A Multi-Camera Method for 3D Digitization of Dynamic, Real-World Events

Peter Rander
rander @cs.cmu.edu

CMU-RI-TR-98-12

May 1998

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890 USA

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical and Computer Engineering

Thesis Committee
Takeo Kanade, chair
José Moura
Martial Hebert
Mel Siegel
Narendra Ahuja, University of Illinois

Abstract

This thesis presents a method for the 3D digitization of dynamic, real-world events. This task requires sufficient temporal and spatial sampling to capture the entirety of an event, as well as the estimation of 3D shape and appearance over time. Direct sensing of global 3D structure is not feasible because of the motion of the scene, and even range scanning systems usually sample too coarsely or too slowly to accurately capture dynamic events. This thesis presents a method of 3D digitization that overcomes this sensing problem through the use of a synchronized collection of a large number of calibrated video cameras.

Our 3D digitization method decomposes 3D shape recovery into the estimation of visible structure in each video frame followed by the integration of visible structure into a complete 3D model. Visible surfaces are extracted using the multi-baseline stereo (MBS) algorithm. This implementation of MBS efficiently supports any number cameras in general positions through a novel rectification strategy for general camera configurations that do not allow the rectification of all images to a single 3D plane. Stereo-computed range images are then integrated within a volumetric space using a novel integration strategy. Each range image is converted into a 3D mesh, and then into an implicit surface embedded in the volumetric space by encoding the signed distance between each 3D volume sample (voxel) and the 3D mesh. Multiple range images are integrated by accumulating the signed distance at each voxel. The resulting global surface model is then extracted by applying the Marching Cubes implicit-surface extraction algorithm.

The digitization of scene appearance uses the estimated 3D structure to determine visibility and sampling of color in the original video images. A compact, global texture map is computed by mixing the color estimates, emphasizing the color estimates obtained from cameras viewing the local surface structure most directly. Alternatively, an image-based representation is derived from the global model by reprojecting the global structure itself back into the original cameras to generate a visible surface model in each real camera. New views are synthesized by separately rendering three of these models and mixing the rendered views of the models directly on the virtual image plane.

This thesis also presents extensive results of digitizing real events recorded in the *3D Dome,* a recording studio employing a synchronized collection of 51 calibrated video cameras mounted on a 5-meter diameter geodesic dome. This facility provides a workspace of approximately 8 cubic meters, sufficient to capture the entirety of motion of 1-2 people performing athletic actions including swinging a baseball bat, bumping a volleyball, and passing a basketball. Results are presented for 5 different events, each 1-2 seconds long. Video is captured at 240x320 image resolution, with a volumetric modeling resolution of 1 cubic center. The resulting models are used to generate both simple camera motions near the original camera viewpoints as well as camera motions deep into the event space and into views nearly impossible to capture with real cameras.

Acknowledgments

Without the direction, support, and encouragement of many people, this thesis and the work it is built upon would never have occurred. Without a doubt, my advisor, Takeo Kanade, has taught me more in the last seven (!) years that I would have thought possible in a lifetime. Despite his ever-increasing responsibilities as a researcher, director of the Vision and Autonomous Systems Center (VASC), and then director of The Robotics Institute, he still managed to provide both high-level direction as well as detailed insights into the unexplained phenomena that drove further research. His enthusiasm and dedication have been instrumental at motivating me to fight through the long periods of seemingly little progress. His honesty and integrity have strengthened my belief that a researcher can and should admit limitations as well as the highlights of new discoveries.

I have found many of these same characteristics in many other educators along the way to this Ph.D. My Ph.D. committee members, José Moura, Martial Hebert, Mel Siegel, and Narendra Ahuja (at the Beckman Institute of University of Illinois), for example, have been exemplary in their flexibility, guidance, and critical analysis. From José's guidance during my teaching internship to Mel's advice about thesis writing, this crew has helped shape my thinking in general and this document in particular. One other person who deserves special mention is my undergraduate advisor, Mark Paulik (at the University of Detroit-Mercy). His ability to challenge me to stretch myself has helped me tremendously during my Ph.D. studies.

Of course, many other people here at CMU and VASC have had a major impact on me and my work as well. P.J. Narayanan is a friend, fellow virtualizer, and co-conspirator on most of the ideas presented in this thesis. Without him, little would have actually worked, and I would have lacked a good friend. We grew up together on this project, PJ as a Project Scientist and me as a student. Since PJ left CMU, Sundar Vedula and Hideo Saito have come in on the project, and together we have had a lot of fun and made great research progress, too. One of the first people I met in VASC was Jim Rehg, who did a great job of opening my eyes to computer vision through reading groups, interesting discussions in the lab, and general inquisitiveness. He also helped me survive my PhD qualifier and proposal. There are too many others to name, but here's an attempt: Bill Ross and Jim Moody (thanks for

keeping those computers running!), Mei Chen, Michael Smith, Rich Madison (office mates!), Conrad Poelman, Omead Amidi, Carol Novak, Mark Wheeler, Yoichi and Imari Sato, Pauletta Pan, Stephanie Riso, Carolyn Ludwig, Jennie Kay, Todd Williamson, Reg Willson, Harry Shum – the list really is too long to name everyone!

On the more social side, Chris and Nicole Newburn are unparalleled in the time and energy they devoted to our friendship. I feel like a brother to them, and love them both dearly. And then there are the fellowship people, Dave and Sandy Thuel, Steve and Marge Frezza, Mikko and Erica Lipasti, Sean and Nancy Smith, Shawn and Trisha Bushway, Keith and Tami Kiser, Kevin and Pam Schmaltz, Bill Courtright, Chris Novak, and Pascal Meier, and Mark and Nancy Werner. I can’t say enough about how much you all have helped me through the tough times and have celebrated the good times.

Perhaps the people who deserve the greatest acknowledgment are my family. Without them, I would never have been able to undertake graduate studies in the first place. My mother and father, through their love and encouragement, helped me see that I had talent and had a responsibility to develop and use it wisely. With a pitch fork and a dirty barn, they also emphasized the importance of hard work, humility, and a sense of humor!

My wife, Marylin, has also helped me beyond measure. Her love and encouragement are boundless. She has supported me through the highs and lows of the rocky road to a Ph.D. with more grit than her mild demeanor might suggest. She also was the vessel by which our greatest joy and blessing, our daughter Rachel, came into this world. They are the reason why I was able to maintain sanity through thesis writing and, at the same time, the reason why I would have given it all up if the need arose. I will love them both with all that I am for the rest of my days.

Finally, and most importantly, I owe all that I am and all that I have to the grace and mercy of God the Father through Jesus Christ, His Son, by the power of the Holy Spirit. It is He who entrusted me to a loving family, led me through my undergraduate education, into graduate studies when I thought all options had been eliminated, and to a great advisor and an exciting research project. It is He who led my wife and me to meet, who led us into marriage. and who blessed us with His greatest gift, His child Rachel. And it is by His grace and for His glory that I have completed this thesis.

Table of Contents

List of Figures

Chapter 1

Introduction

Television and motion pictures are tremendously successful because of the ease with which these media convey the vision of remote events to millions of viewers. This success has been achieved using purely 2D imaging systems and with viewers having no control over their viewpoint. While most of us have adapted to these limitations, our visual systems are designed to see stereoscopically and we normally expect to be able to walk around the places we visit.

The potential of extending these media into 3D and with user viewpoint control has dramatic implications on the immersive capabilities of these media. 3D structure would allow stereoscopic viewing, increasing the visual cues of structure and relative scale. With user-controlled viewpoints, television could become significantly more interactive. Imagine chasing Michael Jordan down the basketball court as he plays in a live NBA basketball game, or running with Barry Sanders as he eludes would-be tacklers in a live NFL football game. Consider the power of training in such an environment, for example as a medical student studying a real surgical procedure – even from the surgeon's perspective – without interfering with the real operation. Such a training video could be frozen in time, giving the student the power to study intricate operations one step at a time while still allowing complete navigation through the frozen event.

This thesis addresses one major challenge in reaching this goal: the recovery, or digitization, of 3D shape and scene appearance of dynamic events. By digitizing 3D shape, new viewpoints can be synthesized using standard projection of this shape into the desired viewpoint. Digitizing scene appearance allows this projection to create a photorealistic view of the scene. In order to demonstrate this ability, this thesis also addresses this view synthesis problem.

The most significant obstacle to digitizing dynamic 3D shape and appearance stems from the motion of the scene. If the scene were guaranteed to be motionless, then it would be possible to move a single, reliable, high-accuracy sensor or sensor system around the scene and build up a model to nearly arbitrary quality. Even with dynamic scenes, a single sensor could be used if that sensor could see *through* the scene, such as with MRI. These sensors do not capture visible light, however, so they would be unable to reproduce the visible appearance of the scene at any viewpoint. With dynamic events, then, many sensors must be used simultaneously to “freeze” the event in time. By sampling these sensors at high rates, the scene motion can be well represented.

In order to resolve the dynamic event digitization problem, this thesis employs a synchronized collection of calibrated video cameras. These sensors directly capture the visual appearance of dynamic events from specific viewpoints, allowing the estimation of scene appearance from any viewpoint. In addition, the collection of views from all cameras at any point in time can be used to estimate complete 3D scene structure. Finally, video cameras capture snapshots of the scene at temporal rates sufficient to create the illusion of continuous motion to a human observer, so the event motion itself is well represented by this temporal sampling for visual display.

1.1 Limitations of Current Methods

The visual reconstruction problem addressed in this thesis can be broken down into two broad steps: modeling from images and rendering from models. To achieve maximum flexibility, both steps will rely on physically-based models. The modeling phase will recover all the physical parameters of the scene that contribute to scene appearance, such as 3D scene shape, spectral and spatial distributions of light, scene inter-reflection, and material properties such as reflectance. Then computer graphics rendering techniques will synthesize new views using these physically based dynamic scene models, even allowing arbitrary changes in any of the modeling parameters.

Great progress has recently been made in realizing this vision. On the modeling side, many recent methods (including one presented in this thesis) have been developed to combine multiple range images into 3D models. Methods by Curless and Levoy [10], Rander et. al. [47], Wheeler [67], Hilton et. al. [19], and Martins and Moura [37] all have found reliable, robust solutions by dealing with the merging problem in a discretized volumetric space rather than as a surface problem [57][61][65]. Some progress has also been made in estimating material properties from images. Sato et. al. [50] estimate scene reflectance by applying color-space analysis to a set of observations (images) of object surfaces whose shape is estimated with great accuracy. On the rendering side, ray tracing and radiosity methods have proven to be powerful techniques for creating realistic images, with lighting, shading, translucency, and inter-reflection.

This ideal modeling approach has two serious drawbacks, however. First, the estimation of many of the model parameters cannot yet be done reliably, especially in relatively uncontrolled, dynamic settings. Even the promising results of Sato et. al. [50] come with the qualification that the generated reflectance models capture only a portion of the full reflectance of real materials, which are more fully modeled by the bidirectional reflectance distribution function (BRDF). Second, high-quality ray-traced and radiosity-based renderings can often take hours, or even days, to compute, eliminating the possibility (at least in the near future) of using this rendering method for interactive viewpoint control.

1.1.1 Image-Based Modeling

In light of these observations, recent work has focused on ways of avoiding the complexities of complete, explicit modeling and rendering. These approaches are generally referred to as image-based modeling and rendering. Typically, the goal of these techniques is to create new views of a scene from real images of that same scene. The space of new images is significantly smaller than the potential space of varying all model parameters in a full modeling approach, but is still interesting because it enables at least limited navigation through the environment. In addition, these approaches avoid most of the complexities of modeling physical properties of the scene that give rise to the real images.

One of the simplest and yet surprisingly effective approaches is the object movie of Apple's QuickTime VR [5]. In this strategy, an object is modeled only by the images themselves, with knowledge of the viewpoints from which those images where taken. When a viewer manipulates the object, the system simply displays the image from the closest real viewpoint. The viewer can zoom into the scene as well, giving the appearance of motion into the scene, although perspective effects can not be correctly simulated. A typical object movie has several hundred images, with angular camera separation on the order of 10 degrees.

Most other image-based models usually fall into two categories: ray-based approaches and explicit correspondence-based methods. Ray-based methods deal with images as a set of rays in space. Ray-based methods can be thought of as generalizing the object movie in order to synthesize nearly any viewpoint rather than just switching among the images. First, these approaches convert the set of input images into rays in space, as shown by Katayama et. al [29]. This space can be modeled as a 4D function, commonly referred to as a lightfield [34] or lumigraph [15]. This lightfield is often parameterized by two 3D planes, with a ray defined by its two intersection points on the two planes. Next, the scene is assumed to be flat, and positioned on one of the planes of the lightfield. Using this formulation, new views can be synthesized by sampling the viewing rays that comprise the image. The flat-world assumption allows each unknown ray to be constructed by interpolating among known rays. If the actual object geometry is known, then the rays can be

depth corrected [15], creating much more realistic views, especially when fewer real images are available. Still, these methods have been demonstrated only with thousands of images, which would require a tremendous video capture system for dynamic events.

Correspondence-based methods use pixel-wise correspondences among images as estimates of scene structure, which then allow new view synthesis by mapping this geometry into the new viewpoint[6]. Interpolation of correspondences will not generate physically correct views for general image pairs [53], so knowledge of the epipolar geometry must be exploited. Many methods have used interactive techniques along with epipolar constraints to determine correspondences and generate correct imagery [7][38][54]. Many others have used automatic correspondence detection to varying degrees of success [24][25][33][39][40][56]. These approaches suffer from two problems, though. Some of these methods restrict viewer motion to lie within the 3D surface containing the original viewpoints. More importantly, all of these methods rely on good correspondences: inaccuracies can greatly distort the synthesized views. Stereo methods [12] are able to generate data sufficient for navigation, but generating pixel-accurate, dense correspondences is still out of reach.

1.1.2 Explicit Shape Recovery from Images

The logical extension of correspondence-based methods is to extract full 3D structure from the scene. This approach reworks the correspondence problem in a global framework, allowing the simultaneous use of more information to resolve ambiguity and to increase precision. The range image integration strategies already discussed can be considered a form of this problem. Another approach is to detect and integrated silhouettes, or occluding contours [1][9][58][60]. Immersive Video [22][30][41] has successfully used this approach with a known background. It has been shown, however, that these methods can only extract the line hull of the scene. For example, even perfect silhouettes of a coffee cup would lead to a model with a closed top, since no silhouette could distinguish the inside of the cup.

Debevec and Malik [11] use an interactive modeling system to develop high-quality 3D models of architecture. This system takes advantage of the fact that the scenes contain architecture, allowing the use of relatively few, simple primitives to represent the scene. They then use model-based stereo to refine the alignment between the hand-drawn model and real images of the scene. Seitz and Dyer [55] devised a strategy to test the occupancy of each point in a 3D volume based on color consistency. Each volume element, or voxel, is projected into all the images to get a set of pixel colors. If the variance of these colors is low, then the voxel is assumed to be occupied. By positioning all cameras on one side of the 3D volume, the voxels can be traversed in a front-to-back order so that visibility can be used to eliminate projections that are known to be occluded. Like many correspondence-style methods, this approach is sensitive to specularities and regions of low texture. In addition, mistakes in occupancy at one voxel layer can have a tremendous impact on the results of voxel layers further away.

A different class of 3D model-building methods is shape from motion (SFM). In these methods, usually a single camera is moved through a static environment to create a model of the scene. Because the camera is moving relatively slowly, the inter-frame motion is usually quite small, so feature-tracking algorithms can be used to track features through long camera motions. These features can then be used to estimate shape, for example using the Factorization method [63][46] or non-linear optimization [59]. These approaches usually work only with sparse features, and therefore face the challenge of finding surfaces to connect the feature points.

1.1.3 Dynamic Event Modeling

One way to model dynamic events is to approximate them with sequences of independent static scenes – the same approach used in television and video formats such as NTSC and PAL, which encode each video frame as an independent image and sample at a rate sufficient to create the illusion of motion to human observers. In principle, all of the static scene modeling techniques in the previous section can be extended to model dynamic events in the same way, i.e. by repetition at each temporal sample of the event. Practically,

however, two problems make this extension unwieldy. First, many of these systems rely on (intelligent) human operators to solve some of the most difficult 3D digitization problems. Whatever level of automation that is in these systems simply refines or interpolates from the operator's initialization. For a few static scenes, this interaction is tolerable. For long dynamic events, however, this interaction would be extremely time consuming.

The second problem with extending current approaches for static object modeling is the sensors used in these systems. Many use active range scanners, which illuminate the scene to estimate depth, usually with a laser or light stripes. For dynamic scenes, multiple sensors must operate simultaneously to obtain multiple perspectives of the same scene pose. Without extreme care in system design, the illuminations of these multiple sensors could interfere with one another, and scaling such a system to include more sensors would make the design progressively more difficult. In addition, active scanning samples different parts of the scene at different moments, which can distort the appearance of objects in motion. Since many range scanners requires several seconds of exposure time, the distortion occurs even for slow scene motions. SFM approaches use feature tracking through video sequences captured by camera motion through a fixed scene. To apply these approaches to dynamic scenes, a collection of cameras could be treated as positions of the "moving" camera through the fixed scene of a single image from all cameras. The feature tracking algorithms usually require very small feature motion from frame to frame, however, so the real camera spacing would have to be small enough to simulate inter-frame camera motion – which could easily amount to (tens of) thousands of images.

1.2 Approach

This thesis presents a new technique to recover 3D geometry and appearance of dynamic events from real video sequences that overcomes these problems. This approach combines strengths both of explicit 3D modeling and of image-based rendering. Multi-camera video sequences of dynamic events are used to automatically estimate global scene structure cor-

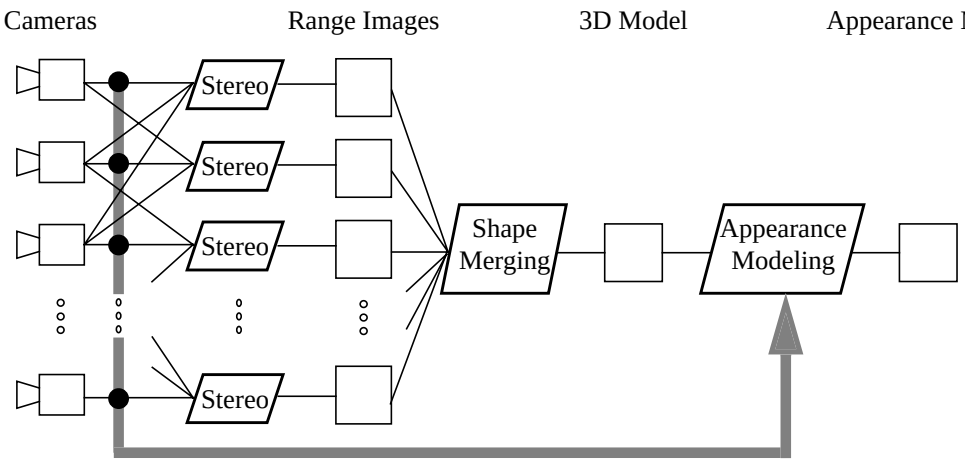
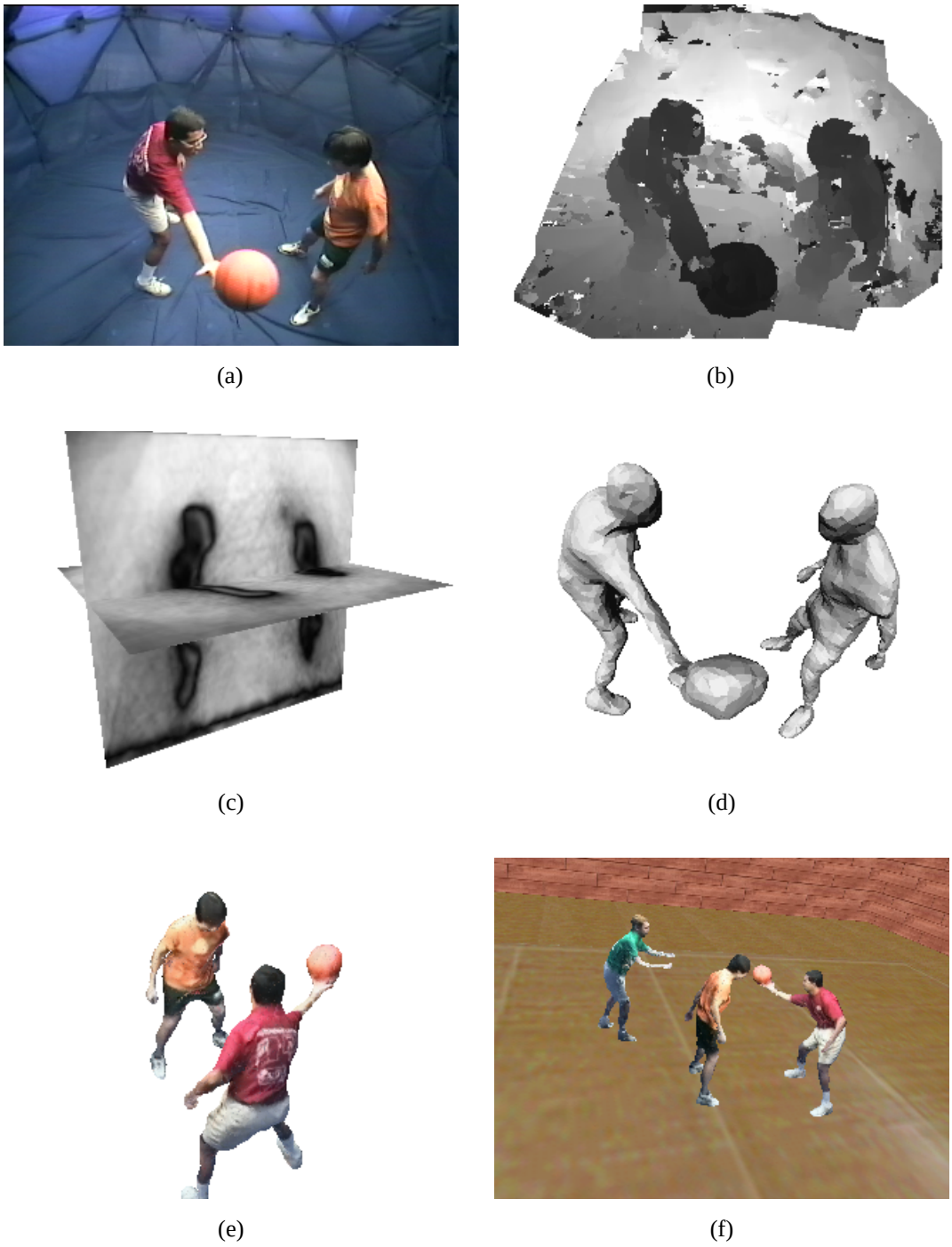


Figure 1.1: Overview of 3D shape and appearance digitization algorithm. Stereo extracts visible surfaces in the set of images captured at one moment. These surfaces are fused to create a single, consistent, global 3D model. The original images are then used to recover an appearance model of the scene. Repeating over time, the algorithm constructs models of dynamic events, which can be used to synthesize arbitrary camera trajectories through the event space.

responding to each frame of video, resulting in a 3D triangle mesh. The real images can be mapped back to this structure using texture mapping to add realistic visual appearance. The technique is repeated over time to capture complex dynamic events.

This shape and appearance digitization algorithm can be decomposed into two steps: first, recover 3D shape, and then estimate scene appearance. Shape digitization itself is decomposed into the estimation of visible surfaces in each video image via stereo, and the merging of these estimates via volumetric integration into a single global 3D model. This process is shown graphically in Figure 1.1. Decomposing shape and appearance helps avoid problems found in voxel coloring, in which mistakes early in the process hinder both shape and appearance modeling. Decomposing shape digitization allows local information to propagate to the global structure in a hierarchical framework, increasing the scope of the modeling process and decreasing noise as the hierarchy is traversed.

An example of this process is shown in Figure 1.2. An example input image for one frame of the event is shown in Figure 1.2(a). The range image corresponding to this view appears in Figure 1.2(b). As with all range images in this thesis, brighter pixels correspond to surfaces further from the camera. Figure 1.2(c) shows slices through the 3D volumetric space created by merging 51 range images from different viewpoints. These slices show how far



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

Figure 1.2: Results of 3D shape and appearance digitization algorithm. (a) example input image. (b) range image computed with stereo (darker = closer to camera). (c) slices through the volumetric space integrating multiple range images (darker = closer to 3D surface). (d) 3D model extracted from volumetric space. (e) rendered view of model with texture. (f) rendered view of model integrated into a virtual set and combined with another digitized model.

each voxel thinks it is from the nearest surface, with black being on the surface and white being far away. The body and arm of the player with the basketball are visible on the left of these planes, while the other player is visible on the right. Figure 1.2(d) shows the 3D triangle mesh surface extracted from the volumetric space, rendered as a shaded surface. Figure 1.2(e) contains a view of the 3D surface model with a global texture map computed from the original images, and Figure 1.2(f) shows a different view of this model integrated with another digitized player and with a virtual basketball court.

1.3 Dissertation Overview

This thesis begins in Chapter 2 by studying how multiple video images can be used to model dynamic events. While the image set can be used directly as a model of the event, this approach requires a dense set of views to simulate realistic viewer motion. Even sparse camera sets can provide much more extensive modeling capabilities when the implicit geometric structure of the event is explicitly extracted from the video images using stereo. With sparse image sets, however, the possibility of not observing an important surface increases, so there are constraints on the sparsity and positioning of the cameras in the work space.

Next, Chapter 3 presents a novel method of fusing the range images computed from stereo into a single, global 3D model. The structure estimated from stereo tends to have highly inaccurate estimates at times, which causes many other model building algorithms to fail. This new approach uses a volumetric integration strategy, which avoids problems of surface topology and mesh stitching while suppressing the inaccuracies typical of correlation-based stereo – poor localization of depth discontinuities and unreliable estimates in regions with little texture.

With the structure computed, Chapter 4 explores two approaches to estimate global scene appearance. The first approach directly uses real images as textures on the global model, preserving view-dependent visual effects captured in the image sequence. The second approach fuses the real video images into a single global texture map. Although this process eliminates any viewpoint-dependent variations in appearance, it still preserves the

overall appearance of the scene. With global models and global texture maps, the motion sequence can even be directly encoded in common 3D file formats such as Open Inventor and VRML 2.0.

Chapter 5 presents ways in which the resulting sequence of digitized 3D shape and appearance models can be used. The most obvious use is to allow the viewer to directly navigate through the digitized event. Because the modeling is fully 3D, it is also possible to combine multiple digitized models into more complex environments, as well as to add purely virtual objects into the digitized model.

Chapter 6 compares the methods developed in this thesis to other related approaches. In particular, we explore image-based modeling and rendering, 3D modeling from range images, and 3D modeling from visual images.

Chapter 7 summarizes the contributions of this work, in addition to looking at future directions for research.

Chapter 2

Recovering Visible Surfaces From Video Images

The first step in digitizing 3D shape and appearance is the estimation of partial scene structure from video images. This structure is encoded as a range, or depth, image for each input video image. Because of (self-)occlusion, range images will contain depth only for the visible structure of the scene. Although the range image contains point samples of depth, grouping these samples into surfaces allows the recovery of the *Visible Surface Model (VSM)*. The first two sections of this chapter address the way this entity is recovered from video images alone.

The concept of a VSM intentionally highlights the importance of visibility in the modeling process. In order to model all surfaces in a real scene, a collection of VSMs may be used, but care must be taken to ensure that each real surface is represented in at least one VSM. If a real surface is unobserved by all VSMs, then this surface will never appear in any views synthesized from the VSMs. Even if a surface is observed in a VSM, it may be severely foreshortened, reducing the quality of the VSM. The last section of this chapter explores the challenges of designing a testbed in light of these visibility constraints.

2.1 Recovery of Visible Surface Models

The Visible Surface Model (VSM) contains depth for every pixel in an image. As already discussed in Section 1.1.3, a variety of methods can be used to estimate depth for static scenes, including laser scanning, light-stripe range finding, and even contact-based digitization. For dynamic events, however, all of these methods fail because of the time required to sample depth of a full scene. The only practical options at this time are methods based on defocus or on stereo. Depth from defocus requires specialized hardware in order to capture multiply focused images at the same viewpoint, though, making a large-scale system costly to implement. Stereo, on the other hand, can use standard video capture equipment.

The most practical approach to range estimation for dynamic events, then, is stereo. The remaining problem is deciding on the best algorithm to use in this context. Given decades of nearly continuous work, the options are extensive – see [12] for a review of many of the possibilities. Feature-based stereo algorithms provide only sparse correspondences, creating a new challenge of grouping features into surfaces and identifying discontinuities. A better alternative in this case is correlation-based approaches, which generate dense correspondences. Within this class of algorithms, the Multi-Baseline Stereo (MBS) algorithm stands out because of its ability to simultaneously search for correspondences in multiple images, reducing ambiguity and increasing precision of depth estimates as compared to typical two-camera (binocular) methods. In addition, the algorithm is relatively fast, as demonstrated by the real-time implementation of Kanade et. al. [26].

The MBS algorithm has been shown to perform better when the scene is actively illuminated by a pattern with high spatial-frequency [28]. This illumination reduces the probability that the images of the scene will contain textureless regions. Since correlation-based methods such as MBS perform poorly in those regions, the illumination therefore improves performance by artificially adding texture to the scene. The drawback of such illumination is that the images no longer accurately represent the appearance of the scene

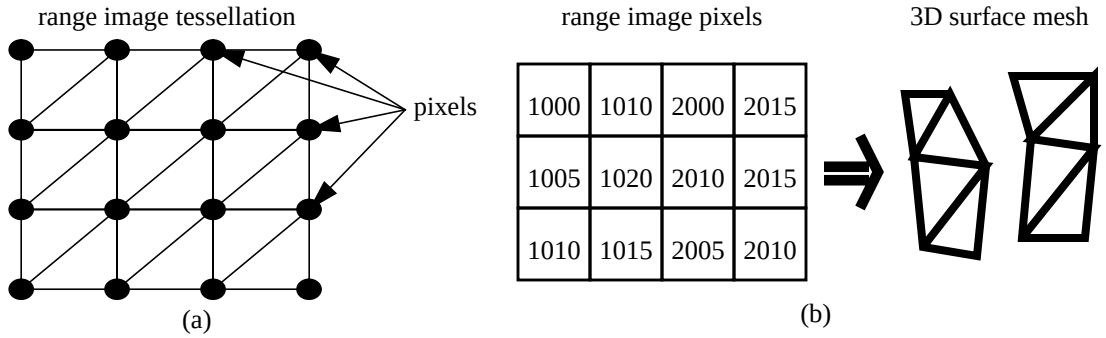


Figure 2.1: Conversion of range image depths to surfaces. (a) Each 2x2 pixel region in the range image is tessellated with two triangles. (b) Tessellation with discontinuities, and the resulting 3D mesh.

under normal lighting. Since the video images are used both for stereo and for appearance modeling, this alteration of scene appearance is unacceptable, so only ambient illumination is used.

2.1.1 Estimation of Surfaces from Dense Range Images

Dense range images computed with stereo (and with many other techniques) actually compute point samples of depth rather than continuous depth across surfaces. To extract surfaces from the range images, we tessellate the range image so that every *2x2* patch of pixels in the range image form two triangles by connecting all vertical and horizontal edges and one diagonal of the patch, as shown in Figure 2.1(a). For simplicity, we always use the same diagonal. Next, we eliminate triangles that have large depth variations. That is, we measure the edge lengths of each triangle edge, and eliminate triangles with an edge longer than a certain threshold, as shown in Figure 2.1(b). A similar style of test would be to eliminate triangles whose surface normals deviate significantly from the direction of the optical axis, as was done in [67].

2.2 MBS: Multi-Baseline Stereo

Stereo algorithms compute depth using triangulation, i.e., by intersecting the imaging rays of corresponding pixels from each camera in 3D. The strong calibration of the cameras facilitates computing the intersection in a common Euclidean (“world”) coordinate system for each pixel, given its correspondence(s) in the other image(s). We use a version of the

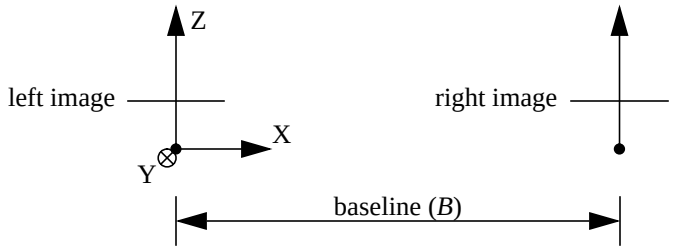


Figure 2.2: Parallel, two-camera geometry. The left camera is the reference, and its axes define the world coordinate system. The distance between the cameras is referred to as the baseline.

multi-baseline stereo algorithm (MBS), developed by Okutomi and Kanade [45], that has been extended to general camera configurations (i.e., non-parallel cameras) by incorporating the Tsai camera model [62].

2.2.1 Parallel, Two-Camera Stereo

To understand the MBS algorithm, first consider a simple stereo system with two parallel cameras. Assume that both cameras are perfect pinhole projectors with the same focal length *F*, with the focal length indicating the distance from the center of projection to the image plane, and is measured in units of pixels. The first camera is positioned at the world origin with its optical axis looking down the positive Z axis and with the X and Y axes running to the right and down, respectively (see Figure 2.2). The second camera is parallel to the first and shifted down the negative X axis by *B*, which is referred to as the baseline between the two cameras. A world coordinate point $\bar{p}$ is then projected into the two images according to

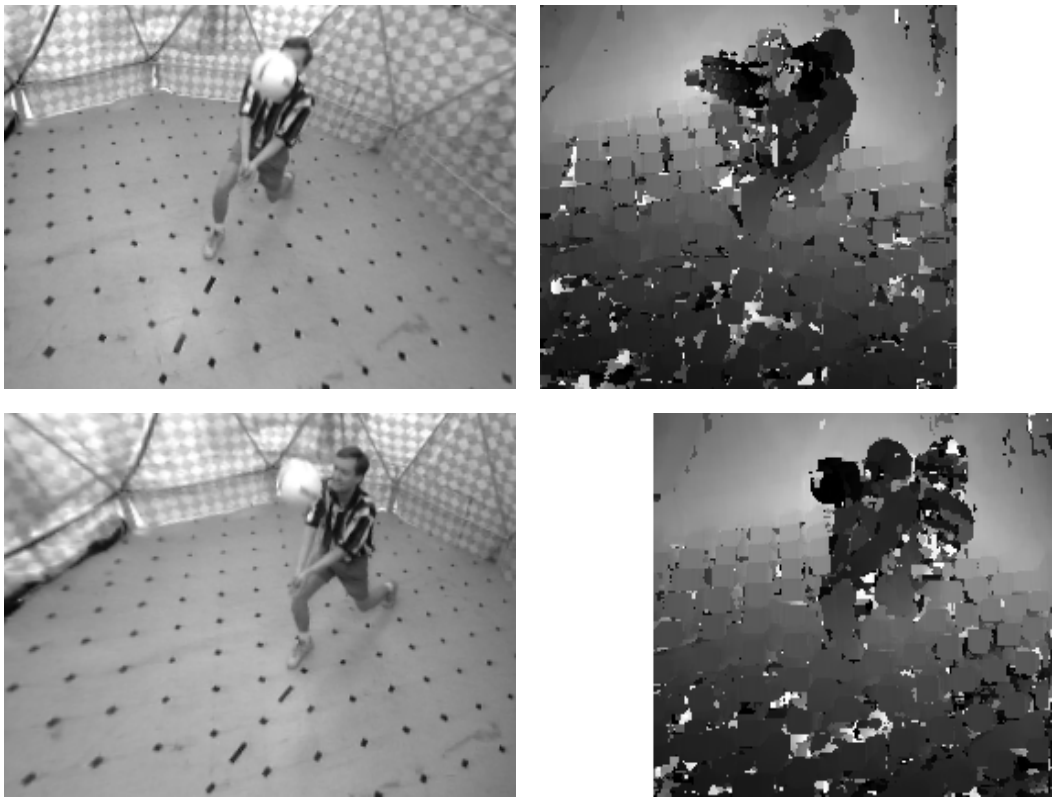
$$\bar{p} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \qquad \bar{q}_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{F}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \qquad \bar{q}_1 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \frac{F}{Z}\begin{bmatrix} X + B \\ Y \end{bmatrix} \tag{2.1}$$

The goal of stereo is to estimate the depth *Z* to the scene point $\bar{p}$ corresponding to each pixel $\bar{q}_0$ in the reference camera. This requires the determination of the corresponding image point $\bar{q}_1$ in the second camera for each pixel in the reference camera. For any pair of corresponding points, the difference between these image points gives the disparity *d*:

$$\bar{q}_1 - \bar{q}_0 = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{F}{Z}\begin{bmatrix} B \\ 0 \end{bmatrix} = \begin{bmatrix} d \\ 0 \end{bmatrix} \qquad \text{where} \qquad d = BF\frac{1}{Z} \tag{2.2}$$

To find corresponding points, stereo searches discrete disparities and selects the disparity yielding the best match (determined using Sum of Squared Differences, i.e. SSD, for example – see Section 2.3.5) between the reference image and the image being searched. Using disparity in this search process guarantees that the image will be evenly sampled, which minimizes the computational costs of the search. In addition, disparity combines the effects of focal length and baseline, so the stereo algorithm can ignore these calibration parameters during the correspondence search. Once a correspondence has been made, depth can be recovered by inverting the depth-disparity relation in Equation (2.2). An example result is shown in Figure 2.3, with two real images shown in (a) and the corresponding range images for each camera in (b). The range images encode nearer surfaces with darker colors. Note that in areas of occlusion, e.g. left of the person in the top view and right of the person in the bottom view, the two-camera stereo algorithm performs poorly because occlusion blocks the non-reference camera's view of the correct correspondence. In addition, the depths of regions with low texture, especially between the dots on the floor, are poorly estimated.

From Equation (2.2), it is clear that increasing the baseline B amplifies the disparity between corresponding image points. Since the correspondence search is performed over discrete values of disparity, increasing the baseline therefore increases the number of disparities that will be searched for a fixed range of depths. As a result, the precision of the estimated distance Z increases as the baseline between the cameras increases. Increasing the baseline, however, also increases the likelihood of matching points incorrectly because of several factors, including the change in perspective and occlusion.



(a) (b)

Figure 2.3: Example of two-camera stereo. (a) Input images. (b) depth maps, with black and white being closest to and furthest from the camera, respectively.

2.2.2 Extension to Multiple Baselines

Multi-baseline stereo attempts to improve matching by computing correspondences between multiple pairs of images, each with a different baseline. Since disparities are meaningful only for a pair of images, we reformulate Equation (2.2) to derive a parameter that can relate correspondences across multiple image pairs:

$$\frac{d}{BF} = \frac{1}{Z} = \varsigma \tag{2.3}$$

The search for correspondences can now be performed with respect to the inverse depth ς, which has the same meaning for all image pairs with the same reference camera, independent of disparities and baselines. The match metric from each individual baseline (e.g. Sum of Squared Differences, or SSD) can then be combined at each inverse depth to create

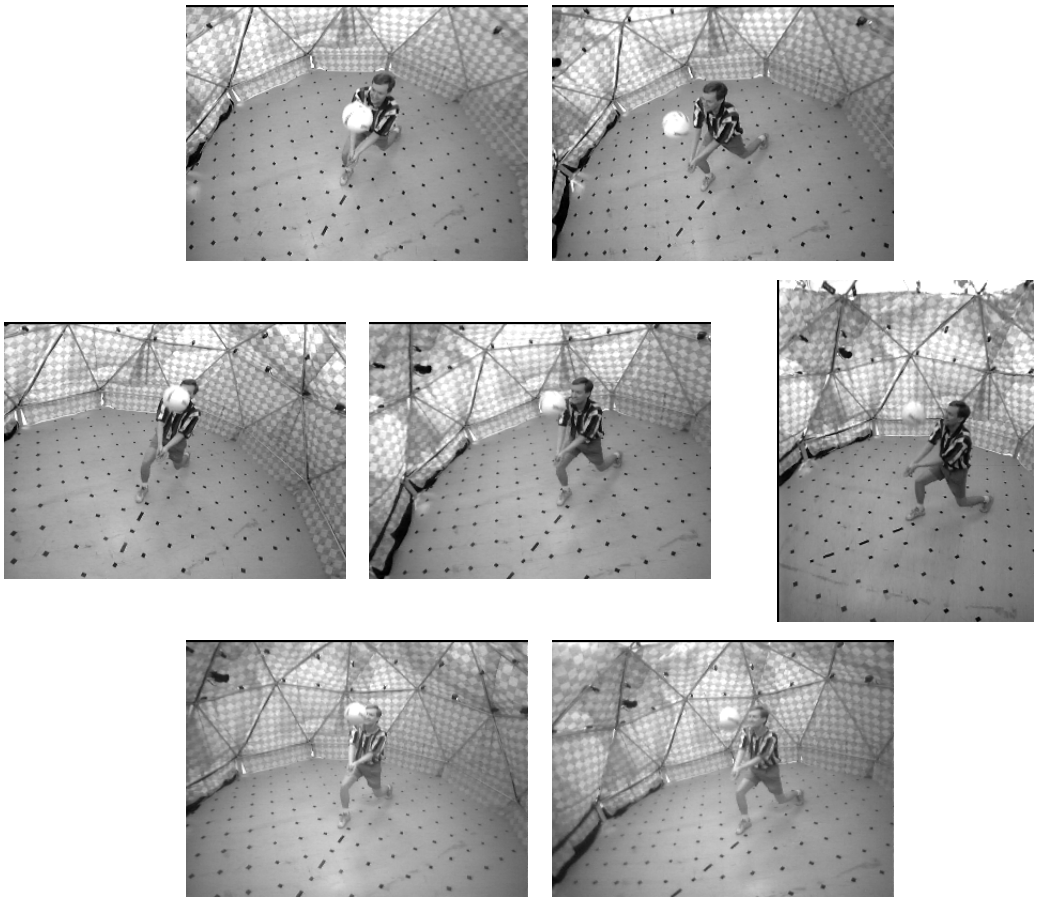
a multi-baseline match metric (e.g., Sum of SSD, or SSSD – see Section 2.3.5). The estimated inverse depth is computed by finding the discrete inverse depth with the best multi-baseline match metric. The resulting correspondence search combines the correct correspondence of narrow baselines with the precision of wider baselines. Note that inverse depth is linear in image space, just as with normal stereo disparity, allowing fast and efficient implementations. An example result of the MBS algorithm is shown in Figure 2.4. The last two the input images are the same ones used in the two-camera example of Figure 2.3. Note how MBS has reduced the number of errors, successfully handling large areas occluded in at least one input image and areas with poor texture.

2.3 GMBS: MBS with General Camera Model

This formulation of multi-baseline stereo is limited to sets of cameras that mutually satisfy the simple parallel-camera geometry. We now extend this algorithm to general camera configurations. In the proposed system, this extension provides greatly needed flexibility in placing the cameras around the environment.

2.3.1 General Camera Model

We assume that a camera can be modeled by 10 parameters: rotation angles (3), 3D translational displacements (3), focal length (1), aspect ratio (1), and image center (2). (Note that we actually use the Tsai camera model [62], described in Appendix A, which includes a parameter for lens distortion. If the lens distortion is first removed, then the following discussion applies here, too.) The perspective projection of a 3D world point $\bar{p}_w$ into image point $\bar{q}$ can then be represented as a series of matrix operations in projective coordinates, with the projective image point q:



(a)

(b)

Figure 2.4: Example of multi-baseline stereo. (a) Input images. (b) depth map corresponding to middle input image, with black and white being closest to and furthest from the camera, respectively

$$\underline{q} = \begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = \begin{bmatrix} aF & 0 & C_x \\ 0 & F & C_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} & \\ R & \overline{T} \\ & \end{array}\right] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = N\underline{P}_w \qquad \bar{q} = \frac{1}{q_3}\begin{bmatrix} q_1 \\ q_2 \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.4}$$

where

$$\underline{P}_w = \begin{bmatrix} \bar{p}_w \\ 1 \end{bmatrix} = \begin{bmatrix} X_w & Y_w & Z_w & 1 \end{bmatrix}^T \qquad N = \begin{bmatrix} aF & 0 & C_u \\ 0 & F & C_v \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} & \\ R & \overline{T} \\ & \end{array}\right]$$

R, a rotation matrix, and $\overline{T}$, a translation vector, relate the world and camera coordinate systems, while the aspect ratio a, focal length F, and image center (C_x, C_y) define the camera-to-image projection. Note that this projection is linearly projective and can be represented by the 3x4 projection matrix N. This model degenerates into the simple projection of Equation (2.1) when there is no rotation ($R = I$), no translation along the Y or Z axes ($T_Y = T_Z = 0$), equal focal lengths across all cameras, a unity aspect ratio ($a = 1$), and an image center (C_x, C_y) at (0,0).

2.3.2 Stereo with General Camera Model

In this general configuration, the disparity-depth relationship is much more complicated than the relationship of Equation (2.3). To derive this expression, and without loss of generality, we assume camera 0 is the reference camera, with its coordinate system defining the world coordinate system. For simplicity we also assume that $C_x = C_y = 0$ and $a = 1$ for all cameras. From Equation (2.4), the disparity is given by

$$\begin{aligned} \bar{q}_1 - \bar{q}_0 &= \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{F_1}{Z_{C,1}}\begin{bmatrix} X_{C,1} \\ Y_{C,1} \end{bmatrix} - \frac{F_0}{Z_{C,0}}\begin{bmatrix} X_{C,0} \\ Y_{C,0} \end{bmatrix} \\ &= \frac{F_1}{(r_3\bar{p}_w) + T_Z}\begin{bmatrix} r_1\bar{p}_w + T_X \\ r_2\bar{p}_w + T_Y \end{bmatrix} - \frac{F_0}{\overline{Z}}\begin{bmatrix} X \\ Y \end{bmatrix} \end{aligned} \tag{2.5}$$

where $\bar{r}_i$ is the row vector corresponding to row i of the rotation matrix R. This general form can be simplified if we consider that a stereo search keeps the reference pixel ($\bar{q}_0$) fixed. The 3D point $\bar{p}_w$ can then be represented as

$$\bar{p}_w = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{Z}{F_0}\begin{bmatrix} x_0 \\ y_0 \\ F_0 \end{bmatrix} = \left(\frac{Z}{F_0}\right)\bar{p}_{w0} \qquad \text{where} \qquad \bar{p}_{w0} = \begin{bmatrix} x_0 \\ y_0 \\ F_0 \end{bmatrix} \tag{2.6}$$

Combining Equations (2.5) and (2.6) yields

$$\begin{aligned} \bar{q}_1 - \bar{q}_0 &= \frac{F_1}{(\bar{r}_3\bar{p}_{w0})\frac{Z}{F_0} + T_Z}\begin{bmatrix} \bar{r}_1\bar{p}_w + T_X \\ \bar{r}_2\bar{p}_w + T_Y \end{bmatrix} - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} \\ &= \frac{1}{Z(\bar{r}_3\bar{p}_{w0}) + F_0T_Z}\begin{bmatrix} Z(F_1\bar{r}_1\bar{p}_{w0} - x_0\bar{r}_3\bar{p}_{w0}) + F_0(F_1T_X - x_0T_Z) \\ Z(F_1\bar{r}_2\bar{p}_{w0} - y_0\bar{r}_3\bar{p}_{w0}) + F_0(F_1T_Y - y_0T_Z) \end{bmatrix} \end{aligned} \tag{2.7}$$

The right-hand side of Equation (2.7) is now only a function of depth (Z), since all other variables are fixed during the search for a given reference image pixel, so we rewrite it as

$$\bar{q}_1 - \bar{q}_0 = \frac{1}{K_1Z + K_2}\begin{bmatrix} K_3Z + K_4 \\ K_5Z + K_6 \end{bmatrix} \tag{2.8}$$

where the terms K_i are constant for a given reference image pixel $\bar{q}_0$. This function can easily be evaluated as a function of depth. However, unlike the multi-baseline, parallel-camera configuration of Section 2.2.2, it is not possible to linearly search all the images as a function of inverse depth while also directly relating the evaluations across all cameras, because the constants K_i vary from camera to camera. As a result, the search must be performed as a function of depth, with non-uniform sampling in the images. This method is less efficient than directly searching in image space because the sampling will be finer than necessary in some areas in order to maintain consistency with other cameras. There is also the danger of sampling too little, in which case the actual corresponding point may be

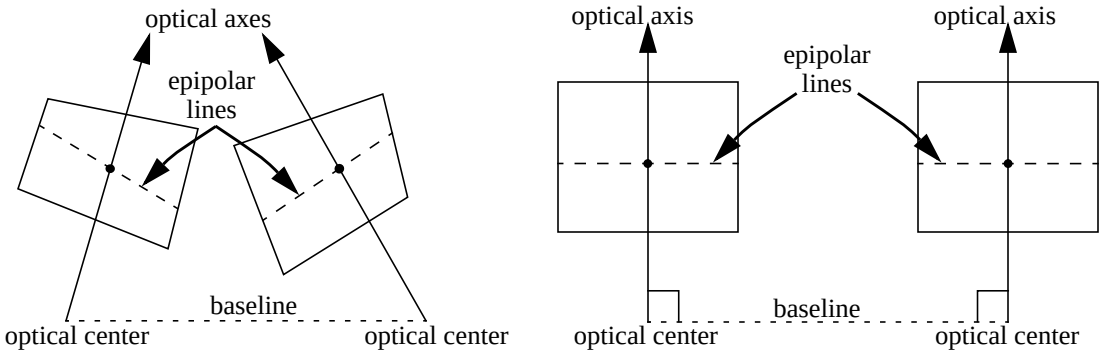


Figure 2.5: Rectification of two cameras. (a) input camera geometry. (b) rectified camera geometry. The cameras are rotated about their optical centers until the optical axes are parallel to one another and perpendicular to the baseline joining the optical centers. The scale factors (focal length, aspect ratio) in each camera are also forced to match.

missed. The search can be performed linearly in the images if the error functions are then resampled to a common depth sampling. This method is more efficient, but requires the interpolation of the error functions, which may be inaccurate away from the true solution.

2.3.3 Rectification

The solution to this complication relies on rectifying the input images so that they satisfy the parallel-camera configuration. In rectification, two input images are resampled to obtain images that correspond to cameras having parallel-camera geometry, as shown in Figure 2.5. This operation can alter both the intrinsic and extrinsic camera parameters under the constraint that the optical centers of the cameras must remain fixed. Physically, rectification can be thought of as rotating two cameras so that their optical axes are parallel to each other and perpendicular to the line (baseline) joining their optical axes. This rotation guarantees that epipolar lines will be horizontal everywhere in the images. (Note that for general cameras, this rotation is underconstrained. See Appendix C for details on how we sufficiently constrain the problem.) To force the epipolar lines to lie on scan lines, the focal length and aspect ratio must be set identically in the two images. Even with these constraints, the image center is free to change, which is helpful to ensure maximal overlap between the input and rectified images. Without this change, the field of view in the input may have little overlap with the rectified image.

The rectification process can also be viewed as a homography which projectively maps pixels in the input image to pixels in the rectified image. To see this, consider the mapping of a single pixel *(x,y)* in the input image into the rectified image. For this purpose, let the unrectified camera define the world coordinate frame, generating the camera model

$$\underline{q} = \begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = \begin{bmatrix} aF & 0 & C_x \\ 0 & F & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I \mid \bar{0} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} = A\bar{p}_w \qquad \text{and} \qquad \bar{p}_w = A^{-1}\underline{q} \tag{2.9}$$

$$\text{where} \qquad A = \begin{bmatrix} aF & 0 & C_x \\ 0 & F & C_y \\ 0 & 0 & 1 \end{bmatrix}$$

Rectification allows arbitrary rotations and changes to the intrinsic parameters of the camera, but requires that the image center remain fixed, i.e. $\overline{T} = \overline{0}$. The rectified camera model is therefore given by

$$\underline{q}' = \begin{bmatrix} x'w' \\ y'w' \\ w' \end{bmatrix} = \begin{bmatrix} a'F' & 0 & C'_x \\ 0 & F' & C'_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} & | & \\ R' & | & \bar{0} \\ & | & \end{bmatrix} \begin{bmatrix} \bar{p}_w \\ 1 \end{bmatrix} = \begin{bmatrix} a'F' & 0 & C'_x \\ 0 & F' & C'_y \\ 0 & 0 & 1 \end{bmatrix} R'\bar{p}_w = S\bar{p}_w \tag{2.10}$$

$$\text{where} \qquad S = \begin{bmatrix} a'F' & 0 & C'_x \\ 0 & F' & C'_y \\ 0 & 0 & 1 \end{bmatrix} R'$$

Combining Equation (2.9) with Equation (2.10) yields

$$\underline{q}' = S\bar{p}_w = SA^{-1}\underline{q} = H\underline{q} \qquad \text{where} \qquad H = SA^{-1} \tag{2.11}$$

Thus, the matrix H is a 3x3 homography mapping the pixels in the unrectified image to the pixels in the rectified image.

In general, it is possible to rectify only two cameras at a time if the goal is strict parallel-camera, horizontal disparity rectification. If vertical disparity is allowed, then it is always possible to rectify three cameras by rotating all three optical axes to be perpendicular to

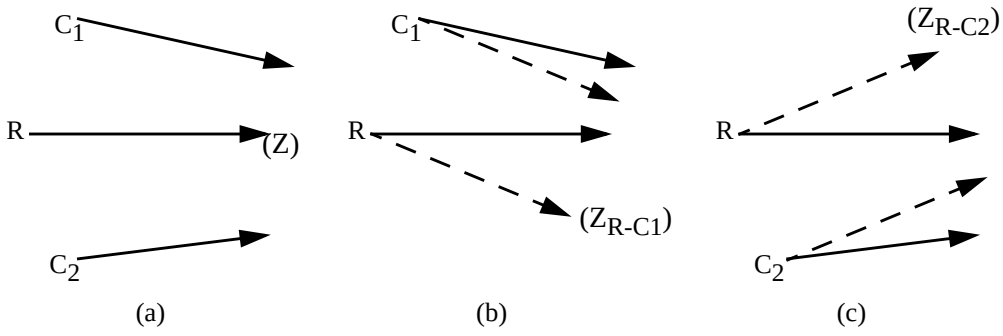


Figure 2.6: Rectification of a set of three cameras. (a) input camera geometry, with arrows indicating optical axes of the cameras and tails indicating camera positions; (b) rectification of the top camera with the reference camera; (c) rectification of the bottom camera with the reference cam-

the plane containing the optical centers of the three cameras. If the cameras fall into certain degenerate configurations (e.g., all the optical axes fall on a plane in space) then more than three cameras can be rectified, although these cases rarely occur in practice.

To deal with multiple cameras in arbitrary positions, we pairwise rectify each non-reference camera with the reference camera. This gives us a new set of non-reference cameras and corresponding images that have the simple parallel-camera geometry with respect to rectified versions of the reference camera. Figure 2.6 shows an example involving three cameras. Note that this process could create multiple versions of the reference image – in fact, there would be as many versions as there are input images in the stereo set. Rather than computing all of these versions, we instead compute only the rectification parameters and simply keep the original reference image. We then use the rectification parameters to relate the input and rectified reference image coordinate systems.

2.3.4 GMBS with Rectification

During each correspondence search, the reference pixel in the input reference image is mapped to a rectified non-reference camera by first computing the pixel position in the corresponding rectified reference camera, and then using the simple parallel-camera geometry of the rectified pair and the (inverse) depth being tested to map from the rectified reference camera to the rectified non-reference camera. Note that the depth itself must

be transformed too, because the change in orientation of the reference camera changes the way depth is measured because the optical axis rotates, as shown in Figure 2.6. This process is repeated for each of the test disparities (inverse depths).

Algorithm: Naive GMBS Search

```
for each pixel (row,col) in reference image
    bestzed [row,col] = 0;
    for each inverse depth zed
        sum[zed] = 0;
        for each camera cam
            (rectrow, rectcol) = unrect_to_rect (row,col,cam)
            rectzed = unrect_to_rect_zed (row,col,cam,zed)
            testrow = rectrow
            testcol = rectcol + rectzed * B_cam
            sum[zed] += comparison {I_ref[row,col], I_cam[testrow,testcol]}
        end
        if (sum[bestzed[row,col]] is worse than sum[zed])
            bestzed[row,col] = zed
        endif
    end
end
```

To speed up this process, we observe that the parallel-camera geometry can be exploited to avoid the lengthy transformation from input reference image to rectified non-reference image. The simple geometry allows us to step uniformly through the non-reference images without the need for repeated pixel transformations. In order to exploit this property, however, we must account for the distortions introduced by rectification, such as rotation.

First, recall that the rectification of one camera with another introduces a rotation of the optical axis, but the optical center is fixed. This 3D transformation can be represented with a single rotation matrix M which relates coordinates between the unrectified and rectified versions of a given camera. For the reference camera, and using Equation (2.6), the rectified coordinates p_w^r are given by

$$\bar{p}_w^r = \begin{bmatrix} X^r \\ Y^r \\ Z^r \end{bmatrix} = M\bar{p}_w = M\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = M\left(\frac{Z}{F}\bar{p}_{w0}\right) = \left(\frac{Z}{F}\right)M\bar{p}_{w0} \tag{2.12}$$

The rectified camera pair is in the standard parallel-camera configuration, so we can apply Equation (2.2):

$$d = B^r F^r \frac{1}{Z^r} = \left(\frac{B^r F^r F}{\bar{m}_3 \bar{p}_{w0}} \right) \frac{1}{Z} = C \frac{1}{Z} \qquad \text{where} \qquad C = \frac{B^r F^r F}{\bar{m}_3 \bar{p}_{w0}} \tag{2.13}$$

Equation (2.13) relates the depth Z in the unrectified reference camera to disparity in the rectified image pair. This expression has the same form as Equation (2.2), and is identical if $\bar{m}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$, i.e. the optical axis of the reference camera is already perpendicular to the baseline between the two cameras so rectification does not change this axis. As before, regrouping to isolate inverse depth ζ enables us to relate the search spaces across multiple images. This algorithm is the one that actually computed the range image in Figure 2.4.

Algorithm: Efficient GMBS Search

```
for each pixel (row,col) in reference image
    bestzed [row,col] = 0;
    for each camera cam
        (rectrow[cam], rectcol[cam]) = unrect_to_rect (row,col,cam)
    end
    for each inverse depth zed
        sum[zed] = 0;
            testrow = rectrow[cam]
            testcol = rectcol[cam] + zed * C_cam
            sum[zed] += comparison {I_ref[row,col], I_cam[testrow,testcol]}
        end
        if (sum[bestzed[row,col]] is worse than sum[zed])
            bestzed[row,col] = zed
        endif
    end
end
```

2.3.5 Practical Correspondence Detection

A popular method to compute correspondences between a pair of images is to compare a small window of pixels from one image to corresponding windows in the other image. The matching process for a pair of images involves shifting this window as a function of ς and computing the match error, usually with normalized correlation[1] (NC) or sum of squared differences (SSD), over the window at each position. For two images I_0 and I_i, and for image point (u_0, v_0), the SSD and NC are given by

$$SSD_i(x_0, y_0, \varsigma) = \sum_{x=-w}^{w} \sum_{y=-w}^{w} [I_0(x_0 + x, y_0 + y) - I_i(x_0 + x + \varsigma B_i F, y_0 + y)]^2$$

$$\nabla C_i(x_0, y_0, \varsigma) = \sum_{x=-w}^{w} \sum_{y=-w}^{w} \frac{[I_0(x_0 + x, y_0 + y) - \mu_{I_0}][I_i(x_0 + x + \varsigma B_i F, y_0 + y) - \mu_{I_i}]}{\sigma_{I_0}\sigma_{I_i}} \quad (2.14)$$

where μ_{I_k} and σ_{I_k} are the mean pixel value and the variance of the pixel values within the window of size (2w+1) x (2w+1) centered at the image point (x_0, y_0) in image I_k. B_i is the baseline between the reference camera (which generates image I_0) and the second camera (which generates image I_i). MBS computes the sum of SSDs (SSSD, or the sum of NCs, SNC) across all cameras:

$$SSSD(x_0, y_0, \varsigma) = \sum_{i=1}^{N_{cameras}} SSD_i(x_0, y_0, \varsigma) \quad (2.15)$$

The depth estimate $\hat{\varsigma}$ is the ς that minimizes this error:

$$\min_{\varsigma} \; SSSD(x_0, y_0, \varsigma) \quad (2.16)$$

This matching approach makes two important assumptions about the scene. First, it assumes that the intensity variation (i.e., texture) within the window will be strong enough for the search to produce reliable results. Second, it assumes that there is no depth variation within the reference window – that is, that the world is flat within the window. In image regions with low texture, the depth estimate will have low precision because the window will usually match equally well over a wide range of depths. Along depth discontinuities, the planarity assumption breaks down because the reference window includes pixels at two (or more) different depths. Because of perspective projection, these 3D points will no longer lie next to one another in images from other viewpoints, and changes in visibility may cause some of the 3D points to be occluded. As a result, the SSSD function may not have a minimum at the correct depth. Even if a local minimum exists at the

1.Normalized correlation actually computes similarity, not error. We use the additive inverse of NC as an error measure. NC will outperform SSD when the images have large differences in intensity across cameras; otherwise, SSD will perform at least as well.

correct depth, other local minima will occur at the depths of the other 3D points represented within the window, and none of these minima will be as strong as when the planarity assumption is satisfied. As a result, the global minimum may lie at some depth for which random alignment of the image textures happens to generate the lowest error. Part of the motivation to develop the global integration strategy in Chapter 3 was to overcome some of these problems, which are more easily addressed in a global framework.

2.4 Scene Modeling with Visible Surface Models

One key constraint in modeling a scene from images is ensuring that the images contain observations of all (relevant) scene structure. If an important part of the scene is not observed in the set of images, then there is no way to reconstruct that feature without exploiting content-specific knowledge. Because of the possibility of (self-) occlusion, though, this constraint can never be fully met for an unknown scene. That is, for a given configuration of cameras, it is always possible to construct a scene in which occlusion prevents the observation of part of the scene. In addition, increasing the number of cameras will help reduce the problem but will also drive up system cost. We therefore seek a camera system that does a reasonable job of balancing the need for many viewpoints with the desire for minimizing system costs.

Numerous other factors play into the design of a real camera system. For example, in order to relate the video of one camera to the video of another, the cameras must be synchronized (i.e., exposures occur at the same time) and time-stamped. In order to apply stereo, or any other 3D extraction process, the cameras must be calibrated so that a consistent mapping between world and image and among different images is known. Stereo also performs more accurately when the distance between cameras is small, which reduces the likelihood of mismatching during correspondence estimation. And a lighting system must be designed so that the scene is well lit but not “blinding” the cameras. For a more thorough discussion of these issues, see Appendix B.

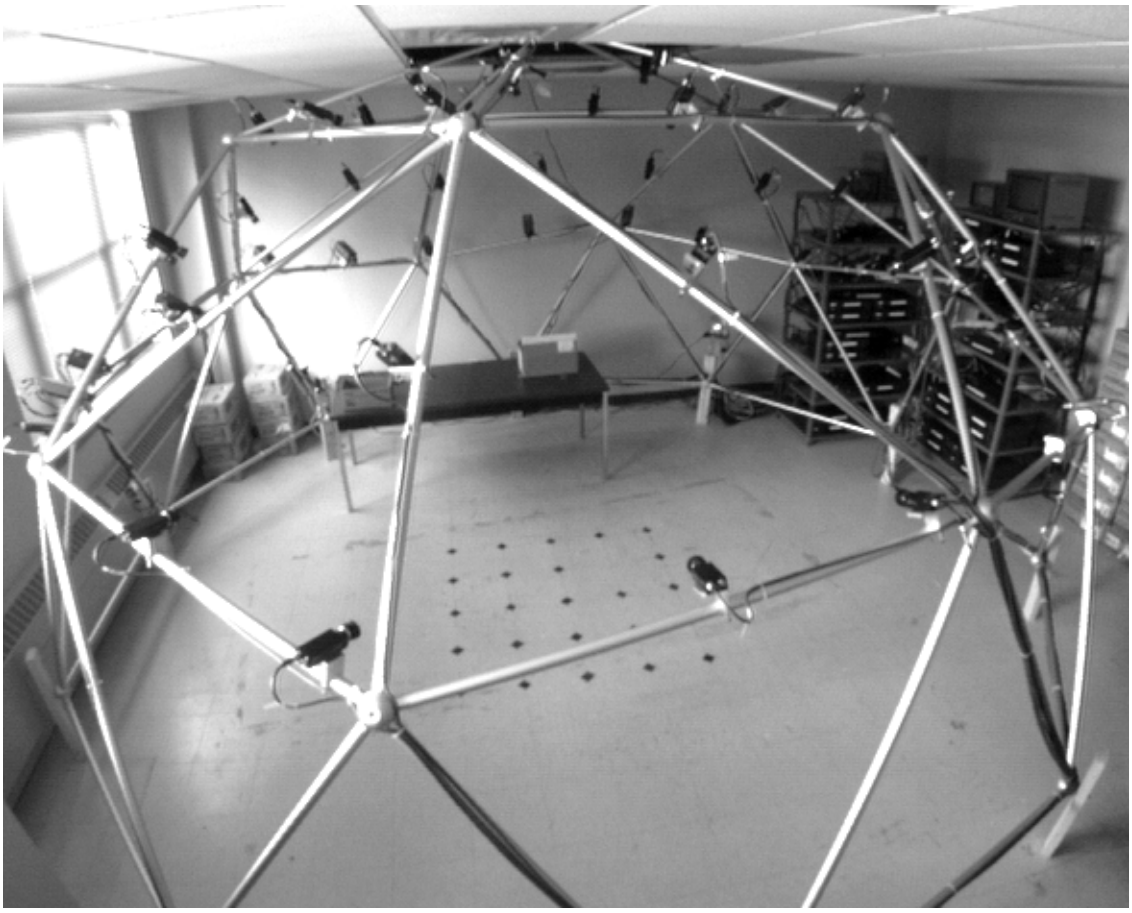


Figure 2.7: Camera system. 51 video cameras are mounted on a 5-meter diameter geodesic dome.

The solution taken in this work is to disperse 51 cameras over a geodesic dome 5 meters in diameter, providing nearly a hemisphere of viewpoints surrounding the workspace, as shown in Figure 2.7. This system provides a roughly cubical working volume, 2 meters on a side, sufficient to a allow a person to perform athletic actions such as swinging a baseball bat. From the center of the dome, the cameras have roughly uniform angular separation. This uniformity was chosen because it minimizes the possibility of occlusion for an unknown scene. The angular separation between cameras is roughly 18^{o}. Based on empirical analysis, this separation is small enough to avoid significant occlusion for many events.

Chapter 3

Global Structure From Partial Surfaces

A Visible Surface Model (VSM) extracted from video images represents the 3D scene surface geometry visible in the corresponding video image. Collections of VSMs represent full scene geometry without an explicit, global 3D model. The VSM structure extraction process, however, has significant limitations because it operates only on local information, in which only a few images with similar viewpoints are available. As a result, surface estimates tend to be rough and at times incorrect. This chapter presents a method for overcoming these problems by transforming the surface recovery problem from a local framework to a global one, using many images from highly separated viewpoints. By expanding the focus, this strategy recovers complete, global scene structure rather than the local structure of the VSMs. We call this structure the *Complete Surface Model*, or *CSM*.

Moving to global structure recovery may introduce new problems, however. Intuitively, one might think of stitching the range images together like a quilt to form the global surface, since each range image provides one patch in the global surface. This approach has been attempted a number times, with reasonable success on relatively low-noise data

[57][61][65]. When the ranges estimates are noisy, however, the surface patches do not align well, leading to a bigger problem: estimating the underlying topology of the scene. Depending on the expected topology of the scene, the surface patches can fit together in many ways, all with approximately similar errors. In addition, with unknown topology, it is difficult to integrate multiple samples of the same surface, which reduces the effectiveness of using multiple observations to reduce noise.

This chapter addresses the global surface construction problem by working with a volumetric representation rather than with the surfaces themselves. This framework virtually eliminates the problem of topology while supporting a simple method of integrating multiple estimates to reduce noise. In addition, surfaces can easily be extracted from the volumetric space with little degradation in geometric quality.

3.1 Volumetric Fusion

One of the first (and simplest) volumetric structures used in computer vision is the occupancy grid [9]. This structure decomposes a volume into solid volume elements, or voxels. Each voxel contains a single binary value indicating whether or not the voxel is occupied. When used to represent object shape, the occupancy grid suffers from the strict binary quantization of the real surface. This effect diminishes as the density of voxels increases, but the quantization will always be present at some resolution.

A more powerful volumetric representation can be achieved by changing the information stored at each voxel. Rather than storing a binary value, each voxel can store the (point-sampled) value of a function $f(x,y,z)$ defined over the volume. An occupancy grid can be computed from this volume by thresholding the voxels at some level f_0, but this will still suffer from all the problems of the occupancy grid. Lorensen and Cline [35] observed that an alternative approach is to extract the isosurface of the volume defined by the same threshold, i.e., $f(x,y,z) = f_0$. The advantage of this approach is that the position of the isosurface can be estimated to sub-voxel precision, creating smoother models than are possible with an occupancy grid at the same resolution.

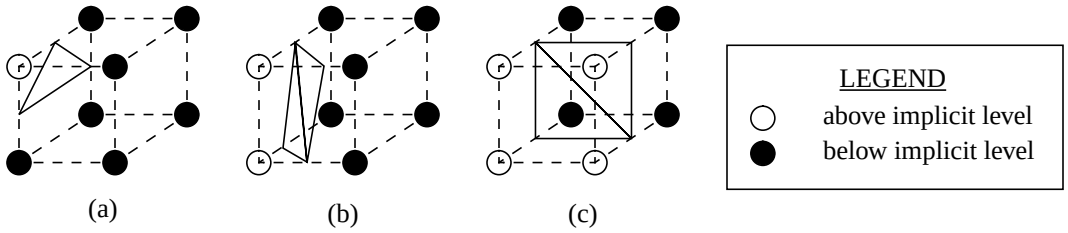


Figure 3.1: Example of Marching Cubes strategy. Each voxel is treated as a point sample of an implicit function in space, shown graphically as a circle. When the voxels are aligned on a regular grid, a 2x2x2 set of voxels encloses a cube of space. (a) When the value of one voxel is above the implicit surface level and all others in the cube are below this level, then the surface intersects the cube through the 3 cube faces containing the voxel above the implicit surface level. (b) Two neighboring voxels are above the level, all others are below. (c) 4 neighboring voxels are above, the other 4 are below.

The other important observation they made was that the isosurface can be quickly and easily extracted from sampled volumes by “marching” through the volume in a single pass, which led to the name of the algorithm: Marching Cubes [35]. By approximating the isosurface with a 3D polygonal mesh, the location of a surface is dependent only on the local information in the volumetric space. By treating the voxels as point samples of the function on a 3D grid, the presence of an isosurface in a particular part of the volume can be determined directly from the 8 voxels which lie at the corners of a cube surrounding the volume being tested. If all voxels are above or below the isosurface level, then the isosurface can not pass through this space. If one voxel of the cube lies above (or below) the isosurface level while all others lie below (or above) it, then the isosurface must slice through just the one corner of the cube, separating this one voxel from the other 7 voxels, as shown in Figure 3.1(a). Two other cases are shown in Figure 3.1 (b) and (c), corresponding two possible configurations of two and four voxels above (or below) the isosurface level, respectively. Since each corner takes on a binary value (i.e., either above or below the level), $2^8 = 256$ different combinations exist, although many of these are symmetric. For each configuration, the tessellation can be pre-computed, with the actual vertices computed by interpolating the voxel values along each cube edge intersecting the isosurface.

The key insight from volumetric methods of range image fusion, including this chapter, is that the surface recovery problem can be posed as the estimation of an implicit surface in 3D, followed by implicit surface extraction with Marching Cubes to return to a surface description [10][19][20][47][67] ([36][37] integrate in a volumetric space, but do not

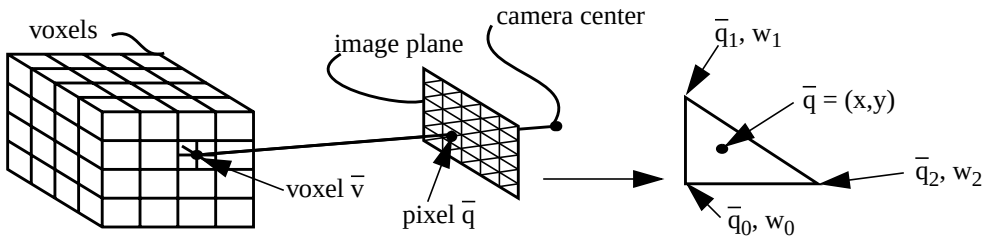


Figure 3.2: Basic operation in computing signed distance. Each voxel is projected into the image plane of each camera using the camera models already computed for stereo. The depth of the voxel can be extracted from this process. The range image is interpolated to compute the depth from the camera to the surface. The signed distance from the voxel to the surface is computed by subtracting these depths.

extract the isosurface). In this strategy, voxels accumulate the signed distance to the observed surfaces in the range images. This implicit encoding of the surfaces overcomes the problems of topology that have plagued surface-stitching algorithms. In addition, the volume approach easily accumulates multiple estimates of surface position to reduce noise in the final structure.

3.2 Computation of Signed Distance

In order to implement this general volumetric approach, the mapping from range images to signed distance must be specified. Treating voxels as 3D points, we define the signed distance function $f(\bar{v})$ at a voxel $\bar{v} = (X,Y,Z)$ to be the difference between the depth $D_{\bar{v}}$ of the voxel from the viewpoint of the VSM and the depth D_{VSM} of the VSM surface that lies along the viewing ray passing through the voxel $\bar{v}$:

$$f(\bar{v}) = D_{\bar{v}} - D_{VSM} \tag{3.1}$$

The signed distance $f(\bar{v})$ is computed in three steps (see Figure 3.2). First, the voxel depth $D_{\bar{v}}$ is computed by projecting the voxel $\bar{v}$ into the image plane of the VSM using Equation (2.4) (reproduced here for convenience), with $\underline{q}$ representing the projective image coordinate position of the current voxel:

$$\underline{q} = \begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = \begin{bmatrix} aF & 0 & C_x \\ 0 & F & C_y \\ 0 & 0 & 1 \end{bmatrix} \left[\begin{array}{c|c} & \\ R & T \\ & \end{array}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = N\bar{v} \tag{3.2}$$

$$D_{\bar{v}} = w$$

The third component of $\underline{q}$, w, gives the voxel depth $D_{\bar{v}}$ in the camera coordinate system. Next, the depth of the VSM surface that lies along the viewing ray intersecting the voxel $\bar{v}_w$ is computed by observing that the voxel and the surface both project to the same image point $\bar{q} = (x,y)$. The depth D_{VSM} can then be interpolated from the tessellation of the range image discussed in Section 2.1. Assume that the image points $\bar{q}_i=(x_i,y_i)$, i = 0,1,2 are the pixel coordinates of the facet of the tessellation on which $\bar{q}$ projects, as shown in the right part of Figure 3.2, and that the depth at pixel $\bar{q}_i$ is w_i. Defining plane containing the triangular facet in terms of the image coordinates, i.e.,

$$Z = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.3}$$

allows the plane parameters to be estimated directly from known vertices:

$$\begin{bmatrix} w_2 & w_1 & w_0 \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x_2 & x_1 & x_0 \\ y_2 & y_1 & y_0 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} a & b & c \end{bmatrix} M$$
$$\begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} w_2 & w_1 & w_0 \end{bmatrix} M^{-1} \tag{3.4}$$

By careful selection of M, the inverse can be computed with little effort. To make this selection, translate the image coordinates so that the point (x_0,y_0) it at the origin. (x_1,y_1) and (x_2,y_2) now correspond to (0,1) and (1,0), respectively:

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad M^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix} \tag{3.5}$$

The plane equation is therefore given by

$$\begin{bmatrix} a & b & c \end{bmatrix} = \begin{bmatrix} w_2 - w_0 & w_1 - w_0 & w_0 \end{bmatrix} \tag{3.6}$$

This plane can then be projected back into the image. By using the constraints from the known projective pixel coordinates q_i, the VSM depth is determined to be

$$D_{VSM} = \frac{w_0}{1 - \left(\frac{w_2 - w_0}{w_2}\right)x - \left(\frac{w_1 - w_0}{w_1}\right)y} \tag{3.7}$$

where (x,y) is now the image location relative the known point (x_0,y_0). Finally, the signed distance $f(\bar{v})$ is computed by applying Equation (3.1).

3.2.1 Accumulating Signed Distance Across Multiple VSMs

In order to integrate estimates from multiple VSMs, each voxel accumulates the signed distance relative to each VSM. If the voxel values are called $V(\bar{v})$, then

$$V(\bar{v}) = \sum_{i=1}^{N_{\text{cameras}}} f_i(\bar{v}) = \sum_{i=1}^{N_{\text{cameras}}} [D_{\bar{v}_i} - D_{VSM_i}] \tag{3.8}$$

If the estimated structure in the VSMs have unequal reliability, the accumulation can weight the signed distance by that confidence, possibly with a different weight for each 3D point in every VSM (see Figure 3.3(a)):

$$\begin{aligned} g_i(\bar{v}) &= w_i(x,y)\, f_i(\bar{v}) \\ V(\bar{v}) &= \sum_{i=1}^{N_{\text{cameras}}} g_i(\bar{v}) \end{aligned} \tag{3.9}$$

This accumulation process, repeated for each voxel and for each VSM, converts the explicit geometry of the individual VSMs into the zero-level implicit surface embedded in the volume (see Figure 3.3(b)). The surface geometry is then recovered by extracting this

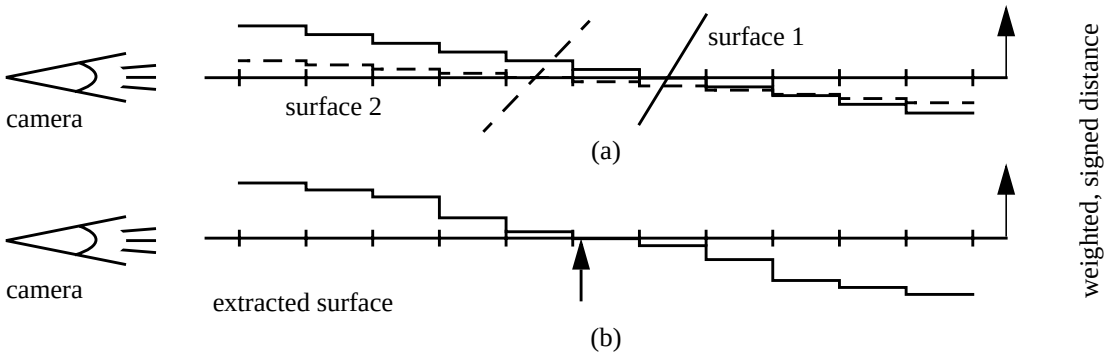


Figure 3.3: 1D example of volumetric fusion. (a) Two surfaces are added to the volume. Surface 1 has higher weight and therefore contributes more to the final result. (b) Final surface is at the zero crossings of accumulated values in the voxels.

implicit surface with an algorithm such as Marching Cubes. (Our actual implementation is based on freeware implementation of the implicit surface extractor of Bloomenthal [4], modified to support the fast marching process.)

3.2.2 Limiting Influence of Individual VSMs

While this approach can be directly applied without concern for the absolute distance between a voxel and the VSM surface, it is usually better to divide the volume into three classes for each VSM based on their visibility from that view: empty, near-surface, and occluded. Empty voxels lie in the viewing frustum of the of the VSM between its origin (viewpoint) and the closest surface contained in it, corresponding to negative values of $f(\bar{v})$. The voxels in the near-surface volume are within some threshold distance (without sign) from the surface ($-\varepsilon<f<\varepsilon$, $\varepsilon>0$). Finally, beyond the near-surface volume lies the occluded volume, in which voxels are hidden from view of the VSM by the VSM surface (positive $f(\bar{v})$).

Given these three classes of voxels, the accumulation algorithm can now be more precise in updating the voxel space. Near-surface voxels are updated using the signed distance functions just described. For occluded voxels, however, the update is skipped because they may lie on a real surface occluded from the view of the VSM under consideration. Finally, empty voxels must be updated in some way in order to reject incorrect surfaces with the

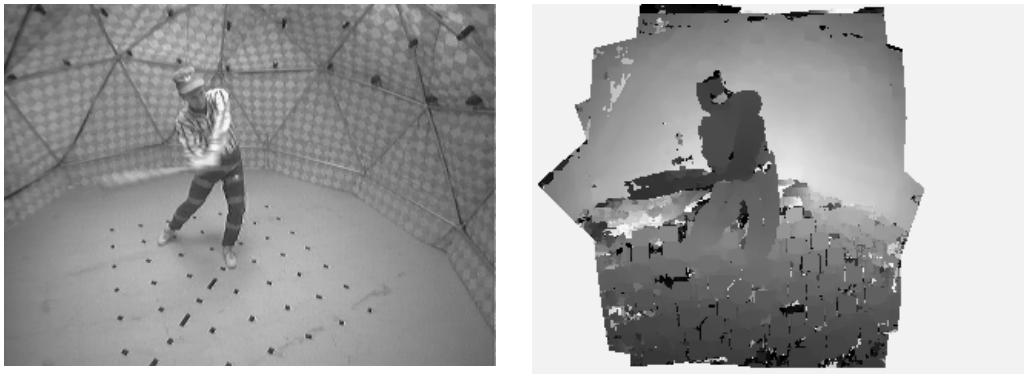
VSMs. Updating with the full weighted, signed distance, however, can easily overwhelm the contribution of all other VSM estimates in a particular area because the signed distance can grow quite large. In order to reduce the error associated with this update while still retaining some ability to "erase" erroneous surfaces, we clamp the weighted, signed distance to lie within the limited range $[-\delta, \delta]$:

$$h_i = \begin{cases} g & \text{if } -\delta < g < \delta \\ \delta & \text{if } g > \delta \\ -\delta & \text{if } g < -\delta \end{cases} \qquad V(\bar{v}) = \sum_{i=1}^{N_{\text{cameras}}} h_i(\bar{v}) \tag{3.10}$$

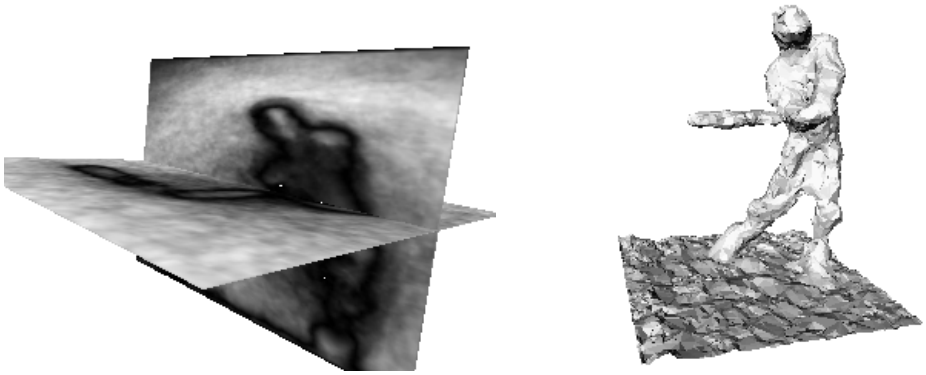
This approach introduces two errors, however. First, correct surfaces can be eliminated if only a few VSMs contain them, since a few erroneous estimates can overwhelm the correct ones. Second, this approach can introduce a bias into the voxel space, shifting the reconstructed surface away from its true position, when voxels near a real surface are determined to be empty from the perspective of at least one VSM. On our data, this effect is actually less serious than that of the noise present in the VSM surface models, so the global geometry is still a significant improvement of that of the VSMs. The refinement techniques in Section 3.5 also reduce this distortion. Figure 3.4 shows an example of the output for one scene, with the parameter δ set to 100 mm. The size of the floor patch is 1500 mm by 1500 mm, and the person is approximately 1700 mm tall. Part (a) shows one of the 51 input images, and (b) shows the corresponding range image computed with stereo. Part (c) shows two slices through the volumetric space after integration of all 51 range images. Black corresponds to zero distance from the isosurface, with increasing brightness indicating greater absolute distance to the surface. Part (d) shows the extracted isosurface as a polygonal mesh, with no shading interpolation across facets.

3.3 Removing Unobservable Surfaces

One interesting effect of applying this algorithm on noisy data with erroneous, or "hallucinated" surfaces, is that some surfaces may appear *behind* others in the final model. These surfaces are unconnected to other real surface patches, but can not be eliminated during



(a) (b)



(c) (d)

Figure 3.4: Results of volumetric merging. (a) An original image. (b) The range image corresponding to the image in (a). (c) two slices through the volumetric space, with black representing voxels on the isosurface and increasing brightness corresponding to increasing distance from the surface. (d) The extracted surface model.

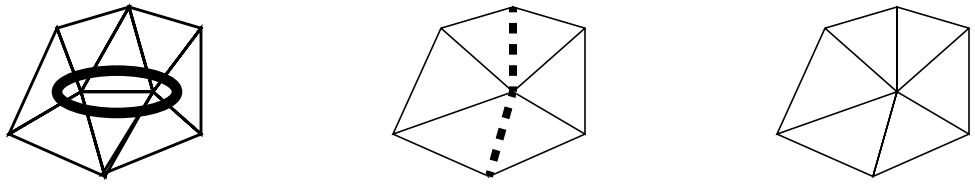
volumetric integration because other views consider that portion of the voxel space to be unobservable. In order to eliminate these surfaces in a principled way, we add an extra step after the isosurface extraction that determines the visibility of each independent polygonal mesh in the polygonal model. If a given surface mesh is not observable from any VSM, then it is eliminated from the final model. Note that this test is applied to entire meshes rather than to individual triangles. Applying this test to individual triangles may eliminate triangles that close gaps, or holes, in meshes that are mostly visible. Although these surfaces are not directly observed, keeping them creates models with more continuous surfaces, which are usually more appealing to a viewer.

3.4 Decimation of Meshes

One practical problem with the Marching Cubes algorithm [35] is that the tessellation of the implicit surface grows with the number of voxels. Consider, for example, how it handles a simple planar implicit surface. Assume that the voxel space is of size NxNxN and that the surface is parallel to the voxel grid. With this configuration, only the NxN cubes will intersect the plane. Because the surface is parallel to the voxel grid, each of these voxel cubes will generate 2 triangles. Repeating across all cubes intersecting the plane will generate almost $2N^2$ triangles. If N=100, for example, then nearly 20000 triangles will be generated. Because the surface is a plane, however, we could perfectly represent it by only 2 triangles that together completely span the voxel space.

We address this problem by applying a mesh decimation algorithm, which approximates one mesh by another than has fewer triangles. There are a number of approaches to achieve this objective, of which [14][21][23][52][64] are just a few. An important distinction among these algorithms is the types of topology supported. Some methods are specifically designed to decimate height fields, while others support nearly any topology. Since height fields are topologically equivalent to a plane, algorithms designed to work specifically with them may not work for general meshes such as those produced from our merging system. We therefore require an approach that support unknown topologies.

Another important distinction among decimation algorithms is how they handle mesh topology. Some decimation algorithms allow arbitrary changes to surface topology, attempting to make better use of the triangles in other parts of the scene, while others maintain the input topology. Our 3D models must support viewing from nearly any viewpoint and nearly any distance, so topology preservation is critical. If the topology changes, the realism of the resulting model may diminish. Imagine, for example, that a thin object such as a book is collapsed into a plane. From the front or back, the view might look believable, but a side view would reveal that the book has no (physical) depth (we will leave the question of literary depth for the reader!).



(a) (b) (c)

Figure 3.5: Mesh decimation by edge collapse. (a) A mesh to be decimated. The edge to be collapsed is circled. (b) The mesh after collapsing the edge. Redundant edges are drawn as dashed lines. (c) The mesh after edge collapse and redundant edge removal.

Based on these restrictions, we chose an algorithm [23] that supports and preserves arbitrary topology. This algorithm is an edge-collapse method, which reduces mesh size by progessively collapsing the mesh edges that introduce minimal shape distortion in the mesh. The basic operation is to collapse an edge joining two vertices into a single vertex, with the new vertex gaining the connectivity of the two vertices at the end of the collapsed edge. For example, the central edge in Figure 3.5(a) is collapsed to form the mesh in part (b). Any redundant edges in the new connectivity are then removed. These edges occur when the two vertices on the collapsed edge both connect to other vertices. Figure 3.5(b) highlights two of these edges. Figure 3.5(c) shows the final result after one edge collapse. The results of applying this algorithm to a real 3D model are shown in Figure 3.6. The original model, with 176520 triangles, is shown in Figure 3.6(a) with a close-up view of the back of the person's right shoulder. Similar views of the decimated model, with 22698 triangles, are shown in Figure 3.6(b).

3.5 Shape Refinement

The recovery of explicit 3D structure relies on merging structural information in the VSMs, which themselves were constructed from stereo range estimates. This process allows the global structure to emerge in a bottom-up strategy, without any pre-conceived notions about the scene shape. Without any knowledge of scene shape, though, the stereo problem has proven to be extremely challenging [12]. The result is that the global modeling process is working with extremely low-quality estimates of scene structure. Once the modeling process is complete, though, a good estimate of surface shape is available. This model can be fed back to guide earlier processes, leading to more accurate shape recovery.

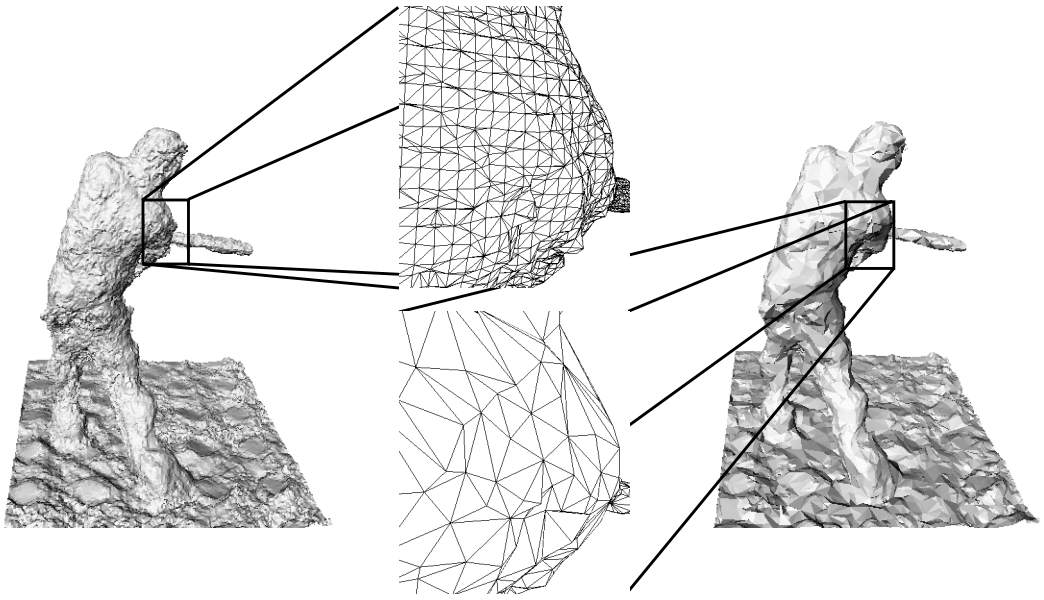
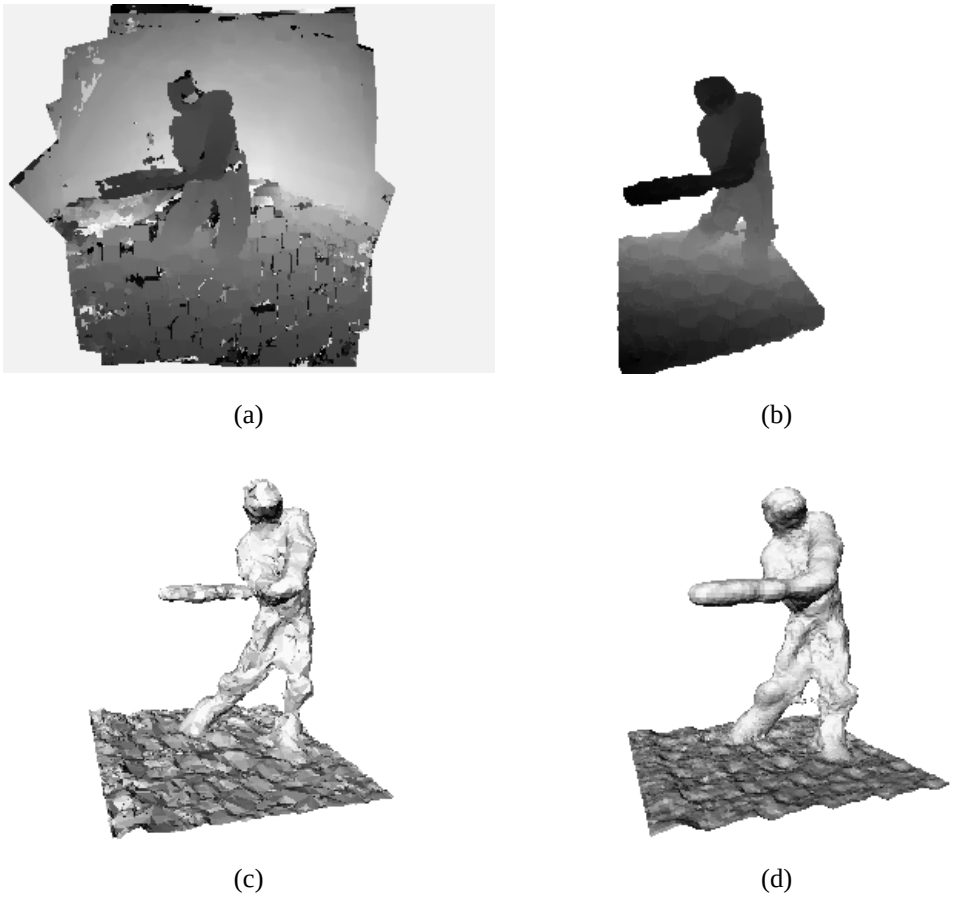


Figure 3.6: Example of decimation applied to a 3D digitized shape model. (a) An input mesh, with a close-up of a portion of the mesh. (b) A decimated mesh, also with a close-up view.

3.5.1 Cleaning VSMs

One way to realize this improvement is to re-evaluate the existing VSMs. The global model can be rendered back into each camera, creating a “global” range image. This range image can then be compared to the input stereo range image. Because of the merging process, correctly located surfaces in the two range images may not match exactly but should be fairly close to one another. Incorrect stereo estimates, however, should be eliminated in the merging process, so these two range images will differ greatly when stereo has made a mistake. This information can be used to filter out the gross errors made by stereo. The test can be based directly on comparing the range of corresponding pixels in the stereo and global range images. If the difference between the two estimates is small, then the stereo value is kept. If the estimate is grossly wrong, the point can be eliminated.

Although this strategy eliminates incorrect stereo range estimates, it also creates gaps in the cleaned range image. These gaps appear because the cleaning process makes decisions based entirely on pixel comparisons. The global model provides knowledge of the underlying surfaces in the range data, though, so the cleaning algorithm can interpolate across



(a) (b)

(c) (d)

Figure 3.7: Results of cleaning range data and re-merging. (a) An original stereo range image. (b) The cleaned range image. (c) The original 3D model. (d) The cleaned model.

gaps in the range images. When a gap in encountered, the algorithm can determine the surface in the global model that projects to that gap. This surface can then be mapped to valid range estimates near the gap. From these estimates, the gap can be interpolated directly from the stereo range data. A slightly simpler strategy is to simply replace the unknown stereo estimate with the range from the current global model, although this step may introduce slightly more roughness, since the model may not align precisely with the stereo range data. Note that interpolation from range samples is commonly done to create surfaces from sparse range data. These approaches have no knowledge about the true surface structure, however, and so must make intelligent guesses about feature connectivity or must rely on a human operator to provide this structure. Unlike these methods, our approach bases the interpolation on the globally estimated surfaces, greatly increasing the validity of the interpolation. An example of this process is shown in Figure 3.7.

3.5.2 Re-Computing Stereo

A more aggressive strategy than cleaning range images is to completely re-compute stereo. The original stereo process had no knowledge of the global structure, so it made the best estimate it could looking over the entire depth space. With the global model, however, approximate range is already known. Using this range as an initialization, stereo can search near this range to improve the surface estimates. In addition, stereo can use the approximate local surface structure to avoid the flat-world assumption in the window-based correspondence detection.

Tailoring the search range can be integrated into the stereo algorithm with little effort. The fixed depth search range in the original stereo process is replaced with a minimum and maximum depth for each pixel. These bounds are computed by rendering the global model into each camera to get the approximate depth. Then, at each pixel, a fixed depth is added and subtracted to get the maximum and minimum bounds, respectively. Stereo then computes the best depth estimate within the allowable range for each pixel.

The local surface structure can also remove the flat-world assumption in stereo. Recall from Section 2.3.5 that the stereo window is assumed to be at a constant depth with respect to the reference camera. The true structure, however, frequently violates this assumption, reducing the precision of correspondence detection. The global model can provide approximate local structure within the window, and can determine visibility of the elements of the window in the non-reference views. The resulting correspondence search should yield higher precision estimates.

Of these two improvements, only the first has been implemented. The results of this process are shown in Figure 3.8. Note how the overall shape is now much smoother than the original, since the range estimates are now working cooperatively to reconstruct the true surfaces.

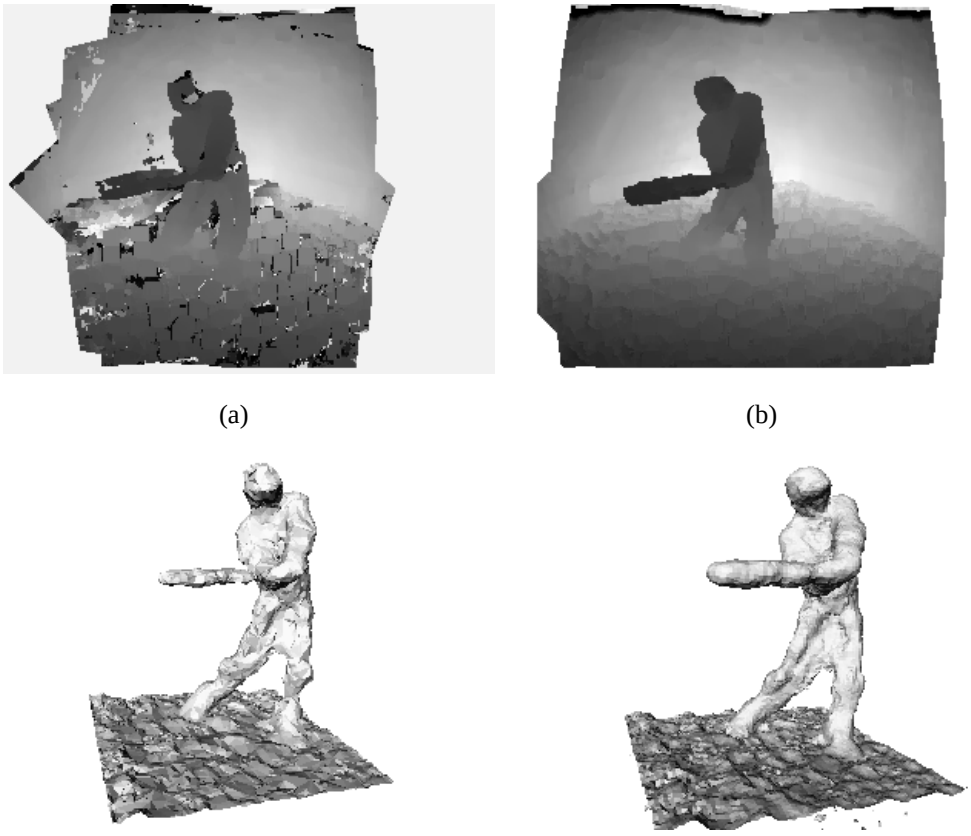


Figure 3.8: Results of recomputing stereo based on approximate global shape. (a) An original stereo range image. (b) The re-computed range image. (c) The original 3D model. (d) The improved model.

Chapter 4

Scene Color From Structure and Images

In order to synthesize realistic appearance of a 3D model, geometric structure must be complemented by some representation of the visual appearance of the scene. This chapter presents two approaches to modeling this appearance by combining global structure with the original video images.

One important issue that must be addressed is the meaning of “scene appearance.” A real camera observing a real scene measures the scene radiance. Scene radiance is a function of many factors, including 3D scene geometry, spectral and spatial distributions of illumination, and material properties such as reflectance. One way to model scene appearance, then, is to model the scene radiance by estimating all of these parameters from the images. Sato et. al. demonstrated that with high-quality 3D models and with multiple observations with controlled lighting, it is possible to separate and estimate diffuse and specular reflectance for real objects [50][51]. Beyond this analysis, it may be desirable to explicitly model the full reflectance properties with a bidirectional reflectance distribution function (BRDF), which captures the full reflectance as a function of the spectrum of the illumina-

tion. It would also be desirable to eliminate the need for controlled lighting. As the degrees of freedom grow, however, solving this problem becomes more difficult. In fact, no one has devised a practical method for complete BRDF modeling of general surfaces.

Even if this complexity problem can be addressed, unraveling the intricacies of the scene radiance frequently requires more effort than is needed to create the desired visual effect. For example, correct modeling of all the physical parameters can provide great flexibility in changing scene parameters such as lighting. However, in many cases these parameters will remain fixed during the synthesis of new views, implying that there was no need to recover these parameters. Estimating the underlying physical parameters may hurt quality, too, since errors in the physical parameters could seriously impact the appearance of the model. For many situations, then, it may be preferable to avoid recovering the parameters that model scene radiance.

We present two methods to avoid explicit modeling of physical surface parameters. One method is to model scene appearance with a global appearance (or texture) map. The map describes the overall color, or *appearance,* of the scene as viewed in the original images, rather than the true amount and spectral content of light radiating from each point in the scene. This approach is discussed in Section 4.1. The second approach is to model scene appearance directly with a collection of images. By preserving the original images, many view-dependent effects can be preserved. We will explore these issues in Section 4.2.

4.1 Global Appearance

One of the simplest ways to approximate scene appearance is with a single, global image mapped directly onto 3D scene geometry. With geometry encoded as polygonal meshes, displaying the model can be accomplished using the standard computer graphics technique of texture mapping. The main issue in this approach is recovering the global appearance image given a set of input images of the scene. This problem can be broken down into two parts: projecting one image onto the 3D model, and integrating the overlapping components of multiple projections. The following sections address each of these issues.

4.1.1 Mapping One Image onto Global Model

Estimating scene appearance from an image and a 3D model can be thought of as the inverse of creating an image from the model with known appearance. In image formation, or rendering, the image plane samples the light rays entering the camera from the scene. Two basic techniques are used to simulate these light rays: forward projection of surfaces from the scene into the camera, and back projection of rays from the camera into the scene, called ray tracing. In the forward projection of surfaces, each surface in the model is projected into the image, updating the color of the image if that surface is visible. In ray tracing, each pixel is mapped to a ray (or set of rays) that intersect the scene. The color of the pixel is set based on the appearance of that surface. In either case, the surface color can be determined by underlying material properties, or by texture mapping, which models the surface color with an image, called the texture map.

In order to reconstruct the appearance of the scene itself, the sampling of the light rays must be inverted. That is, rather than determining the color of a pixel from the known colors in the scene, we must recover the colors in the scene from the known colors in the image. As with rendering, there are two approaches to addressing this problem: ray splatting and surface mapping. Ray splatting traces rays from the image out into the scene. When the ray intersects an object, the color of the ray is cast onto the surface – an operation that we will refer to as *splatting*. Surface mapping projects surfaces into the image. The color of each surface is then set to the color of the pixels to which the surface projects.

4.1.1.1 Ray Splatting

Ray splatting projects rays out from the camera through all the image pixels – the inverse of the projection that formed the image. As each ray moves out from the camera, it encounters a surface visible to the camera. The color of the pixel corresponding to the ray can then be applied to the first surface in the scene (assuming that the object is opaque). Implementing this approach requires two steps: determining the first surface encountered by each ray, and then applying the color of that ray to the global texture map. Determining the first surface intersecting the ray, called the ray intersection test in ray tracing, can be

solved using standard techniques [13]. Once the point of intersection on this surface is determined, the color of the ray must be applied to the portion of the global texture map corresponding to this surface. The ray will usually not land directly on a pixel within this texture map, however, so the color must be blended, or splatted, into several samples in the global texture.

This method has two difficulties, closely related to problems in ray tracing. First, determining the surface intersecting the ray, i.e. the ray intersection test, is time consuming. Second, the rays themselves will not uniformly cover the global texture space. As a result, splatting an image into the global texture space may leave gaps in the global texture between the projections of the rays. By splatting rays that lie between pixel sample points, this problem can be reduced, but this solution further increases computation time. Another solution is to consider the perspective distortion (foreshortening) present in the image being back-projected. If each ray becomes a “solid” ray subtending a solid angle in space, then the color of the ray can be spread out to each surface observable in that solid angle. This approach requires the use of a weighting function to model the splatting of the ray color, much like a point spread function models the blending of color in image sampling. Without this weighting, the color would change abruptly at the boundaries of the solid angles corresponding to neighboring solid rays. Both the complexity and computational costs make this method unattractive.

4.1.1.2 Surface Mapping

Surface mapping projects surfaces of the scene into the camera, onto the image plane. Z buffering is used to determine the visibility of each surface. If the surface is visible, then the color of the pixels to which it projects can be pasted onto the projected surface patch. For 3D triangle meshes representing scene geometry, each surface is a single triangle facet. As a result, the mapping from texture space to image space is a homography in image coordinates. To see this, consider the 3D plane $Z_p = 1$. The plane is indexed by the position (X_p,Y_p) within the plane. This plane can represent any 3D plane if its points are transformed by a 3D projective transformation P:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = P \begin{bmatrix} X_p \\ Y_p \\ Z_p = 1 \\ 1 \end{bmatrix} = P \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ 1 \end{bmatrix} \tag{4.1}$$

Equation (2.4) defines the projection of points on this plane into the image, with the matrix A representing the transformation induced by the intrinsic parameters of the camera:

$$\begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = A \begin{bmatrix} & | & \\ R & | & T \\ & | & \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = A \begin{bmatrix} & | & \\ R & | & T \\ & | & \end{bmatrix} P \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ 1 \end{bmatrix} \tag{4.2}$$

which can be reduced to

$$\begin{bmatrix} xw \\ yw \\ w \end{bmatrix} = H \begin{bmatrix} X_p \\ Y_p \\ 1 \end{bmatrix} \qquad \text{where} \qquad H = A \begin{bmatrix} & | & \\ R & | & T \\ & | & \end{bmatrix} P \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

Using this homography, each texture pixel, or texel, can be efficiently projected into the image. The (possibly interpolated) color at the resultant image coordinates is then copied into the texel. This formulation avoids cracks in the global texture since each texel can be directly sampled. The one potential problem occurs if the texel sampling rate drops below the image sampling rate, potentially introducing aliasing into the global texture. This problem can be addressed by computing the footprint of each texel in the image, and applying a filter with this footprint in the image to generate the texel value [17][18].

4.1.2 Integration Functions

Once the mapping of a single image onto the global model is defined, the next question to address is how to integrate multiple observations of a given texel. We define a general mixing function *mix* that combines the multiple observations into a single color estimate:

$$C = mix(c_1, c_2, ..., c_N) \tag{4.4}$$

where C is the global color estimate and c_i is the color observation from camera i. Because of occlusion, some cameras will not see the surface being considered, and so the mixing function actually operates only on the set of observations in which the surface is visible:

$$C = mix\{c_i \mid \text{surface is visible to camera } i, \quad i\varepsilon[1, N]\} \tag{4.5}$$

We now examine two classes of mixing functions: linear combinations and samples of the observations. For simplicity, the following discussion will drop the notation about visibility, although visibility must be used in the actual algorithms. One class of methods for mixing the observations is to compute linear combinations of the observations:

$$C = \sum_{i=1}^{N} w_i c_i \tag{4.6}$$

If the weights w_i are all set equal to one another, the resulting texture is simply the average of all observations. An example of this method is shown in Figure 4.1(a). The weights can also be set to emphasize observations that may carry more information. For example, weighting views that see surfaces more directly should yield better results than (unweighted) averaging. One way to set the weights to achieve this effect is with the cosine of the angle between the viewing ray $\hat{e}_i$ from camera i and the surface normal $\hat{n}$:

$$w_i = \hat{e} \bullet \hat{n} \tag{4.7}$$

An example of this approach is shown in Figure 4.1(b). Not surprisingly, this method outperforms the unweighted averaging process. The difference in performance grows with increasing errors in geometry and calibration, since the weighting mechanism emphasizes the contributions of a small number of observations, limiting the amount of interference that can occur among more observations.

A second class of methods selects a single real sample from the set of all observations. Because all colors in the global texture map will be real samples, no blurring of information in the original images can occur. With each texel determined from a single image, the resulting texture image can retain the detail of the original images. We present two meth-

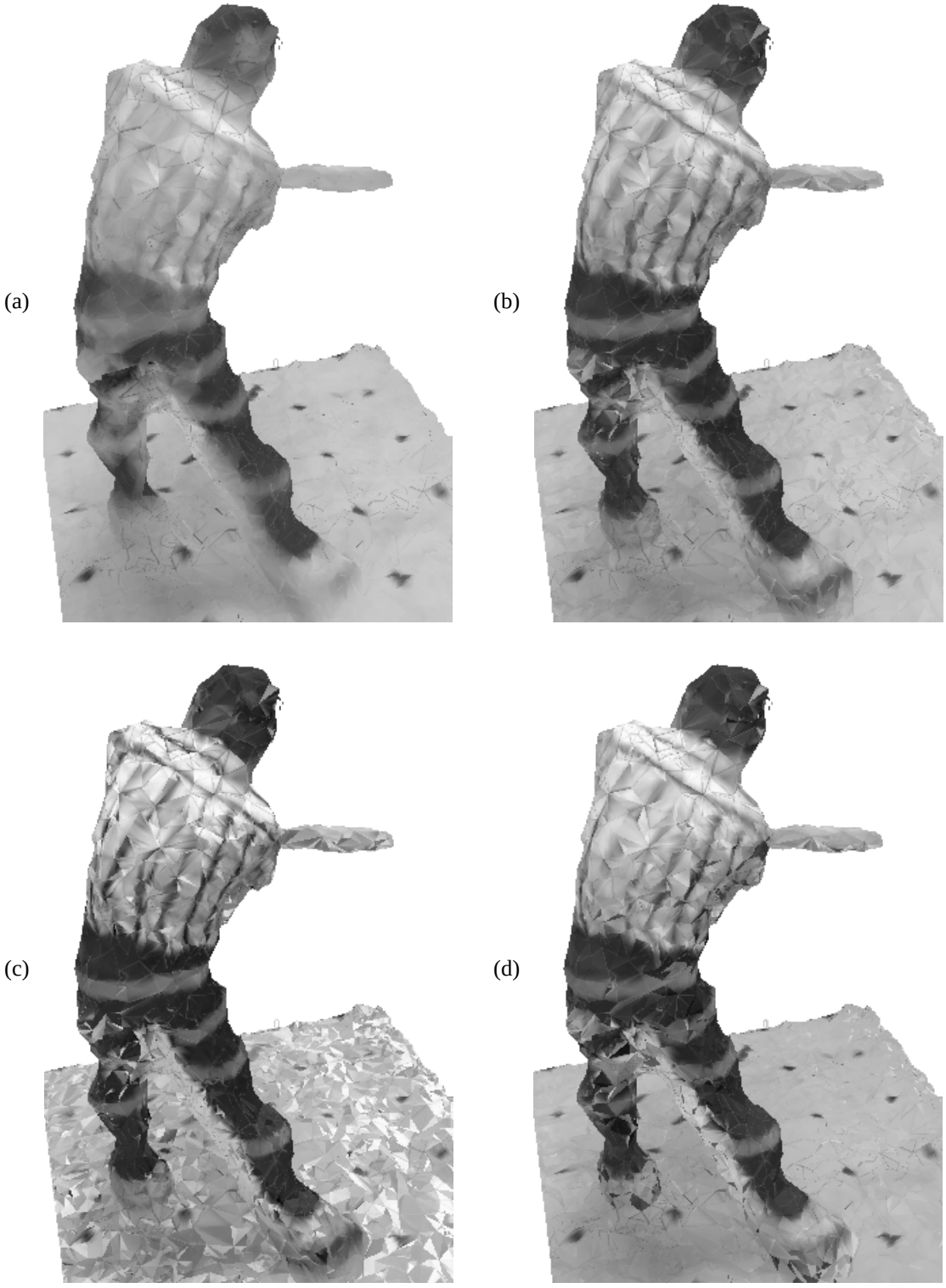


Figure 4.1: Renderings of a model with different global scene appearance. The input observations are used to create the global texture by (a) unweighted averaging., (b) weighted averaging, (c) selecting observation with maximum weight, (d) median filtering of observations with weight greater than 0.7.

ods of selecting the single sample from the set of observations: maximum-weight selection and median intensity filtering. For maximum-weight selection, the texel color is set to the observation with maximum weight, as computed from Equation (4.7):

$$C = c_k \qquad \text{where} \qquad k = \arg\left\{ \max_i (w_i) \right\} \tag{4.8}$$

An example with this method is shown in Figure 4.1(c). Renderings using this texture map exhibit sharp contrast edges across what should be smoothly shaded surfaces. This phenomenon occurs because the input images have different imaging characteristics, leading to inconsistent brightness in observations of the same surface, even if geometry and calibration are perfect.

A different method is to compute the median intensity of all observations at each texel. This approach should still provide some of the detail from the original images but will select an intensity that is more representative of the set of observations. To be most effective, this approach should incorporate knowledge of the weights so that only the best views are considered. An example is shown in Figure 4.1(d), with the median computed from observations with a weight of at least 0.7.

A generalization of these approaches would be to formulate this problem in a probabilistic framework, as done by Martins and Moura [36][37] for the determination of voxel color. This strategy would allow the explicit use of a sensor model with the probabilistic update of surface color, and would overcome the limitations of sampling resolution for voxel color. This reformulation has not been attempted.

4.1.3 Mapping Global Textures to 3D Models

An important issue that has not been addressed yet is how to map the global texture to the 3D model. Unless the scene is homomorphic to a plane, the mapping is not obvious. Even with planar topology, developing a strategy that yields approximately uniform resolution across the entire scene can be difficult.

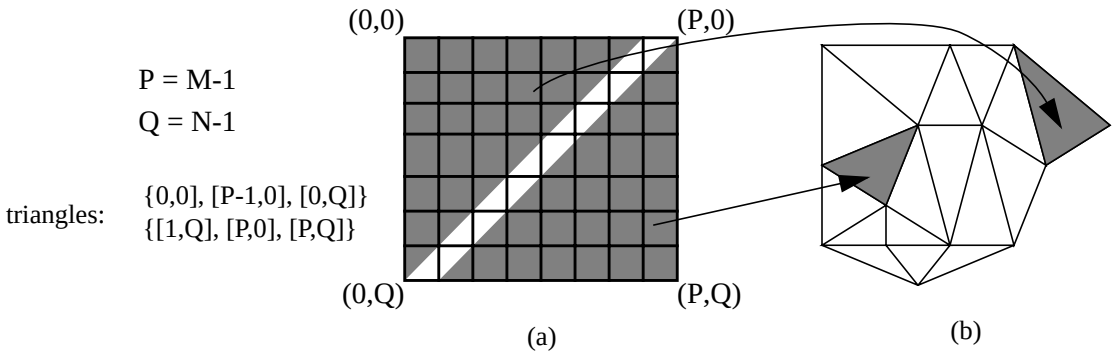


Figure 4.2: Decomposing the texture image into independent triangles. (a) This MxN section of the texture map contains two texture triangles, one represented by horizontal lines and one by vertical lines. The intersection of grid lines indicates pixel values. (b) These two texture triangles can be assigned to any two 3D triangles in the 3D mesh.

To avoid these issues, we devised a strategy that ignores scene topology entirely. We tile the texture image area into independent triangular regions, as shown in Figure 4.2. We then assign each triangular region of the texture image to one triangle in the 3D scene model. The simple strategy that we implemented makes this assignment based only on the order in which the 3D triangles occur, so the texture map itself has no apparent structure outside of the individual triangles. A typical texture map is shown in Figure 4.3.

If the display system understands how the texture was divided, it can directly use this texture map without any other concerns. In many hardware-accelerated texture mapping systems, though, the underlying algorithms may misuse the texture information. For example, bi-linearly interpolating the texture image along the boundary of texture triangles will lead to “cross-talk” between the two independent textures. In order to avoid this problem for bi-linear interpolation, two extra columns can be inserted between triangles sharing a diagonal in texture space. These extra texels, set to the same color as the neighboring triangles, will serve as padding to avoid the interpolation problems.

Another problem that can develop is with mip-mapping. The mipmap is a texture map stored under various resolutions. In computer vision, this structure is usually referred to as an image pyramid. The texture mapping hardware selects the mipmap level that best matches the resolution of the real surface in the rendered viewpoint. This selection reduces the amount of texture computation needed, especially as objects with high-resolution texture maps move far into the distance of the virtual camera. The problem is that if the sys-

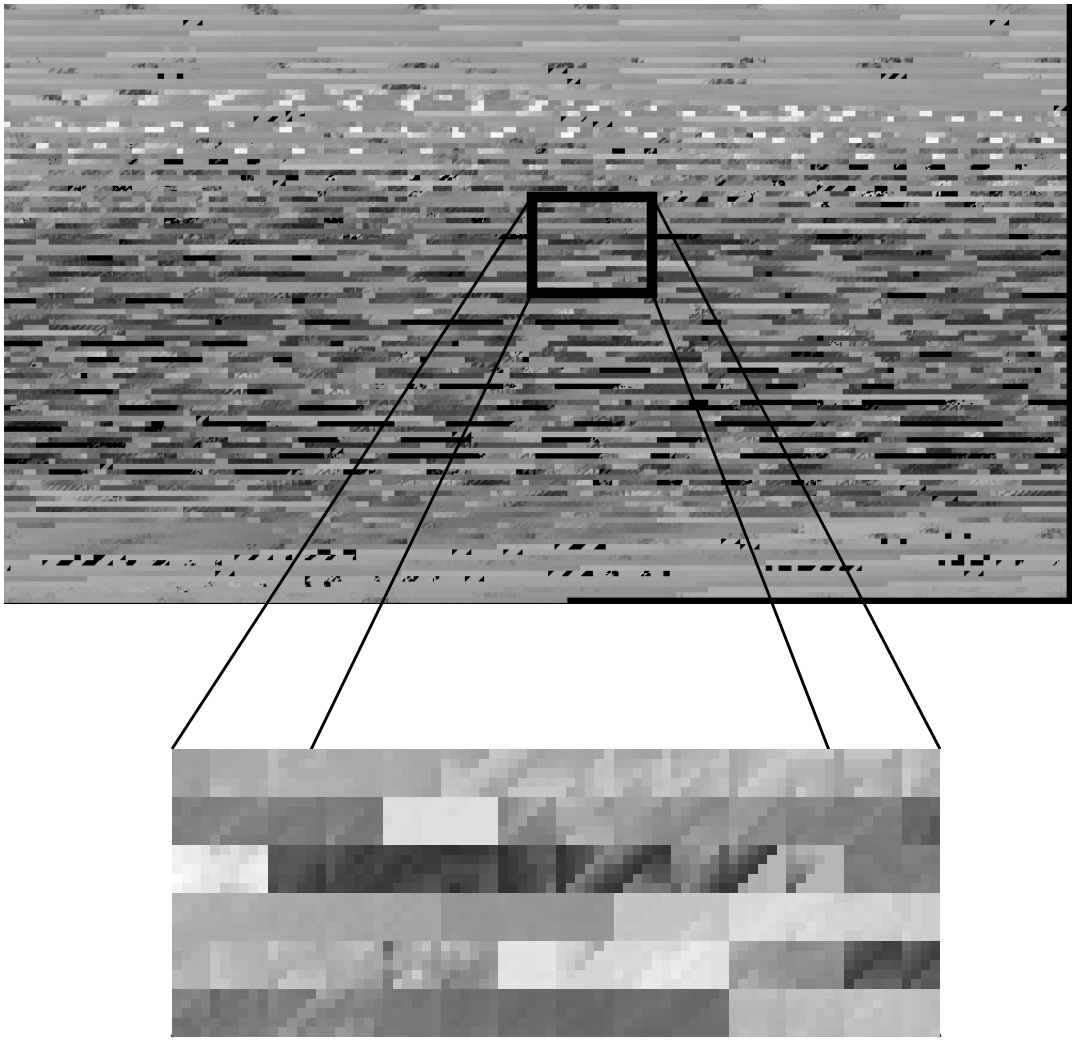


Figure 4.3: The actual texture map used to synthesize the image of Figure 4.1(d). The top image is the global texture map computed with the unweighted average technique. The bottom image is a close-up of a small portion of the global texture, highlighting the independence of neighboring texture triangles.

tem automatically computes the mipmaps from the input texture map, it will almost certainly blur across independent triangle textures. The obvious solution to this problem is to pre-compute the mipmaps with knowledge of the underlying texture map structure.

Slightly changing the ordering of triangles within the texture image can also have powerful effects. If every pair of triangles sharing a diagonal in the texture map share the same diagonal in the 3D model, then the texture should be continuous across the edge. This

strategy completely avoids the bi-linear interpolation problem. In addition, if the texture will be compressed with block-based compression strategies, the continuous texture block should yield better compression than with random triangles sharing a diagonal.

4.2 Image-Based Appearance

An alternative to the global appearance strategy is to model scene appearance directly with a collection of images. Keeping the original images avoids the problems associated with constructing global texture maps such as blurring. In addition, with global modeling, all view-dependent effects present in the image set are removed. By preserving the original images, these effects can be preserved, potentially creating more realistic images. For a slowly moving viewpoint or a slowly changing scene, these changes in visual appearance may be particularly important because the observer may have expectations for changes of scene appearance that would not appear with the global method. This inconsistency could degrade the impression of immersion. For fast viewpoint changes, these effects will probably be too subtle to be detectable, since the viewpoint change itself will produce a large change in visual appearance. The cost of using this approach is that new rendering techniques must be developed to support these models.

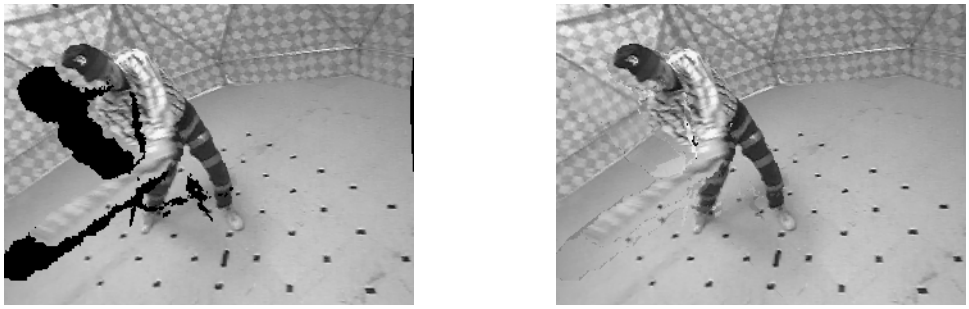
The concept of using images as a scene appearance model was demonstrated in the View Interpolation algorithm of Chen and Williams [6]. In this system, disparities between pairs of images were used as models of the underlying scene geometry. To synthesize a viewpoint partway between the two input images, the viewpoint was assigned an interpolation factor based on the distance to the two real images. The disparities relating the images were then scaled by this factor, and the pixels in each image were forward-projected in a splat-style operation into the virtual camera. The splatting suffered from the sampling problems discussed in global texture creation, namely that cracks could appear in the synthetic output. The gaps were filled by interpolating from nearby pixels that were occupied. As Chen and Williams observed, the interpolation of disparity vectors is geometrically

correct only for a few special camera configurations such as the parallel-camera geometry discussed the context of stereo (cf. Section 2.2.1). Seitz and Dyer pointed out that by rectifying the images first, this strategy will always produce physically correct views [53].

In addition to the sampling problems, this strategy suffers from two other problems. First, camera motion is restricted to interpolation of the input image set. If the images all lie on a plane, then the virtual camera cannot leave that plane. Second, the strategy is fundamentally based on image pairs, which will grow at a rate of N_c^2 for N_c images if every image is connected to every other one. In a mesh of widely separated cameras, we obviously would not want to connect all images to all others, but each image might have 5-6 neighbors, requiring 5-6 correspondence maps per image.

One way to reduce this storage problem is to use a single depth image at each viewpoint rather than the set of correspondence maps. With fully calibrated cameras, disparities and depths are interchangeable, so no information is lost in this conversion. The power of this approach is that once depth is available, the virtual camera is free to navigate anywhere, regardless of the configuration of the real cameras. Laveau and Faugeras actually use weakly calibrated cameras and projective structure in this way [33]. The problem with this approach is that the virtual viewpoint must be specified by 4 point correspondences between the virtual camera and the input images, making navigation a tedious task. Laveau and Faugeras also use an inverse sampling technique, mapping the pixels in the virtual viewpoint back to the original images. While this approach can avoid gaps by interpolating shape in the input range images, it is much slower than forward mappings because a search is necessary along the epipolar lines in the input images

The approach in this thesis is similar to Laveau and Faugeras in that we use depth information rather than disparities. We assume full camera calibration, however, allowing easy navigation through the event space. In addition, we use a forward mapping, which allows the use of hardware-accelerated rendering. In order to avoid sampling problems that create gaps in the synthesized image, we assume that pixels with similar depths are part of the same surface, which allows us to render triangles rather than splatting pixels. This assumption is actually identical to the way Visible Surface Models are constructed from



(a) (b)

Figure 4.4: Holes from occluding contours. (a) Holes appear when virtual camera moves away from a VSM. (b) Supporting VSMs can fill the holes.

range images (see Section 2.1). In fact, our implementation of these ideas pre-computes range images from the global 3D model, and Visible Surface Models are constructed on the fly as the range and color images are read into the program. For convenience, we will use the term VSM throughout the following discussion unless the distinction between the range image and the true VSM is important.

4.2.1 Synthesis of Virtual Images From Real Images

A single VSM represents all surfaces visible within the viewing frustum of the camera. As we will show, it can be used to render the scene from locations near this real camera, but the synthesized image will have holes when the virtual camera moves away from the exact real camera position because occluded areas become exposed, as shown in Figure 4.4(a). However, given a sufficiently dense distribution of VSMs, we can typically find other VSMs to fill the holes. Intuitively, when the virtual camera moves away from the VSM in one direction, a neighboring VSM that lies “beyond” the virtual camera can fill the holes created. Thus a combination consisting of a small number of neighboring VSMs can provide nearly hole-free synthesized views as in Figure 4.4(b).

The VSM “closest” in approach to the virtual camera provides the most promising model and should play the central role in synthesizing the rendered image. We call this the *reference* VSM. It will typically not be hole-free on its own; we therefore include two neighboring VSMs to fill the holes. They are called the *supporting* VSMs. These are selected so as to collectively cover all regions that are uncovered when the virtual camera moves away

from the reference VSM. The combination of one reference plus two supporting VSMs works well for our current arrangement of cameras, which approximately lie on the surface of a hemisphere. A suitable combination of a small number of VSMs should be used for other arrangements of the VSMs.

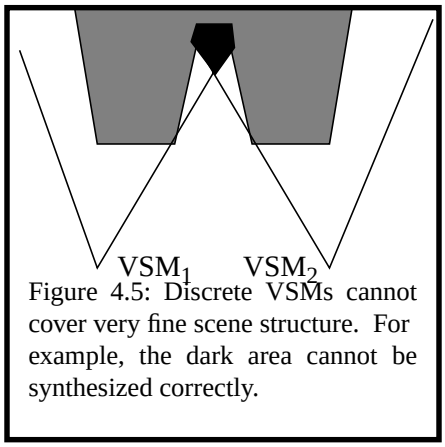


Figure 4.5: Discrete VSMs cannot cover very fine scene structure. For example, the dark area cannot be synthesized correctly.

Note that the problem of covering all possible holes with a finite number of VSMs has no good solutions. It is possible to create a scene and a virtual camera location that will not be hole free for any given arrangement of VSMs as shown in Figure 4.5. The dark area is occluded from both VSMs and cannot be synthesized in any view. Our approach works well under the following conditions: First, every part of the scene to be modeled must be visible in some VSM. In other words, there is no structure too fine for the distribution of the VSMs. This condition is satisfied when the distribution of the VSMs is fairly dense and uniform with the main event space as their focus. Second, the virtual camera should be pointed roughly in the same region. This condition is satisfied because all the objects of interest are in the central region. These conditions are reasonable for any setup and lead to our design decisions; they are however not limitations of the system as it can be extended to other situations easily.

4.2.2 Selection of reference camera

Finding a good definition of “closeness” for the selection of the reference VSM is a complex problem because of the possibility of occlusion. Intuitively, the usefulness of a VSM increases as the virtual camera moves closer (in physical space) to it. But the physical distance alone as a closeness metric is insufficient because the intersection of the VSM’s and the virtual camera’s extents (or fields of view) may be small. Using direction of gaze alone also has clear problems. In theory, the choice of the reference VSM as well as the number

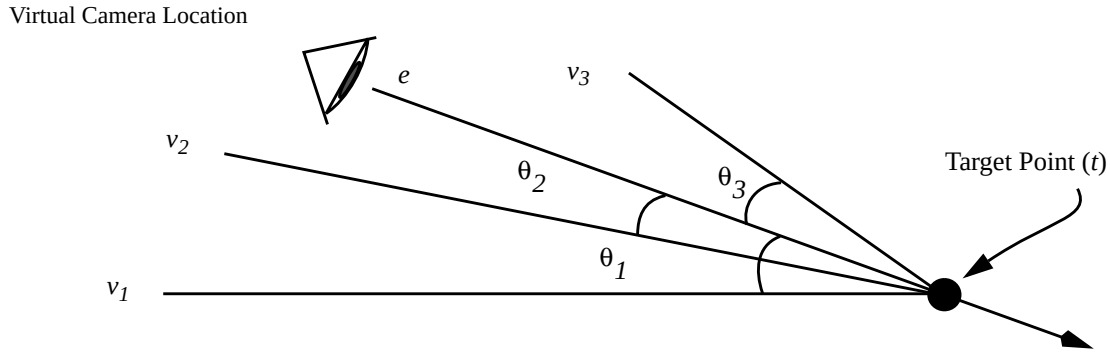


Figure 4.6: Selecting the reference VSM. θ_i is the angle between the virtual camera's line of sight and the line joining the target point with the position V_i of VSM i. The VSM with the smallest angle θ_i is selected as the reference VSM.

and combination of other VSMs used during rendering should depend on the field of view of the virtual camera and the extents of the VSMs. All the information to support an alternative strategy is available in our system, however.

We use a closeness metric based on the assumptions about the distribution (3D placement) and orientation (field of view) of the VSMs as well as about the general regions of interest in a typical scene. Consider, for example, the example of Figure 4.6. The viewpoint of the virtual camera is specified by an eye point (*e*) and a target point (*t*). The virtual camera is positioned at the eye point and oriented so that its line of sight passes through the target point. Our measure of closeness is the angle θ_i between this line of sight and the line connecting the target point to the 3D position of the VSM. The VSM with the smallest angle is chosen as the reference VSM. Since the cosines of these angles have exactly the opposite relative ordering as the angles themselves, we can actually evaluate these instead, choosing the VSM with the maximum cosine. If we define $\bar{e}$ as the vector from *e* to *t*, and $\bar{v}_i$ as the vector from v_i to *t*, then this angle is given by

$$\cos\theta_i = \frac{\bar{V}_i \bullet \bar{e}}{|\bar{V}_i||\bar{e}|} \tag{4.9}$$

Since the magnitude of *e* is the same across all potential viewpoints, we instead compute

$$\frac{\bar{V}_i \bullet \bar{e}}{|\bar{V}_i|} = C_i \tag{4.10}$$

where C_i is now the closeness metric to be maximized. For an arbitrary virtual viewpoint, all VSMs must be considered, and so the VSM selected is the one with maximum C_i:

$$\text{ref VSM index} = \arg\{\text{Max}|_i \; C_i\} = \arg\left\{\text{Max}\left|_i \; \frac{\bar{V}_i \bullet \bar{e}}{|\bar{V}_i|}\right.\right\} \tag{4.11}$$

This selection process can be encoded as shown in algorithm FindReferenceVSM.

```
Algorithm: FindReferenceVSM
    MaxCos = -1.0; RefVSM = -1
    for each VSM i
        c = v̄_i • ē / |v̄_i|
        if (c > MaxCos) then
            maxCos = c;
            RefVSM = i
        endif
    end
    if (MaxCos > 0) then
        return RefVSM
    else
        return FAILURE
    endif
```

This measure works well when both the virtual viewpoint and all the VSMs are pointed at the same general region of space. In our system, this assumption holds for the VSMs by design, which tends to focus the user's attention on this same region. Other metrics of closeness are also possible, for instance, the angle the line of sight of the virtual camera makes with the line of sight of the camera corresponding to a VSM.

4.2.3 Selection of supporting cameras

The supporting VSMs are used to compensate for the occlusions in the reference VSM. These VSMs therefore must have different viewpoints from the reference, but they must be close enough to the reference VSM that the textures will match well. Choosing a viewpoint far away may change the appearance of a surface because of foreshortening, for example. We therefore only consider VSMs that neighbor the reference VSM. In addition, we require that the two supporting VSMs must also be neighbors. Consider, for example,

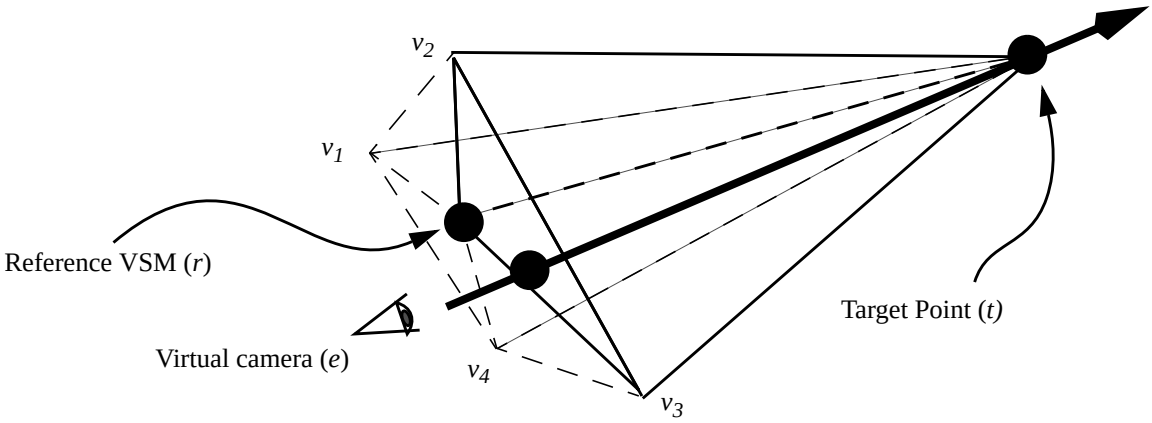


Figure 4.7: Selection of supporting VSMs. Given a reference VSM, the supporting VSMs are determined by the triangle pierced by the line of sight of the virtual camera.

the example in Figure 4.7, in which the reference camera has 4 neighboring VSMs. Now consider the triangles formed by the reference VSM and all adjacent pairs of its neighboring VSMs. In general, if the VSM has n neighbors, there are n such triangles. To select the supporting VSMs, first determine which of these triangles is pierced by the line of sight of the virtual camera using the available geometry. The non-reference VSMs that form this triangle are selected as the supporting VSMs. Intuitively, the reference and supporting views "surround" the desired viewpoint, providing a (nearly) hole-free local model for the virtual camera. The holes created by the virtual camera moving away in any direction from the reference VSM are covered by one of the supporting VSMs as they collectively lie "beyond" the virtual camera when viewed from the reference VSM.

To determine the triangle pierced by the line of sight of the virtual camera, let $\bar{e}$ be the viewing vector from virtual eye point e to target point t, let $\bar{r}$ be the vector from the reference VSM to t, and let $\overline{rv_i}$ be the vector from reference VSM position r to neighboring VSM i. The triangles can be described by rv_iv_j, where

$$j = \begin{cases} i+1 & \text{if} \quad i < n \\ 1 & \text{if} \quad i = n \end{cases} \tag{4.12}$$

Rather than computing intersections with each triangle, we can instead consider the planes formed by the reference VSM *r*, the target point *t*, and the possible supporting VSM *i*. If the virtual camera position *e* lies on the same side of these planes as the triangle rv_iv_j, then VSMs *i* and *j* are selected as the supporting VSMs. The tests are quite simple: if VSM *i* and virtual camera position *e* lie on the same side of plane rtv_j, and if VSM *j* and *e* lie on the same side of plane rtv_i, then VSMs i and j are selected. We specify the planes in terms of their surface normals $\bar{n}_i$, given by

$$\bar{n}_i = \bar{r} \times \overline{rv}_i \tag{4.13}$$

The sign of the dot product of a point with the surface normal of a plane determines the side of the plane on which the point lies. Therefore the tests needed are simply comparisons of signs of dot products. Let t_1 and t_2 represent the dot products of $\bar{n}_i$ with $\overline{rv}_j$ and $\bar{n}_i$ with $\overline{re}$, respectively:

$$t_1 = \bar{n}_i \bullet \overline{rv}_j \qquad t_2 = \bar{n}_i \bullet \overline{re} \tag{4.14}$$

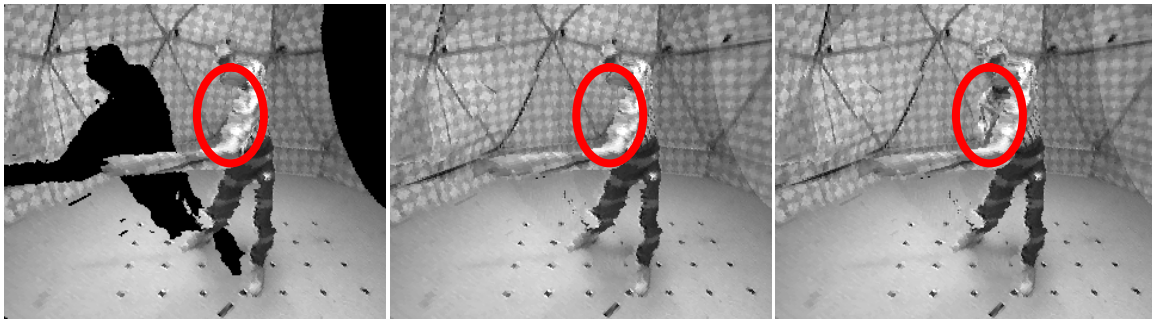
The VSM *j* and point *e* lie on the same side of plane rtv_i if t_1 and t_2 have the same sign. This process is encoded in the pseudocode below.

```
Algorithm: FindSupportingVSMs
    For each neighboring VSM i = [1,NumNeighbors]
        j = i + 1
        if (j > NumNeighbors) then
            j = 1
        endif
        n̄i = r̄ x r̄vi
        t1 = n̄i • r̄vj
        t2 = n̄i • r̄e
        if (sign(t1) = sign(t2)) then
            n̄j = r̄ x r̄vj
            t1 = n̄j • r̄vi
            t2 = n̄j • r̄e
            if (sign(t1) = (sign(t2)) then
                SupportingVSMs = {i, j}
                return
            endif
        endif
    end
```

4.2.4 Rendering with VSMs

The selected VSMs represent a dynamically selected scene model that contains a hole-free description of the scene from the virtual camera's viewpoint. It, however, contains redundant descriptions of a large section of the scene. It is possible to geometrically fuse the multiple triangle meshes into one mesh before rendering, but at considerable computational expense for each combination of VSMs. An alternative to geometric fusion is fusion in the image performed *after* rendering each component VSM separately. The supporting VSMs are used only to fill the hole pixels in the rendering of the reference VSM. The demand for computing power is minimal in this approach. A variation of this approach could have the renderings of the component VSMs *cross-dissolved* (i.e., weighted averaged) based on the distances of the virtual camera from the VSMs.

Merging the renderings of the VSMs in the image requires the hole pixels to be identified. A simple method would be to start with a designated background color that is absent in the scene and render the reference VSM over it. The untouched pixels are the hole pixels. This approach, while being simple to implement, does not correctly identify holes to be filled in some situations. For instance, a thin foreground object will project a thin hole on the background object which will be exposed when the virtual camera moves away from the origin of the reference VSM. If the virtual camera moves far enough, the background that lies on one side of the foreground object (from the perspective of the reference VSM) will appear on the other side of the object from the perspective of the virtual camera. While this reversal in ordering is geometrically correct, it cannot account for surfaces that may lie between the foreground object and the background because the background that has switched sides will fill in the pixels that should contain these hidden surfaces. These hidden surfaces may be part of the same object (such as the additional surface of a sphere that is visible as one moves) or could be independent objects that are occluded by the foreground object. The first situation is illustrated in Figure 4.8(b) where the baseball player's right shoulder is not filled properly.



(a) (b) (c)

Figure 4.8: Various hole-filling strategies. (a) Hole identified using the background color only. (b) Filled image where the right shoulder is incorrect. (c) Filled image where the holes are identified from hole triangles. The shoulder is filled properly.

To overcome this problem, we use a two-fold approach to identify the hole pixels. The pixels that are not written over when rendering the reference VSM are marked as holes, as in the simple algorithm. In addition, we create *hole triangles* of the VSM. These triangles are the ones that were eliminated from the VSM because they have large depth variation along at least one edge (c.f. Section 2.1). These triangles are not real surfaces, but they indicate the space into which the reference VSM can not see. The hole triangles are therefore rendered into a separate hole buffer, marking each pixel that is touched as a hole pixel, even if filled from the reference VSM. The intensity and z buffers are not changed while rendering the hole triangles (although a cross-dissolve could be accomplished by alpha blending during this rendering operation). The holes are identified geometrically using this approach, correctly filling the shoulder of the person in Figure 4.8(c).

Thus, rendering using VSMs is a three step procedure: First, the scene is rendered from the virtual camera's location using the reference VSM. Next, the hole triangles of the reference VSM are rendered into a separate buffer to explicitly mark the hole pixels. Lastly, the supporting VSMs are rendered, limiting their rendering to the hole region. Figure 4.4 and Figure 4.8 show the results of rendering with this method.

4.2.5 Pre-computation of Fill Triangles

The basic VSM-based rendering algorithm makes all filling decisions on the virtual image plane. An analysis of the scene geometry will reveal, however, that the reference VSM's holes and the hole-filling surfaces of the supporting VSMs are independent of the virtual camera view. The hole-filling regions are easily identified by rendering the hole triangles of the reference VSM into the supporting VSMs. The portions of the supporting VSMs touched by the rendered hole triangles are the hole-filling regions. This observation leads to a more efficient rendering algorithm which renders only the hole-filling surfaces from the supporting VSMs. The determination of the hole-filling surfaces for a particular reference VSM can be performed dynamically during rendering, whenever the reference-supporting VSM set changes, or can be pre-computed, allowing faster and more uniform rendering of the digitized scene. The cost of this strategy is increased complexity and storage. Work by Gortler et. al. [16] suggests that the amount of new information added in by neighboring views is actually much less than the size of the reference image itself, though, so the complexity may be the dominant cost in implementing this approach.

A similar strategy can also reduce the effects of the limited field of view of the reference camera by extending the scope of the supporting VSMs to beyond the viewing frustum of the reference camera. This can be done by mapping the entire reference VSM to the supporting VSMs. The pixels of the supporting VSMs not touched in this process contain the portion of space outside the reference model's viewing frustum but visible to the supporting VSM. The field of view can be expanded by rendering this region along with the reference model.

4.2.6 Decimation of meshes

A typical triangle mesh computed from a range image contains over 100,000 triangles for a 240x320 image/depth map as no effort is made to recognize planar patches of the scene. For example a flat section covering 50x50 pixels will generate 2500 triangles instead of two. Such finely tessellated meshes lend themselves easily to decimation. However, the decimation needs to be done carefully to preserve the visual realism of the original model.

A well-known visual effect is that humans can tolerate significant surface errors over continuous surfaces, but errors at occlusion boundaries (or silhouettes) are quite objectionable. This property suggests that aggressive decimation of interior mesh surfaces can be performed with little loss in visual quality, so long as boundaries are well preserved. A boundary point in this discussion is a point that could be on the silhouette of a foreground object against a background when viewing using the model.

We use a simple edge-collapse decimation method [23] that eliminates triangles subject to a number of constraints on the error between the original and the simplified models. The process can be tuned in a number of ways, for instance, by controlling the priority of boundary errors relative to the interior errors. (Boundary errors result from moving boundary points while decimating. Interior errors result from moving the interior points of the surfaces.) This controllability allows us to match the mesh to the human eye's ability to discern detail. We give maximum priority to maintaining the boundary points while substantially decimating interior surfaces. The model is typically decimated from 100,000 triangles to 4000 triangles without appreciable deterioration in the rendered image quality. Maintaining the occlusion silhouette while decimating the mesh is easy in an oriented model like the VSM as the depth discontinuities in the model correspond closely with the visibility in the final rendered images.

4.3 Evaluation of Image Synthesis

The images that have been generated to this point have been evaluated for qualitative correctness and for the presence of visual artifacts. Other methods of evaluation are also useful, such as the signal to noise ratio between a predicted and real image, and psychophysical studies with human observers. In fact, this type of work warrants a large analysis itself, especially as other methods become available. We therefore restrict our evaluations here to the qualitative types, and leave the larger evaluation as open work.

Chapter 5

From Static Scenes to Dynamic Events

Given the techniques for 3D digitization of single frames of motion presented in the previous chapters, replication at successive moments can create the illusion of continuous motion. The first part of this chapter displays the results of applying the digitization process to events along with novel views synthesized with the techniques already described. In addition to re-displaying the original scene, the recovery of 3D structure allows the integration of virtual content with the event and the combination of multiple events into a single, larger one. The second section of this chapter demonstrates these capabilities with real examples.

5.1 3D Digitization of Dynamic Events

Given the ability to 3D-digitize single snapshots of an event, this process can be replicated over time to create sequences of models. This section demonstrates this capability on five example data sets, using the algorithms already described. In order to be believable, the models must be consistent enough from frame to frame to present consistent motion to the

viewer. If a ball is flying through the air, for example, the viewer will expect it to behave in a particular way. If this expectation is violated, then the realism will suffer. We will therefore discuss the quality of the results specifically regarding this property, in addition to overall clarity of scene reproduction at each moment.

It is worth noting here that the most powerful way to determine the realism of these effects is to observe the motion in a nature form, such as through a pre-recorded video or better yet through an interactive viewing system. Since these media are not available in this forum, the following sections will display temporal sequences of images spatially across the page. With this type of display, however, it is nearly impossible to make critical evaluations of properties such as continuity of motion and other temporally varying artifacts.

5.1.1 Example 1: Baseball

The first example is a baseball player swinging a baseball bat. One frame of this sequence was used to demonstrate the modeling techniques discussed in earlier parts of this thesis. Figure 5.1 shows 10 frames of this sequence, sampled at a rate of 6 frames per second, from one of 51 viewpoints. Figure 5.2 shows wire-frame renderings of the 3D models extracted using stereo at each of the 51 viewpoints and with volumetric merging. No shape refinement was performed. From these renderings, it is clear that the overall structure of each frame is represented in the model, with the notable exception of the baseball bat. The bat is disconnected from the player through most of the sequence, and disintegrates into small, disconnected surfaces near the beginning of the sequence. These artifacts are a result of difficulty in stereo, which had difficulty because of the bat's thinness and its appearance, which blended into the background in many images.

In order to evaluate the model in a visual way, we have rendered views from several different viewpoints using the image-based appearance methods of Section 4.2. Figure 5.3 shows a fixed viewpoint, different from any of the original positions. Figure 5.4 shows a viewpoint moving near the cameras. Figure 5.5 shows a more aggressive motion, dropping into the scene to a point just below the reach of the bat. Finally, Figure 5.6 shows a much more aggressive sequence that attempts to produce the viewpoint of a virtual baseball

pitched into the scene and hit into the air. From these views, it is clear that the models are able to generate realistic views when the virtual camera stays near the original views, or across smooth surfaces. With a viewpoint very different from the input views, however, the errors in the recovered shape degrade the scene appearance along occluding contours, especially when those contours separate regions of highly dissimilar intensity.In addition, intensity differences among the real images can exaggerate this degradation when the view-dependent rendering strategy does not blend across occluding contours.

5.1.2 Example 2: Volleyball

The next example is a volleyball player “bumping” a volleyball into the air. Figure 5.7 shows 10 frames of this sequence, sampled at a rate of 15 frames per second, from one of 51 viewpoints. Figure 5.8 shows shaded renderings of the 3D models extracted using stereo at each of the 51 viewpoints and with volumetric merging. No shape refinement was performed. As with the baseball data, the overall structure is recovered fairly well, although the ball is not round and actually changes shape over time. A less obvious error is that the player’s arms are actually modeled as one surface without a gap. The arms still appear separated in many views because there is a deep depression between the arms, which hides the error. The problem here is that many of the stereo views could not see through the gap, so the volumetric integration fused the arms together.

The complete visual modeling is demonstrated in two different camera paths. The rendering strategy used was again the image-based approach. Figure 5.9 shows a viewpoint moving into the scene but flying around the player. Figure 5.10 shows a viewpoint flying very close over the shoulder of the player. As with the baseball sequence, the viewpoints are believable when the camera stays near the original viewpoints. As the camera moves more aggressively into the scene, the errors in the underlying geometry and especially in the occluding contours are revealed.

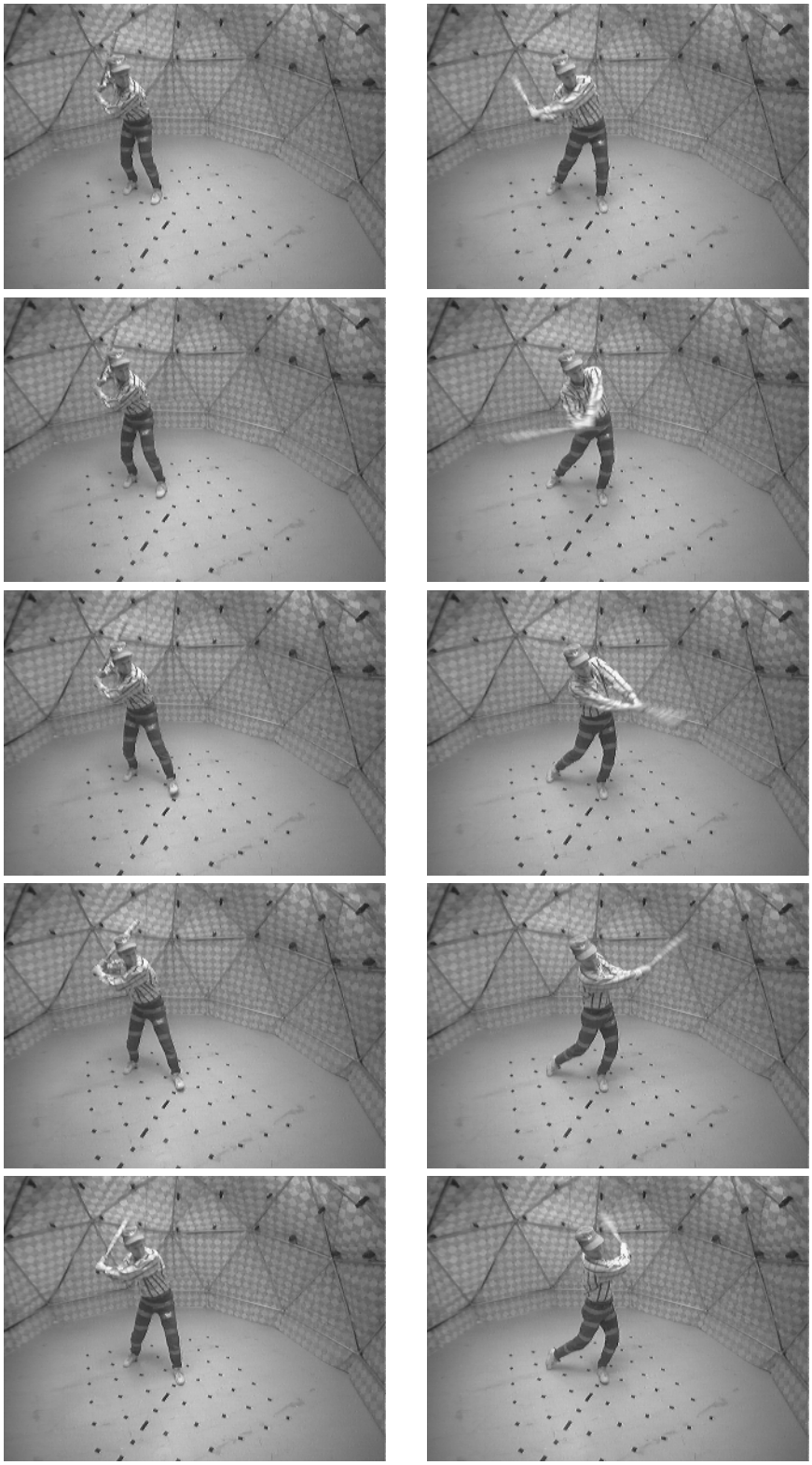


Figure 5.1: Sequence of 10 baseball images from one camera viewpoint. Time moves top to bottom, then left to right.

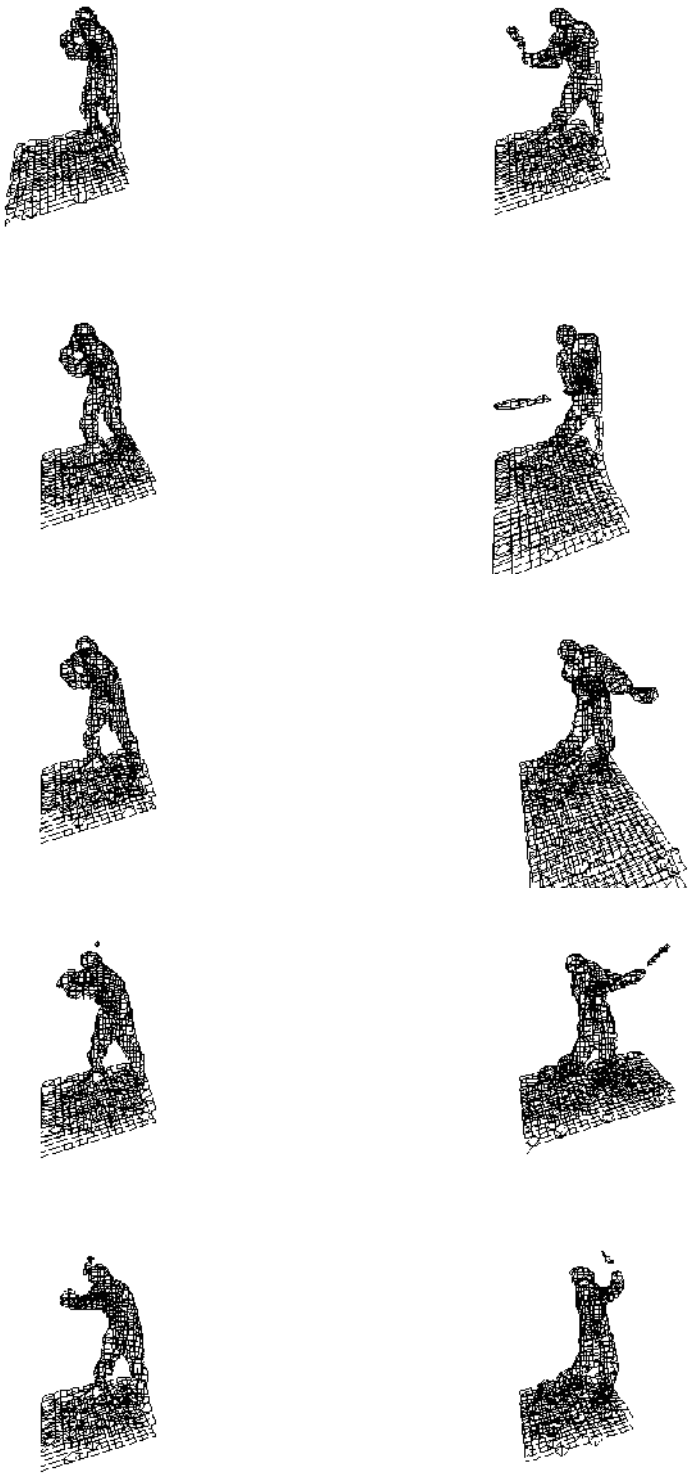


Figure 5.2: Sequence of 10 baseball models from one viewpoint. The models are displayed as wire-frame meshes. Time moves top to bottom, thenleft to right.

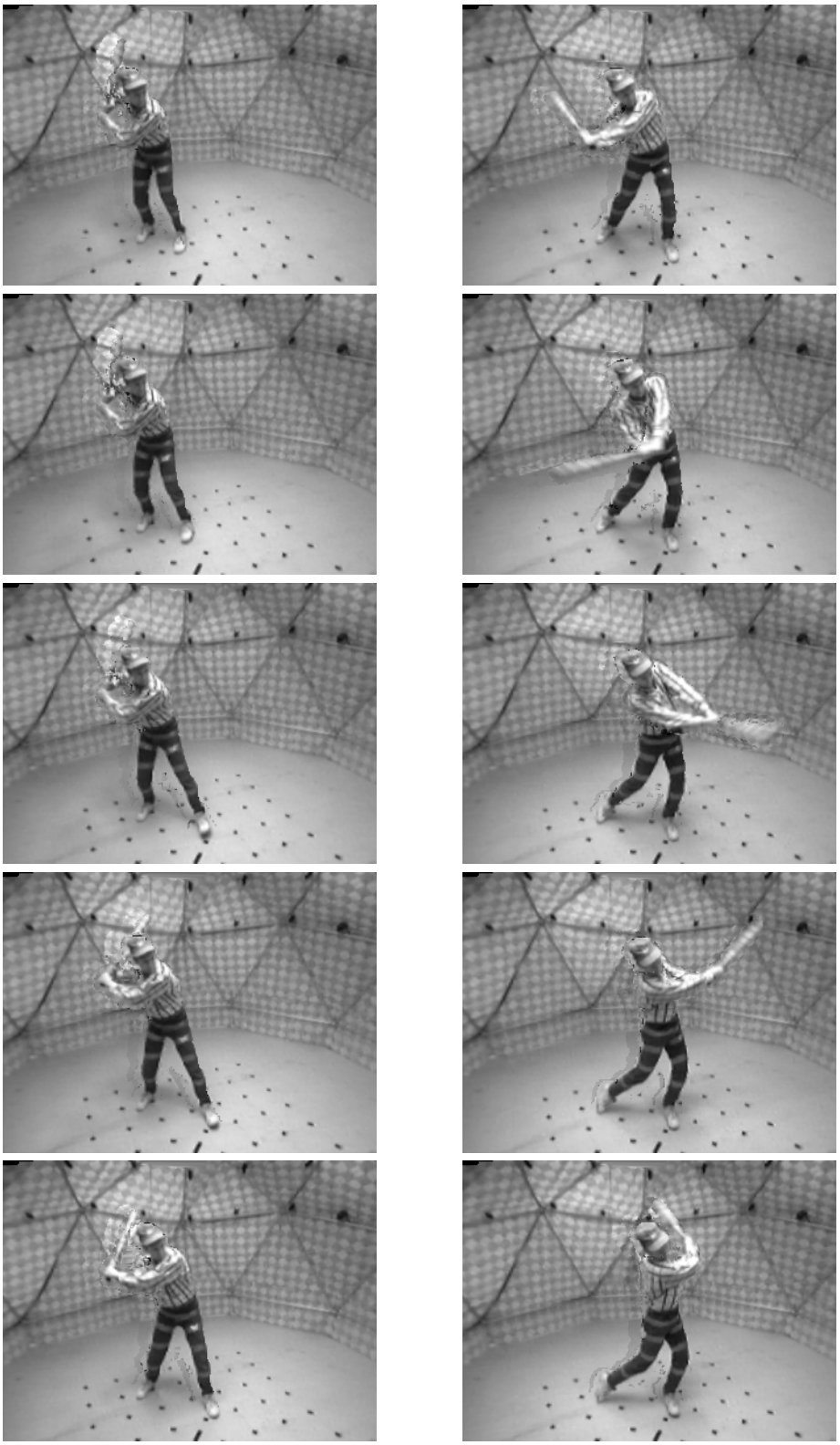


Figure 5.3: Sequence of views from a static viewpoint, different from the input viewpoints. Time moves top to bottom, then left to right.

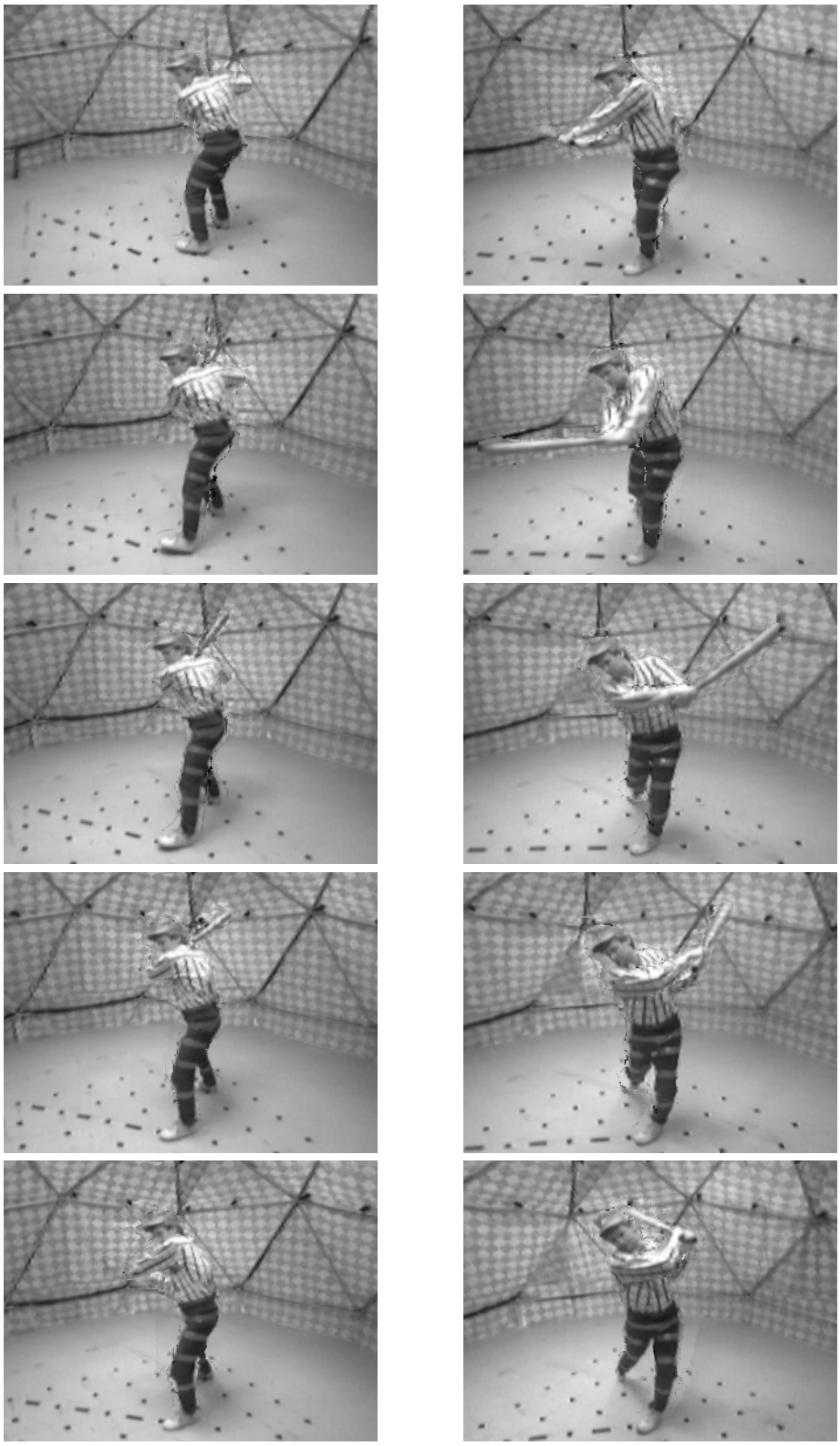


Figure 5.4: Sequence of views from a camera moving around the scene. Time moves top to bottom, then left to right.

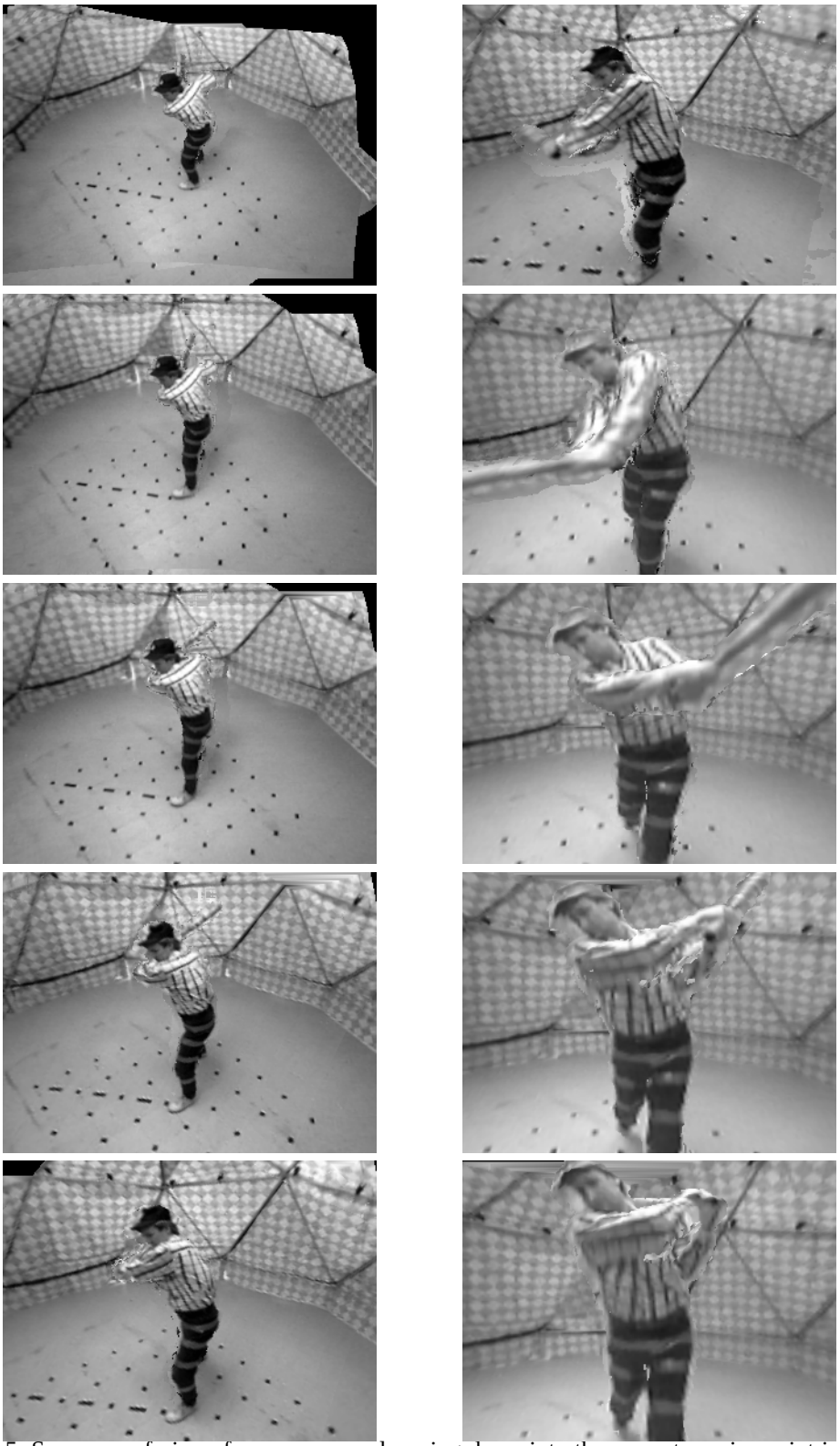


Figure 5.5: Sequence of views from a camera dropping down into the scene to a viewpoint just below the trajectory of the bat. Time moves top to bottom, then left to right.

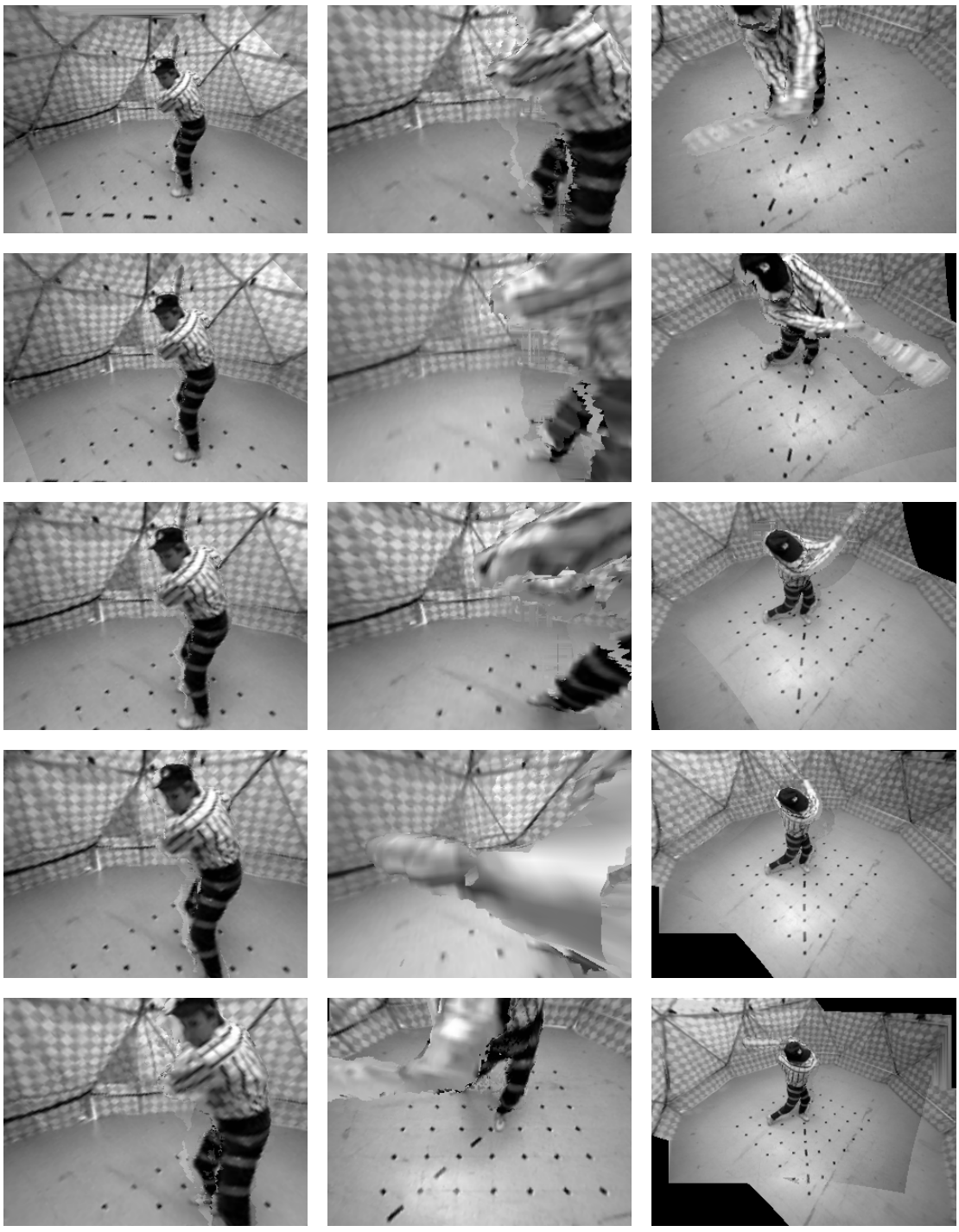


Figure 5.6: Sequence of views from a camera moving along the trajectory of a (virtual) baseball thrown at the batter and hit into the air. Time moves top to bottom, then left to right.

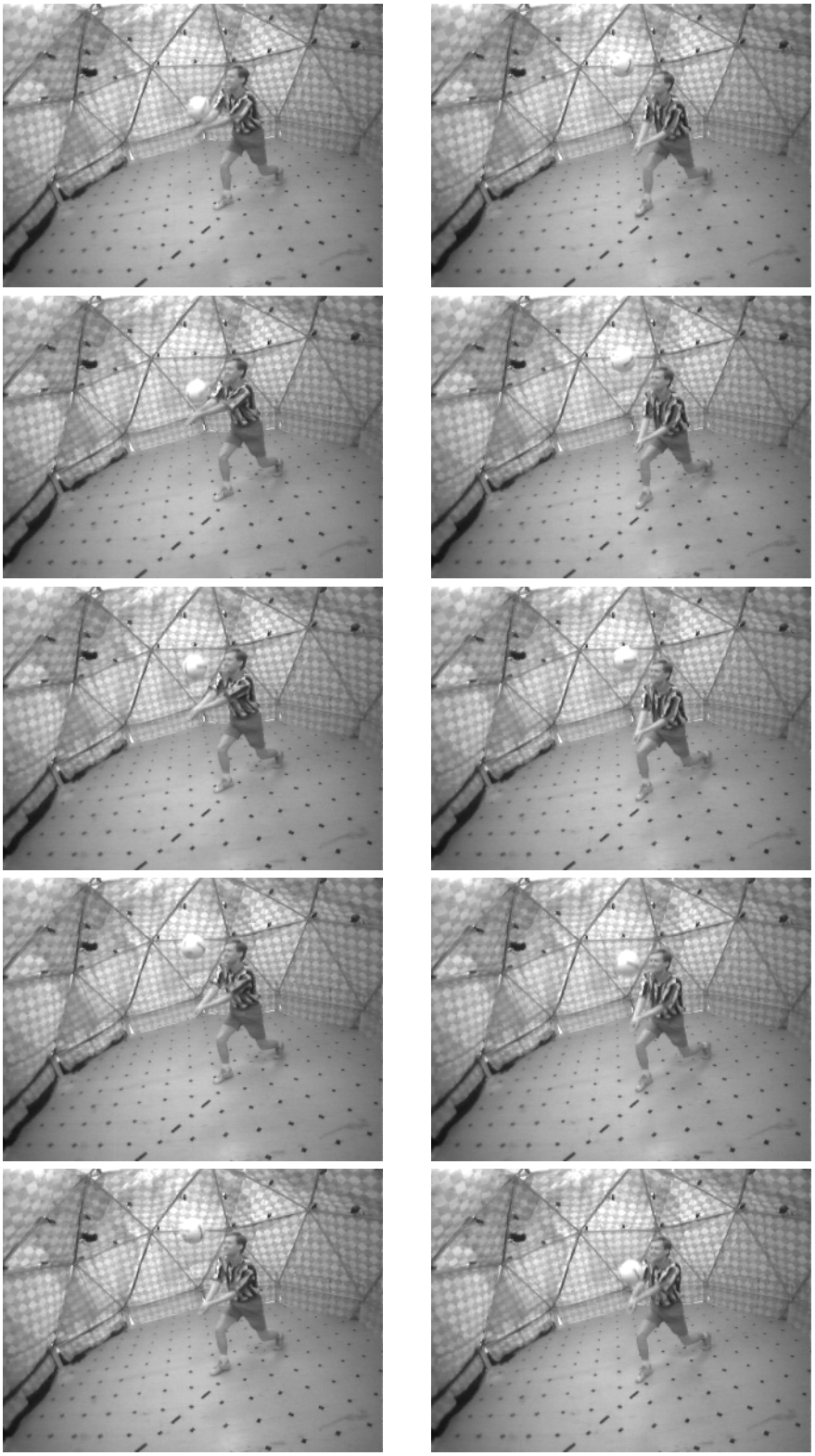


Figure 5.7: Sequence of 10 real volleyball images from one camera viewpoint. Time moves top to bottom, then left to right.

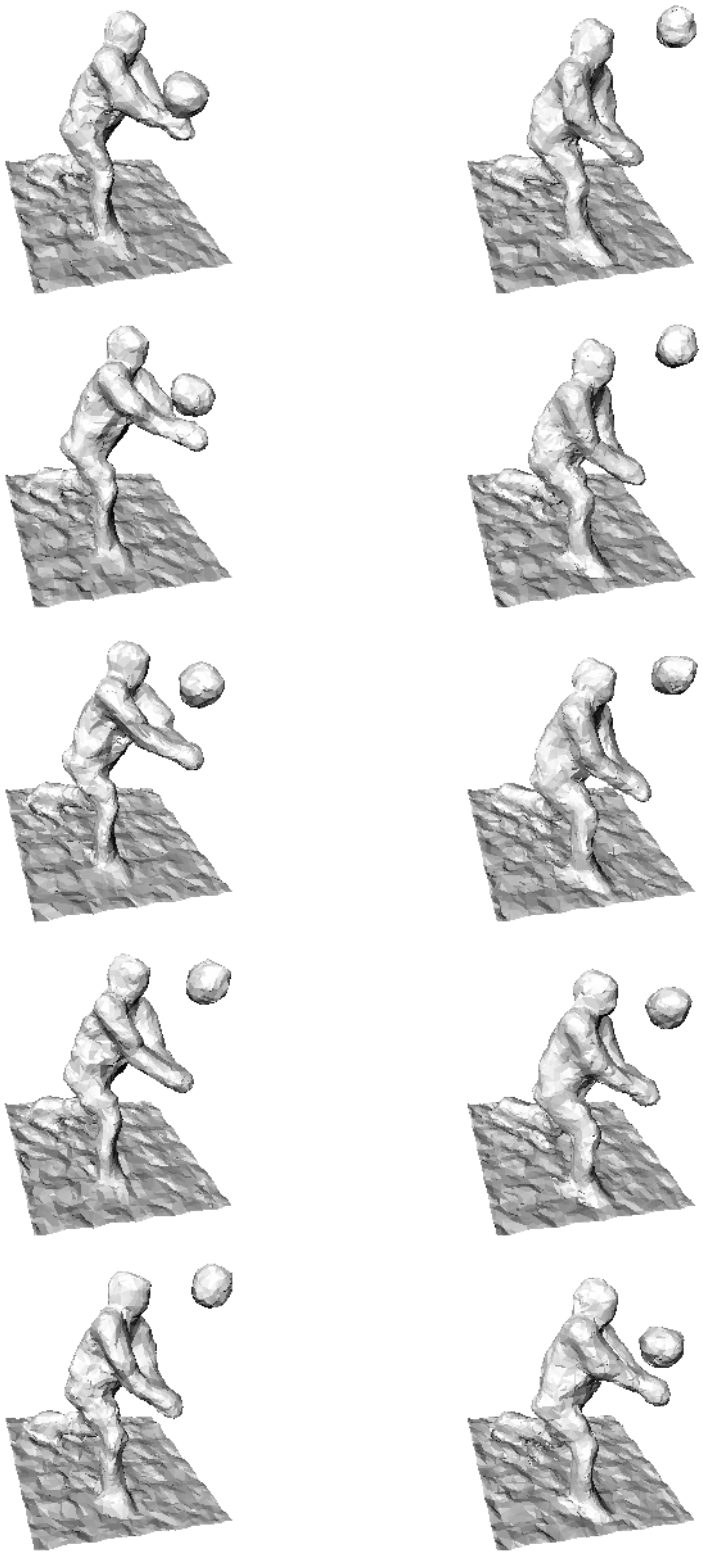


Figure 5.8: Sequence of 10 volleyball models from one viewpoint. The models are displayed displayed as shaded 3D models. Time moves top to bottom, then left to right.

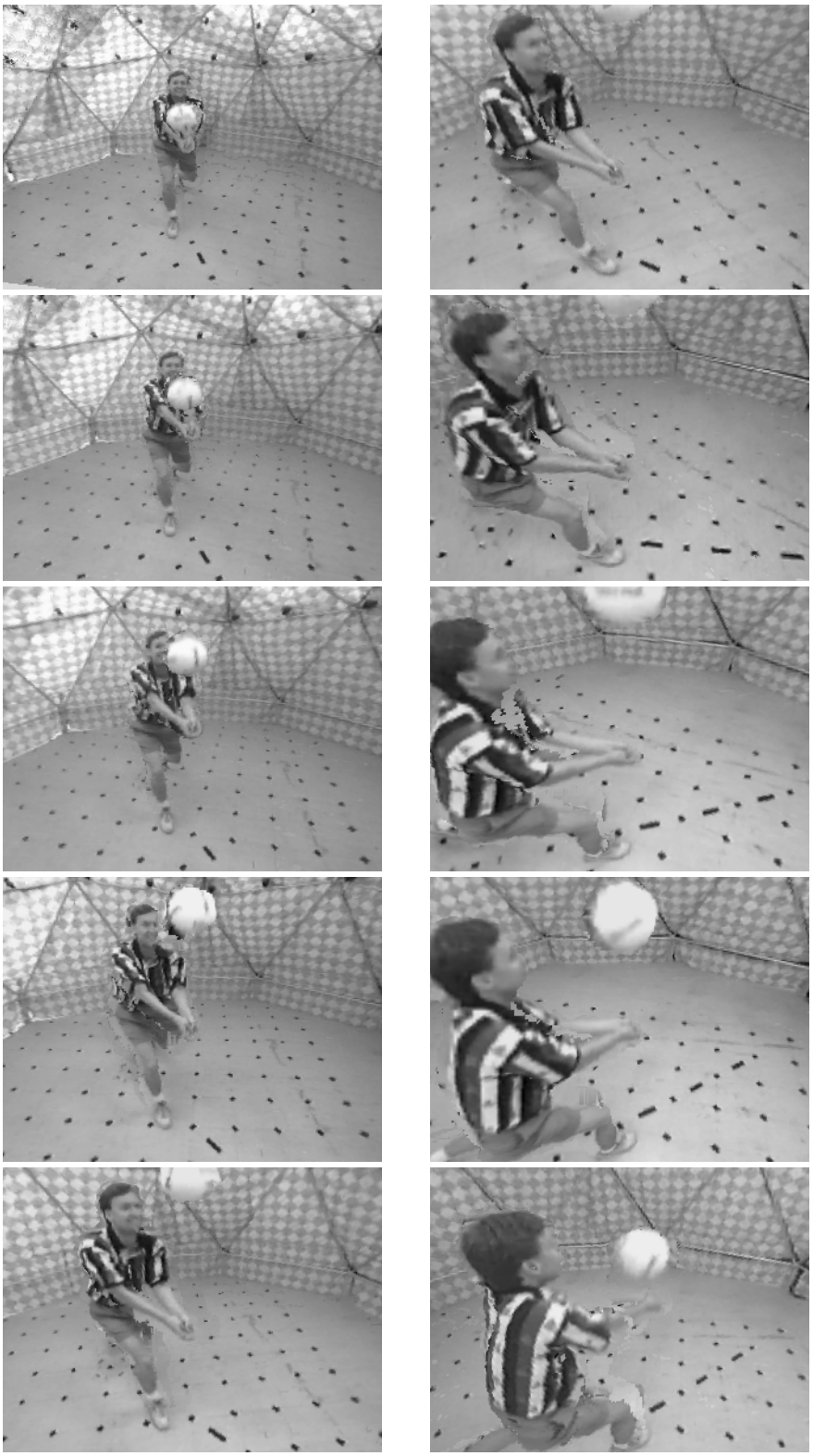


Figure 5.9: Sequence of views from a camera flying past the player. Time moves top to bottom, then left to right.

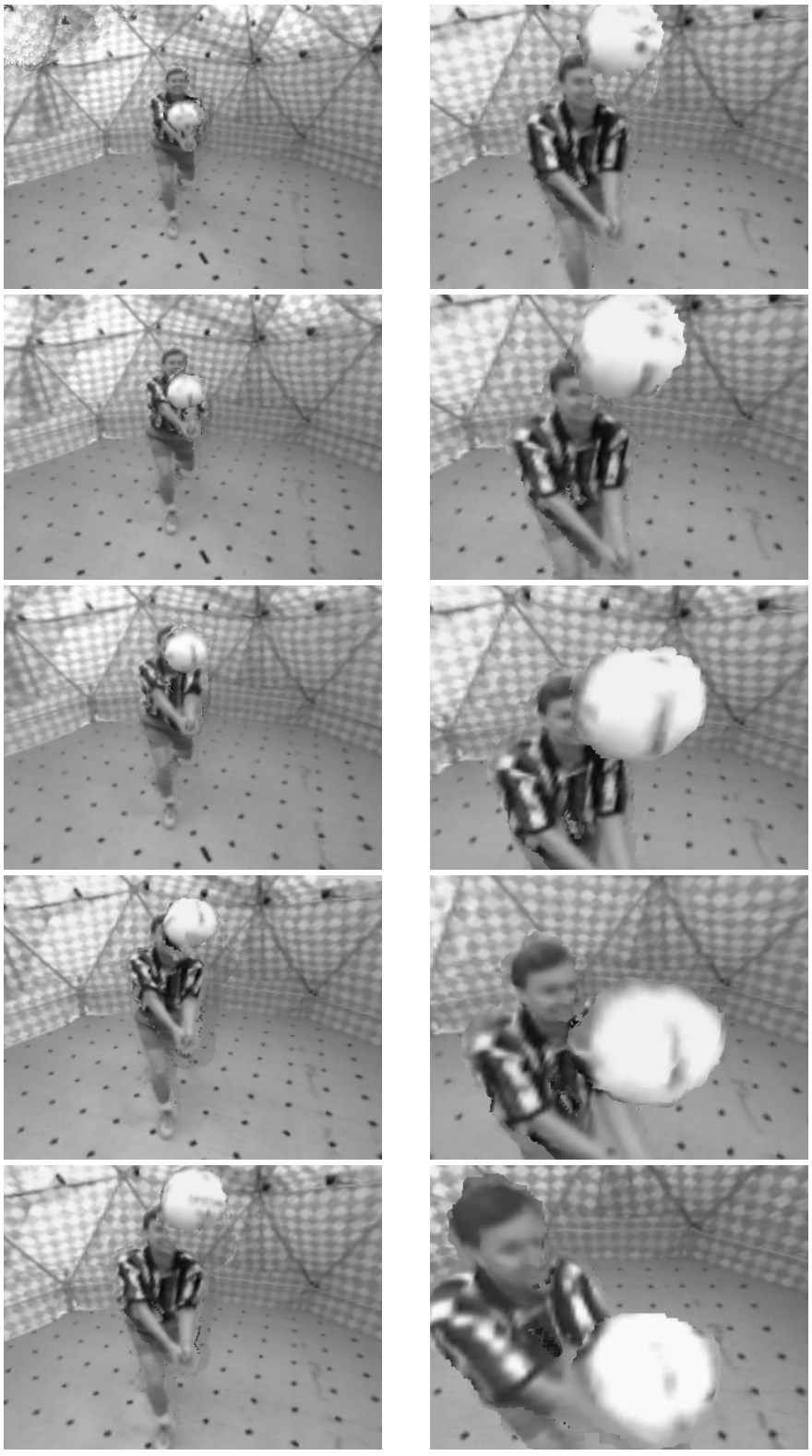


Figure 5.10: Sequence of views from a camera flying very close to the player. Time moves top to bottom, then left to right.

5.1.3 Example 3: Basketball I

The third example contains two people playing basketball. Figure 5.11 shows one real viewpoint with 15 of the 28 frames in this sequence. The full sequence was sampled at a rate of 15 frames per second. Figure 5.12 shows shaded renderings of the corresponding 3D models. No shape refinement was performed. Once again, the modeling process captures the main surface structure of the scene, but the occluding contours are poorly preserved.

5.1.4 Example 4: Basketball II

Given the difficulty of recovering the occluding contour, we experimented with the inclusion of hand-edited segmentation of these contours. These contours separated the foreground scene content from the background environment. The contours were projected out into the volumetric space before the isosurface extraction process was applied. Any voxel lying outside the foreground region was reset so that the isosurface extraction process would treat it as lying outside the isosurface. Voxels within the foreground mask were left untouched. This process essentially uses the contours to carve out portions of the model that extend beyond the contours, while allowing concave surfaces to remain untouched. This process was used in the next two examples.

One of these examples contains another sequence with two people playing basketball. Figure 5.13 shows one real viewpoint of the 15 frames in this sequence, sampled at a rate of 15 frames per second. Figure 5.14 shows shaded renderings of the corresponding 3D models. These models were recovered using the shape refinement method of re-computed range images, followed by the contour carving approach, which used contours in 11 of the 51 input views. In this example, the contouring processing affects two areas: the gap between a player's arms and his body, and between his legs. These gaps often are too narrow for stereo to see through, resulting in a closed surface. The contours force the gap to reappear.

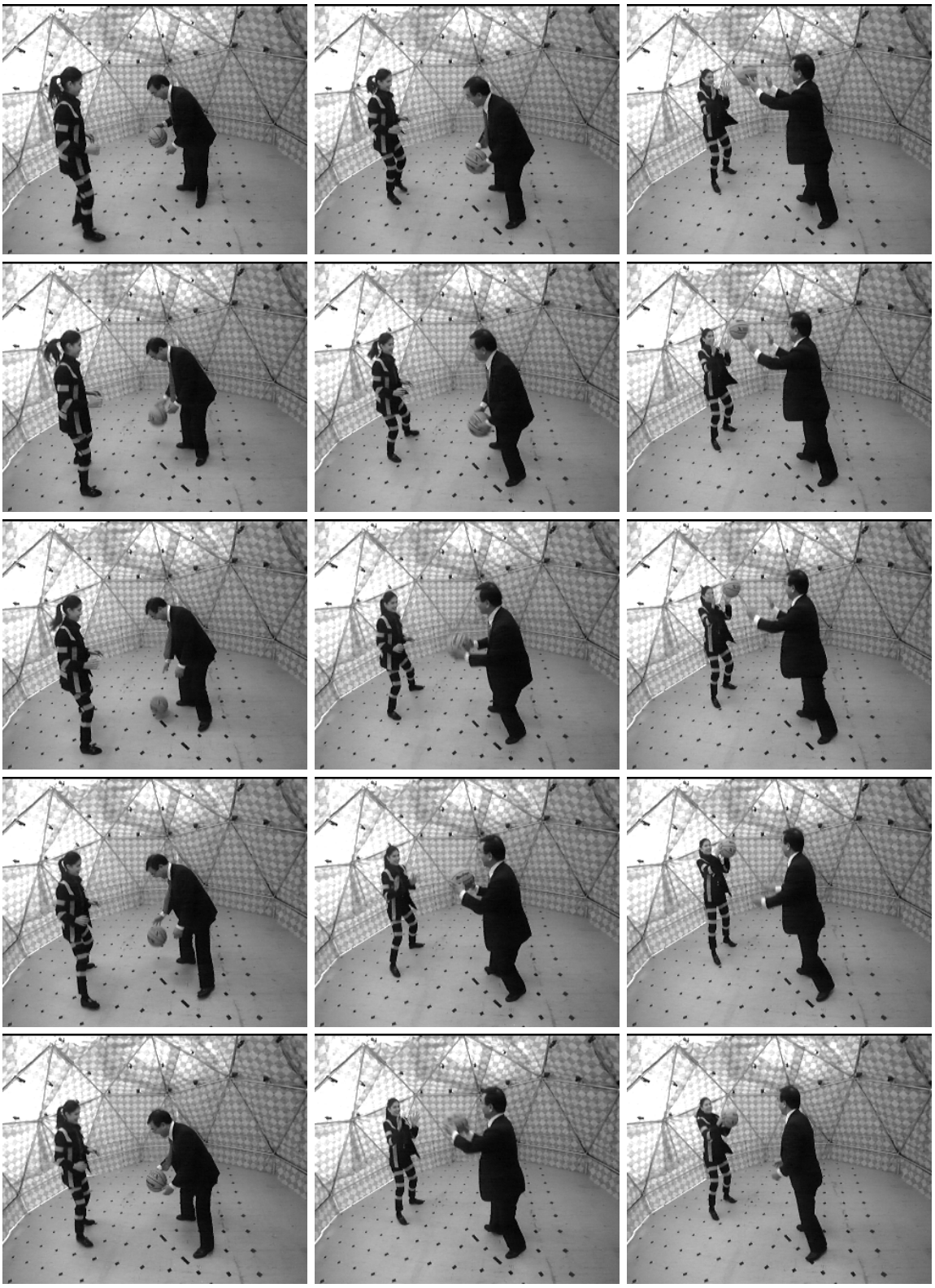


Figure 5.11: Sequence of 15 real basketball images from one camera viewpoint. Time moves top to bottom, then left to right.

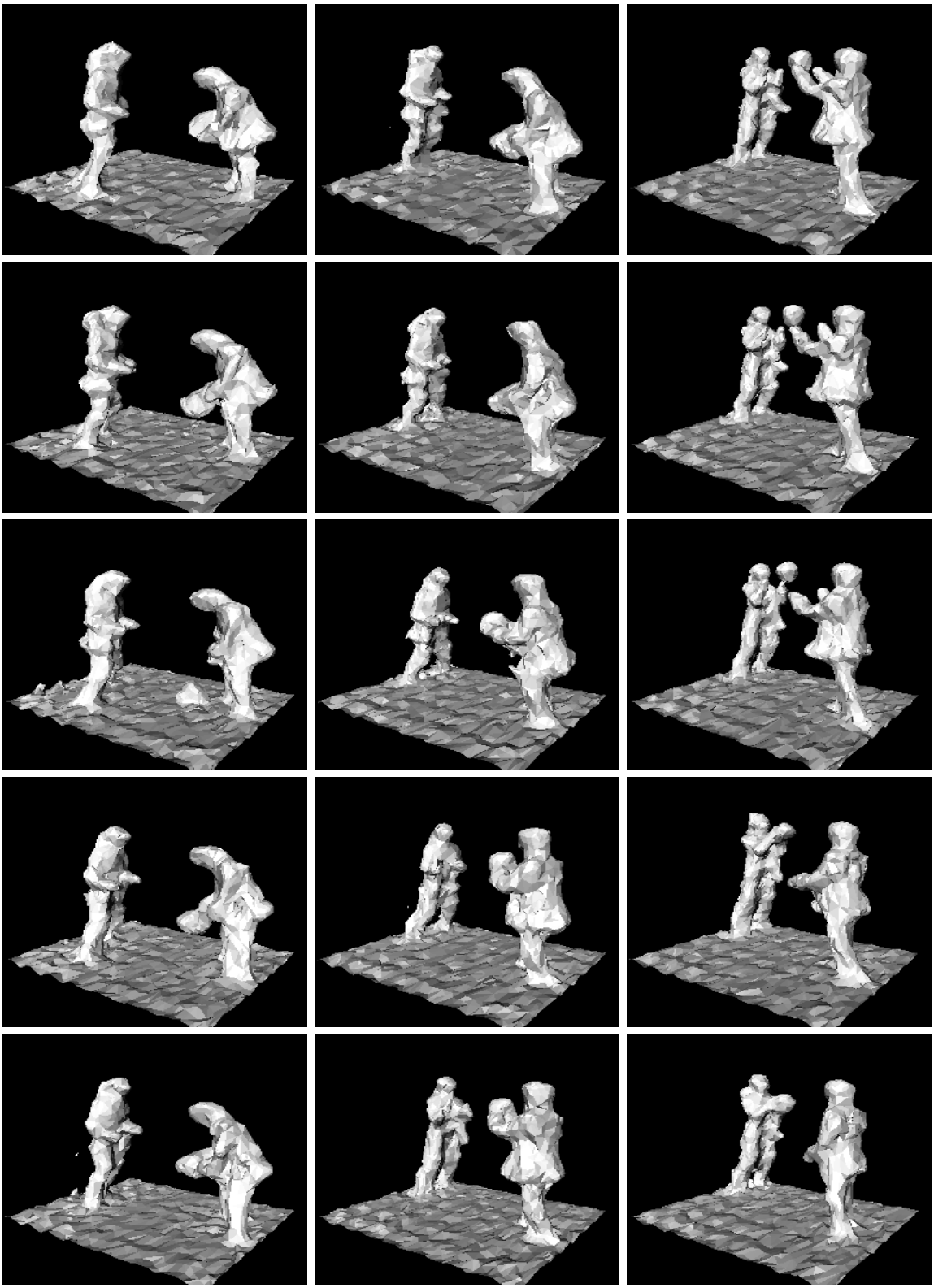


Figure 5.12: Sequence of 15 basketball models from the full 28-frame sequence, displayed as shaded 3D models, from one viewpoint. Time moves top to bottom, then left to right.

Figure 5.13: Sequence of 15 real basketball images from one camera viewpoint. Time moves top to bottom, then left to right.

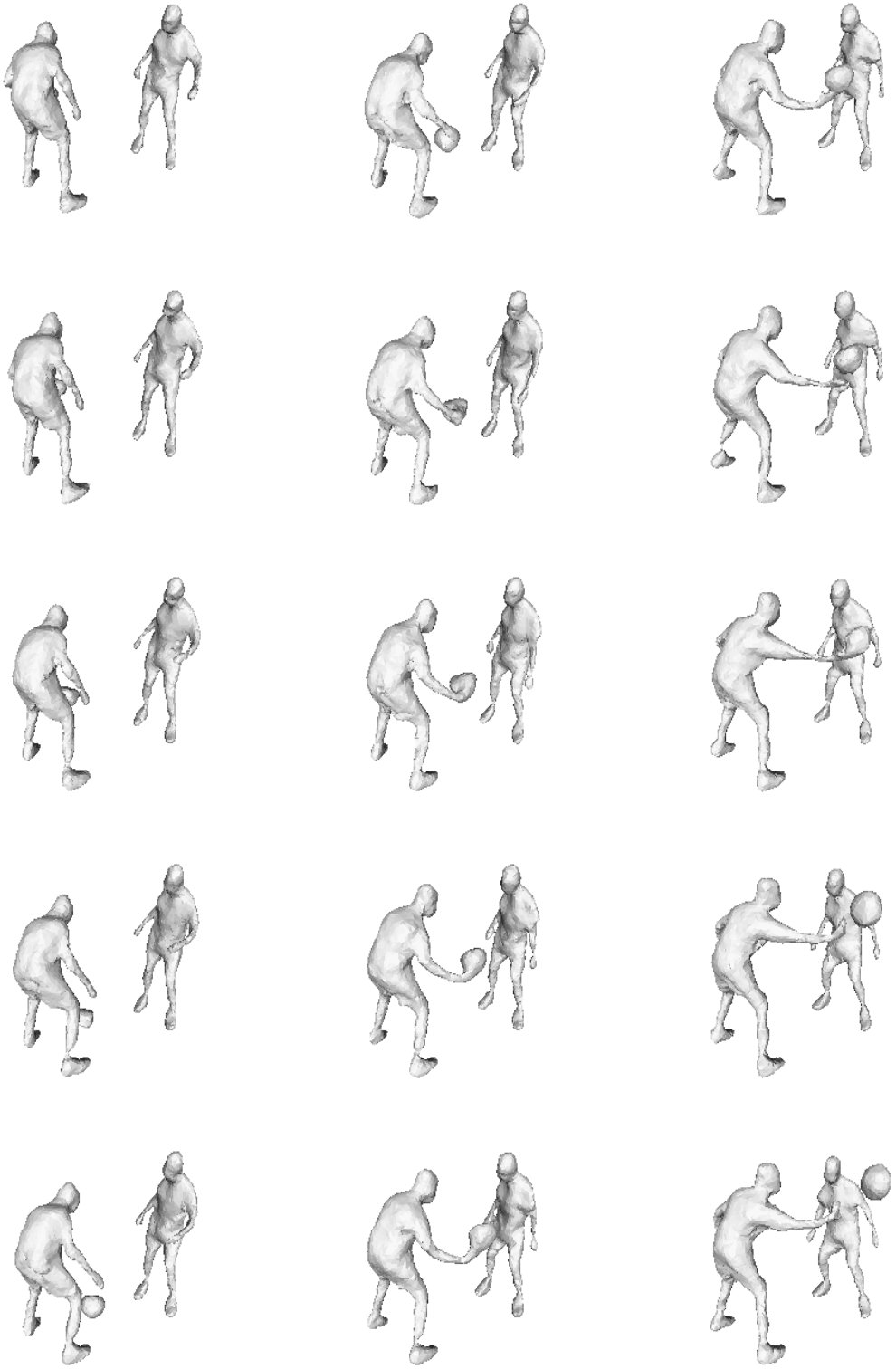


Figure 5.14: Sequence of 15 basketball models from one viewpoint. The models are displayed as shaded 3D models. Time moves top to bottom, then left to right.

5.1.5 Example 5: Basketball III

The last example shows a single basketball player catching a basketball thrown in from a player outside the modeling area. Figure 5.15 shows one real viewpoint of 15 frames in this 24-frame sequence, which was sampled at a rate of 15 frames per second. Figure 5.16 shows shaded renderings of the corresponding 3D models. These models were recovered using the shape refinement method of re-computed range images and the contour carving approach just discussed. The contour carving process uses contours from 11 of the 51 views. As with the previous example, the contours help separate the legs, and the arms from the body.

5.2 Integration with Virtual Models

Beyond merely re-displaying events from arbitrary viewpoints, 3D digitization of dynamic events allows the event to be integrated with other virtual models. For example, a virtual baseball can be added to the scene shown in Figure 5.1, to create the illusion that the batter actually was swinging at the object. Another example is to combine two digitized events in the same 3D space to create a larger space with more action, or events can be concatenated just as traditional video editing concatenates independent video clips. This section explores different ways in which these effects can be implemented as well as the requirements for making these effects realistic.

5.2.1 Combining Real and Virtual

The event shown in Figure 5.1 contains a baseball player swinging a baseball bat. To make this event more interesting, we would like to combine the digitized model with a virtual baseball to create the illusion that the player was swinging at the baseball. In order to achieve this effect, the ball must be inserted with the appropriate scale and must move in a trajectory that is consistent with a ball flying through the air. If the ball is to make contact with the bat, then the trajectory must intersect the bat trajectory and then change directions in a realistic way. We hand-adjusted these parameters to create the sequence in Figure 5.17. The ball is colored red to highlight its addition into the scene. From the figure, it is

Figure 5.15: Sequence of 15 real basketball images from a 24-frame sequence for one camera viewpoint. Time moves top to bottom, then left to right.

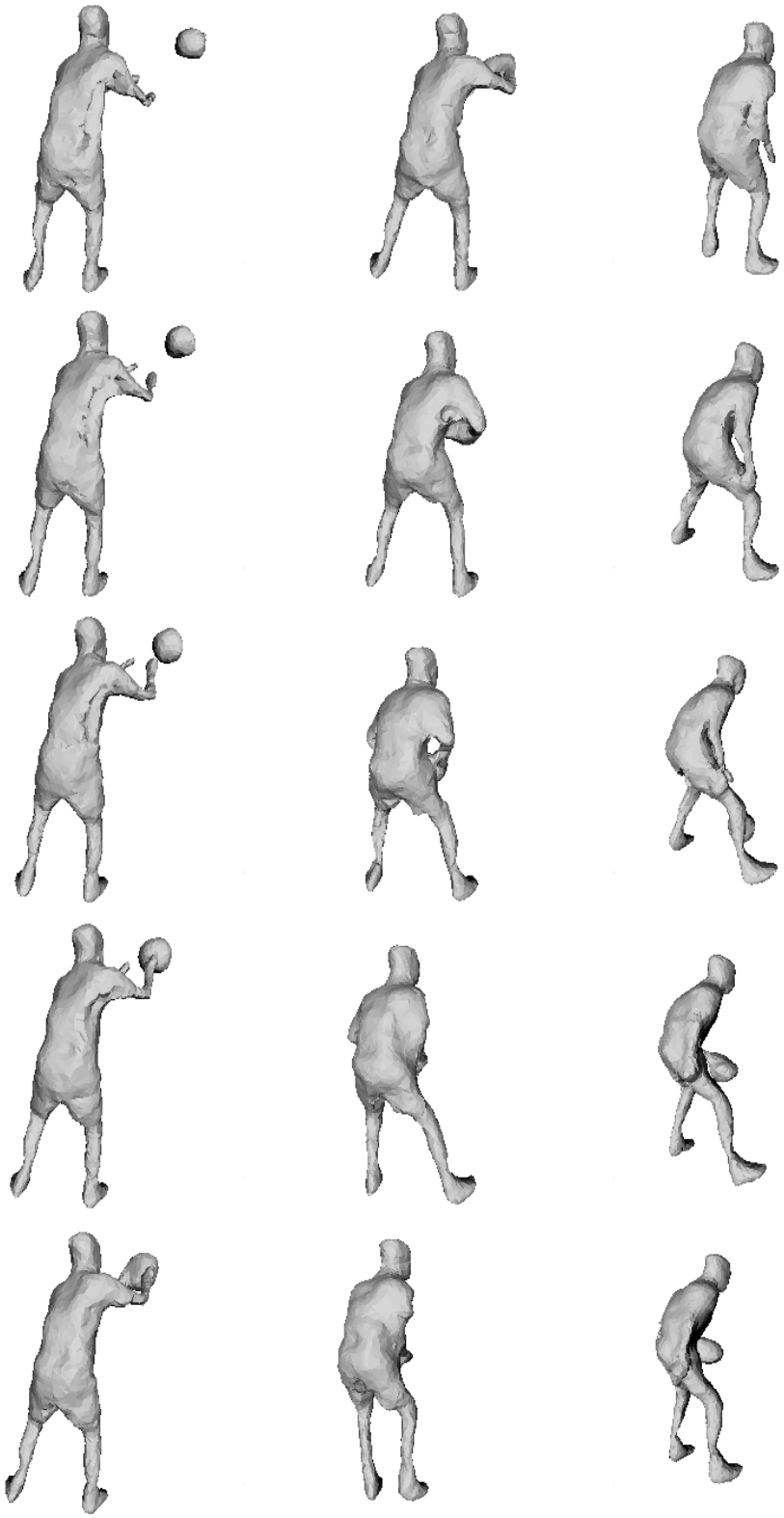


Figure 5.16: Sequence of 15 basketball models from the full 24-frame sequence, displayed as shaded 3D models, from one viewpoint. Time moves top to bottom, then left to right.

clear that to obtain more realistic images, the ball should be lit in much the same way that the digitized event was, and that the motion of the ball should create motion blur in the images, just as the bat does.

5.2.2 Combining Real and Real

Combining real digitized events into a single event can create powerful new visual effects. As with standard 2D video cuts, events can be concatenated to create longer events. In addition, multiple events can be placed next to one another – the equivalent of split-screen techniques in 2D video – or even inter-mingled, much like chroma-keying attempts to do. In each of these cases, the 3D digitized event provides far more power than the 2D equivalent because of the explicit encoding of visibility (i.e., 3D shape) and the ability to smoothly alter viewpoints.

To demonstrate these capabilities, we combined the two-player event in Figure 5.13 with the single player in Figure 5.15. In the event of Figure 5.13, the basketball flies away from one of the players, while in Figure 5.15, the player appears to catch a pass from someone. In typical 2D video editing, these two events could be put together to create the *illusion* that the toss from event 1 is a pass to the player in event 2. This cut, however, would never be able to show both players at the same time. With the 3D digitized event models, on the other hand, the two events can be placed side by side in the same virtual space and concatenated in time to create a more complete illusion. To place them in the same virtual space, we had to introduce a 3D rotation and translation of the coordinate systems so that the ball moves in a reasonable trajectory. In order to get the timing right, we froze the 3D model of one event while the other one was moving. The result is shown in Figure 5.18.

5.2.3 Combining Real and Real and Virtual

The combination of multiple digitized events can also be complemented with the addition of virtual objects. In the previous example combining two basketball events, for example, we can add a virtual basketball court for the players to play on. This effect is shown in Figure 5.19. Note that this virtual set was designed simply to show the ability to integrate

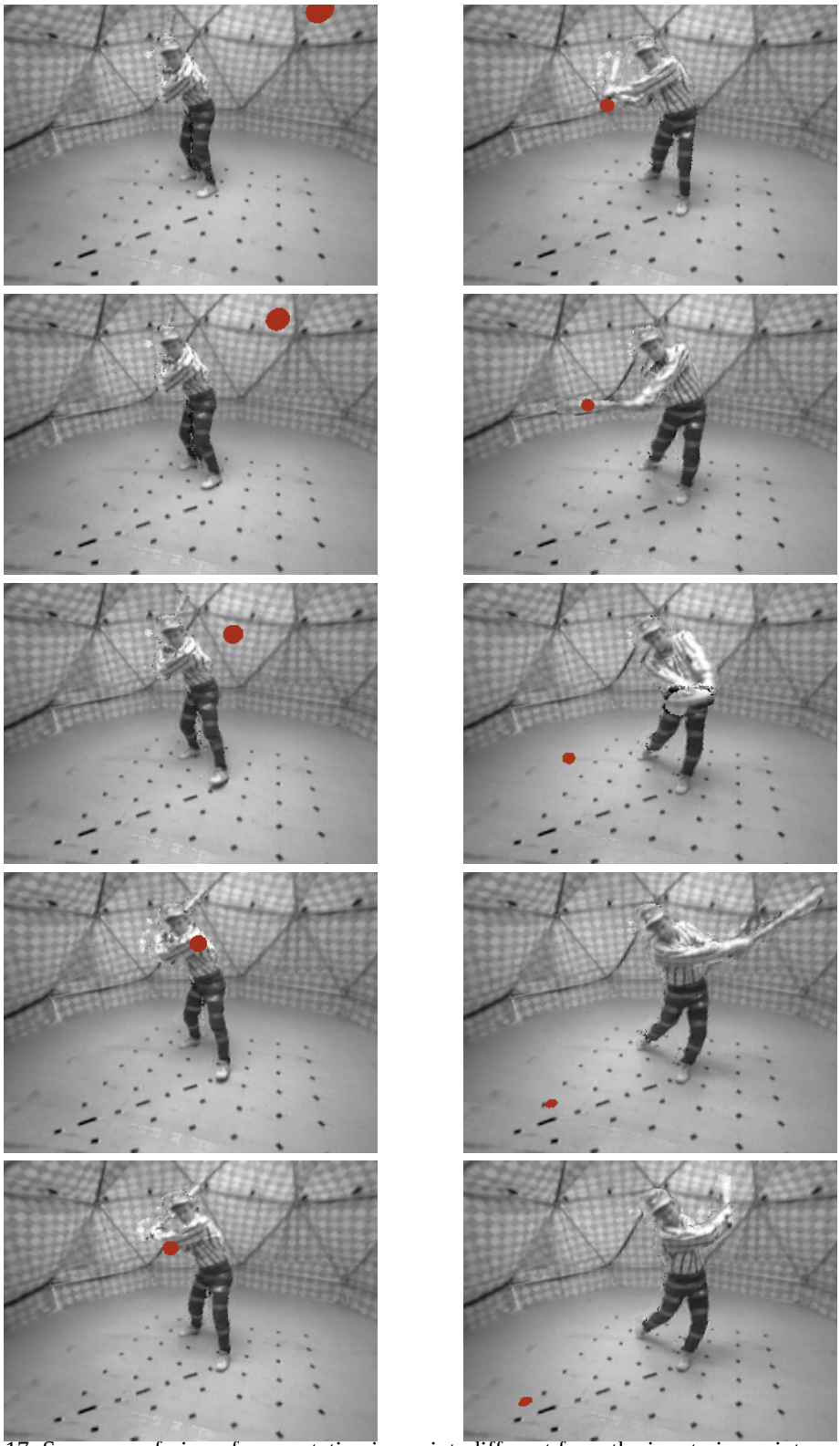


Figure 5.17: Sequence of views from a static viewpoint, different from the input viewpoints, with a virtual baseball inserted into the event. The ball is pitched to the batter, hits the bat, then rolls on the floor. Time moves top to bottom, then left to right.

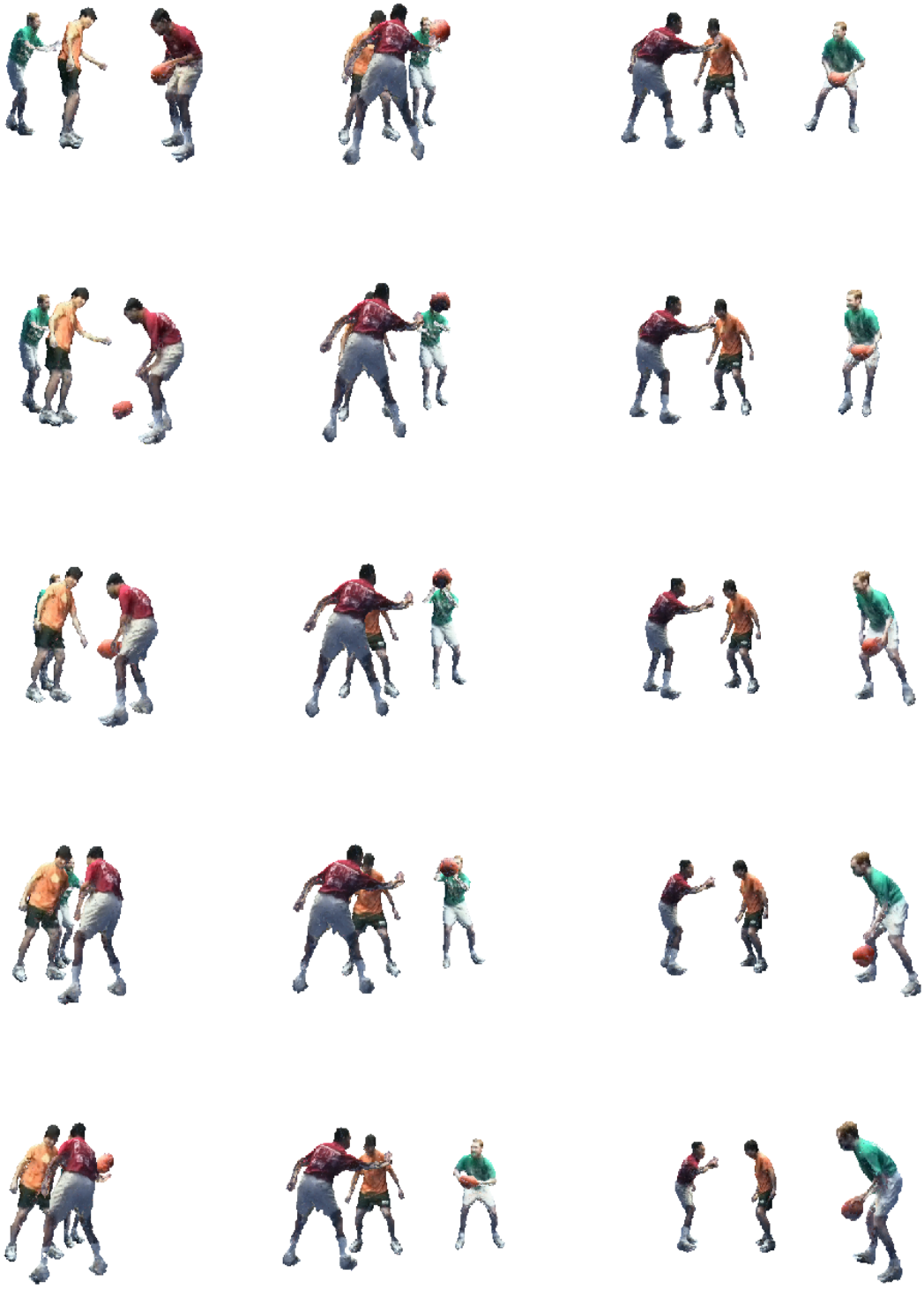


Figure 5.18: Sequence of 15 views from the full 39-frame sequence of a camera moving around the basketball players. The player in green and white was digitized separately. Time moves top to bottom, then left to right.

multiple digitized events with entirely virtual content; its realism can be improved by using a more detailed model. In order to be more realistic, the lighting of the virtual set should approximate the real lighting, and shadows should appear on the floor.

Figure 5.19: Sequence of 15 views from the full 39-frame sequence of a camera moving over the basketball players. The player in green and white was digitized separately, and the basketball arena is a virtual set. Time moves top to bottom, then left to right.

Chapter 6

Related Work

Elements of this thesis have relations to a number of different fields, each with large bodies of research. Part of these related efforts were introduced in Section 1.1 as part of the motivation for the methods presented in this thesis. This chapter explores this body of work in more detail to place this thesis in a broader context. We divide the related work into three broad categories: 3D modeling from range data, 3D modeling from visual data, and image-based modeling.

6.1 3D Modeling from Range Images

One problem that has received much attention is the construction of global shape models from a set of range images. Until recently, most approaches addressed this task as a mesh stitching problem: given a set of 3D meshes, piece these meshes together to form one large "mosaic" mesh. Methods by Turk and Levoy [65] and by Soucy and Laurendeau [57] are among the most successful of this approach. The challenge in this framework is resolving topology of the underlying model and integrating overlapping estimates into a high-precision mesh. These methods do not respond well to the presence of significant noise and gross errors.

The method presented in this thesis is based on the concept of volumetric integration. Each sample of the volumetric space estimates the signed distance to the surfaces in each range image. Four other approaches have used similar strategies. The closest algorithm to our own in the one of Curless and Levoy [10]. Both methods compute signed distance along the line of sight of the camera. This approach utilizes surface observations to carve out free space between the sensor and the estimated surface. This approach also allows iterative updates to the volumetric space simply by adding in the contributions from new viewpoints. This property makes the method amenable to parallel implementation, since each voxel can be updated independently of all others.

The main difference between the two methods is the way space is carved. Curless and Levoy assume that the range data contains only additive noise, in which case all estimates are near the correct surface. Under this assumption, they carve empty space by adding a zero-weight signed distance contribution at each voxel far in front of the surface. The zero weight allows this contribution to be overwritten should any other estimate alter the value at a given voxel. In the presence of gross range errors and large noise, however, this method will fail to eliminate false surfaces. Our approach is to make actual weighted updates to each voxel far in front of the surface. In order to allow strong evidence from other views to override this single contribution, we clip the weighted, signed distance.

Three other volumetric approaches compute the signed distance in object space, independent of the viewpoints from which the data came. These methods compute the surface normal of the range data and measure distance along this direction. Hoppe et. al. [20], using unorganized 3D points, first had to estimate local surfaces, and then updated the volume based on that local evaluation. Hilton et.al. [19] and Wheeler [67] both work with surface estimates, so the surface normal is computed directly from that data. Hilton’s algorithm requires only a single estimate of surface structure to update the volume, and so it is likely to degrade in the presence of range errors. Wheeler’s Consensus Surface algorithm, on the other hand, requires consistency among several surfaces in order for a voxel to be labeled. This method is much more robust to noise, then, since false surfaces are basically ignored.

Setting the threshold for consensus is problematic, though, and the algorithm has proven to be sensitive to this value. All of these methods that use surface normals suffer when the data is noisy, since the computation of the normals amplifies the noise.

Finally, Martins and Moura [36][37] present a volumetric framework with Bayesian updating of volumetric tessellations of space, without implicit surface extraction. The lack of isosurface extraction requires the use of volume rendering techniques to avoid sampling problems. Given the debate about the relative merits of surface rendering vs. volume rendering, this point may not be a true limitation. However, a major limitation in explicit use of the volumetric space is the resolution available for a given amount of memory, since the volume grows much faster than the surface area. Still, the framework makes explicit use of sensor models and provides a well-founded Bayesian updating process for the occupancy at each voxel. Proper selection of the sensor model may allow this approach to handle large amounts of range image noise, and is worthy of further research.

6.2 3D Modeling from Video Images

The methods of the previous section assumed that range data was already available, usually captured with various “direct” range capture systems such as laser scanners and light-stripe range finders. In this thesis, we actually construct range estimates from stereo, and then use these range estimates to derive global structure. A similar strategy was used by Kang and Szeliski [27]. They capture panoramic images from multiple viewpoints and applied stereo to these images. Stereo in this framework has worked fairly well, especially in the highly textured environments that they used. In addition, the epipolar geometry can be estimated very well since the field of view is so large. For dynamic scenes, however, the panoramic imaging technique of stitching perspective views together will not work. In addition, this method by itself does not resolve the problem of visibility into a complex scene. Debevec and Malik [11] use an interactive modeling system to develop high-quality 3D models of architecture. This system takes advantage of the fact that the scenes contain architecture, allowing the use of relatively few, simple primitives to represent the scene.

They then use model-based stereo to refine the alignment between the hand-drawn model and real images of the scene. The amount of human interaction required makes this approach unattractive for dynamic event modeling.

An alternative to this two-step process is to directly reconstruct global shape from images. Three general approaches to this problem are Shape from Motion (SFM), space carving from occluding contours, or silhouettes, and voxel color consistency. In SFM methods, usually a single camera is moved through a static environment to create a model of the scene. Because the camera is moving relatively slowly, the inter-frame motion is usually quite small, so feature-tracking algorithms can be used to track features through long camera motions. These features can then be used to estimate shape, for example using the Factorization method [63][46] or non-linear optimization [59]. These approaches usually work only with sparse features, and therefore face the challenge of finding surfaces to connect the feature points. In addition, with a limited number of sensors, the feature tracking problem becomes more difficult because of the large viewpoint change between images.

In Immersive Video [22][30][41], intersecting foreground masks carve out 3D models of objects moving within a known background. This strategy is similar to fairly well-known approaches to modeling by silhouettes [1][9][58][60]. It has been shown that these method can only extract the line hull of the scene. For example, even perfect silhouettes of a coffee cup would lead to a model with a closed top, since no silhouette could distinguish the inside of the cup. Another challenge in a general scene is the extraction of the occluding contours. Immersive Video has attempted to use image motion to extract these regions, but motion detection fails to capture exact occluding contours.

The last class of methods for direct shape from images is the use of color consistency. Khalili [31] proposed the use of color variance at each voxel to determine occupancy. His algorithm tested each voxel by projecting it into all the images to get a set of colors. If the voxel is occupied, then the different observations should have low variance. If, on the other hand, the voxel is unoccupied, then the variance will be high. Processing all voxels this way and then thresholding, Khalili generated occupancy grids useful for navigation. Seitz and Dyer [55] devised a similar strategy, but added a constraint to help deal with vis-

ibility. By positioning all cameras on one side of the 3D volume, the voxels can be traversed in a front-to-back order so that visibility can be used to eliminate projections that are known to be occluded. Like many correspondence-style methods, these approaches are sensitive to specularities and regions of low texture. In addition, with the Seitz and Dyer algorithm, mistakes in occupancy at one voxel layer can have a tremendous impact on the results of voxel layers further away. Finally, the color consistency test is only a sufficiency test, unlike stereo algorithms which search for an optimal answer. As a result, the threshold level is critical to good performance.

6.3 Image-Based Modeling

Image-based models usually fall into two categories: explicit correspondence-based methods, and ray-based approaches. One of the simplest and yet surprisingly effective approaches is the object movie of Apple's QuickTime VR [5]. In this strategy, an object is modeled only by the images themselves, with knowledge of the viewpoints from which those images where taken. When a viewer manipulates the object, the system simply displays the image from the closest real viewpoint. The viewer can zoom into the scene as well, giving the appearance of motion into the scene, although perspective effects can not be correctly simulated. A typical object movie has several hundred images, with angular camera separation on the order of 10 degrees.

Ray-based methods can be thought of as generalizing the object movie in order to synthesize nearly any viewpoint rather than just switching among the images. First, these approaches convert the set of input images into rays in space, as shown by Katayama et. al [29]. This space can be modeled as a 4D function, commonly referred to as a lightfield [34] or lumigraph [15]. This lightfield is often parameterized by two 3D planes, with a ray defined by its two intersection points on the two planes. Next, the scene is assumed to be flat, and positioned on one of the planes of the lightfield. Using this formulation, new views can be synthesized by sampling the viewing rays that comprise the image. The flat-world assumption allows each unknown ray to be constructed by interpolating among known rays. If the actual object geometry is known, then the rays can be depth corrected

[15], creating much more realistic views, especially when fewer real images are available. Still, these methods have been demonstrated only with thousands of images, which would require a tremendous video capture system for dynamic events.

Correspondence-based methods use pixel-wise correspondences among images as estimates of scene structure, which then allow new view synthesis by mapping this geometry into the new viewpoint. From correspondences alone, this mapping was demonstrated by Chen and Williams [6] for synthetic imagery as a method of quickly approximating the rendering process on low-cost hardware. This was based on the developments of image morphing [3], which uses correspondences between two images of different objects to smoothly transition between them. Seitz and Dyer demonstrated that these views will not, in general, be physically correct unless the images are first rectified [53]. Related work by McVeigh et. al [39][40] and Siegel et. al. [56] in 3D stereoscopic image coding and display also showed how automatic disparity computations in digital video coding can be used to synthesize intermediate views.

Because of the formulation of these methods, though, the virtual camera is constrained to lie on the surface containing the real cameras. Laveau and Faugeras [33] demonstrated that arbitrary views can be synthesized by considering epipolar geometry between two real cameras and between each of those cameras and the virtual camera. Navigation was cumbersome, though, because the epipolar geometry had to be specified by identifying corresponding points between the desired viewpoint and the real images. Similar results using interactive modeling systems has been demonstrated by McMillan and Bishop [38], Seitz and Dyer [54], and Chen and Medioni [7]. More recently, the tri-linear tensor has also been shown to be an effective tool for mapping pixels from two known views directly into the virtual viewpoint [2]. Using fully calibrated cameras also allows free navigation, as Kanade et. al. demonstrated [24][25].

One common difficulty that has been ignored in many of these techniques is the difficulty in acquiring good correspondences. No matter how clever the display algorithm, gross errors in correspondence across large areas of the images will distort the synthesized views. For static scenes, shifting this process into a semi-automated system produces both

better correspondences and faster solutions, since a human can quickly determine correctness while the computer can determine precision. For dynamic scenes, however, interactive methods are cumbersome and laborious.

Chapter 7

Conclusions

This thesis has explored the 3D digitization of dynamic, real-world events using multiple video cameras. We have developed new algorithms for merging multiple range images and for modeling scene appearance from the images, and have incorporated support for arbitrary camera positions into the Multi-Baseline Stereo algorithm. We have also developed the *3D Dome*, a testbed for synchronously capturing multiple video streams of dynamic events, and validated the algorithms using this system. These efforts demonstrate that events can be modeled by multiple video sequences along with scene structure extracted from those videos. The first section of this chapter explains the main contributions of these efforts.

The success achieved within this system also points the way to continued development of modeling techniques. The improvement of refining shape in Section 3.5, for example, suggests that explicit use of surface structure during stereo would yield better range images, further improving the quality of 3D modeling. This improvement also suggests that any scene structure known a priori could be exploited to improve modeling. The evaluation of color in Chapter 4 suggests that the final model can be refined directly from the original images. Finally, explicit use of temporal consistency can improve model quality, inter-frame consistency, and efficiency of representation. The second section of this chapter explores these issues in more detail.

7.1 Contributions

This thesis makes two main contributions:

- Development of algorithms for the 3D digitization of dynamic, real-world events
- Implementation of a full-scale system demonstrating feasibility of the algorithms

The next two sections discuss each of these contributions in more detail.

7.1.1 3D Digitization Algorithms

One main contribution of this thesis is the development of algorithms for the 3D digitization of dynamic, real-world events. We have developed a novel method of determining 3D structure from calibrated video imagery. This method uses stereo to compute approximate scene structure from multiple viewpoints, and then uses a novel merging strategy for extraction of the global model. The method is linear in the number of cameras used, providing a scalable solution to the modeling problem. In addition, it has proven to be extremely robust even in the presence of large range errors. We have also shown how this model can be fed back into the stereo process to improve the quality of range estimates. In support of this structure recovery method, we combined the Multi-Baseline Stereo algorithm with a general camera model to allow range estimation from general collections viewpoints.

Given an approximate 3D model, we have shown how calibrated video images can be used to model scene appearance. We show that a global texture can be recovered by “mixing” the original views into a single texture. The quality of the texture is highly dependent on the function used to “mix” the original images, especially in the presence of errors in the surface shape. We also show how the original images can be used directly during rendering to better preserve the resolution of the input images. This method perfectly reproduces the original viewpoints while providing graceful quality reduction as the viewpoint moves away.

7.1.2 3D Digitization System

The second main contribution of this work is the development of a full-scale 3D digitization system. The video capture sub-system incorporates 51 video cameras in a hemisphere of viewing positions which cover the event space. In addition to addressing practical issues such as camera synchronization and video frame time-stamping, the development of this system also explored the trade-offs among resolution, workspace size, visibility, and system complexity. Using video captured with this system, the algorithms presented in this thesis were successfully applied. This complete, end-to-end system demonstrates that the algorithms developed for 3D digitization are in fact feasible.

7.2 Future Work

The approach presented in this thesis can be extended in a number of ways. The following sections describe a number of these possibilities. Just as the efforts summarized in this thesis span a full end-to-end system, so too do the extensions, from calibration up through view synthesis.

7.2.1 Weak Calibration via Correspondences

The calibration of the recording system currently uses a known 3D object placed at known 3D positions in the space. While this process has proven effective, it has two important drawbacks. First, the calibration will suffer as the quality of 3D positioning decreases. Second, the process itself is cumbersome, and some applications may not allow the use of such a procedure. Recent efforts at weakly calibrating cameras suggest that it may be possible to eliminate the need for this type of procedure. These methods use only corresponding points between images to determine the epipolar geometry relating the two images. The advantage of such approaches in stereo is that the epipolar geometry is precisely the information needed to efficiently search the images. Full calibration methods, on the other hand, fit the camera model to optimize the relationship between the (possibly noisy) 3D structure and the images. As a result, the epipolar geometry may be lower in quality com-

pared to that of weakly calibrated systems. Once the system is weakly calibrated, knowledge of several 3D constraints (such as distances and angles) allow the recovery of full Euclidean calibration.

The main stumbling block to using weak calibration in our system is lens distortion. All weak calibration algorithms assume that the camera is well modeled by a pinhole camera. Lens distortion violates this assumption, and with the amount of distortion in our images, the quality of weak calibration is far too low to be useful. Recent work by Collins [personal communication] has shown that observations of a set of straight lines can be used to calibrate lens distortion directly, without concern for focal length or extrinsic camera parameters. This algorithm can therefore be applied before weak calibration to pre-correct for radial lens distortion. With undistorted data as input, the weak calibration process becomes much more effective.

7.2.2 Surface-Based Stereo

The shape refinement of Section 3.5 took one step toward a more complete approach to range refinement. To go further, the explicit surface being refined can be modified directly, rather than sampling the surface on the image grid. Explicit use of surfaces can allow incorporation of very wide baselines, since visibility can be directly determined from the surface description. In addition, this formulation seems well suited to refining the occluding contour, since changes in the contour change visibility. This approach could still search the space as with stereo, or it could be formed as an optimization problem to take advantage of gradient descent techniques.

7.2.3 3D Model Refinement

The surface optimization framework from the previous section can actually be used directly in the global framework as well. That is, given an approximate 3D shape model, we could modify the shape to increase the consistency of the model with respect to all the images. One formulation of this approach is to move the vertices of the approximate model to minimize the variance of a global texture map computed with (weighted) averag-

ing. This problem could also be considered a shape-from-focus problem, where the focus is now the consistency among different views. The surface would come in to optimal focus precisely when the shape is most consistent with the images. An alternative is to formulate this optimization in a super-resolution framework similar to Cheeseman et. al. [8]. In this approach, the super-resolved model is computed by adjusting camera parameters with known shape, and then shape is modified by assuming known camera parameters. This two-phase optimization would generate improved shape as well as a super-resolved texture, even higher resolution than the original images.

7.2.4 Volumetric Merging

The current volumetric merging strategy uses only the range images from stereo, ignoring the implicit structural information in the original images. Work by Khalili [31] and by Seitz and Dyer [55] demonstrate that color consistency can be a powerful cue to shape. The simultaneous use of range and color could be even more powerful than either one alone. The advantage of using range is that the search process looks for an *optimal* solution rather than a *sufficient* solution in the case of thresholding. In addition, the range provides an initial estimate of visibility that is independent of the merging process. As a result, the merging process is freed from the burden of determining surface location and visibility at the same time. In addition, the algorithm could be applied to camera configurations for which no visibility traversal order exists.

7.2.5 Temporal Integration

The methods presented in this thesis model dynamic events as a series of static ones. Extending this approach to incorporate some notion of scene dynamics can simultaneously improve quality, computational cost, and storage efficiency. Quality can improve because the inter-frame constraints provide more information about structure, and because accurate motion models would allow compensation for motion blur. Computational costs drop because constraints on physical motion help focus computation only on the volume likely to contain structure. And storage efficiency improves because the coding can explic-

itly model the frame-to-frame consistencies such as color, and can model changes with delta encoding rather than absolute positioning. An added benefit of this extension is that the temporal sampling of the event could be modified just as the current system allows spatial viewpoint modifications. Unlike traditional scan conversion in analog video, this process would have explicit knowledge of scene motion and could therefore create more accurate conversions. This ability could also be a powerful virtual slow motion mechanism, allowing the replay of the event at nearly any speed.

7.2.6 Application Specific Models

In many situations, much will be known a priori about the event being modeled. In these situations, it is advantageous to incorporate models appropriate for this context. For events containing rigid objects, for example, these objects can be modeled beforehand, and then localization strategies can be used to determine the positions of these objects in the space. Kinematic models could be powerful for many other objects, including humans. These models greatly restrict the configurations of the objects being modeled, and help separate independent surfaces that are in contact from connected surfaces. Incorporating such models could reduce computational costs as well as improve quality, in addition to improving storage efficiency.

Appendix A

The Tsai Camera Model

The Tsai camera model [62] describes a camera as a pinhole projector combined with radial lens distortion and is completely defined by 12 parameters:

- (3) 3D rotation
- (3) 3D translation
- (1) focal length
- (2) lens distortion
- (1) aspect ratio
- (2) image center

Tsai observed that lens distortion is usually modeled well with only one parameter, and so the actual model used has 11 parameters.

To project a 3D world point $\bar{p}_w$ into an image, the 3D coordinate is first rotated and translated into camera coordinates, yielding $\bar{p}_c$:

$$\bar{p}_c = R\bar{p}_w + \bar{T} \qquad \text{where} \qquad R = R_{\theta_z} R_{\theta_y} R_{\theta_x} \qquad \bar{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \qquad \bar{p}_\alpha = \begin{bmatrix} X_\alpha \\ Y_\alpha \\ Z_\alpha \end{bmatrix} \tag{A.1}$$

R_{θ_k} is a 3x3 rotation matrix, rotating about coordinate axis k by angle θ_k, and T_k is a translation along coordinate axis k. The six camera parameters used here, θ_k and T_k, are collectively referred to as extrinsic parameters. After this 3D transformation, $\bar{p}_c$ is perspectively projected into undistorted sensor coordinates (x_u, y_u), using the focal length f:

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = \frac{f}{Z_c} \begin{bmatrix} X_c \\ Y_c \end{bmatrix} \tag{A.2}$$

Next, the sensor coordinates are radially distorted, using the distortion parameter κ_1, to acquire distorted sensor coordinates (x_d, y_d):

$$\begin{bmatrix} x_u \\ y_u \end{bmatrix} = (1 + \kappa_1 r^2) \begin{bmatrix} x_d \\ y_d \end{bmatrix} \qquad r^2 = x_d^2 + y_d^2 \tag{A.3}$$

Note that this equation is formulated as an inverse mapping; solving for distorted coordinates requires the solution of a cubic polynomial. The image coordinates (x_f, y_f) are computed by applying the aspect ratio a and image center (C_x, C_y):

$$\bar{q} = \begin{bmatrix} x_f \\ y_f \end{bmatrix} = \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \end{bmatrix} + \begin{bmatrix} C_x \\ C_y \end{bmatrix} \tag{A.4}$$

This model degenerates into the general camera model of Equation (2.4) when there is no lens distortion (i.e., $\kappa_1 = 0$), and further degenerates into the simple projection of Equation (2.1) where there also is no rotation ($R = I$), no translation along the Z axis ($T_z = 0$), equal focal lengths across all cameras, a unity aspect ratio ($a = 1$), and an image center (C_x, C_y) at (0,0).

Appendix B

3D Dome: A 3D Digitization Testbed

Real experimentation into 3D digitization of dynamic events requires a system to record multiple video streams. These video streams must be synchronized, and the cameras must surround the scene to be modeled. This chapter explores the design and development of *3D Dome*, our facility for 3D digitization. This facility is designed to allow unknown, free-form, human-scale objects to move around freely in the work space. The design scales linearly with the number of cameras and uses only standard video equipment to minimize cost. The actual system was implemented with 51 video cameras on a 5-meter diameter geodesic dome.

B.1 Specifications

Any system for 3D digitization of dynamic, real-world events must satisfy three important requirements. First, the system must not interfere with the normal appearance or activity of the event, since the images will be used to construct both photometric and geometric scene models. This requirement prohibits the use of active lighting, which can improve

geometric modeling quality at the expense of distorting scene appearance. Second, the system must achieve sustained real-time performance in order to capture meaningful events. The precise definitions of *sustained* and *real-time* are context dependent. Third, the video must be captured digitally so that the 3D extraction processes can be applied.

In addition to these general system requirements, we added two objectives. First, the system should allow unknown, free-form, human-scale objects to move around freely in the work space. Human motion can occur relatively quickly, so we define *real-time* operation to mean full NTSC frame rate video capture of multiple video streams. We define *sustained* performance to mean video capture with no missed frames for at least one minute, enough to allow several repetitions of simple motions without restarting the system. Second, the system should be scalable so that nearly any number of cameras can be used.

B.2 Multi-Camera Video Capture Architecture

One obvious approach to solving this design problem is to directly capture digital video to some high-bandwidth, high-capacity storage device(s). However, synchronous digital acquisition of multiple video streams is a difficult task due to the magnitude of data involved. Just a single color camera generates 27 MBytes of raw data for 30 frames of size 480x640 per second. A 50-camera system recording for one minute would generate about 80 GBytes of data. Although specialized systems can provide the necessary bandwidth for a few video streams, extending these systems to many cameras would be prohibitively expensive. (Note that although lossy image compression would reduce the amount of data, the information loss may degrade the performance of the subsequent modeling processes. Lossless compression may help reduce the bandwidth, but usually only by a factor of 2-3, and at the cost of relatively large computation.)

Our alternative to direct digital acquisition is to use real-time analog acquisition and off-line digitization. Although the final capacity requirements remain unchanged, the digital acquisition can now occur at whatever rate is available. More importantly, the recording stage can use common analog recording hardware such as standard CCD video cameras and analog VCRs, greatly reducing overall system cost. The quality of the video can be

controlled by the quality and recording format of the analog video, providing a range of solutions with variable cost. We use low cost equipment (consumer-grade SVHS VCRS), so the image quality is relatively low. One disadvantage of this approach is that the digital video capture process is longer, since each analog video must be digitized separately. For our purposes, this cost was tolerable. A more detailed report on the design and implementation of this architecture is also available [42].

B.2.1 Real-Time Multi-Camera Analog Video Recording

The video capture system must capture multiple video streams in real time. Standard VCRs record a single video channel in real time for long duration, so using one VCR per video camera ought to be an effective design strategy: the cost is low, the components are reliable, and the system would scale easily. The main obstacle to implementing this approach is that video from independent VCRs will not be synchronized, so there will be no way to relate video frames from one camera to those of another.

Our solution to this problem requires two steps. First, all the cameras are sent an electronic synchronization signal so that all the cameras open their shutters simultaneously, which will synchronize the video streams. Next, every field of each camera's video is time stamped with a common Vertical Interval Time Code (VITC) before being recorded onto video tape. The VITC allows the video to be re-synchronized after digitization. This timecode contains the hour (00-23), minute (00-59), second (00-59), frame (00-29), and field (0 or 1) for each video field.

B.2.2 Off-line Multi-Camera Video Digitization

At the conclusion of a multi-camera analog video recording session, the video must be digitized. The common VITC across all video cameras greatly simplifies this process. The operator selects a range of VITC codes to be digitized, and then runs an automatic digitization program, which digitizes only the desired video frames. The video digitizer itself does not need to support VITC timecode itself, since the software can read and interpret the VITC in real time.

B.3 Camera Placement

The placement of cameras around the workspace can affect overall system performance in a number of ways. Most importantly, if no camera sees a part of the scene, then the 3D digitization system will be unable to model that scene element. The placement also affects the quality of the geometric and appearance models.

B.3.1 Clustered vs. Uniform Camera Distributions

One aspect of camera placement is the distribution of cameras around the work space. One approach is to cluster cameras together, which can improve the correctness of stereo correspondence by reducing changes in visibility and appearance from camera to camera within a cluster. A uniform camera distribution will make the correspondence problem in stereo more difficult, but will reduce the visibility problem, i.e., the risk of not observing an important element of the scene. Since visibility is critical to modeling process, we chose a nearly uniform camera distribution.

B.3.2 Effects of Camera Placement on Image Resolution

Another aspect of camera placement is its impact on image resolution. As the camera is moved closer to the work space, the field of view must increase in order to capture the full space. For objects near the front of this space, the difference in resolution is small, but the resolution changes dramatically for objects near the back of the work space, as shown in Figure B.1. If we assume that the workspace is defined by a cube with one face parallel to the image plane, and that the optical axis of the camera intersects this face at its center, then the variation in resolution can be expressed as the square of the ratio of the depth of the near cube face to the far cube face:

$$K = \left(\frac{d}{d+w}\right)^2 \tag{B.1}$$

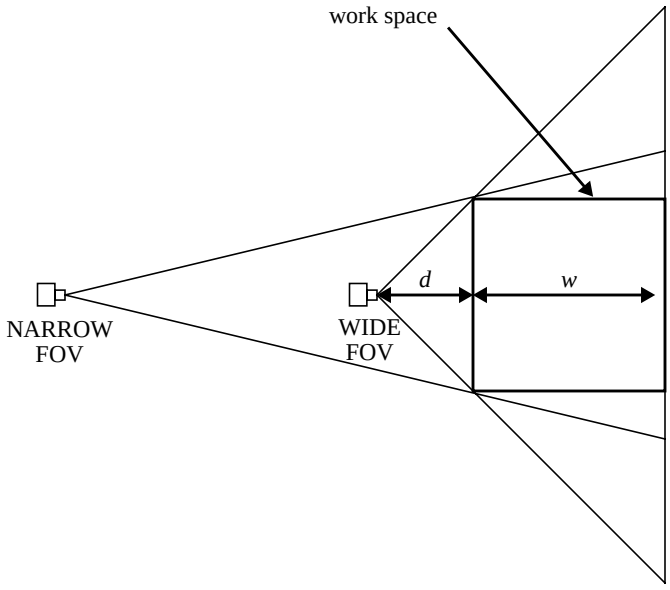


Figure B.1: Variation in image resolution across the work space. A camera close to the workspace must have a wide field of view (FOV) to see all of the near space. As a result, the resolution of objects near the back of the work space goes down rapidly compared to a camera with a longer distance *d* and narrower field of view.

where *d* is the depth to the near surface, *w* is the width of the work space, and K is the ratio of image pixels filled by the back side of workspace to pixels filled by the front side. Because of the constraints of the dome, our cameras had to be placed very close to the work space, with $d \sim 1$ meter, and $w \sim 2$ meters, yielding $K \sim 0.111$. (The camera labeled "WIDE FOV" actually approximates this configuration.) Thus, the far side of the work space has nearly an order magnitude lower resolution than the near side.

B.3.3 Density of Cameras

One of the most difficult factors to determine in designing a real multi-camera system is the number, or density, of cameras that are actually needed to observe the scene. Ideally, we want an optimal solution, not just a sufficient one. However, it is not obvious even how to define this sampling problem, let alone solve it. We therefore limit ourselves here to listing factors that must affect the solution. The most important factor is the 3D scene itself. Large areas of self-occlusion require a dense camera distribution to be able to see into the occlusion. Without some knowledge of the contents of the scene, little can be said about the camera density. Another major factor is image resolution, since it determines the

smallest resolvable (i.e., observable) 3D feature. As image resolution decreases, the smallest feature size increases. If we approximate scenes with occluded regions that are smaller than the smallest feature by similar scenes that close, or fill in, these occluded regions, then the image resolution bounds the complexity of the camera system necessary to completely observe the scene. The final major factor is the impact of camera spacing on the stereo algorithm. Changes in camera spacing can affect accuracy, precision, and coverage in stereo, since the correspondence search becomes increasingly difficult with wider camera separation. Finally, for scenes with minimal self-occlusion, the field of view of the cameras also may be important, since very few cameras are necessary to see the foreground structure. In this situation, the camera number may drop so low that portions of the background are not observed. If the system objective is only to observe the foreground, this factor can be ignored, and with reasonably complex scenes, this factor is negligible in comparison to the others. Although beyond the scope of this paper, this topic is worthy of a more detailed analysis.

In the design of 3D Dome, we predominantly focused on the needs of stereo while keeping system cost to reasonable levels, resulting in a camera spacing of approximately 60-70 cm. This baseline distance is relatively large for the 1-5 meter distance between each camera and the working volume of the dome – for example, the stereo machine [26] used a baseline that was an order of magnitude smaller for a similar range of depths – but our experiments have verified that stereo successfully operate on this narrow baseline. For a simple object such as a sphere, each surface point is observed by as many as all 51 cameras. With complex self-occlusion as in the examples shown throughout this thesis, at times only a few cameras see into highly complex regions.

B.4 Calibration

The video capture system provides only raw video for future processing. In order to make use of this data, the camera system must be calibrated. Each camera is modeled with the Tsai camera model, discussed in Appendix A. The calibration process fits the parameters of the general camera model to each real camera. This process requires a set of 3D points

in known positions along with the image projections of these points for each camera. In systems with only a few closely spaced cameras, this information can be collected by imaging a known 3D object in all cameras. Because of visibility constraints, however, this method does not extend easily to the calibration of a large volume surrounded by cameras.

We have developed two methods for dealing with this problem. The first approach is to decouple the calibration of intrinsic and extrinsic camera parameters. If the intrinsic parameters are known, then a single planar set of dots can be used to calibrate the extrinsic parameters. Since all cameras can see the floor of the dome, markers put directly on the floor in known positions can be used as the known 3D points on a plane for extrinsic parameter calibration. The second approach is to design and use a calibration object that avoids most of the visibility problems inherent in the multi-camera system. Then this object is moved through the volume to create a set of known 3D calibration points.

B.4.1 Decoupling of Intrinsic and Extrinsic Parameters

The decoupling approach involves two steps. First, to get the intrinsic parameters, each camera is removed from the dome and placed on a controlled imaging platform mounted in front of a small calibration plane. This plane has dots at known positions, and can be moved along the surface normal to create a set of 3D points. The plane is square, 40 cm on a side, and is moved 40 cm along its surface normal. By imaging the plane at several points along this motion, the projections of the 3D points can be determined. This data is fed into the calibration process to compute a full camera model. Second, the camera is then placed into its final position on the dome and used to image dots on the floor of the dome. These known 3D points and their image projections are then fed back into the calibration process, which now holds the intrinsic parameters fixed while optimizing the extrinsic camera parameters.

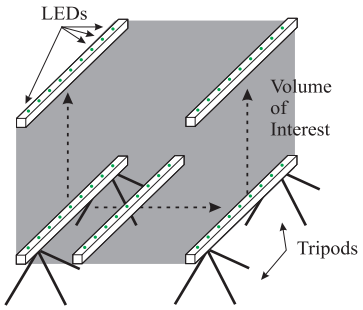


Figure B.2: The calibration bar assembly. The calibration bar contains 11 LEDs spaced 20 cm apart. The bar is mounted on two tripods, which provide known vertical positioning. Sliding the tripods orthogonally to the bar provides the remaining motion to define the 3D volume of calibration points.

B.4.2 Direct Calibration

In order to calibrate all cameras directly in their desired positions, the scene must be instrumented with a known 3D object. The object should span the 3D volume to obtain best results, but such an object is likely to occlude itself in many viewpoints. Our strategy for addressing this problem was to build a simple object with minimal self-occlusion and then build a mechanism to move this object through the volume to known 3D positions. This approach is philosophically identical to the method of calibrating the intrinsic parameters in the previous method, but the scale is much larger. The intrinsic method calibrated a volume of about 0.1 m^3, while this method must calibrate a volume of almost 10 m^3. Because of the physical dimensions involved, we chose to build only a bar rather than a full planar object. The bar also has less self-occlusion, since it must be viewed nearly end-on to obscure itself, while the plane could occlude itself from viewpoints near the plane.

The bar defines one horizontal dimension of the 3D volume. In order to define the vertical dimension, the bar is mounted on tripods whose vertical positions have been calibrated to raise the bar to known heights. The final dimension is defined by lateral translation of the bar-tripod assembly, with known positions marked on the floor. The bar has 11 LEDs mounted 20 cm apart. The floor is calibrated with markers 20 cm apart in both dimensions, as is the vertical motion of the tripods that the bar is mounted on. During calibration, this bar is swept through the volume along the lateral and vertical axes to generate known 3D points imaged in all cameras (see Figure B.2). Since we have three dimensional calibration data, we can now combine the extrinsic and intrinsic calibration steps into one.

Appendix C

Rectification

The rectification procedure discussed in Section 2.3.3 approaches the rectification problem from the perspective of camera rotations about the optical centers of the cameras. This rotation must align the optical axes (i.e., make them parallel) and mutually perpendicular to the baseline (i.e., 3D line) connecting the camera centers. By itself, this process leaves the direction of the optical axes under-constrained, free to rotate in the plane whose surface normal is given by the baseline. In order to constrain this degree of freedom, we require that the rectified optical axes have minimum deviation from the average of the two unrectified optical axes. To implement this strategy, we compute the average direction of the two input optical axes, and then remove the portion of that direction that lies along the computed baseline. This difference is orthogonal to the baseline and minimally different from the average direction. The rectification procedure is then given below. The camera centers are assumed to be specified in the world coordinates, as all vectors are 3D vectors in the world coordinate system.

Algorithm: Rectify Camera Pair

$\overline{baseline} = \overline{camcenter1} - \overline{camcenter2}$;
$\overline{Xaxis} = normalize\ (\overline{baseline})$;

$\overline{avgZaxis} = normalize\ (\overline{Zaxis1} + \overline{Zaxis2})$;
$coef = \overline{Xaxis} \cdot \overline{avgZaxis}$;
$\overline{tmpaxis} = \overline{avgZaxis} - (coef * \overline{Xaxis})$;
$\overline{Zaxis} = normalize\ (\overline{tmpaxis})$;

$\overline{tmpaxis} = \overline{Zaxis} \times \overline{Xaxis}$;
$\overline{Yaxis} = normalize\ (\overline{tmpaxis})$;

$$R = \begin{bmatrix} Xaxis(1) & Xaxis(2) & Xaxis(3) \\ Yaxis(1) & Yaxis(2) & Yaxis(3) \\ Zaxis(1) & Zaxis(2) & Zaxis(3) \end{bmatrix}$$

$\overline{translation1} = -\boldsymbol{R} \cdot \overline{camcenter1}$;
$\overline{translation2} = -\boldsymbol{R} \cdot \overline{camcenter2}$;

Bibliography

[1] N. Ahuja and J. Veenstra. Generating Octrees from Object Silhouettes in Orthographic Views. *IEEE Trans. PAMI*, Vol. 11 No. 2, pp. 137-149, 1989.

[2] S. Avidan and A. Shashua. Novel View Synthesis in Tensor Space. *Proc. IEEE CVPR*, June 1997.

[3] T. Beier and S. Neely. Feature-based Image Metamorphosis. SIGGRAPH'92, pp. 35-42, July 1992.

[4] J. Bloomenthal. An Implicit Surface Polygonizer. *Graphics Gems IV*, ed. P. Heckbert, pp. 324-349, 1994.

[5] E. Chen. QuickTime VR – An Image-Based Approach to Virtual Environment Navigation. SIGGRAPH'95, pp. 29-38, 1995.

[6] E. Chen and L. Williams. View Interpolation for Image Synthesis. *SIGGRAPH'93*, pp. 279-288, 1993.

[7] Q. Chen and G. Medioni. Image Synthesis from a Sparse Set of Views. *IEEE Visualization 97*, October 1997.

[8] P. Cheeseman, B. Kanefsky, R. Kraft, J. Stutz, R. Hanson. Super-Resolved Surface Reconstruction from Multiple Images. *Maximum Entropy and Bayesian Methods*, ed. G. R. Heidbreder, Kluwer, the Netherlands, pp. 293-308, 1996.

[9] C.H. Chien, Y.B. Sim, and J.K. Aggarwal. Generation of Volume/Surface Octrees from Range Data. *Proc. IEEE CVPR*, pp. 254-260, 1988.

[10] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. *SIGGRAPH '96*, August 1996.

[11] P. Debevec, C. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-based Approach, *SIGGRAPH'96*, August 1996.

[12] U.R. Dhond and J.K. Aggarwal. Structure from Stereo – a Review. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, pp. 1489-1510, 1989.

[13] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics: Principles and Practice*, second edition. Addison-Wesley Publishing Company, 1993.

[14] M. Garland and P.S. Heckbert. Surface Simplification Using Quadric Error Metrics. *SIGGRAPH '97*, pp. 209-216, August 1997.

[15] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen. The Lumigraph. *SIGGRAPH'96*, August 1996.

[16] S. J. Gortler, L. He, Michael F. Cohen, Rendering Layered Depth Images. *Microsoft Technical Report*, March 1997.

[17] P. Heckbert and H. Moreton. Interpolation for Polygon Texture Mapping and Shading. *State of the Art in Computer Graphics: Visualization and Modeling*, Springer-Verlag, 1991.

[18] P. Heckbert. *Fundamentals of Texture Mapping and Image Warping*. M.S. Thesis, UCB/CSD 89/516, CS Division, U.C. Berkeley, June 1989.

[19] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. Reliable Surface Reconstruction From Multiple Range Images. *Proceedings of ECCV'96*, pp. 117-126, April 1996.

[20] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface Reconstruction from Unorganized Points. *SIGGRAPH'92*, pp. 71-78, 1992.

[21] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization. *SIGGRAPH'93*, pp. 19-26, 1993.

[22] R. Jain and K. Wakimoto. Multiple perspective interactive video. *Proceedings of IEEE Conference on Multimedia Systems*, May 1995.

[23] A. Johnson. Control of Mesh Resolution for 3D Computer Vision. *Robotics Institute Technical Report*, CMU-RI-TR-96-20, Carnegie Mellon University, 1996.

[24] Takeo Kanade, P.J. Narayanan, and P.W. Rander. Virtual*ized* Reality: Concept and Early Results. *IEEE Workshop on the Representation of Visual Scenes*, June, 1995.

[25] Takeo Kanade, P.J. Narayanan, and P.W. Rander. Virtual*ized* Reality: Being Mobile in a Visual Scene. *International Conference on Artificial Reality and Tele-Existence and Conference on Virtual Reality Software and Technology*, Japan, Nov 1995.

[26] T. Kanade, A. Yoshida, K. Oda, H. Kano, M. Tanaka. A Stereo Machine for Video-rate Dense Depth Mapping and its New Applications. *Proc. IEEE CVPR*, June, 1996.

[27] S.B. Kang and R Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. *International Journal of Computer Vision*, 25(2):167-183, November 1997. Appeared in shorter form in *Proc. IEEE CVPR*, June 1996.

[28] S.B. Kang, J. Webb, L. Zitnick, and T. Kanade. A Multibaseline Stereo System with Active Illumination and Real-Time Image Acquisition. *International Conference on Computer Vision*, pp. 88-93, 1995.

[29] A. Katayama, K. Tanaka, T. Oshino, and H. Tamura. A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images. *SPIE Proc. Vol. 2409: Stereoscopic Displays and Virtual Reality Systems II*, pp.11-20, 1995.

[30] A. Katkere, S. Moezzi, D.Y. Kuramura, and R. Jain. Towards Video-based Immersive Environments. *Multimedia Systems*, vol. 5, no. 2, pp. 69-85, 1997.

[31] P. Khalili. Forming a Three-Dimensional Environment Model for Autonomous Navigation Using a Sequence of Images. *PhD thesis*, University of Michigan, 1994.

[32] K.T. Kim, M.W. Siegel, and J.Y. Son. Synthesis of a High Resolution 3D-Stereoscopic Image from a High Resolution Monoscopic Image and a Low Resolution Depth Map. *Proc. SPIE Vol. 3295A: 3D Displays, Stereoscopic Displays and Applications IX*, 1998.

[33] S. Laveau and O. Faugeras. 3-D Scene Representation as a Collection of Images. *Proceedings of ICPR'94*, 1994.

[34] M. Levoy and P. Hanrahan. Light Field Rendering. *SIGGRAPH'96*, August 1996.

[35] W. Lorensen and H. Cline. Marching Cubes: a High Resolution 3D Surface Construction Algorithm. *SIGGRAPH'87*, 163-169, July 1987.

[36] F.C.M. Martins and J.M.F. Moura. 3-D Video Compositing: Towards a Compact Representation for Video Sequences. *Proc. ICIP'95*, pp. 550-553, 1995.

[37] F.C.M. Martins and J.M.F. Moura. Video Representation with Three-Dimensional Entities. *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 1, pp. 71-85, Jan. 1998.

[38] L. McMillan and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *SIGGRAPH 95*, pp. 39-46, 1995.

[39] J.S. McVeigh, M.W. Siegel, and A.G. Jordan. Algorithm for automated eye strain reduction in real stereoscopic images and sequences. *Proc. SPIE Vol. 2657: Human Vision and Electronic Imaging* pp. 307 - 316, 1996.

[40] J.S. McVeigh, M.W. Siegel, and A.G. Jordan. Intermediate view synthesis considering occluded and ambiguously referenced image regions. *Signal Processing: Image Communication 9*, pp. 21 - 28, Elsevier 1996.

[41] S. Moezzi, A. Katkere, D.Y. Kuramura, and R. Jain. Reality Modeling and Visualization from Multiple Video Sequences. *IEEE Computer Graphics and Appplications*, vol. 16, no. 6, pp. 58-63, 1996..

[42] P.J. Narayanan, P.W. Rander and T. Kanade. Synchronizing and Capturing Every Frame from Multiple Cameras. *Robotics Institute Technical Report*, CMU-RI-TR-95-25, Carnegie Mellon University, 1995.

[43] P.J. Narayanan, P.W. Rander and T. Kanade. Constructing Virtual Worlds Using Dense Stereo. *IEEE International Conference on Computer Vision*, Bombay, 1998.

[44] S.K. Nayar. Catdiotric Omnidirectional Camera. *Proc. IEEE CVPR*, pp. 482-488, June 1997.

[45] M. Okutomi and T. Kanade. A multiple-baseline stereo, *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 15(4):353-363, 1993.

[46] C. Poelman and T. Kanade. A Paraperspective Factorization Method for Shape and Motion Recovery. *Proc. of ECCV*, pp. 97-108, 1994.

[47] P.W.Rander, P.J. Narayanan and T. Kanade. Recovery of Dynamic Scene Structure from Multiple Image Sequences, *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Washington, D.C., Dec. 1996.

[48] P.W.Rander, P.J. Narayanan and T. Kanade. Virtualized Reality: Constructing Time-Varying VIrtual Worlds from Real World Events. *IEEE Visualization 97*, October 1997.

[49] M. Rutishauser, M. Stricker, and M. Trobina. Merging Range Images of Arbitrarily Shaped Objects. *Proc. IEEE CVPR*, pp. 573-580, 1994.

[50] Y. Sato, M. D. Wheeler, and K. Ikeuchi. Object shape and reflectance modeling from observation. *SIGGRAPH'97*, pp. 379-387, August 1997.

[51] Y. Sato and K. Ikeuchi. Reflectance Analysis for 3D Computer Graphics Model Generation. *Graphical Models and Image Processing*, 1996.

[52] W.J. Schroeder and J.A. Zarge. Decimation of Triangle Meshes. *SIGGRAPH'92*, pp. 65-68, 1992.

[53] S.M. Seitz and C.R. Dyer. Physically-Valid View Synthesis by Image Interpolation. *Proc. Workshop on Representation of Visual Scenes*, pp. 18-25, 1995.

[54] S.M. Seitz and C.R. Dyer. View Morphing, *SIGGRAPH 96*, pp. 21-30, 1996.

[55] S.M. Seitz and C.R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring, *Proc. IEEE CVPR*, pp. 1067-1073, June 1997.

[56] M.W. Siegel, S. Sethuraman, J.S. McVeigh, and A.G. Jordan. Compression and Interpolation of 3D-Stereoscopic and Multi-View Video. *Proc. SPIE Vol. 3012: Stereoscopic Displays and Virtual Reality Systems IV*, p. 227 - 238, 1997.

[57] M. Soucy and D. Laurendeau. Multi-Resolution Surface Modeling from Multiple Range Views. *Proc. IEEE CVPR*, pp. 348-353, 1992.

[58] R. Szeliski and R. Weiss. Robust Shape Recovery from Occluding Contours Using a Linear Smoother. *Proc. IEEE CVPR*, pp. 666-667, 1993.

[59] R. Szeliski and S.B. Kang. Recovering 3D Shape and Motion from Image Streams Using Nonlinear Least Squares. *Journal of Visual Communication and Image Representation*, 5(1):10-28, March 1994.

[60] R. Szeliski. Rapid Octree Construction from Image Sequences. *CVGIP: Image Understanding*, 58(1):23-32, July 1993.

[61] R. Szeliski and D. Tonnesen. Surface Modeling with Oriented Particle Systems. *SIGGRAPH'92*, 26(2):185-194, July 1992.

[62] R. Tsai. A Versatile Camera Calibration Technique for High-accuracy 3D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323-344, 1987.

[63] C. Tomasi and T. Kanade. Shape and Motion from Image Streams under Orthography: A Factorization Method. *International Journal of Computer Vision*, Vol. 9 No. 2, pp. 137-154, 1992.

[64] G. Turk. Re-tiling polygonal surfaces. *SIGGRAPH '92*, pp. 55-64, July 1992.

[65] G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. *SIGGRAPH'94*, pp. 311-318, July 1994.

[66] T. Werner, R. D. Hersch and V. Hlavac. Rendering Real-World Objects Using View Interpolation. *IEEE International Conference on Computer Vision*, Boston, 1995.

[67] M. Wheeler. *Automatic Modeling and Localization for Object Recognition.* PhD thesis, Carnegie Mellon University, 1996.