequency require more components than areas of similar average color. The maximum number of components for this view is 10, the average is 5. This is typical for all of our data.

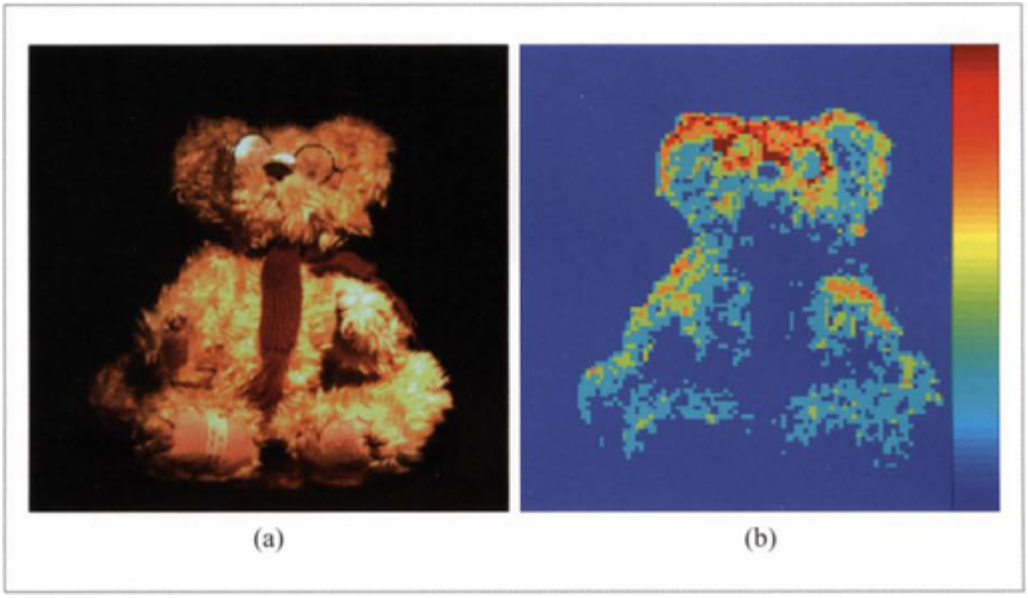

Figure 3.34 (*a*) Original view. (*b*) Visualization of the number of PCA components per block (Max. = 15, Mean = 5).

High-resolution Illumination

Next, we present the results of using the combined low-resolution reflectance fields and high-resolution environment mattes. Figures 3.35 and 3.36 show different models in new illumination environments.

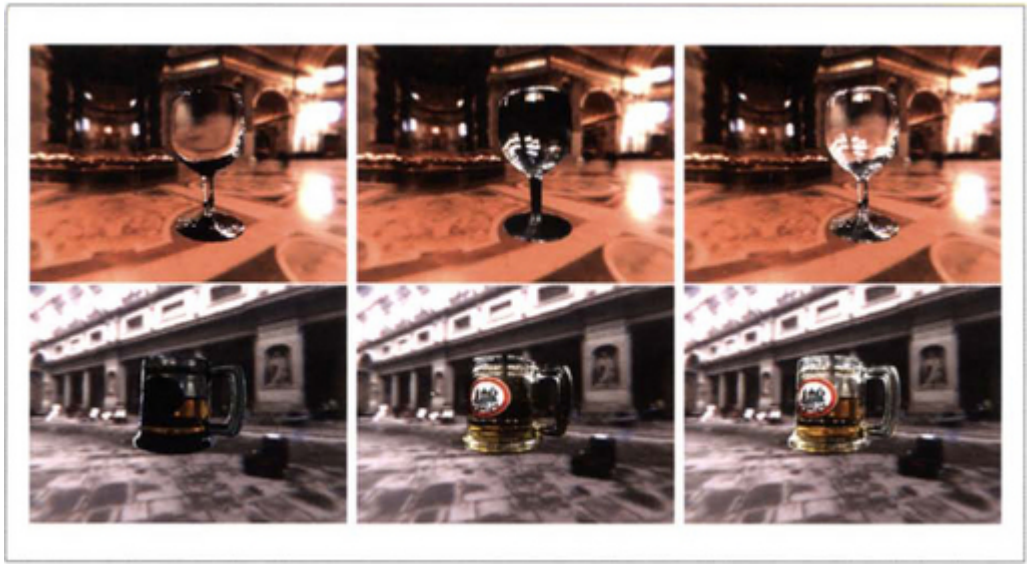

Figure 3.35 ***Left***: High-resolution reflectance field from the environment mattes. ***Middle***: Low-resolution reflectance field from the reflectance images. ***Right***: Combined.

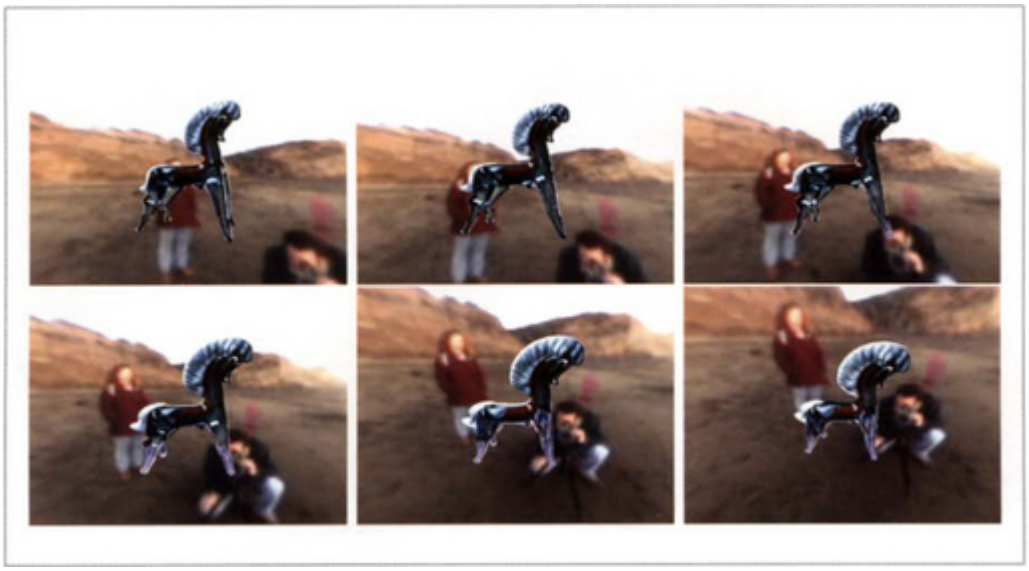

Figure 3.36 Frames from an animation with rotating viewpoint.

We used the light-probe images available from Paul Debevec's website as environment maps. All objects are rendered from novel viewpoints that are not part of the acquired image sequence.

Figure 3.35 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 3.36 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.

In addition to low-resolution surface reflectance fields we need to acquire environment mattes from different viewpoints. For each viewpoint we capture 300 images to estimate an environment matte. Because the aspect ratio of our monitor is 16:9, we use a different number of backdrop images for each direction: 125 in diagonal, 100 in horizontal, and 75 in vertical direction. We capture environment mattes using only three of the six cameras. The cameras are roughly spaced 10° apart in elevation and 5° angular turntable increments. For a full 360° view of the object, this corresponds to 3 × 72 = 432 viewing positions. All environment matte and reflectance images are captured using the HDR technique. For each viewpoint, we take four pictures with different exposure times. The total number of pictures is 3 × 72 × 300 × 4 = 259,200. The additional acquisition time is about 18 hours, or 5 minutes per viewpoint.

3.3.9 CONCLUSION

This section described a fully automated and robust 3D photography system optimized for the generation of high-quality renderings of objects. The basic premise of our scanning approach is to use large amounts of radiance and opacity information to produce accurate renderings of the object instead of relying on accurate geometry. We have also presented a method for acquisition and rendering of transparent and refractive objects. Using a point-based 3D scanning system with color monitor backdrops, we are able to scan transparent objects that would be extremely difficult or impossible to scan with traditional 3D scanners.

We have introduced the opacity hull, a new shape representation that stores view-dependent alpha parameterized on the visual hull of the object. Opacity hulls combined with surface reflectance fields allow us to render objects with arbitrarily complex shape and materials under varying illumination from new viewpoints. We have shown that a parameterization of surface reflectance fields into high- and low-resolution areas offers a practical method to acquire high-quality models.