Virtual Worlds using Computer Vision

P. J. Narayanan
Centre for Artificial Intelligence & Robotics
Raj Bhavan Circle, High Grounds
Bangalore, INDIA 560 001.
E-mail: pjn@cair.res.in

Takeo Kanade
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890. U.S.A.
E-mail: tk+@cs.cmu.edu

Abstract

Virtual Reality has traditionally relied on hand-created, synthetic virtual worlds as approximations of real world spaces. Creation of such virtual worlds is very labour intensive. Computer vision has recently contributed greatly to the creation of the visual/graphical aspect of the virtual worlds. These techniques are classified under image-based -- *as opposed to* geometry-based -- *rendering in computer graphics. Image based rendering (IBR) aims to recreate a visual world given a few real views of it. We survey some of the important image-based rendering techniques in this paper, analyzing their assumptions and limitations. We then discuss* Virtualized Reality, *our contribution to the creation of virtual worlds from dynamic events using a stereo technique that gives dense depth maps on all-around views of the event. The virtualized world can be represented as multiple view-dependent models or as a single view-independent model. It can then be synthesized visually given the position and properties of any virtual camera. We present a few results from* 3DDome, *our virtualizing facility.*

1 Introduction

Virtual worlds for VR are mostly hand created using sophisticated software tools such as MultiGen, 3DStudio etc. This is an expensive process and produces only a synthetic approximation of the real world that lack in fine detail for walk-through and other applications. Texture mapping improves the fidelity of the virtual world, but the enormous manual effort involved is one of diminishing returns as the model gets finer.

View synthesis – the generation of new views of scenes – has traditionally used hand-created geometry as the basic model underlying scene representations. Mapping textures collected from the real world using a camera improved the photorealism of virtual worlds that use such models. Substitution of synthetic texture by real-world texture brought the following question: Can the geometry -- or equivalent structure -- be inferred from images of the real world also? Is there a need to manually specify the scene geometry if carefully selected views of the scene can be obtained?

Recently, several techniques have been proposed to synthesize arbitrary views of a scene from a collection of photographs rather than from an explicit geometric model. The superior visual realism these methods achieve even when the implicit geometric model is imperfect makes them particularly attractive. These techniques are commonly referred to as *image-based rendering* (IBR) techniques. These techniques tread a common ground between computer vision and computer graphics.

We present some of the image-based rendering techniques in the next section and compare them on the basis of the type of input they require and the restrictions that places on the view synthesis. We then present in some detail Virtualized Reality, our contribution to the field of IBR [9][18][20].

Virtualized Reality closes the gap between images and traditional geometric models by computing the latter from the former. Range/depth map is the fundamental scene structure used by Virtualized Reality. It is recovered from a few carefully placed viewpoints using a multibaseline stereo method. This can be translated into textured geometric models of all surfaces visible from the viewpoint. The virtualized event can be immersively interacted with fully using a representation of it in terms of multiple Visible Surface Models (VSM) [18]. We also devised a method to merge these multiple partial models of the scene into a volumetric space to generate a view independent Complete Surface Model (CSM) of the scene [20]. The CSM is fully compatible with traditional geometric models used in virtual reality.

The completeness and smoothness of reconstruction of the virtualized event depends on the distribution of the VSMs and hence the cameras. Our virtualized events are generated using the *3DDome*, a facility consisting currently of 51 cameras covering a space enclosed by a geodesic dome of 5 meters diameter. The arrangement of cameras provides nearly uniform view of the dome from all angles. We present the theory of virtualized reality briefly in this paper as well as some results from the *3DDome*.

2 Image-based Rendering

The various image-based rendering techniques in the literature differ from one another significantly in the extra-image information used, such as the parameters of the imaging system and the sophistication of the models extracted from the images. In order to understand them better, we classify them on the basis of (a) the type of calibration of the imaging system necessary, (b) the need for pixel correspondences between pairs of images, (c) the nature of the underlying "model" used and the restrictions this model places on the position of the virtual camera and (d) the capacity to extend synthesis to dynamic events. Table 1 summarizes the image-based rendering algorithms and their properties.

2.1 Camera Calibration

Different view synthesis methods make different assumptions about the calibration parameters of the imaging system. Camera calibration parameters fit a general camera model to an actual imaging system, relating the camera's origin and image plane to the 3D world. Traditional camera calibration, or *strong calibration*, computes the full camera model, providing its imaging geometry in a Euclidean framework. The *intrinsic* parameters of the camera relate each pixel coordinate of its image

Method	Camera Calibration	Pixel Correspondence	Type of Model	Virtual camera placement	Dynamic Scene Handling
View Interpolation [2][25]	None	Required	Implicit Shape	Limited by input views	Enormous software effort
View Morphing [23]	Like weak calibration	Required	Projective Shape	Limited by input views	Enormous software effort
Projective Reconstruction [5][21]	Weak calibration	Required	Projective/ Affine Shape	Specified projectively	Enormous software effort
View Transfer [13]	Weak	Required	Projective Shape	Specified projectively	Enormous effort
Tensor-based Reconstruction [21]	Three-view weak	Required, in triplets	Like Projective Shape	Specified using few ref. points.	Enormous software effort
Plenoptic Modeling [16] Omnidirectional Stereo [11]	(Scaled) Strong	Required	(Scaled) Metric Shape	Anywhere	Moderate effort
Field Based (e.g., Lightfield, Lumigraph) [6][12][14]	Strong	No	Metric Field of Light Rays	Outside convex hull of objects	With prohibitive hardware setup
Multi Perspective Interactive (MPI) Video [7]	Strong	No	Metric Shape	Anywhere	Straightforward
Virtualized Reality	Strong	Required	Metric Shape	Anywhere	Straightforward

Table 1: Classification of view synthesis techniques

to a three-dimensional imaging ray in a coordinate system with the camera's optical center as the origin. The *extrinsic* parameters orient the camera's internal coordinate system used above with respect to a world coordinate system. Thus, it is possible to compute the imaging ray equations of all cameras in a common reference frame from the strong calibration parameters. This makes the recovery of a metric model of the scene possible from the images. *Weak calibration*, on the other hand, only computes enough parameters, usually in the form of a *fundamental matrix*, to relate pixels of one image to lines in another (the epipolar geometry). The structure of the scene can only be computed up to an unknown projective transformation using weak calibration.

The view interpolation techniques require no explicit camera calibration as long as image flow or pixel correspondence is given. Automatic computation of pixel correspondence between a pair of images is a difficult problem *without* calibration data to constrain the search. The proponents of these methods do not address this issue and usually compute correspondence by hand. This makes the method quite general and applicable even when no information is available on the imaging systems (such as old photographs), but can introduce unwanted distortion of the geometric structure if the cameras are not parallel as pointed out by Seitz and Dyer [22]. View morphing does not use conventional calibration information but extracts something like weak calibration data in order to rectify the images to a common (unknown) plane. The projective reconstruction methods and the view transfer technique require pair wise weak calibration data. Since these methods also specify the target camera being synthesized projectively, they require weak calibration for all pairs of the triad of views: the two input views and the view being synthesized. The tensor-based approach does not explicitly calibrate the cameras. They instead compute equations that directly constrain point matches in three images. The method can be thought of as three-view weak calibration as the computation is strictly projective.

In plenoptic modeling and omnidirectional stereo, the panoramic image construction essentially computes the strong calibration data up to a scale factor. The latter also shows a simple way to determine the scale factor. The field based techniques and MPI Video require strong calibration to relate the imaging ray directions in a world coordinate space. View-dependent texture mapping and Virtualized Reality need strong calibration to recover metric scene structure using stereo. For most of these algorithms, a calibration procedure such as Tsai's is usually necessary [24].

Calibration parameters are usually computed by solving the concerned camera equations using a few known data points in images. Weak calibration between a pair of cameras, for instance, requires the simultaneous identification of a few – a minimum of 7, but usually larger for reliability – points identified in a common scene. Strong calibration requires the identification of a few points with known 3D positions in each image. Both calibration methods assume an ideal pin-hole camera. In reality, however, the pin-hole assumption is often violated, especially by low focal length lenses whose optical properties can distort the images systematically. The weak calibration model cannot handle these non-ideal situations whereas the strong cal-

ibration model has been extended to correct them [24]. This is a practical advantage of methods using strong calibration over those using weak calibration, though the latter makes less assumptions in theory.

2.2 Pixel Correspondences

Pixel correspondences, or image flow vectors, between a pair of images of the same scene relate pixels in one image with corresponding pixels in the other image. Correspondences are in general difficult to compute but form the heart of all structure from motion and stereo techniques. Camera calibration helps correspondence finding by limiting the search space to the epipolar line rather than having an unconstrained search across the entire image plane. Conversely, correspondences can be used to weakly calibrate the imaging system.

Most view synthesis algorithms require pixel correspondences. View interpolation, view morphing, and plenoptic modeling use correspondences to map pixels directly to the synthetic image. View transfer uses it to compute the curve defined by the epipolar line of each output pixel. The tensor-based reconstruction requires correspondence for all pixels in the reference image and three-way correspondence involving the reference views and the target view for a few points in order to compute the tensor. Omnidirectional stereo and view-dependent texture mapping use correspondences to compute structure. Virtualized Reality also uses pixel correspondences to compute scene structure.

Another class of synthesis techniques eliminates the need for pixel correspondences by densely sampling the viewing space, possibly interpolating missing views. Katayama et. al demonstrated that images from a dense set of viewing positions on a plane can be used to generate images for arbitrary viewing positions [12]. Levoy and Hanrahan [14] and Gortler et al. [6] extend this concept to construct a four-dimensional field representing all light rays passing through a 3D surface that is the convex hull of the objects of interest in the scene. New view generation is posed as computing the correct 2D cross section of this field. The main drawback of these methods is the large number of images necessary. In [14], for example, as many as 8000 images are used to build just a partial model for a single object.

An alternative to correspondence-based analysis is to perform model-based motion analysis to a set of image sequences, a technique used in Multiple Perspective Interactive (MPI) Video [7]. In this system, motion is detected by subtracting a "background" image from each video frame. Three-dimensional models of the moving objects are computed by intersecting the viewing frustums of the pixels that indicate motion in a volumetric space. Complete 3D environments are then built by combining these dynamic motion models with *a priori* environment models (for the structure of the background). This approach is well suited to applications with only a few small moving objects and with a known stable environment.

2.3 Model Type

An implicit or explicit model of the world is necessary for view synthesis. The techniques presented here model either the object shapes or the field of light rays passing through a region of space in some form. The characteristics of the model used may limit the mobility of the virtual camera or restrict how the viewpoint can be specified. The type of the model also affects how it can be manipulated *post facto*. This is important if view synthesis is not the only goal of the system.

2.3.1 Virtual Camera Position

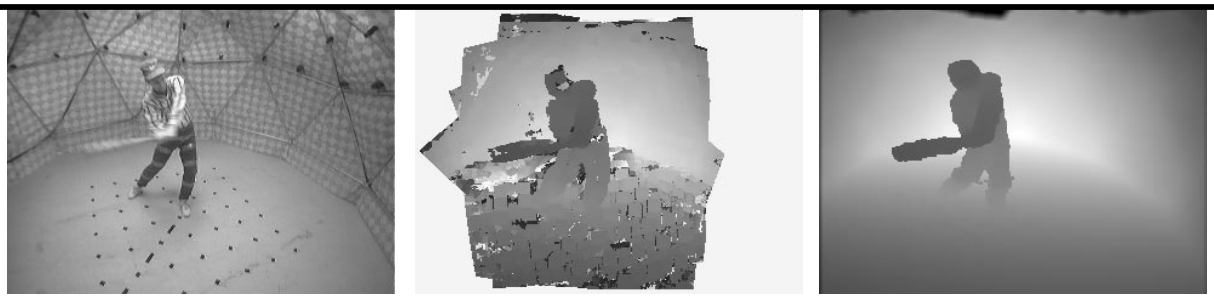
All correspondence-based methods model the scene geometry implicitly or explicitly. View interpolation and view morphing use only pixel correspondences as an implicit model of the scene geometry, without the explicit knowledge of how these correspondences relate to shape. As a result, synthesis is limited to a space that is a linear combination of the input image locations. Projective methods synthesize the scene only up to an unknown projective transformation; the virtual camera can therefore be anywhere in the projective space. The view transfer method also recovers the projective structure in effect. The virtual camera could be anywhere, but its position must be specified in terms of epipolar relationships with the input cameras. The trilinear tensor is an extension of this to three views, with the synthesized viewpoint being specified using the motion of a few reference pixels in the two reference images. The virtual camera can not be specified in a Euclidean coordinate system in all these methods.

Plenoptic and omnidirectional maps recover the space visible from a few closely spaced viewpoints and can be used to synthesize views from any point as long as occluded areas in the space are not exposed. The possible lack of absolute scale is usually unimportant for viewing, since many viewing interfaces already adjust for it. The field based techniques recover a field that is defined over a (metric) space outside the convex hull of the objects in the scene. The virtual camera can be placed anywhere in that space, but in general can not lie within the convex hull. The model as such can be considered to be an exhaustive enumeration of all possible light rays through that region. MPI Video, view-dependent texture mapping and Virtualized Reality recover a global metric model of the event, allowing the virtual camera to be anywhere in space. Occlusions from specific viewpoints in the space can be handled as long as every part of the event space is recovered by at least some real cameras.

2.3.2 Model Editing

The ability to edit the model after capture enables more than passive viewing of a scene captured using images. The editing could involve adding objects into the scene or removing objects from it. In both of these cases, the potential difficulty is resolving visibility of objects within the new environment. Editing can also involve modifying the scene, such as changing the costumes of the actors or the background texture. In this case, the potential difficulty is locating the same physical surfaces and objects in the multiple views.

The first five methods in Table 1 have at best a projective model of the scene and can manipulate it only projectively. This severely restricts the editability. Plenoptic models are metric up to a scale factor and can be edited. The field-based methods do not support editing in any meaningful way. Editing is possible with the model computed for MPI. The virtualized models can be edited in the 3D space using both representations we use. The CSM representation can also take advantage of model building/editing software used in virtual reality for the editing.



(a) (b) (c)

Figure 1: Visible Surface Model of one image. (a) intensity image, (b) depth map computed by MBS (distant surfaces are brighter), (c) depth map computed by reprojecting the global model (CSM) into the camera.

2.4 Dynamic Scene Handling

We now consider the dynamic event capabilities of these methods. MPI Video and Virtualized Reality have, by design, addressed the dynamic issues. Each time instant can be virtualized automatically (and independently) as the component processes -- like stereo and volumetric merging -- are performed without manual intervention. It is also possible to exploit the temporal consistencies in the scene either at the time of stereo or at the time of the model building.

Other techniques have been demonstrated only for static scenes. In addition, most have used the same imaging system, moving the camera or the object to get multiple views of static scenes. Extension to time-varying imagery would require hardware modifications but few algorithmic changes, but the quality of correspondences computed will suffer when using different imaging systems. In practice, the effort involved in the human-assisted correspondence computation used in the first six methods will make them essentially unusable in dynamic situation. The field-based approaches extend easily algorithmically, but several thousand cameras will be necessary to provide the numerous views it requires, making it prohibitively expensive.

3 Visible Surface Model

The fundamental scene structure is recovered using stereo in our system. Stereo gives a range/depth map listing the distances to each point in the intensity/colour image. This 2-1/2 D structure is converted to a Visible Surface Model (VSM) of all surfaces visible from a camera's viewpoint. This section describes the recovery and representation of the VSMs from input images, as well as the causes and effects of errors in that recovery.

3.1 Multibaseline Stereo

We adapted the multibaseline stereo algorithm (MBS) [19] for a various number of cameras in a general (i.e., non-parallel) configuration by incorporating the Tsai camera model. The choice of MBS was motivated primarily by two factors. First, MBS recovers dense depth maps, with a depth estimate for every pixel in the intensity image. Second, MBS takes advantage of the large number of cameras we use to improve the depth estimates. Figure 2(b) shows a depth map computed by MBS, aligned with the reference image shown in Figure 2(a). The farther points in the depth map appear brighter. We apply stereo to compute a depth map at each camera, with 3 to 6 neighboring cameras providing the baselines required for MBS. Adding more cameras arbitrarily may not improve the quality of the recovered structure because of the increased difficulty in matching.

3.2 Computing Visible Surface Model

A textured geometric model of all surfaces in the scene visible from a camera can be constructed from the aligned depth and intensity information provided by stereo and the camera calibration parameters. The steps for the construction are given below.

1. The *(X, Y, Z)* coordinates of each pixel are computed in a world coordinate system using its depth *d*, pixel coordinates *u* and *v* and camera calibration parameters as follows. The intrinsic camera parameters specify the ray corresponding to the pixel *(u, v)*. The camera-entered *(x, y, z)* coordinates can be obtained by intersecting the ray with the $z = d$ plane. The extrinsic calibration parameters orient the camera based coordinate system in the world coordinate frame giving the *(X, Y, Z)* coordinates.
2. The resulting cloud of 3D points is converted to a triangle mesh assuming local connectivity of the pixel array. That is, every 2x2 section of the array is converted to two triangles by including one of the diagonals.
3. The triangles with a large difference in depth along any side lie on occlusion boundaries (or on extremely foreshortened surfaces which are approximately equivalent). They are identified using a threshold on the depth difference along any edge and are marked as *hole* triangles, not to be rendered.
4. The aligned image from the camera is used to texture map the mesh. Texture coordinates are easy to compute as the triangle vertices fall on image pixels. Thus, the texture for a triangle is the section of the image that falls on it.

We call the resulting textured triangle mesh the *Visible Surface Model (VSM)* of the scene. The VSM is an oriented, local description of the scene. It has the optical center of the corresponding camera as its *origin* and the viewing frustum of the camera as its geometric extent. Since the computation of the textured mesh model from the depth/intensity pair is straightforward, we use the term VSM to also refer to the aligned pair of depth and intensity images for a particular camera (i.e., with the calibration parameters implied). The VSM has the following noteworthy properties. (1) It is a textured triangle mesh model of surfaces visible from its origin with the occluded surfaces left blank. (2) A view synthesized using it will be exact

when rendered from its origin irrespective of the errors in the recovered geometry. The rendering remains realistic if the virtual camera is close to the origin. This property has great ramifications in decimating the VSM mesh as will be seen later. (3) The synthesized view has "holes" when rendering from locations away from the VSM's origin as occluded regions get uncovered. (4) The VSM contains $O(N^2)$ triangles for an NxN depth map. The triangles are often tiny when using the above construction.

3.3 Effects of Errors in Correspondence

Errors in computing the pixel correspondences manifest themselves as incorrect depth in the depth map. Systematic errors in correspondence computation result in systematic distortion of the scene. For example, the periphery of the image tends to have lower quality depth estimates compared to the center in general as the region is visible to a fewer number of cameras participating in the stereo computation. Two other sources of systematic errors are camera calibration and scene occlusion.

Camera calibration is fundamental to the structure extraction process because calibration parameters determine the search space (i.e., epipolar geometry) for corresponding points across multiple images. Inaccuracies in the camera model will manifest themselves as serious distortions of the recovered model. The effects of inaccuracies in calibration on the recovered model merit a systematic study.

A section of computer vision researchers believe that the dependence on camera calibration should be kept to a minimum. The charm of the image-based rendering methods that either do not require camera calibration or require only a weaker form of calibration is their potential immunity from this source of systematic distortion. As already discussed in Section 2.1, these approaches have their own drawbacks, such as the inability to directly handle lens distortion and other deviations from perfect perspective projection. They also cannot provide a Euclidean model of the scene that is most intuitive to handle.

Area based methods for (automatic) correspondence computation fare poorly near occlusion boundaries in the scene. Usually, the effect of this violation is either the foreground surface expanding into the background or vice versa, a process we call *fattening* of the range image surfaces. VSMs computed from stereo inherit this shortcoming. We developed two solutions for these problems. The first is a human supervised editing step that corrects the inconsistencies in a single VSM, that typically takes a couple of minutes per VSM. The effort involved is far less than computing correspondences manually and the effects of errors in this manual operation are far less critical. The second solution involves the volumetric merging step described in Section 5. Fattening is reduced in the individual VSMs by enforcing global geometric consistency through volumetric merging. Figure 1(c) shows the depth map created by reprojecting the global model back into the camera. Clearly, the global model building process reduces noise, improves localization of depth discontinuities, and increases the effective field of view.

3.4 Decimating the VSM Mesh

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.Such finely tessellated meshes lend themselves easily to decimation. However, the decimation needs to be done carefully. A well-known visual effect is that humans can tolerate significant errors over continuous surfaces, but errors at occlusion boundaries (or silhouettes) are quite objectionable. This 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 [8] 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, giving maximum priority to the boundary points. 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. As will be shown later, this is a unique advantage of the VSM not shared by more global models of the scene which have no preferred direction for isolating the occlusion boundaries.

4 Rendering Using VSMs

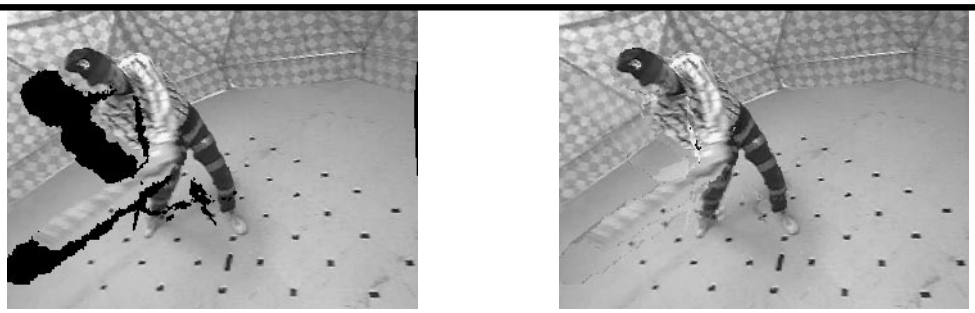
One representation of the virtualized environment is as a collection of VSMs for each time instant. We describe in this section how virtual camera views of the scene can be synthesized seamlessly from any location using this representation. Moreover, all interactions with the virtual environment that a conventional global representation permit are possible using this representation.

4.1 The Representation

A single VSM represents all surfaces within the viewing frustum of the camera visible from the camera location. It can be used to render the scene from locations near its origin, but the synthesized image will have holes when the virtual camera moves away from the origin as occluded areas become exposed, as shown in Figure 2(a). However, given a sufficiently dense distribution of VSMs, we can typically find other VSMs to fill these holes. Intuitively, when the virtual camera moves away from the origin of the VSM in one direction, a neighboring VSM that lies "beyond" the virtual camera, to which the exposed regions are visible, would 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 2(b). We call this combination the *View-dependent Local Model (VLM)* of the scene.

4.2 View-dependent Local Model

The combination of VSMs in a VLM used for rendering should provide a hole-free view from the virtual camera location. The VSM "closest" in approach to the virtual camera provides the



(a) (b)

Figure 2: (a) Holes appear when virtual camera moves away from a VSM. (b) Supporting VSMs can fill the holes.

most promising model and should play the central role in the VLM by providing most of the rendered image. We call this the *reference* VSM of the VLM. It will not be hole-free on its own; we therefore include **two** neighboring VSMs in the VLM 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 came a suitable combination of a small number of VSMs should be used for another arrangements of the VSMs.

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. Our approach works well under the following conditions: First, every part of the scene to be modeled must be visible in some VSM. 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 focussed 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 our setup; they are however not limitations of the system as it can be extended to other situations easily.

4.2.1 Selecting Reference VSM

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 or the direction of gaze are not sufficiently good measures of closeness. 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. The viewpoint of the virtual camera in our system is specified by an eye point and a target point. The virtual camera is situated at the eye point and oriented so that its line of sight passes through the target point. Our measure of closeness is the angle between this line of sight and the line connecting the target point to the 3D position of the VSM, as shown in Figure 3. The VSM with the closest angle with the virtual camera's viewing direction is chosen as the reference VSM. 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.

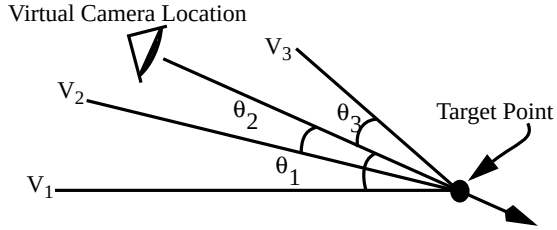


Figure 3: Selecting the reference VSM. θ_i is the angle between the virtual camera's line of sight and the line joining the target point with the position V_i of VSM i. The VSM with the smallest angle θ_i is selected as the reference VSM.

4.2.2 Selecting Supporting VSMs

The supporting VSMs are used to compensate for the occlusions in the reference VSM. Given a reference VSM, consider the triangles formed by its origin and all adjacent pairs of its neighboring VSMs, as shown in Figure 4. If the VSM has *n* neighbors, there are *n* such triangles. 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. This strategy gives hole-free rendering in practice, though not guaranteed in theory.

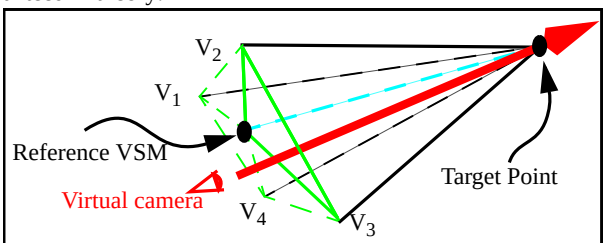


Figure 4: Selection of supporting VSMs. Given a reference VSM, the supporting VSMs are determined by the triangle pierced by the line of sight of the virtual camera.

4.3 Rendering Using a VLM

A VLM is a dynamically selected scene model that contains a hole-free description of the scene from the virtual camera's viewpoint, consisting of a reference and two supporting VSMs. How do we combine the VSMs effectively? We decided to fuse them in the image *after* rendering each component VSM separately. The supporting VSMs are used only to fill the hole pixels in the rendering of the reference VSM. A variation of this approach could have the renderings of the component VSMs *cross-dissolved* (i.e., weighted averaged) based on, say, a closeness metric or the distances of the virtual camera from the VSM origins. Merging the renderings of the VSMs in the image requires the hole pixels to be identified. A simple method could be to mark all pixels untouched while rendering as hole pixels. This approach 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.

We use a two-fold approach to identify the hole pixels to overcome this problem. The pixels that are not written over when rendering the reference VSM are marked as holes. In addition, the hole triangles of the VSM (as discussed in Section 3.2) are rendered into a separate hole buffer marking each pixel that is touched as a hole pixel, even if filled from the reference VSM.

Thus, rendering using a VLM 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. The contributions of the reference and supporting VSMs could also be cross-dissolved using alpha blending. We have not implemented this as of now. Figure 2 shows the results of rendering a VLM.

4.4 Alternative Local Models

The view-dependent local model consisting of one reference and a few supporting VSMs has the advantage that only a fixed number of VSMs (in our case 3) need to be rendered for any view. This approach, however, can leave small holes in the synthesized view depending on the scene geometry even when another VSM in the system (but not in the selected VLM) can fill it. Strategies using a variable number of supporting VSMs can eliminate all holes in regions visible in at least one VSM. We do not currently employ this strategy.

The 3D event space that generates the holes in any view given a reference VSM is the space *not visible* from its origin; it does not depend on the position or orientation of the virtual viewpoint. It is, therefore, possible to identify the subset of another VSM that covers this space by mapping it to the VSM, instead of recomputing it for each virtual camera position. (The spatial ordering induced by the pixel ordering of the VSM's depth image makes it straightforward to "circumscribe" a given 3D space in it, unlike a general geometric model.) This can be done by mapping the hole triangles into the supporting views using the relative camera geometry. The surface model induced by the pixels touched in this process belongs entirely to the hole region of the reference VSM. The holes of each VSM can then be mapped to each (possible) supporting VSM *off-line*. While synthesizing a virtual view, the reference VSM is rendered first based on the virtual viewing angle. Next, the supporting VSMs are rendered in a sorted order of distance and direction from the reference VSM, based on the hole pixels left unfilled. This strategy ensures better filling of holes. The rendering will have only one pass as the hole triangles need not be rendered; it will also be faster as only the relevant subset of the supporting models need to be rendered. We currently do not use this strategy because of the increased complexity and storage, although a system truly desiring maximum performance should consider this approach.

5 Complete Surface Model

The multi-VSM representation of an event has a 3D representation of it and is capable of full immersion and interaction like any global model. Because the VSM-based model never propagates local information into other views, the errors in one VSM only impact view synthesis when the virtual camera is near that VSM. In addition, this approach guarantees exact reproduction of the input images when the virtual camera is aligned with a real one. However, the local errors are not suppressed by the information in other views that clearly contradicts them. Thus, the random errors that can get removed using the number of views remain.

It may therefore be profitable to build a global model of the scene. Two factors motivate the development of a global model. First, global models are easier to manipulate than local models. For example, removing an object from an image-based model requires finding the object in every image. Performing the same operation on a global model involves nothing more than scissoring the object from the model. Second, the availability of a (nearly) invertible transformation between an image-based model and a traditional "geometric" model adds great flexibility in real applications. For example, since most existing graphics systems do not support image-based models, a designer must be able to convert these models into traditional ones in order to make use of these techniques. Even if future systems directly support image-based models, there may still be times when the global representation is preferable.

We devised a procedure to merge the set of VSMs into a global scene model with a single geometric description (a 3D triangle mesh, or set of 3D meshes for unconnected objects) and a single texture map. We call this global representation a *Complete Surface Model* (CSM). To recover the CSM geometry, we merge models of the VSMs using an adaptation of the Curless and Levoy volumetric integration algorithm [3]. This technique has shown tremendous resilience even to the gross errors common in stereo-computed range images [20]. Other techniques, such

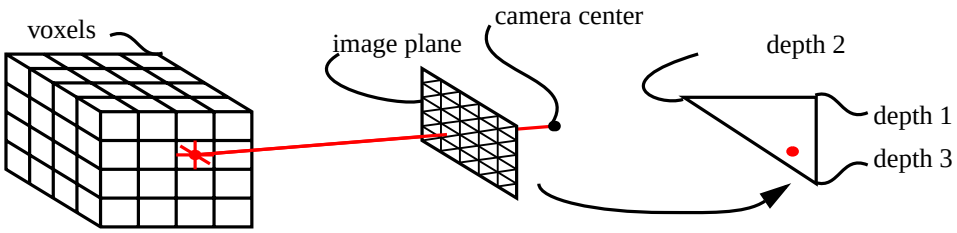


Figure 5: Basic operation in computing signed distance. Each voxel is projected into the image plane of each camera using the camera models already computed for stereo. The range image is interpolated to compute the distance from the camera to the surface, from which the signed distance from the voxel to the surface is computed.

as those that fuse 3D points or triangle meshes, tend to perform poorly in the presence of even relatively small errors. CSM texture is computed by back-projecting the real images onto the CSM geometry.

5.1 Volumetric Merging of VSMs

The volumetric integration algorithm fuses range images within an object-centered volume divided into voxels, or volume elements. Each voxel accumulates the signed distance to the surfaces in the VSM. The signed distance is computed in three steps (see Figure 5). First, the voxel is transformed into camera coordinates using the following equation, with $\bar{\mathbf{p}}_\mathbf{w}$ and $\bar{\mathbf{p}}_\mathbf{c}$ now representing the world and camera coordinate positions, respectively, of the current voxel and $\mathbf{R}$ and $\bar{\mathbf{T}}$ representing the rotation and translation respectively of the camera in the world coordinate frame:

$$\bar{\mathbf{p}}_\mathbf{c} = \mathbf{R}\bar{\mathbf{p}}_\mathbf{w} + \bar{\mathbf{T}}$$

Next, the camera coordinate $\bar{\mathbf{p}}_\mathbf{c}$ is projected into the image to derive the image coordinates $(\mathbf{x_f}, \mathbf{y_f})$ of the voxel. Finally, the signed distance $\mathbf{g}$ is computed by subtracting the depth $\mathbf{d}$ at pixel $(\mathbf{x_f}, \mathbf{y_f})$ (linearly interpolated from the vertices of the triangle on which it falls) from the depth of the voxel, $\mathbf{Z_c}$:

$$\mathbf{g}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w}) = \mathbf{Z_c} - \mathbf{d}(\mathbf{x_f}, \mathbf{y_f})$$

The voxel $\mathbf{V}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w})$ accumulates the signed distance across its projections into all cameras:

$$\mathbf{V}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w}) = \sum_{\mathbf{i}=1}^{\mathbf{N_{cameras}}} \mathbf{g_i} = \sum_{\mathbf{i}=1}^{\mathbf{N_{cameras}}} [\mathbf{Z_{c_i}} - \mathbf{d_i}(\mathbf{x_f}, \mathbf{y_f})]$$

If the estimated structure in the VSMs have unequal reliability, the signed distance can be weighted by that confidence, possibly with a different weight for each 3D point in every VSM (see Figure 6(a)):

$$\mathbf{h_i}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w}) = \mathbf{w_i}(\mathbf{x_f}, \mathbf{y_f})\mathbf{g_i}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w})$$

$$\mathbf{V}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w}) = \sum_{\mathbf{i}=1}^{\mathbf{N_{cameras}}} \mathbf{h_i}(\mathbf{X_w}, \mathbf{Y_w}, \mathbf{Z_w})$$

This projection process, repeated for each voxel and for each VSM, converts the explicit geometry of the individual VSMs into an implicit surface embedded in the volume. In particular, since the signed distance is zero for points on the real surface, the volume's isosurface of level zero represents the surface geometry in the scene (see Figure 6(b)). This geometry can be recovered by extracting this implicit surface. Isosurface extraction is well studied and has standard solutions such as the Marching Cubes algorithm [1][15], which tessellates the implicit surface into a triangle mesh. This is the method we use to extract the geometric component of the CSM.

We divide the volume of voxels 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 and the closest surface contained in it, corresponding to negative values of $\mathbf{g}$. The voxels in the near-surface volume are within some threshold (absolute) distance from the surface $(-\varepsilon < \mathbf{g} < \varepsilon, \varepsilon > 0)$. 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 $\mathbf{g}$). Both our algorithm and the one of Curless and Levoy handle near-surface and occluded voxels in the same way. Near-surface voxels are updated as previously described. Occluded voxels are not updated because they may lie on a real surface occluded from the view of the VSM under consideration.

The main difference between the Curless and Levoy algorithm [3] and our own is the way the empty volume is handled. They update the empty voxels but reduce the weight $\mathbf{w_i}(\mathbf{x_f}, \mathbf{y_f})$ of the VSM contribution to zero in a process they call space carving. This approach allows VSMs to mark voxels as "empty" if they have not already been seen by another VSM, but will not alter any accumulation already stored there. This approach works well for relatively accurate input range images such as those generated using a laser scanner. In the presence gross inaccuracies, however, this approach would propagate the errors into

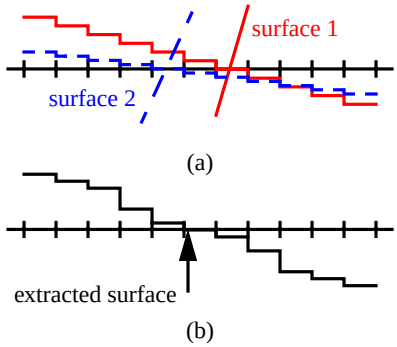


Figure 6: 1D example of volumetric fusion. (a) Two surfaces are added to the volume. Surface 1 has higher weight and therefore contributes more to the final result. (b) Final surface is at the zero crossings of accumulated values in the voxels.

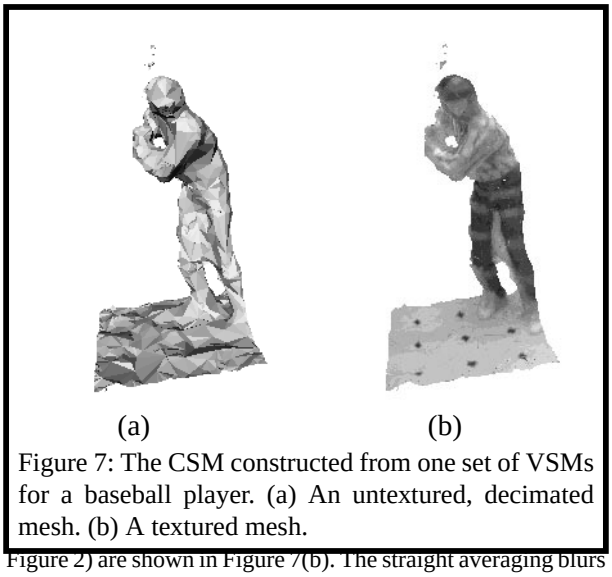
the global model because the zero weighted votes for "empty" would have no effect on the non-zero-weighted erroneous surfaces. Since stereo-computed range images commonly include large errors, we must use a different method to handle this case.

We continue to use the normal weights attached to each 3D point when we encounter "empty" voxels and clamp the weighted, signed distance to lie within the limited range $[-\delta, \delta]$. With this approach, VSMs can vote to "erase" erroneous surfaces if they see any voxel as being empty, while the clamping prevents a single view from overwhelming all others. We also add an extra step to eliminate any surface that is not visible to any of the VSMs. These surfaces are created by VSMs that overestimate the distance to a surface, which frequently places the estimate beyond all visible surfaces in the scene, for instance, "inside" a solid object. Our modifications effectively eliminate false surfaces because most range images will vote strongly against the existence of the false surfaces.

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. An example if the baseball bat in Figure 7(a), most of which has been eliminated. In this case, the baseball player's head blocked the bat's visibility in many images, and even when it was visible, stereo had difficulty because of the thinness of the bat. The integration, however, has performed well, even preserving the correct topology of the player's arms forming a torus. 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. The effect of this on our data is less serious than that of the noise present in the VSM surface models, so the CSM geometry is still an improvement. We are currently exploring ways to preserve the noise rejection property while eliminating these two drawbacks.

5.2 CSM Texture Modeling

The volumetric integration process creates a 3D triangle mesh representing the surface geometry. To complete the CSM, a texture map is constructed by projecting each intensity/color image onto the model and accumulating the results. Several methods can be used to accumulate this global texture map. A simple approach is to average the intensity from all images in which a given surface triangle is visible. It is also possible to weight the contribution of each image so that the most "direct" views dominate the texture computation. We construct a global texture map in which each non-overlapping MxN section of the texture map contains two texture triangles which map to two 3D triangles. The two triangles are defined with a shift of one pixel so that the pixels do not interact. Each texture triangle is applied to its geometric triangle by relating the corresponding vertices. This approach allows each 3D triangle to have a texture map, rather than just a single color. The main drawback of this implementation is that neighboring texture triangles are not necessarily neighboring geometric triangles, and so filtering of the texture map must be done with extreme care. A more efficient approach would be to map contiguous geometric triangles into contiguous texture triangles. The results of texturing one frame of a baseball sequence (with views similar to those in Figure 2) are shown in Figure 7(b). The straight averaging blurs the texture map. The main source of error in both cases is the residual geometric error in the CSM geometry. The model is accurate only to within several pixels when projected into the original images. This is due to a number of sources including calibration error, stereo noise, and bias in the fusion algorithm. This error blurs the back-projected textures like a poorly focused camera. This suggests that a view-dependent texture mapping approach such as our VLM method or the method of Debevec et. al. [4] may provide higher quality in the presence of moderate geometric error.



(a) (b)

Figure 7: The CSM constructed from one set of VSMs for a baseball player. (a) An untextured, decimated mesh. (b) A textured mesh.

5.3 CSM Mesh Decimation

One drawback of the volumetric merging algorithm is the large number of triangles in the resulting models. For example, the dome alone at 1 cm voxels, can create a model with 1,000,000 triangles. The number of triangles in the model is directly related to the resolution of the voxel space, so increasing the voxel resolution will increase the number of triangles in the final model. We can apply the same edge-collapse decimation algorithm [8] to reduce the number of triangles. Since the global model represents the scene independent of any viewpoint, it must preserve the overall structure well. All geometric information is thus of equal importance unlike while decimating the VSMs, when boundary errors were more important than interior errors. Such a natural separation does not exist for a global model as any part of the scene can become a boundary point. The gains from decimation are still large (an order of magnitude is typical), but are less spectacular compared with the reduction of a set of VSMs. The geometric model in Figure 7 is actually a decimated model.

5.4 Rendering Using the CSM

Rendering using the complete surface model of the scene is an easy task as the model is a view-independent geometric description of the scene. The model can easily be converted to a format like the Open Inventor format and manipulated using standard tools. Conventional graphics rendering engines are optimized for this task and render them directly. All the images in Figure 7 were rendered using Open Inventor tools.

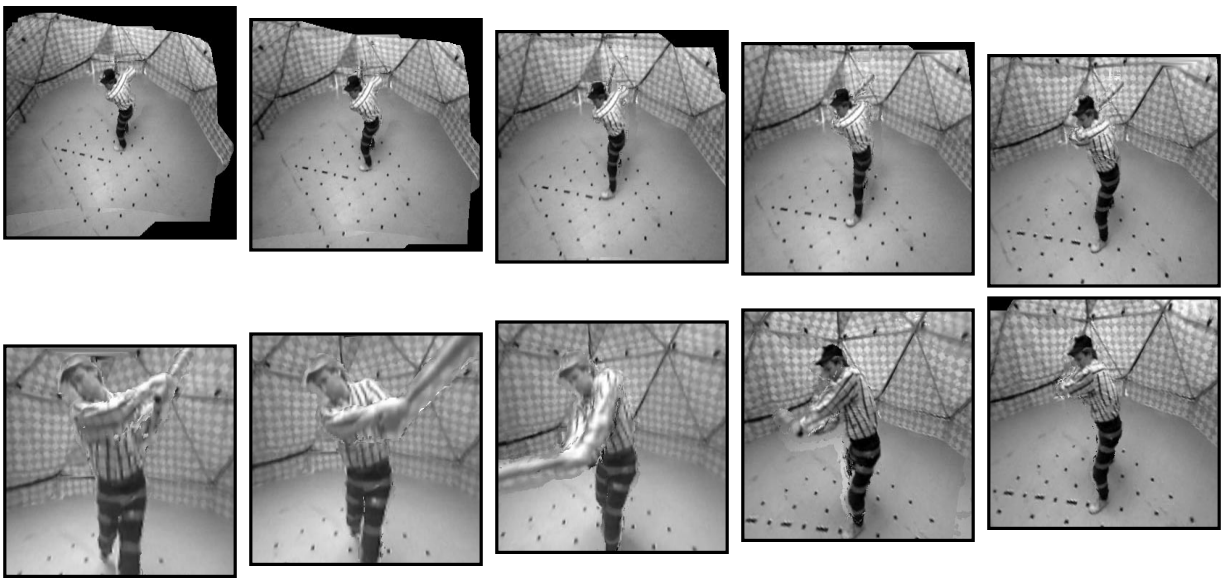


Figure 8: Some views of a dynamic scene. The virtual camera drops down into the scene from high above the batter to a point near the path of the baseball bat.

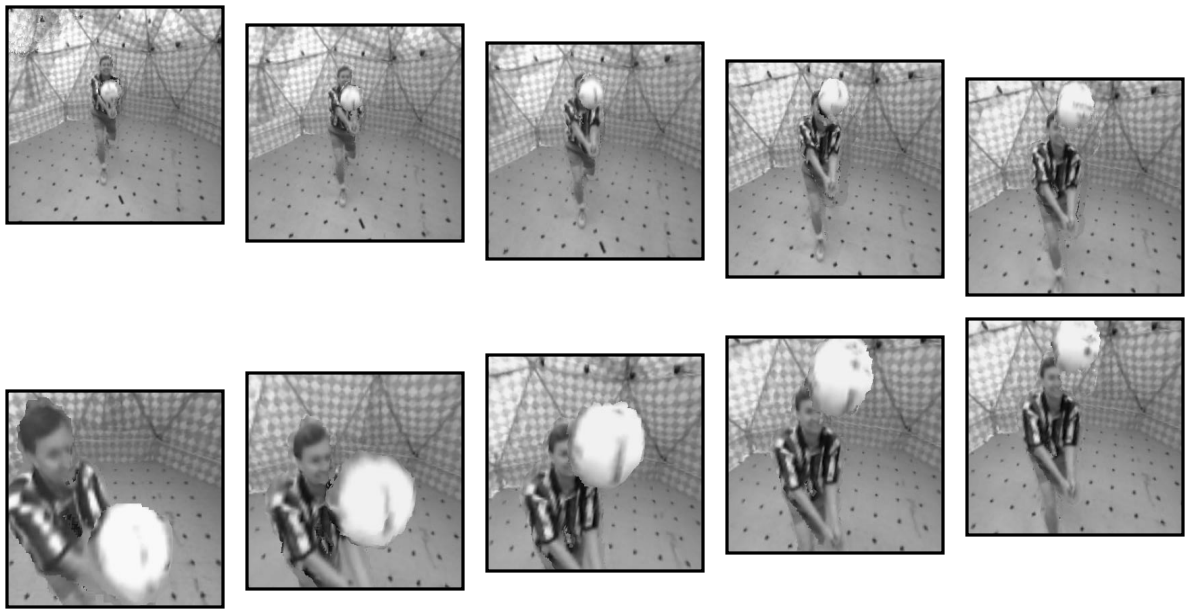


Figure 9: Some views of a virtual camera quickly moving into a scene with a volleyball player bumping a volleyball.

6 Experimental Results

We present a few representative results from 3DDome in this section. The facility currently consists of 51 cameras mounted on a 5-meter diameter geodesic dome, providing viewpoints all around the scene. We currently use monochrome cameras with a 3.6mm lens for a wide (about 90^o horizontally) view. Any arrangement of cameras that provides dense views of the event from all directions will suffice because the calibration procedure will determine the positions of the cameras in a world coordinate system. Each camera view is recorded onto a consumer grade VCR and processed off-line. The cameras are calibrated for their intrinsic parameters independently and for their extrinsic parameters after fixing them in place. The 3DDome captures every frame of the event from each camera angle, maintaining synchronization among the images taken at the same time instant from different cameras. Synchronization is crucial for the virtualization of time-varying events because the stereo process assumes that the input images correspond to a static scene. See [18] for more details on the recording and synchronization mechanisms we use.

6.1 Basic Dynamic Scene Analysis

Figure 8 illustrates the application of Virtualized Reality to a dynamic scene of a baseball player swinging a baseball bat. We currently treat a dynamic event as a sequence of static events, each corresponding to a time instant. At the beginning of the

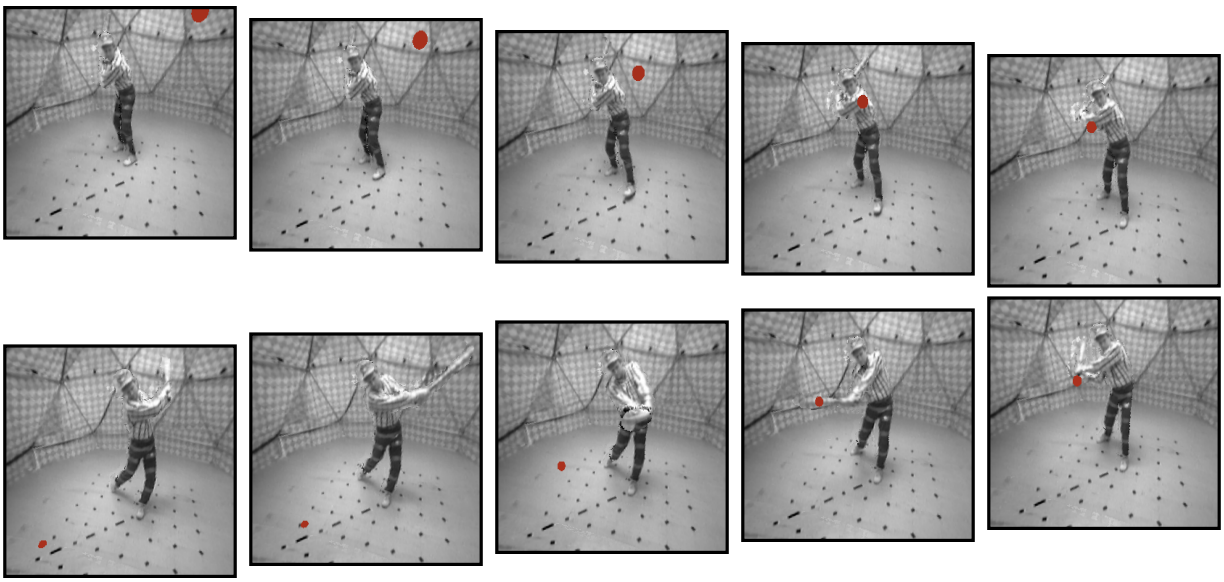


Figure 10: Introducing a virtual ball into a virtualized scene.

swing, the virtual camera is high above the player. As the player swings the bat, the viewpoint drops down and into the motion, approximately to the level of the bat. Figure 9 includes another example, this time with a volleyball player bumping a volleyball. Note that it this case, the ball is floating freely, unconnected to the floor or to the player. Both examples were generated using the VLM rendering approach. The CSM-based approach generates similar images, but the results are more blurred as a result of the geometric distortion of the model.

6.2 Combining real and virtual models

Because a virtualized environment is a metric description of the world, we can easily introduce virtual objects into it. A virtual baseball, for example, is introduced into the virtualized baseball scene, creating the views shown in Figure 10. Note that the rendering of the virtual object can be performed after the synthesis of the virtual camera image without the objects or concurrently with the virtual objects. It is possible to use this approach to extend chroma-keying, which uses a fixed background color to segment a region of interest from a real video stream and then insert it into another video stream. Because we have depth, we can perform Z-keying, which combines the multiple streams based on depth rather than on color [10]. In fact, it is even possible to simulate shadows of the virtual object on the virtualized scene, and vice versa, further improving the output image realism.

6.3 Color Texture on CSMs

We use monochrome cameras currently due to their low costs, though color would improve the correspondence results and hence the depth maps. We can provide color texture for synthesis using only a few color cameras, placed carefully and calibrated to the same world coordinate system as the other cameras. We achieve this by computing a CSM for the scene using the monochrome images, and then replacing the monochrome texture map with a color one computed by projecting the color images onto the recovered global model of the scene. Alternatively, view-dependent texture mapping [4] could be used with only the color images as texture. Figure 11 shows one frame of a sequence of CSMs with texture from 4 color cameras. Gaps in the coverage of the color cameras have been filled in with monochrome texture.

Figure 11: A view of a CSM with (partial) color texture added with 4 color cameras

7 Virtualized Reality and IBR

We now discuss the role of Virtualized Reality in the study of other image-based rendering methods. The virtualized environments contain metric models of the Euclidean space with photorealistic texture. Thus, geometrically correct views of the environment can be generated for any given camera model. Thus, Virtualized Reality can be a tool to generate inputs for IBR techniques. For instance, view interpolation uses two views of a scene with pixel correspondence between them to synthesize any intermediate view on the line connecting them by interpolating the pixel flow. The interpolation is an easy step that can be done on a PC as long as correspondences are available. The new viewpoints must lie in a space defined by a linear combination of the reference views, however. It is possible in theory to arrange the cameras in multiple layers in 3D space to accommodate any user motion, specified *before* recording; it is not practical to arrange them without coming in the way of one another. Virtualized Reality with its metric models can provide

images and pairwise correspondences from any location to extend view interpolation to 3D space. A server can render a few views and associated pixel flows which will be interpolated by a personal view station, which asks for new views from the server only when necessary. Similarly, images with pairwise fundamental matrices for projective reconstruction or images with the trilinear tensors for the triplets for tensor based reconstruction can be generated from a virtualized event. Plenoptic function has the advantage that all-around views can be generated from pixel interpolation alone; the required inputs for that can also be generated from a virtualized event by rendering it onto a cylindrical retina. Virtualized Reality can therefore exploit the advantages of other IBR strategies by subsuming themat the same time extending them to dynamic events. That is very significant to the field based methods that require thousands of simultaneous views of each instant of the event to construct their representation.

8 Conclusions and Future Work

Virtualized Reality provides a significant new capability in image-based scene modeling and rendering by extending it to dynamic events, with no restrictions on the positions from which the view can be synthesized. It closes the gap between VR and real events using imaging and makes it possible to generate a virtual model of a dynamic event its images. A manipulable three-dimensional model of a real event is provided by it.

Virtualized Reality is based on a novel combination of computer vision and computer graphics techniques. On the vision side, we are investigating ways of improving the performance of stereo, especially along occluding contours. We are also exploring alternate calibration strategies to reduce errors due to calibration. On the computer graphics side, many image-based techniques focus on ways of allowing low-end systems to synthesize viewpoint in real time. Virtualized Reality can take advantage of these techniques by incorporating them into the synthesis engine. For example, to use view interpolation, a series of images, with rectified cameras if necessary, and their associated flow vectors can be generated using the virtualized event model.

References

1 J. Bloomenthal. An Implicit Surface Polygonizer. *Graphics Gems IV*, ed. P. Heckbert, 324-349, 1994.

2 E. Chen and L. Williams. View Interpolation for Image Synthesis. *SIGGRAPH'93*, 1993.

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

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

5 O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig? *Proceedings of the European Conference on Computer Vision*, 1992.

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

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

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

9 Takeo Kanade, P. J. Narayanan, and Peter Rander. Virtual*ized* Reality: Concept and Early Results. *IEEE Workshop on the Representation of Visual Scenes*, Boston, June, 1995.

10 T. Kanade, A. Yoshida, K. Oda, H. Kano, M. Tanaka. A Stereo Machine for Video-rate Dense Depth Mapping and its New Applications. *Proceedings of IEEE CVPR'96*, San Francisco, CA, June, 1996.

11 S. B. Kang and R. Szeliski. 3-D scene data recovery using omnidirectional multibaseline stereo. *Proceedings of IEEE CVPR'96*, San Francisco, CA, June 1996.

12 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.

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

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

15 W. Lorensen and H. Cline. Marching Cubes: a High Resolution 3D surface Construction Algorithm. *SIGGRAPH'87*, 163-170, July 1987.

16 L. McMillan and G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. *SIGGRAPH 95*, Los Angeles, 1995.

17 P. J. Narayanan, Peter Rander and Takeo Kanade. Synchronizing and Capturing Every Frame from Multiple Cameras. *Robotics Institute Technical Report*, CMU-RI-TR-95-25, Carnegie Mellon University, 1995.

18 P. J. Narayanan, Peter Rander and Takeo Kanade. Constructing Virtual Worlds Using Dense Stereo. *IEEE International Conference on Computer Vision*, Bombay, 1998.

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

20 Peter Rander, P. J. Narayanan and Takeo 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.

21 Amon Sashua. Algebraic Functions For Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 17(8):779-789. 1995.

22 S. M. Seitz and C. R. Dyer. Towards Image-Based Scene Representation Using View Morphing. *Proceedings of 13th International Conf on Pattern Recognition*, 1996.

23 S. M. Seitz and C. R. Dyer. View Morphing. *SIGGRAPH'96*, August 1996.

24 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.

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