Constructing Virtual Worlds Using Dense Stereo

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

Peter W. Rander, Takeo Kanade
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
E-mail: {rander+, tk+}@cs.cmu.edu

Abstract

We present Virtualized Reality, a technique to create virtual worlds out of dynamic events using densely distributed stereo views. The intensity image and depth map for each camera view at each time instant are combined to form a Visible Surface Model. Immersive interaction with the virtualized event is possible using a dense collection of such models. Additionally, a Complete Surface Model of each instant can be built by merging the depth maps from different cameras into a common volumetric space. The corresponding model is compatible with traditional virtual models and can be interacted with immersively using standard tools. Because both VSMs and CSMs are fully three-dimensional, virtualized models can also be combined and modified to build larger, more complex environments, an important capability for many non-trivial applications. We present results from 3D Dome, *our facility to create virtualized models.*

1 Introduction

Traditional visual media such as motion pictures and television bring spatially and temporally remote visual events to the general public. These media currently 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.

Virtualized Reality is an immersive visual medium that gives a viewer the freedom to control the angle from which to view a *virtualized* event at view time. A real event is virtualized by recording it from multiple directions and computing the 2-1/2D structure of each scene as Visible Surface Models from multiple angles using a stereo technique. Immersive interaction with the virtualized event is possible using this collection of 2-1/2D structures alone. Alternatively, this collection can be merged to provide a 3D description of each scene and interacted with using traditional VR methodology.

View synthesis, or novel view generation, is not the only mode of interaction with the virtualized event. Virtualized models can be edited, modified or combined with other virtual or virtualized models seamlessly. Virtualized Reality thus differs from traditional virtual reality in that it constructs the models from images of the real world, preserving photorealism and fine detail. It is thus possible to create realistic virtualized worlds consisting of complex environments such as an operating room. Thus, one could create a virtual world of a skilled surgeon performing an operation that students could revisit immersively, free to observe from anywhere in the operating room. Or spectators could watch a virtualized basketball game from any stationary or moving point on or off the court.

We introduced the concept of Virtualized Reality and presented some preliminary results using a limited implementation of it with restricted viewer motion in earlier papers [9][10]. In this paper, we present *3D Dome*, the facility to create omnidirectional virtualization. We also present the method of creating complete three-dimensional descriptions of the event from multiple stereo views. We also show how immersive interaction with the virtualized world can occur either with the multiple visible surface models or with the complete surface model.

2 Related Work

Creating 3D models using range images has attracted some attention in the literature [3][6][7][24]. The emphasis there is in fusing existing depth maps together to get a high quality 3D model. These efforts have concentrated on the model building aspect, using high quality range maps as inputs. Image based rendering is another related area that has seen a lot of activity recently. These techniques typically require image flow or pixel correspondence, i.e. the knowledge about where points in one image move to in another image. View interpolation[2][25] interpolates the image flow vectors between two images at each pixel to generate intermediate views for any viewpoint on the line connecting the original two viewpoints. Seitz and Dyer [22] demonstrated that this yields physically valid images only if the source images are rectified. These algorithms restrict the synthetic view to a linear space defined by the reference viewpoints and cannot generate arbitrary views. Laveau and Faugeras [14] present a strategy that produces geometrically correct views for arbitrary viewpoints, but it requires the viewpoint to be specified in terms of epipolar geometry with respect to the original viewpoints. McMillan and Bishop [17] and Kang and Szeliski [12] construct cylindrical panoramic images from planar images and synthesize planar views from correspondences between panoramas. The panorama construction essentially provides calibration, so synthesized images are geometrically correct, except possibly for scale.

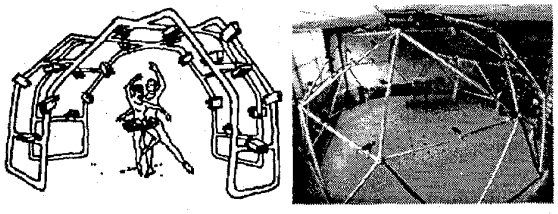


Figure 1: 3D Dome: Conceptual (left) Actual (right).

Another class of view synthesis techniques eliminates the need for pixel correspondences by densely sampling the viewing space, possibly interpolating missing views. Katayama et. al[13] demonstrated that images from a dense set of viewing positions on a plane can be used to generate images for arbitrary viewing positions. Levoy and Hanrahan [15] and Gortler et al. [5] extend this concept to constructing 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. These methods require the full calibration of each input view, but can generate correct synthetic views from any viewpoint outside the convex hull of the scene.

Another interesting approach used in Multiple Perspective Interactive (MPI) Video [18] is to apply motion analysis within each camera view and then build 3D environments by combining *a priori* environment models with dynamic motion models recovered by intersecting the viewing frustums of the pixels that indicate motion. Their modeling strategy requires precise segmentation of moving and stationary objects, a task that motion detection algorithms tend to perform poorly.

3 3D Dome: The Virtualizing Studio

We call our virtualized model-building facility the *3D Dome*. Figure 1 shows the conceptual 3D Dome and the real one. It consists currently of 51 cameras mounted on a 5-meter diameter geodesic dome. 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 3.6mm lenses for a wide view (about 90^{o} horizontally) of the dome. The geodesic dome is a convenient structure to hold the cameras to provide all-around views to an event inside it. However, any arrangement of cameras that provides dense views of the event from all directions can replace the dome structure since we apply camera calibration (see Section 4.2) to determine precise camera positions.

The 3D Dome 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. Note that 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.

To combat this difficulty, we developed a two-step approach to digital acquisition: 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 recording onto a 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 four-camera example is shown in Figure 2. A separate report [20] gives more details on this synchronous multi-camera recording and digitizing system.

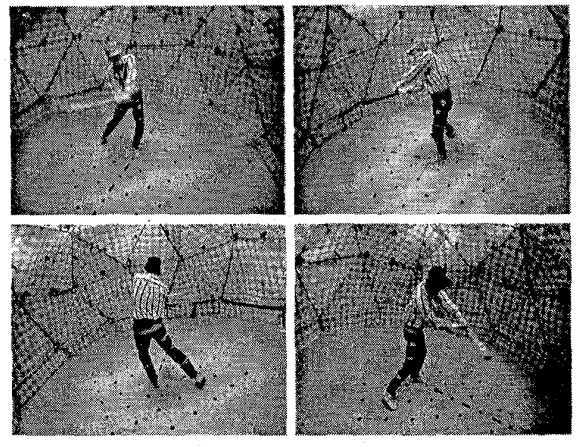


Figure 2: Four images of the same time instant of a dynamic scene captured with the Virtualizing Studio.

4 Visible Surface Models

We compute the geometric structure of each time instant of the scene using a multibaseline stereo technique, using the neighboring cameras to provide the multiple baselines for each camera view. The stereo process computes a dense depth map, aligned with the intensity image, which together encode the visible geometric and photometric structure of the scene. This pair can be translated to a textured polygon model using a simple procedure: convert each 2x2 section of the dense depth map into two triangles in 3D space by connecting a diagonal, with the corresponding section of the intensity image used as texture. (Each depth can be translated to (x, y, z) coordinates using the calibration parameters of the associated camera.) Triangles that span occlusion boundaries can be identified from the large difference in depth between two vertices and marked as "hole" triangles. We call a model so constructed the Visible Surface Model (VSM) of the scene. We interchangeably use the term VSM to refer to the textured triangle mesh model and the aligned depth/intensity pair since the construction of the former from latter is a straightforward step. The VSM is described in a coordinate frame defined by the reference camera of the stereo computation. Hence, the origin of the VSM is at that camera. An aligned pair of depth and intensity of a static scene from one viewing angle is shown in Figure 3.

(a) (b)

Figure 3: Visible Surface Model. (a) Intensity image (b) Aligned depth map, where distant surfaces are brighter.

4.1 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. 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.

4.2 Camera Calibration

Stereo algorithms require fully calibrated cameras to extract Euclidean structure. We calibrate for the camera parameters in two steps, using Reg Willson's implementation [26] of Tsai's method [23] in both steps. We first calibrate each camera's intrinsic parameters – those that affect the projection of points from the camera's 3D coordinates to the image plane – individually using a movable planar calibration object. The calibration process estimates five intrinsic parameters: focal length, aspect ratio. image center (two parameters), and radial lens distortion. We then place the cameras in their positions on the dome and calibrate each camera's six extrinsic parameters – rotation and translation relative to a world coordinate frame – using a set of dots on the floor visible to all cameras. We attach the world coordinate frame to one of these dots.

4.3 Distribution of Visible Surface Models

A VSM encodes the structure of the scene visible to a camera. By rendering the VSM using standard textured triangle mesh rendering algorithms, realistic synthetic views can be created for locations close to the VSM's position. In fact, the synthesized view will be exact when the virtual viewpoint lies at the VSM origin irrespective of any errors in the recovered geometry. As the virtual viewpoint gets farther away from the VSM's origin, unseen or "hole" regions of the VSM will be exposed and errors in the recovered geometry will also become more prominent. Minimizing the distance between the virtual camera and a VSM maximizes the quality of the synthesized images. This implies that VSM density should be uniform and high so that the closest one to any desired viewpoint produces an acceptable quality image.

In fact, if the density is high enough, the need for correspondences is eliminated, allowing direct interpolation of the images using algorithms like the lumigraph [5] or light field rendering [15]. These approaches, however, require an extremely high number of views. For example Levoy and Hanrahan [15] use as many as 8000 images to compute a light field for a single scene. For dynamic scenes, which require all the views to be captured simultaneously, the amount of hardware alone makes this strategy impractical.

The number of cameras is an upper bound on the number of VSMs. The distribution of cameras must take the needs of the stereo process into account as the results are better if the cameras are moved closer. Cost and the complexity of representation are the factors against increasing the number of cameras arbitrarily. We compute a VSM for each camera in the current setup. The number of cameras (51) was fixed to meet the requirements of stereo as well as the constraints of the dome geometry; we fixed a camera at every node and the middle of every bar of the dome.

4.4 Decimating Visible Surface Models

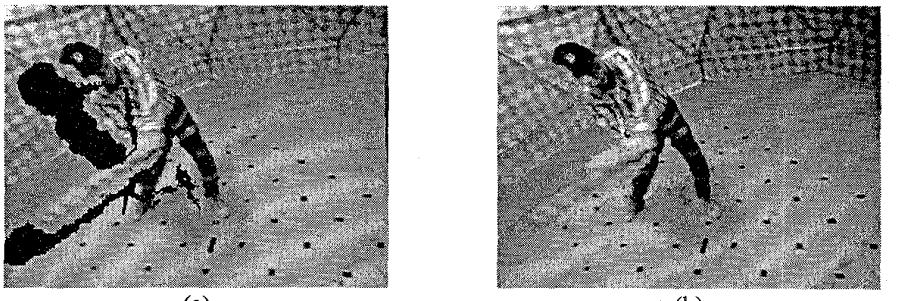
Converting a depth map to a VSM creates a large number of small triangles. An NxN depth map will translate to $O(N^2)$ triangles. The model can be decimated easily to take advantage of the planar regions. We use a method that collapses the edges subject to a number of error conditions, such as the error or deviation from the original model at the interior points or the error at the boundary points [8]. Errors along the occlusion boundary points, i.e. points where the foreground and the background are adjacent, are more noticeable to humans than interior errors. We therefore tune the decimation procedure to keep the boundary errors low. One advantage of a visibility-based model like the VSM is the easy identification of occlusion boundaries in it. It suffices to attach high priority to those boundaries as a VSM will be used only to synthesize views close to its origin, maintaining the occlusion characteristics. Our typical VSMs decimate from about 100,000 triangles to about 4000 triangles without appreciable degradation in the synthesized view. Global models of the scene do not have nicely defined occlusion boundaries and do not lend themselves to this degree of decimation, as will be discussed in Section 6.

5 View Synthesis Using VSMs

Each VSM provides a (partial) local model of the scene; their collection is designed to describe all surfaces in the scene. We now describe how new views of the scene can be synthesized from a viewer-controlled camera using the collection of VSMs. We first discuss view synthesis using a single VSM and then describe how we use other VSMs to improve the quality of the synthesized image.

5.1 Using One VSM

The Visible Surface Model is a standard textured polygon model and can be rendered from any user-given viewpoint using standard graphics techniques. Since the VSM does not contain any triangles from regions that are not visible, the rendered image will have "holes" if its viewpoint is different from the VSM's origin as seen in Figure 4(a). The holes encode the lack of information in the model. One or more of the other VSMs of the scene will have the description of the occluded region and can be used to fill the holes.



(a) (b)

Figure 4: (a) Holes appear using one VSM for rendering. (b) Merged view using two other VSMs to fill holes.

5.2 Merging Multiple VSMs

Holes in the views synthesized using a single VSM correspond physically to the regions occluded from the corresponding camera. These regions are visible in other VSMs, which can be used to fill the holes. Our rendering strategy uses two neighboring VSMs for this purpose. Our complete approach, then, uses three VSMs to construct each synthetic image: the *reference* VSM, i.e., the one "closest" to the desired synthetic viewpoint, and the two *supporting* VSMs for hole filling. Selection of these VSMs is performed dynamically as the user moves through the virtualized space. The actual selection of reference and supporting VSMs is discussed in detail in Sections 5.2.1 and 5.2.2, respectively.

The three selected VSMs must be merged to create a single complete image. The merging can be done in the model itself or in the rendered view. We only need to construct a single specific viewpoint from the combination of VSMs. The merging can then be accomplished with their rendered views rather than with the VSMs themselves. This is a purely 2D problem and hence is faster and simpler than 3D merging. We first synthesize the desired view using the reference VSM. We then render the hole triangles (see Section 4) into a separate buffer to identify hole pixels. Lastly, the supporting VSMs are rendered with the rendering limited to the hole pixels. If any holes remain after this merging, then they are filled by interpolating pixel values from nearby filled pixels. For example, Figure 4(b) shows the results of filling the holes of Figure 4(a) using two supporting views. Notice that the background and the right shoulder and body of the person have been filled properly. The hole pixels need to be identified explicitly as described above -- and not merely from the pixels untouched by the reference VSM -- for the shoulder to be filled correctly.

5.2.1 Selecting the Reference VSM

The VSMs used to synthesize a virtual image should be as "close" as possible to the desired viewpoint. Finding a good general definition of closeness is a complex problem because of the possibility of occlusion. Intuitively, we would expect the usefulness of a VSM to increase as the virtual camera moves closer (in physical space) to it. Since the VSM has a limited field of view, though, this intuitive metric is insufficient. Using direction of gaze alone also has clear problems.

We use a closeness metric based on the viewing angle of the virtual camera and the origins of the VSMs. 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 origin of the VSM, as shown in Figure 5. 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, the assumption about VSMs is true by design, which tends to focus the user's attention on this same region. We use the closest VSM as the reference VSM, which has the most direct view of the scene as the virtual camera.

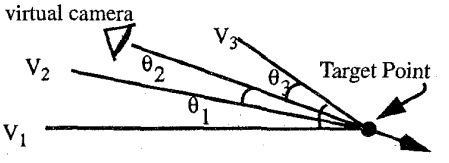


Figure 5: Selecting the reference VSM.

5.2.2 Selecting Supporting VSMs

The supporting angles are used to compensate for the occlusion in the reference description. The general problem of covering all occlusions with a few VSMs relates to the convex covering problem and has no easy solutions. That is, for any given configuration of VSMs, it is possible to construct a scene in which certain surfaces are occluded from all the VSMs. However, it is usually possible to obtain adequate coverage of occluded regions by concentrating only on the neighbors of the reference VSM, especially when the VSMs are densely distributed. Consider the triangles formed by the locations of the reference VSM and all adjacent pairs of its neighbors, as in Figure 6. If the angle has n neighbors, there are n such triangles. We determine which of these triangles is pierced by the line of sight of the virtual camera. The non-reference VSMs that form this triangle are selected as the supporting VSMs. Using this approach, the reference and supporting views "surround" the desired viewpoint, filling holes created by the virtual camera moving in any direction from the reference VSM.

6 Complete Surface Models

The multiple VSM representation of the virtualized scene supports full 3D interaction but only with special tools. An alternative approach is to create an object-based (i.e., view independent) description of each scene by combining the individual

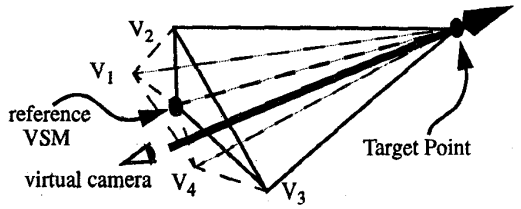


Figure 6: Selection of supporting VSMs. The triangle pierced by the line of sight of the virtual camera determines the supporting VSMs.

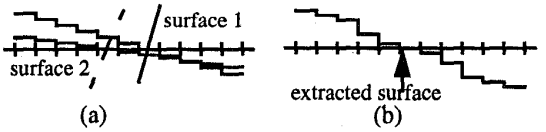


Figure 7: 1D example of the Curless-Levoy algorithm. (a) Surface 1 has higher weight and therefore contributes more to the final result. (b) The final surface is extracted at the zero crossings of the values accumulated in the voxels.

views into a single 3D model. This type of model is fundamental to most existing 3D file formats and can therefore leverage off existing interaction tools. By computing global shape and appearance, we build a complete description of the scene independent of viewpoint, so we call the model the *Complete Surface Model.*

Merging multiple range estimates into one object model has been studied recently, but mainly working with high quality range images [3][6][7][24]. The depth maps generated by stereo under normal scene conditions (i.e., no structured lighting or special textures) suffer from problems inherent in window-based correlation. These problems manifest as imprecisely localized surfaces in 3D space and as hallucinated surfaces that in fact do not exist. Because fusion integrates multiple structure estimates of the same scene, it can improve the precision of surface localization. For example, Curless and Levoy [3] proposed a volumetric integration method which addresses this problem. The method has similarities to two other algorithms [6][7], but is unique in that it sequentially updates the model based on each range image, offering an efficient implementation. In order to overcome the hallucinated surfaces created by stereo, we adapted this algorithm to construct a global 3D surface model of the scene from the depth maps.

6.1 Fusion of Multiple Depth Maps

The Curless and Levoy algorithm uses an object-centered volume decomposed into voxels, or volume elements. Each voxel accumulates the signed distance to the surfaces in the range images. To add a range image into the volume, the image is first converted into a set of triangular surfaces by tessellating the image, connecting each pixel to its neighbors. If neighboring pixels have a large difference in depth, no tessellation occurs between the pixels. 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 sensor providing the current depth map. 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 7(a)). The process is repeated for each voxel. After accumulating across all range images, the voxels implicitly represent the surface by the zero crossings of their values. By finding these zero crossings, the surface is extracted (see Figure 7(b)). Extraction of an implicit surface, or isosurface, is well studied and has standard solutions such as the marching cubes algorithm [1][16], which generates 3D triangle meshes representing the implicit surfaces. One major aspect of this approach is the framing of the surface recovery problem as extraction of an isosurface rather than detection of peaks or ridges in the 3D volume, which has robust solutions [3].

In order to overcome the false surfaces generated by stereo, we made one noteworthy change to the Curless and Levoy algorithm. They limit the extent of each tessellated surface to the voxels near it, while we allow the algorithm to adjust all voxels in front of the surface as viewed from the sensor generating this surface. For voxels far in front of the surface, we clamp the weighted, signed distance contribution of each viewpoint so that this single view does not overwhelm all others in the fusion process. This modification gives significant improvement in the ability of the algorithm to reject the numerous outliers in our range images, while not significantly degrading the recovered shape. One example of the modeling process is shown in Figure 11 (a). For more information on our model construction process, see [21].

6.2 CSM Texture Modeling

The volumetric integration process creates a 3D triangle mesh representing the surface geometry in the scene. To complete the CSM, a texture map can be constructed by projecting each intensity (or 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. The example in Figure 11 (b) uses the simple approach.

6.3 Decimating Complete Surface Models

One drawback of the volumetric merging 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. 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. To reduce the number of triangles, we apply the same edge-collapse decimation algorithm [8] as we use to decimate VSM meshes, as described in Section 4.4. Unlike the VSMs, however, the global model represents the scene independent of any viewpoint and therefore must give equal importance to all geometric information. The amount of reduction in mesh size is therefore smaller than in the VSM case, although the gains are still large. Typically a 1,000,000 triangle model is reduced to one with 100,000 triangles. Note that the example model shown in Figure 11 (a) is actually a decimated model with fewer than 10,000 triangles.

Figure 8: Some views from a spiralling path in a static scene.

6.4 Rendering using a CSM

The CSM is a global polygonal model of the scene with texture. It can easily be converted to a standard model format such as the Open Inventor format. This flexibility is a major advantage of the CSM as standard VR tools can be used to interact with the model or modify it. There is no additional computational overhead to render the view from a specific viewpoint.

6.5 CSM back to VSMs

One use of the global CSM is to improve the quality of the range images. Rendering the CSM back into each of the cameras will result in a new set of depth maps that are globally consistent. The geometric extent of the corresponding VSMs will also be better, without the shortcomings of the VSMs generated by stereo near the borders. In addition, VSMs tend to be lighter models for a given viewing direction as the occluded triangles are eliminated from it.

7 Experimental Results

We now present a few examples using real data, including both static and dynamic scenes. While the earlier sections only discussed static scene analysis, the extension to dynamic scenes is straightforward. We apply the same techniques to each synchronized video frame of the time-varying event. For other examples of virtualized events, visit our Web page at **http://www.cs.cmu.edu/~virtualized-reality/.**

7.1 Static Scene Analysis

We begin by analyzing a static scene virtualized into 51 VSMs from all directions. The scene is a person sitting on a table inside 3D Dome. Figure 8 shows a few views of this scene from a virtual camera moving through the dome. Qualitatively, the images are realistic views that well approximate the motion of the virtual camera. All the holes created by rendering the reference VSM alone have been filled with the information from the supporting VSMs, creating a seamless video sequence.

Two errors are still detectable in the output, however. The first is a "ghosting" effect, something like a shadow around the person. This occurs in the holes present in the rendering of the reference VSM and is caused by inconsistencies in the apparent intensities of the cameras involved. If a VSM with a "bright" image is used to fill holes in a VSM with a "dark" image or vice versa, a ghost results. In the examples shown in this paper, we have applied a normalization technique to minimize this effect; without normalization, the effect is significantly worse. The normalization is simple: adjust the mean intensity and the intensity variance of each image to match a reference value. Even though this normalization does not take into account point-to-point correspondences and only uses a global correction, we have found that it works quite well in most situations.

The second error, appearing only when using the VSM representation, results from the inaccuracies in reproducing motion parallax. The discrepancies in the geometric structure between two adjacent VSMs will result in "jerkiness" of the rendered view when the reference shifts from one to the other. This effect is not usually discernible in side-by-side printed images, but is easily detectable when viewing a sequence containing virtual camera motion. The solution to this problem remains an open research issue.

7.2 Dynamic Scene Analysis

We now move to a dynamic scene containing a baseball player swinging a baseball bat. We processed 11 frames of video, creating a scene representation of 11 static virtualized models each with 51 VSMs. We then synthesized camera motion starting high above the batter and dropping down into the scene to a point just below the trajectory of the baseball bat. The results are shown in Figure 9. The "holes" remaining in the scene result from the limited field of view and the position of each camera (both real and virtual ones), which combine to allow the virtual camera to see more than the VSMs used for view synthesis. This behavior suggests that using more VSMs at specific times may be useful for filling in the holes resulting from the limited field of view of the real cameras.

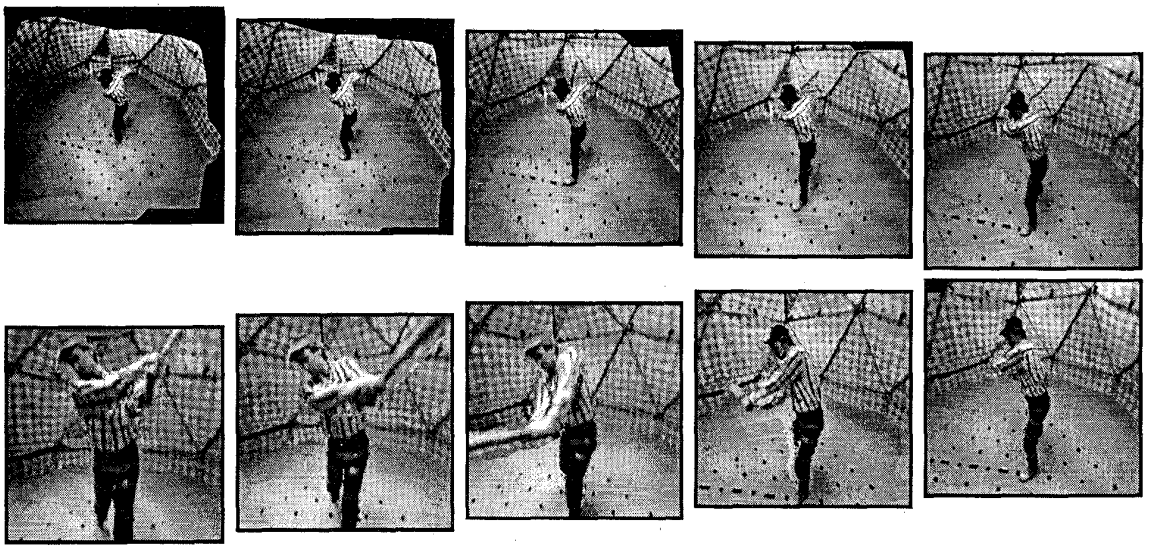


Figure 9: Some views of a dynamic scene. The virtual camera drops down into the scene from high above the batter to a point near the path of the baseball bat.

7.3 Color Texture

We can provide color texture for synthesis using a few color cameras, placed carefully and calibrated in the same way as the other cameras. We achieve this by projecting the color texture from the color cameras into the scene, replacing the monochrome texture map. Alternatively, view-dependent texture mapping [4] could be used with only the color images as texture. The portion of the scene visible to any one of the color cameras gets color texture as a result. Gaps in the coverage of the color cameras will have monochrome texture. One example is shown in Figure 11 (c), in which 4 color cameras have added color to a CSM derived from monochrome images only.

7.4 Combining virtual and virtualized environments

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 discussed in Section 7.2, 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 [11]. 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 and Future Work

We presented a method to create virtual worlds out of real dynamic events using multiple stereo views of it in this paper. We presented two representations of the virtualized world: as multiple Visible Surface Models and as a single Complete Surface Model. True immersive interaction with the virtualized world is possible using either representation. The CSM model is a view independent description of the scene that is compatible with traditional VR tools. The multi VSM model, on the other hand, is more compact for a specific synthetic view. It also is able to preserve the occluding boundaries better as the effects of errors in the recovered geometry remain localized. Our work leads to an immersive visual medium called Virtualized Reality which delays the selection of the viewing angle until view time. It would thus be possible for medical students to view a virtualized surgery from viewpoints of their choice and for fans to view a basketball game from the middle of the court.

The combined effect of imperfect calibration, correspondence finding, and mesh generation on the quality of the final output is an open issue. There is a clear need to improve the rendering to be more resilient to these inaccuracies, which can affect the filling of holes and the motion parallax of a moving virtual camera – a dynamic property that has received almost no attention from the view synthesis community. Developing mechanisms to modify/edit the virtualized worlds, so that they can be transported more easily into other situations, is another promising area of work. For instance, the backdrop for a computer generated animation can be the virtualized model of, say, the Taj Mahal. We intend to pursue these issues in the future.

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

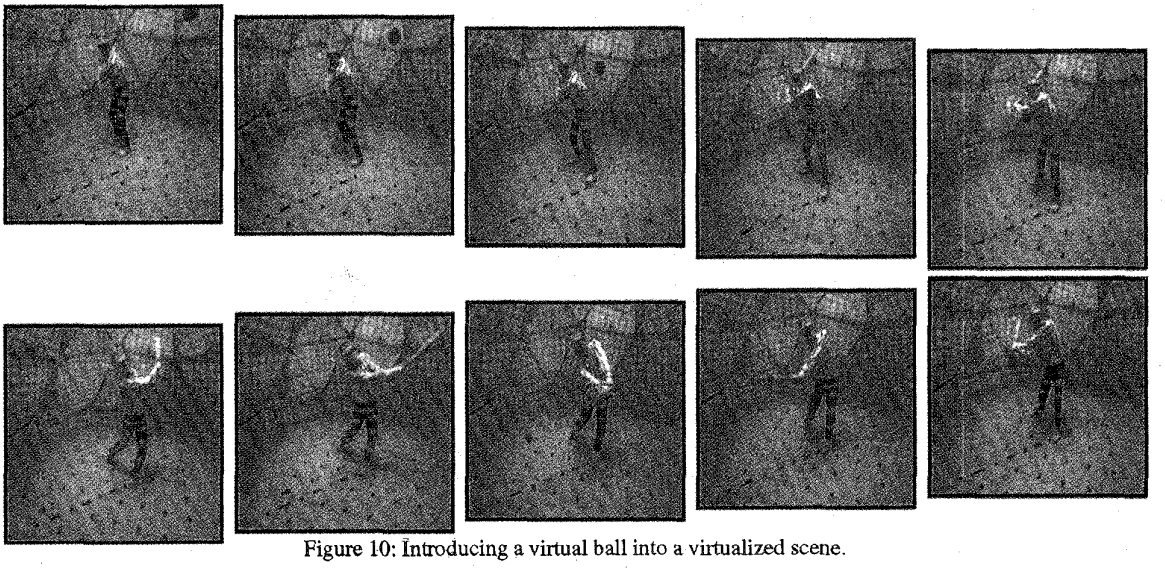


Figure 10: Introducing a virtual ball into a virtualized scene.

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

9 T. Kanade, P. J. Narayanan, and P. W. Rander. Virtualized Reality: Concept and Early Results. *IEEE Workshop on the Representation of Visual Scenes*, Boston, June, 1995.

10 T. Kanade, P. W. Rander, and P. J. Narayanan. Virtualized Reality: Constructing Virtual Worlds from Real Scenes. *IEEE MultiMedia*, **4(1)**, May1997.

11 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.

12 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.

13 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.

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

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

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

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

18 S Moezzi, Li. Tai, and P. Gerard. Virtual View Generation for 3D Digital Video. *IEEE MultiMedia*, **4(1)**, May1997.

(a) (b) (c)

Figure 11: Adding color texture to a model derived from monochrome images only. (a) The untextured model (after decimation). (b) The model with monochrome texture. (c) The model with color texture applied.

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

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

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

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

23 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.

24 G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. *SIGGRAPH'94*, July 1994.

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

26 R Willson. Tsai Camera Calibration Software. http://www.cs.cmu.edu/~rgw/TsaiCode.html.