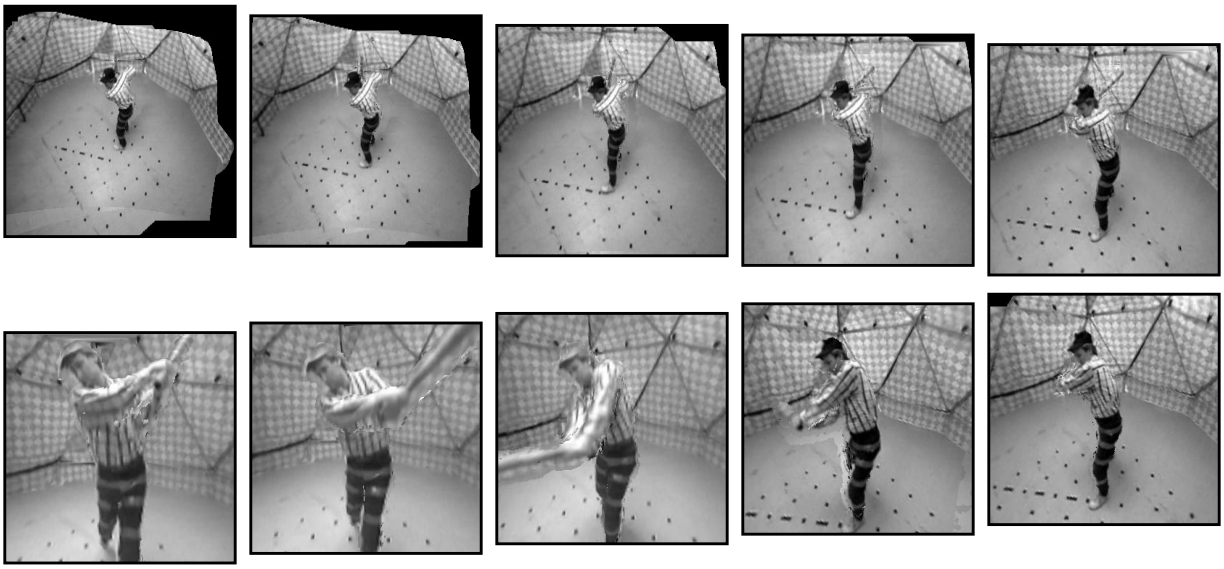


Figure 7: Some views of a virtual camera dropping down into the scene from high above the batter to a point near the path of the baseball bat.

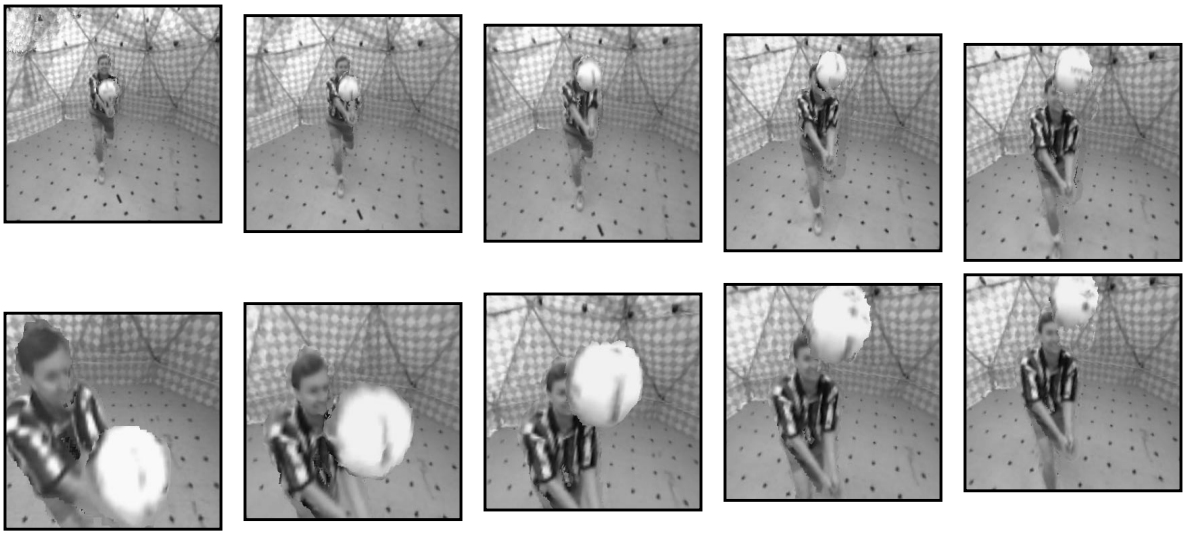


Figure 8: Some views of a virtual camera quickly moving into a scene with a volleyball player bumping a volleyball.

Figure 9: A view of a CSM with (partial) color texture added with just 4 more cameras

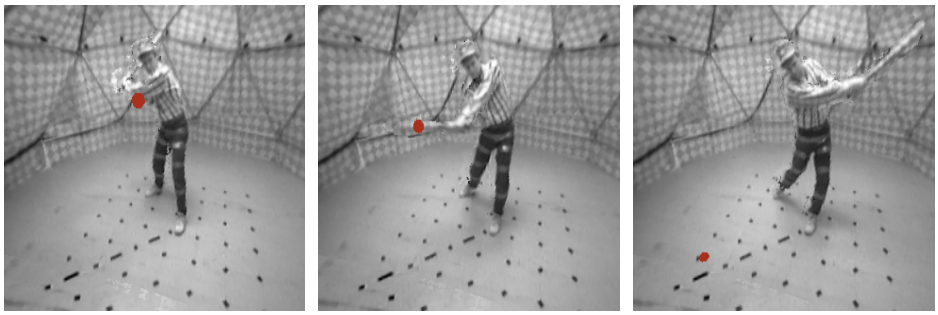


Figure 10: Introducing a virtual ball into a virtualized scene.

7.3 Combining Real and Virtual Models

Because a virtualized environment is a metric description of the world, we can introduce virtual objects into it. A virtual baseball, for example, is introduced into the virtualized baseball scene just discussed, 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 [12]. 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.

8 CONCLUSIONS

Virtualized Reality provides two significant new capabilities in image-based rendering. First, by combining image-based models with 3D scene geometry, our approach allows easy composition of complex 3D scenes from both virtual and virtualized objects. Second, while earlier methods usually worked with only a few views of static events, our approach addresses dynamic events viewed with a large number of cameras, requiring intelligent selection of subsets of views rather than exhaustive use of all available images. Virtualized Reality also provides a way to generate traditional virtual models of real events using a volumetric merging process, closing the gap between VR and real events using imaging.

Virtualized Reality is based on a novel combination of computer vision and computer graphics techniques. By improving the components of the system, the overall image quality ought to improve as well. On the vision side, we are investigating ways of improving the performance of stereo, especially along occluding contours. In traditional computer vision, which has frequently been motivated by autonomous navigation, such detail has been viewed as relatively unimportant. Given the ability of the human eye to quickly spot errors along these contours, however, it is very important to find better ways of detecting the occluding contours.

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. Plenoptic and omnidirectional models can be generated easily by rendering a sequence of images from a fixed viewpoint and then applying the appropriate panoramic modeling algorithm. Similarly, the dynamic event can be converted into a light field representation by synthesizing images from the multiple viewpoints these algorithms require.

9 REFERENCES

1 J. Bloomenthal. An Implicit Surface Polygonizer. *Graphics Gems IV*, ed. P. Heckbert, 324-349, 1994. (**ftp://ftp-graphics.stanford.edu/pub/Graphics/GraphicsGems/GemsIV/GGemsIV.tar.Z**).

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

6 A. Hilton. Reliable Surface Reconstruction From Multiple Range Images. *Proceedings of ECCV’96*, April 1996.

7 H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise Smooth Surface Reconstruction. *SIGGRAPH’94*, 1994.

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

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

10 [The Authors]. Virtual*ized* Reality: Concept and Early Results. *IEEE Workshop on the Representation of Visual Scenes*, Boston, June, 1995.

11 [The Authors]. 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.

12 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.

13 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.

14 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.

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

16 M. Levoy and P. Hanrahan. Light Field Rendering. *SIGGRAPH’96*, August 1996.

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

18 D. Marr. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. W.H. Freeman & Co., San Francisco, CA, 1982.

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

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

21 [The Authors]. Synchronizing and Capturing Every Frame from Multiple Cameras. *Robotics Institute Technical Report*, CMU-RI-TR-95-25, Carnegie Mellon University, 1995.

22 [The Authors]. Recovery of Dynamic Scene Structure from Multiple Image Sequences, *International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Washington, D.C., Dec. 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 G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. *SIGGRAPH’94*, July 1994.

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

(a) (b)

Figure 5: The CSM constructed from one set of VSMs for a baseball player. (The background has been eliminated for clarity.) (a) An untextured, decimated mesh. (b) The same mesh with texture.

5.4 Modeling Analysis

We now discuss several results from this modeling process. We use the example shown in Figure 5, the CSM constructed from one set of VSMs of a baseball player, for reference. First, construction of the CSM is extremely robust, even in the presence of gross and systematic errors in the range images. The process consistently reconstructs the correct topology for even relatively complex scenes. Note, for example, that the player's arms form a loop that is correctly reconstructed in the model.

There are two main problems remaining in the model. First, thin objects like the baseball bat in Figure 5 may be almost completely eliminated. This thin-surface problem occurs because the opposite sides of the object interfere with the integration of each others' estimates. Second, the resulting models are precise only to within several pixels when projected back into the cameras. This residual geometric error has contributions from a number of sources including calibration error, stereo noise, and bias in the fusion algorithm. This geometric error also propagates into the global texture map. Because the model does not exactly match the real images, the averaging process blurs detail out of the real images, creating a texture with less resolution than the original images. The textured model in Figure 5(b), for example, has lower visible resolution than the original images, even though the texture map itself actually has pixels comparable to the original images. This suggests that in the presence of geometric error, view-dependent texture mapping on a global model (as in [4]) may better preserve high resolution.

6 3D DOME: A VIRTUALIZING STUDIO

Figure 6 shows the picture of *3D Dome*, our facility to make virtualized models. It consists currently of 51 cameras mounted on a 5-meter diameter geodesic dome. Note that the dome only holds the cameras and is therefore not required; the cameras could be mounted directly to the walls and ceiling, for example. The cameras are placed at nodes and the centers of the bars of the dome, providing viewpoints all around the scene. We currently use monochrome cameras with a 3.6mm lens for a wide view (about 90^o horizontally) of the dome. *3D Dome* captures every video frame from each camera angle recording the events in the studio, maintaining synchronization among the 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.

6.1 The Studio Setup

Synchronous digital acquisition of multiple video streams is a difficult task due to the size of the data involved. Even a single monochrome camera generates 7.5 MBytes of data per second at 30 images of size 512x512 per second. Specialized systems can provide the necessary bandwidths for a few video streams at a high cost; a system involving a few tens of cameras would be prohibitively expensive.

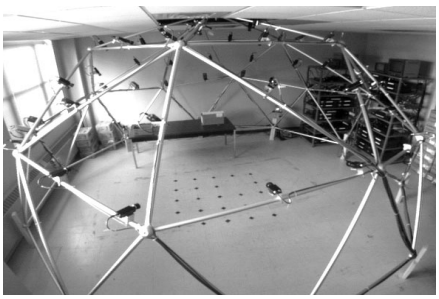


Figure 6: A view of *3D Dome*, our facility for virtualizing dynamic scenes

We perform digital acquisition in two steps: real time recording and off-line digitization. The recording system uses standard CCD cameras, electronically synchronized to a common signal, and consumer-grade VCRs. 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 tapes are digitized individually off-line, under the control of a computer program that interprets the VITC of each field in real time as the tape plays. The computer can identify and capture individual fields of video using the time code, allowing multiple passes (usually no more than four) over the same section of the tape to capture all the required frames or fields. The VITC data is also used to synchronize the video data from different cameras. A separate report [21] gives more details on the synchronous multi-camera recording and digitizing setup.

7 EXPERIMENTAL RESULTS

7.1 Analyzing a Basic Dynamic Scene

Figure 7 illustrates the application of Virtualized Reality to view synthesis a dynamic scene of a baseball player swinging a baseball bat. (Images from this sequence were used as demonstrations throughout the paper.) At the beginning of the 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 8 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.

7.2 Adding 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. Even if we do not replace all monochrome cameras with color ones, though, we can still provide color texture for synthesis using 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 9, for example, shows a view of a CSM with texture from 4 color cameras. Gaps in the coverage of the color cameras have been filled in with monochrome texture.

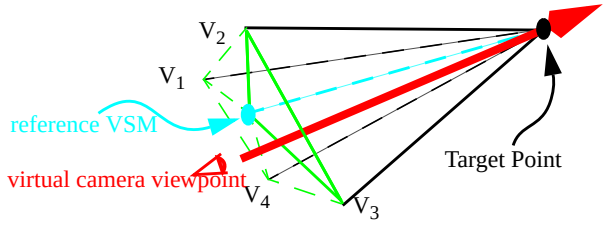


Figure 3: Selection of supporting VSMs. Given a reference VSM, the supporting VSMs are determined by first enumerating the neighboring VSMs (V_i) of the reference and then forming the triangles containing the reference VSM and two of the neighbors. The triangle pierced by the line of sight of the virtual camera determines the supporting VSMs.

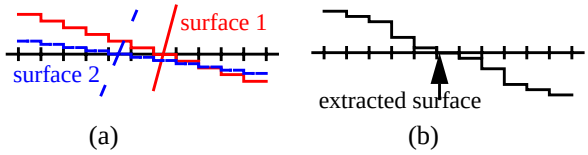


Figure 4: 1D example of the Curless-Levoy algorithm. (a) Two surface estimates contribute to the row of voxels. Each voxel accumulates the weighted signed distance to each surface. Surface 1 has higher weight and therefore contributes more to the final result. (b) The final surface is extracted at the zero crossings of the values accumulated in the voxels.

5 COMPLETE SURFACE MODEL

While image-based rendering techniques hold much promise as models in future graphics systems, existing systems do not support them. It is therefore desirable to find ways of converting image-based models into the single, complete geometric surface models understood by these existing graphics systems. For our system, we must therefore convert a set of VSMs into a global scene model with a single geometric description and a single texture. Since this global description should contain all the surfaces from the Visible Surface Models, we call it a *Complete Surface Model (CSM)*. Construction of a CSM from a set of VSMs involves construction of the geometric model and then of the global texture map.

To construct the geometry of the CSM, we fuse the range images of the VSMs using an adaptation of volumetric integration technique of Curless and Levoy [3]. Other techniques such as those that fuse 3D points [7] or triangle meshes [25] tend to perform poorly in the presence of noise, while the volumetric integration algorithm has shown tremendous resilience even to the gross errors in stereo-computed range images [22].

5.1 Volumetric Range Image Fusion

The volumetric integration algorithm fuses range images within an object-centered volume decomposed into voxels, or volume elements. Each voxel accumulates the signed distance to the surfaces in the range images, so the resulting volume implicitly represents the estimated global surface by the zero-crossings of this space. To update the voxel space, a new range image is tessellated into a set of triangular surfaces, connecting each pixel to its neighbors. The tessellation is broken when neighboring pixels have a large difference in depth, which allows multiple or self-occluding objects to be handled well. (This is the same strategy we use in Section 3.1 to compute a model for rendering with a VSM.) A weight may also be attached to each range estimate, allowing easy incorporation of range estimate reliability into the fusion process. Next, each voxel is projected onto the tessellated surface along the line of sight of the camera corresponding to the range image. From this projection, the signed distance from the surface to the voxel is computed. The weighted, signed distance to the surface is then accumulated at the voxel (see Figure 4(a)). The process is repeated for each voxel. After adding all the range images to the voxel space, the complete surface model is recovered by extracting the implicit surface, or isosurface, of level zero (see Figure 4(b)). Isosurface extraction is well studied and has standard solutions such as the Marching Cubes algorithm [1][17].

Our approach deviates from that of Curless and Levoy in one significant way. In order to carve out empty space, they add a zero-weight value to voxels far in front of the surface in the range image. When the range image has only small amounts of additive noise, this approach works fine. We are, however, working with stereo-computed range images, which also include large errors and erroneous surfaces. Applying this approach to our data would propagate these false surfaces into the final model. To reduce this effect, we must use a non-zero weight on the contribution to voxels far in front of the range image surfaces. (Unfortunately this step may introduce a bias into the voxel space, which can move the reconstructed surface away from its true position, but this effect is considered acceptable in order to reject the range image noise.) This effectively eliminates false surfaces in front of the real ones because almost all range images will vote strongly against the existence of the false surface. To eliminate false surfaces behind real surfaces in the CSM, a simple visibility test can be applied: if a surface is not visible to any of the cameras providing range images, then it can not be a real surface estimated from these cameras and it can therefore be eliminated.

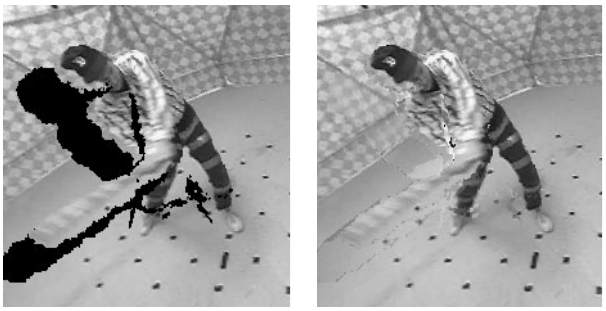
5.2 CSM Mesh Decimation

One drawback of this fusion algorithm is the large number of triangles in the resulting models. For example, fusing range images of our dome itself, with no foreground objects, at 1 cm voxels, can create a model with 1,000,000 triangles. In addition, the number of triangles is directly related to the resolution of the voxel space. To get manageable models, it is necessary to decimate the triangle meshes. We apply an edge-collapse decimation algorithm [9] (the same one used for decimation of the VSM mesh) to reduce the number of triangles. Because the global model must represent the scene from any viewpoint, it must preserve the overall structure well. Unlike the decimation for a VSM, then, all geometric information is roughly of equal importance. The gains from decimation are still large (an order of magnitude is typical), but are more limited than with a set of VSMs.

5.3 CSM Texture Modeling

The volumetric integration process creates a 3D triangle mesh representing the scene. Using this model, we can now construct a texture map by projecting the mesh into each camera and accumulating intensity values. Several methods can be used to accumulate this global texture map. A simple approach is to average the intensity from all images to which a given triangle is visible. It is also possible to weight this averaging by the surface normal of the triangle and by the size of triangle when projected into the image. For simplicity, we only use the straight-average approach. The results of modeling one frame of a baseball sequence (with views similar to those in Figure 1) is shown in Figure 5.

An alternative to constructing a single global texture is to use view-dependent texture mapping [4]. In this approach, a (hand-constructed) global geometric structure is textured by dynamically blending real images based on the visibility of the surfaces from the virtual and real camera positions. We can achieve a similar effect by using the global geometric structure from the CSM and the local textures from the VSMs.



(a) (b)

Figure 1: (a) Holes appear when the virtual camera moves away from a VSM. (b) Supporting VSMs can fill the holes.

of boundary errors relative to the interior errors. This controllability allows us to match the mesh to the human eye's ability to discern detail. A well-known visual effect is that humans can tolerate significant surface errors over continuous surfaces, but errors at 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. Accordingly, the decimation program gave maximum priority to maintaining the boundary points while substantially decimating interior surfaces. The model could typically be decimated from 100,000 triangles to 5000 triangles without appreciable deterioration in the rendered images.

4 VSM-BASED RENDERING

Our virtualized world consists of a collection of VSMs for each time instant. We compute a VSM with each camera used in the recording process as the origin. This is done by initiating a stereo process with each camera as the reference camera and with its neighbors providing the baselines. Scope for economizing the number of VSMs for a given output quality is an issue we want to explore in the future. The virtualized world can be navigated either directly using the collection of VSMs or using the global model computed from the VSMs as described in Section 5. We describe the former method in this section.

A single VSM represents all surfaces visible from its origin, limited to the viewing frustum of the camera. This model can be used to render the scene from locations near its origin as long as the viewing frustum of the virtual camera is within the model's frustum. Holes are created in the synthesized image when the virtual camera moves away from the origin as occluded areas of the VSM become exposed, as shown in Figure 1(a).

We can typically find another VSM to fill the hole given a sufficiently dense distribution of VSMs. 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 would fill the holes created. Thus a combination of a *reference* VSM to provide most of the view plus a small number of *supporting* VSMs to fill the holes can provide nearly hole-free synthesized views, as shown in Figure 1(b). We call this combination the *view-dependent local model (VLM)*.

Another way to look at our method is as geometrically-correct view interpolation, with or without intensity interpolation (i.e., cross dissolve). It also has some similarity to the view-dependent texture mapping [4], as we discuss in Section 5. We can switch to a field-based approach like lightfield or lumigraph if the distribution of the input cameras is really dense. Practically, that would typically mean (tens of) thousands of cameras, though, requiring a significant engineering effort to design the camera system.

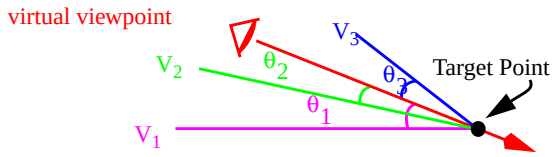


Figure 2: Selecting the reference VSM. θ_i is the angle between the virtual camera's line of sight and the line joining the target point with the position V_i of VSM i. The VSM with the smallest angle θ_i is selected as the reference VSM.

4.1 Computing the VLM

Rendering a model given the virtual camera parameters and the VLM essentially computes a visible surface model from the virtual camera's point of view. Thus, the VLM used for rendering should provide a hole-free view from that location. The VSM "closest" in approach to the virtual camera provides the most promising model and can serve as the reference VSM of the VLM. It will typically not be hole-free on its own, so the VLM includes two neighboring VSMs for support. These VSMs are selected such that they collectively are likely to cover all holes in the virtual camera view when using the reference VSM alone. We explain the process of identifying the VLM below. Note that our earlier work did not address this issue because there were only 2 VSMs available, both of which had to be used at all times.

The problem of covering all possible holes with a finite number of VSMs has no good solutions. It is always possible to create a scene and a virtual camera location that will not be hole free for any given arrangement of VSM origins. Our approach works well when the following conditions are satisfied. First, VSMs are fairly densely distributed with their viewing frustums covering approximately the same event space. The virtual camera also is focussed in the same region (i.e., it is not looking away). These assumptions are reasonable for our setting; they are not to be seen as limitations of the system as our system can be extended to other situations.

4.1.1 Selecting Reference VSM

The viewing direction of the virtual camera is specified in terms of an eye point and a target point. We use the angle between the viewing direction and the line connecting the target point to the VSM origin as the measure of the closeness of the virtual camera to the VSMs, as shown in Figure 2. This measure works well when both the virtual camera and all the VSMs are pointed at the same general region of space.

4.1.2 Selecting Supporting VSMs

The supporting angles are used to compensate for the occlusion in the reference description. Once we select the reference VSM, consider the triangles formed by its origin and all adjacent pairs of its neighboring VSMs, as shown in Figure 3. 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.

4.2 Rendering Using The VLM

We use a three step procedure to synthesize the view of the virtual camera using a VLM. First, we render the scene using the reference VSM. Next, the hole triangles of the reference VSM are rendered into a separate buffer to identify the pixels belonging to the hole region. Lastly, the supporting VSMs are rendered, limiting their rendering to the hole region. This process is shown in Figure 1.

2.3 Number of Images

Intuitively we expect that image-based rendering algorithms prefer larger number of cameras, at least to increase output image quality. Field-based methods, however, *require* many images to work at all. Levoy and Hanrahan [16], for example, use as many as 8000 images to compute the light field for a single scene. Correspondence-based methods, on the other hand, require few images by comparison. Our system, for example, uses only 51 images. Motion-based systems also need relatively few images.

2.4 Scene Model and Virtual Camera Position

All view synthesis methods use an implicit or explicit model of the world. The techniques presented here model either the object shape or the field of light rays passing through a region of space. Because of the limitations of these models, the virtual camera may have limited mobility or may have restrictions on how the viewpoint is specified.

All correspondence-based methods model the scene geometry in one way or another. View interpolation and view morphing use only pixel correspondences as an implicit model of the scene geometry. This model lacks explicit knowledge of how these correspondences relate to shape. As a result, it is valid only in a space limited by the input image locations, which restricts the synthesized viewpoint to a linear combination of the input image locations. View transfer does away with this restriction by explicitly incorporating weak calibration. Because the structure is known only up to a projective transformation, however, the virtual camera can not be specified in a metric coordinate system. Rather, the position of the virtual camera must be specified in terms of epipolar relationships with the input cameras.

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 frequently unimportant, since many viewing interfaces already adjust this parameter. 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. MPI Video starts with an *a priori* metric map of the scene and can synthesize any viewpoint. 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 one real camera.

2.5 Dynamic Scene Analysis

We now consider the dynamic event capabilities of these methods. MPI Video and Virtualized Reality have, by design, addressed the dynamic issues. All other techniques have been demonstrated only for static scenes, usually using the same imaging system (moving the camera or the object to get multiple views of static scenes). Extension to time-varying imagery would require significant hardware modifications but few algorithmic changes, at least in theory. In practice, the human-assisted methods to compute pixel correspondences used in the first five methods would probably have to be replaced by automatic methods. The field-based approaches extend easily algorithmically, but the hardware and software issues of designing a several-thousand-camera imaging system are certainly non-trivial.

3 VISIBLE SURFACE MODEL

We want to compute the model of all surfaces visible from a camera's viewpoint, similar to the 2-1/2 D sketch of the Marr paradigm [18]. Our structure recovery method, multibaseline stereo [20], produces a dense depth map (or range image) that gives the distance in a camera-centered coordinate system to each pixel of the associated image. We first convert the output of the stereo process to a triangle mesh model of the surface.

3.1 Creating the VSM

The *(x, y, z)* coordinates of each pixel can be computed using its depth and pixel coordinates, given the camera calibration information. We convert the resulting cloud of points to a triangle mesh model of the visible surfaces assuming local connectivity provided by the pixel array. The triangles with a large difference in depth across any side lie on occlusion boundaries (or extremely foreshortened surfaces which are approximately occlusion boundaries) corresponding to surfaces and regions that are *not* visible. These triangles are identified using a threshold on the depth difference and marked as *hole* triangles, not to be rendered. The image associated with the depth map is used to texture map the model. Texture coordinates are easy to compute as the triangle vertices fall on image pixels. Since the computation of this mesh is straightforward and reversible, we use the term *Visible Surface Model (VSM)* to refer both to the aligned pair of depth and intensity images and to the corresponding mesh.

The VSM has several interesting properties:

- The VSM serves as a textured triangle mesh model of surfaces visible from its origin with the occluded surfaces left blank.
- The geometry of the VSM guarantees that the synthesized view will be exact when the virtual camera is at the origin of the VSM, even with errors in the recovered geometry. The view remains very realistic as long as the virtual camera is close to the origin. The synthesized view will have holes as the virtual camera moves away from the origin of the VSM.
- The simple procedure to generate the VSM creates a large number of triangles. An $\mathbf{N} \times \mathbf{N}$ depth map will map to $O(N^2)$ triangles.

3.2 Overcoming Correspondence Errors

Stereo programs fare poorly in regions around depth discontinuities because of the difference in what different cameras see in that region. This results in either the foreground surface expanding into the background or the other way around, a process we call *fattening* of the range image surfaces. VSMs computed from stereo inherit this shortcoming. We developed two solutions to it. The first solution, used in our earlier work, is a human supervised editing system that corrects the inconsistencies in a single VSM. This is a step analogous to film editing and typically takes 5 to 10 minutes per VSM. The second solution involves the volumetric merging step described in Section 5. Fattening is reduced in the global model as the merging process enforces geometric consistency. The global model is then "rendered" into each camera to get a cleaner depth map from its point of view. This step increases the spatial scope of the VSM in addition to reducing fattening by replacing badly recovered pixels on the periphery of the image by good depth values from the global model.

3.3 Decimating the VSM Mesh

A typical triangle mesh computed from a range image contains many triangles 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. With such a finely tessellated mesh, decimation can substantially reduce the model complexity with little degradation of the model.

We use a simple edge-collapse decimation program [9] 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

Technique	Camera Calibration	Pixel Correspondence	Number of Images	Scene Model	Virtual Camera Position
View Interpolation [2][26]	None	Required	Low	Non-metric Shape	Limited
View Morphing [23]	Like weak	Required	Low	Projective Shape	Limited
View Transfer [15]	Weak	Required	Low	Projective Shape	Global (epipolar control only)
Plenoptic Modeling [19] Omnidirectional Stereo [13]	(Scaled) Strong	Required	Low	(Scaled) Metric Shape	(Scaled) Global (occlusion limited)
View-Dependent Texture Mapping [4]	Strong	Required	Low	Metric Shape	Global
Field Based (e.g., Lightfield, Lumigraph) [5][14][16]	Strong	None	High	Metric Light Ray Field	Outside convex hull
Multi Perspective Interactive (MPI) Video [8]	Strong	None	Low	Metric Shape	Global
Virtualized Reality	Strong	Required	Low	Metric Shape	Global

Table 1: Classification of view synthesis techniques

2 IMAGE-BASED RENDERING

View synthesis, i.e. the generation of new views of scene, has traditionally used geometry as the basic model underlying scene representations. 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 approaches achieve – even when the implicit geometric model is incorrect – make them particularly attractive. These techniques are commonly referred to as *image-based rendering* techniques. However, they 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 better understand the relationships among these image-based algorithms and Virtualized Reality, we classify them on the basis of (a) the type of calibration information used, (b) the need for pixel correspondences between pairs of images, and (c) the nature of the underlying "model" used and the restrictions this model places on the position of the virtual camera. Table 1 summarizes the image-based rendering algorithms and their properties.

2.1 Camera Calibration

Camera calibration fits the parameters of a general camera model to an actual imaging system. The camera model relates the 3D world to the camera's image plane. Traditional calibration, or *strong calibration*, computes the full camera model and therefore allows recovery of a metric (Euclidean) scene model. *Weak calibration*, on the other hand, only computes enough parameters to relate pixels of one image to lines in the other, i.e., the epipolar geometry, so the structure of the scene is known only up to an unknown projective transformation.

The view interpolation techniques require no explicit calibration data for synthesis, as long as image flow is given. This freedom allows interpolation without knowledge of the imaging systems, but can also introduce unwanted distortion of the underlying geometric structure. View morphing does not use conventional calibration information but extracts something like weak calibration data in order to rectify the images to a common plane. The view transfer technique requires pairwise weak calibration data between the triad of views: the two input views and the view being synthesized.

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].

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 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 imaging systems.

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. 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 [14]. Levoy and Hanrahan [16] and Gortler et al. [5] extend this concept to construct a four-dimensional field representing all light rays passing through a 3D surface. New view generation is posed as computing the correct 2D cross section of this field.

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 [8]. In this system, motion is detected by comparing each video frame to a "background" image and identifying differences. 3D models of the moving objects are computed by intersecting the viewing frustums of the pixels that indicate motion. 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.

Proc. of IEEE Visualization '97, Phoenix, Arizona, Oct. 19-24, 1997, pp. 277-283, 552.

Virtual*ized* Reality: Constructing Time-Varying Virtual Worlds From Real World Events

Peter Rander[1], PJ Narayanan[2], and Takeo Kanade[1]

The Robotics Institute
Carnegie Mellon University[3]

ABSTRACT

Virtualized Reality is a modeling technique that constructs full 3D virtual representations of dynamic events from multiple video streams. Image-based stereo is used to compute a range image corresponding to each intensity image in each video stream. Each range and intensity image pair encodes the scene structure and appearance of the scene visible to the camera at that moment, and is therefore called a Visible Surface Model (VSM). A single time instant of the dynamic event can be modeled as a collection of VSMs from different viewpoints, and the full event can be modeled as a sequence of static scenes – the 3D equivalent of video. Alternatively, the collection of VSMs at a single time can be fused into a global 3D surface model, thus creating a traditional virtual representation out of real world events. Global modeling has the added benefit of eliminating the need to hand-edit the range images to correct errors made in stereo, a drawback of previous techniques.

Like image-based rendering models, these virtual representations can be used to synthesize nearly any view of the virtualized event. For this reason, this paper includes a detailed comparison of existing view synthesis techniques with our own approach. In the virtualized representations, however, scene structure is explicitly represented and therefore easily manipulated, for example by adding virtual objects to (or removing virtualized objects from) the model without interfering with real event. Virtualized Reality, then, is a platform not only for image-based rendering but also for 3D scene manipulation.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Viewing Algorithms; I.4.8 [Image Processing]: Scene Analysis – Time-varying Imagery, sensor fusion, Photometry, Stereo; I.6.5 [Simulation and Modeling]: Model Development – Modeling Methodologies.

Additional Keywords: view synthesis, dynamic scene analysis, modeling from image sequences, computer vision and scene understanding, virtual worlds.

1 INTRODUCTION

Traditional visual media such as motion pictures and television bring spatially and temporally remote visual events to the general public. Despite their near-universal acceptance, these media typically provide only a two dimensional view of the event from a viewing angle determined by a director, limiting the immersive capabilities of the viewer. Virtual Reality (VR), in contrast, provides a viewer-controlled, 3D (stereoscopic) view of a virtual environment essential for an immersive experience. However, VR has found more success with artificially created virtual worlds than on re-created real events.

Recently, image-based rendering techniques have been proposed as a way to bridge this gap between photorealism and interactivity. These techniques model a scene as a collection of real images, from which virtual renderings may be computed. Because this image-to-image mapping does not explicitly require 3D scene structure, these techniques usually avoid this computation. Even without this information, these methods have achieved impressive results at redisplaying the original contents of the scene. Without 3D structure, however, it is difficult to combine these image-based models into larger scenes, which is frequently needed to build large VR worlds.

In contrast, Virtualized Reality is a modeling technique that constructs full 3D virtual representations of dynamic events from multiple video streams. The virtual-*izing* process begins with recording a real event using many cameras to provide video from multiple viewpoints. A multi-camera stereo process then computes a range image for each frame of each video sequence. The combination of an intensity image and its corresponding range image provides a simple model of the surfaces visible from the camera viewpoint at the moment the intensity image was captured. We call this model a vis*ible surface model (VSM)*. A static scene is represented by a set of VSMs captured at the same time. Dynamic scenes are represented as sequences of static scenes, just as video represents dynamic scenes by a sequence of static images. Alternatively, the collection of VSMs representing a single moment in time can be fused into a global 3D surface model of the scene. Since this model should contain all the surfaces visible in any single VSM, we call it the *complete surface model (CSM).*

As with traditional image-based models, synthetic views of the virtualized scene can be constructed from the viewpoint of a user-controlled virtual camera by projecting the virtualized models into the camera. In the virtualized models, however, the explicit encoding of scene geometry makes it easy to change, add to, or remove from the scene by editing in the 3D world rather than in individual images. One simple manipulation is to add color to a scene model computed from monochrome images, which can be achieved by adding only a few color cameras rather than having to replace all monochrome cameras. This same technique could be used to add virtual costumes to characters in the event. More powerfully, virtual objects can easily be added to the virtualized scene without interfering with the real scene. Conversely, virtualized components can be removed from the virtual model without removing them from the real scene, an ability that can be very useful when the real world scenes are cluttered with unnecessary but immovable objects.

We introduced Virtualized Reality and presented preliminary results using only a few cameras (providing only 2 VSMs) in earlier papers [10][11]. This paper contains new results using a virtualizing facility with many viewpoints (providing 51 VSMs) surrounding the event on a hemisphere. This multi-camera synchronous video recording system itself is unique, so this paper includes a brief overview of its design. In addition, while our earlier papers did not address on-the-fly selection of VSMs, this paper explicitly highlights and addresses the issues surrounding this problem, including how to find the real viewpoint "closest" to a desired virtual camera position. It also introduces a second method for generating synthetic images which uses the CSM. Constructing the CSM eliminates the need in the earlier papers to hand-edit range images to correct errors made by stereo; all the results presented in this paper were automatically computed, from the raw images alone.

[1][rander | tk]@cs.cmu.edu
[2]Now with the Centre for Artificial Intelligence and Robotics, Bangalore, India (pjn@cair.ernet.in)
[3]The Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213-3890