Virtualized Reality: Constructing Virtual Worlds from Real Scenes

Takeo Kanade and Peter Rander
Carnegie Mellon University

P.J. Narayanan
Centre for Artificial Intelligence and Robotics

A new visual medium, Virtualized Reality, immerses viewers in a virtual reconstruction of real-world events. The Virtualized Reality world model consists of real images and depth information computed from these images. Stereoscopic reconstructions provide a sense of complete immersion, and users can select their own viewpoints at view time, independent of the actual camera positions used to capture the event.

The different visual media we have today share two shortcomings: director-decided viewpoints and two-dimensional views. *Virtualized Reality* is an immersive visual medium that lets the viewer select a (possibly time-varying) viewing angle at view time, freely moving throughout the *virtualized* event in ways even a mobile camera present at the event site could not. A viewer equipped with a stereo viewing system can even be immersed in a stereoscopic reconstruction of the live or recorded event. Viewers can thus watch a virtualized basketball game from a seat at the center of the court or from a viewpoint moving with the ball.

Like Virtualized Reality, virtual reality (VR) also immerses the viewer in a virtual environment. The two differ, however, in how the virtual world models are constructed. VR environments are typically created using simplistic CAD models and lack fine detail. Virtualized Reality, in contrast, automatically constructs the virtual model from images of the real world, preserving the visible detail of the real-world images. It is thus possible to create realistic virtualized environments of such complex environments as a surgery or a basketball game. Such environments lie beyond the scope of current VR systems.

The Virtualized Reality process involves three phases: transcribing the visual event, recovering 3D structure, and generating synthetic viewpoints. We transcribe a visual event using many cameras surrounding the scene. We compute the 3D structure of the event for some of these cameras using a multi-camera stereo method.[1] The views for which we compute structure are called *transcription angles*. The stereo process generates a depth map, which encodes the scene depth of each image point from the corresponding transcription angle. We call this combination of an image and an aligned depth map a *scene description*.

The virtualized model of one time instant consists of a collection of scene descriptions at that instant, each from a different transcription angle. A time-varying event is represented as a collection of successive time instants. Scene descriptions translate easily into computer graphics models with depth maps providing scene geometry and images providing scene texture. Novel views of the virtualized event can be synthesized easily from these models using graphics hardware.

Possible applications of Virtualized Reality include realistic training in a virtualized work space, true telepresence, and imaginative uses in entertainment. For example, surgical training could be enhanced by virtualizing a rare heart surgery. Students could then observe the operation in 3D from anywhere in the virtualized operating room, potentially standing in the same position as the actual surgeon. To achieve telepresence, reconstruction and display of the remote scene occurs simultaneously with the actual event. For entertainment, Virtualized Reality could make possible a whole new medium that would allow viewers to watch a ballet performance seated on the edge of the stage or a basketball game while standing on the court or running with a particular player.

View synthesis

Visual reconstruction from arbitrary viewpoints is an important component of Virtualized Reality. View synthesis, or view transfer, considers visual reconstruction as an image-to-image mapping designed to generate novel views of a scene given two or more real images of it. Additional knowledge about the scene or the imaging process may also contribute to synthesis.

1070-986X/97/$10.00 © 1997 IEEE

One class of view synthesis techniques requires image flow or pixel correspondence, that is, knowledge about where points in one image move to in another image. Using this information, Tseng and Anastassiou[2] described a codec similar to MPEG-2 that can efficiently transmit a set of discrete viewpoints but cannot construct novel views. View interpolation is an image-based rendering technique that 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. Both Chen and Williams[3] and Werner et al.[4] used linear interpolation as an approximation of perspective viewing because of the simplicity and speed of implementation. Seitz and Dyer[5] demonstrated that this yields physically valid images only if the source images are rectified. Current view interpolation algorithms, however, restrict the synthetic view to a linear space defined by the reference viewpoints and cannot synthesize views from arbitrary viewpoints.

Laveau and Faugeras[6] developed a method that allows arbitrary viewpoints if the viewpoint is specified in terms of epipolar geometry with respect to the original viewpoints. McMillan and Bishop[7] and Kang and Szeliski[8] constructed cylindrical panoramic images from planar images before synthesizing new views from the implicit 3D structure. With no existing real-time cylindrical imaging systems, this approach cannot currently be extended to time-varying imagery. Multiple Perspective Interactive (MPI) Video[9] attempts to give viewers control of what they see using a different approach. This method computes 3D environments for view generation by combining a priori environment models with dynamic motion models recovered by intersecting the viewing frustums of the pixels that indicate motion.

A second class of view synthesis techniques eliminates the need for pixel correspondences by densely sampling the viewing space, possibly interpolating missing views. Katayama et al.[10] demonstrated that it is possible to generate images for arbitrary viewing positions from a dense set of images on a plane. Satoh et al.[11] used this property to develop a prototype 3D image display system with motion parallax. Two similar methods have been proposed recently: lightfield rendering by Levoy and Hanrahan[12] and the lumigraph by Gortler et al.[13] Both methods convert the given views into a four-dimensional field representing all light rays passing through a 3D surface. New view generation is posed as computing the correct 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 of the convex hull of the scene.

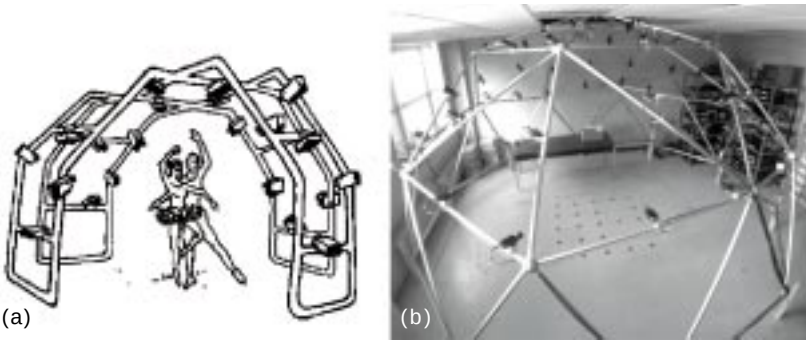


Figure 1. The Virtualized Reality studio: (a) conceptual; (b) 3D Dome.

We introduced Virtualized Reality and presented a few preliminary results in earlier work.[14] This article presents first results from a virtualizing facility that provides all-around coverage, presently consisting of 51 cameras mounted on a geodesic dome 5 meters in diameter.

3D Dome: The virtualizing studio

A *virtualizing studio* is a facility for making virtualized models. Our studio, 3D Dome, transcribes events from many angles to capture all-around scene structure. The studio can provide accurate scene structure for each video field—1/60th of a second in NTSC—of a time-varying event. For structure recovery, the studio captures every frame of each video stream, maintaining synchronization among the images taken at the same time instant from different cameras. Synchronization is crucial to correctly virtualize time-varying events because stereo (the structure extraction process) assumes that the images from the different viewpoints correspond to a static scene. This section provides a brief overview of the actual system. A more detailed discussion of the setup and the synchronous, multi-camera image acquisition system can be found elsewhere.[15]

The studio setup

Figure 1a shows the conceptual virtualizing studio, and Figure 1b shows 3D Dome, the studio we built using a geodesic dome 5 meters in diameter. Fifty-one cameras are placed at nodes and the centers of the bars of the dome, providing views from angles surrounding the scene. Currently, we use monochrome cameras, each equipped with a 3.6-mm lens for a wide view (about a 90-degree

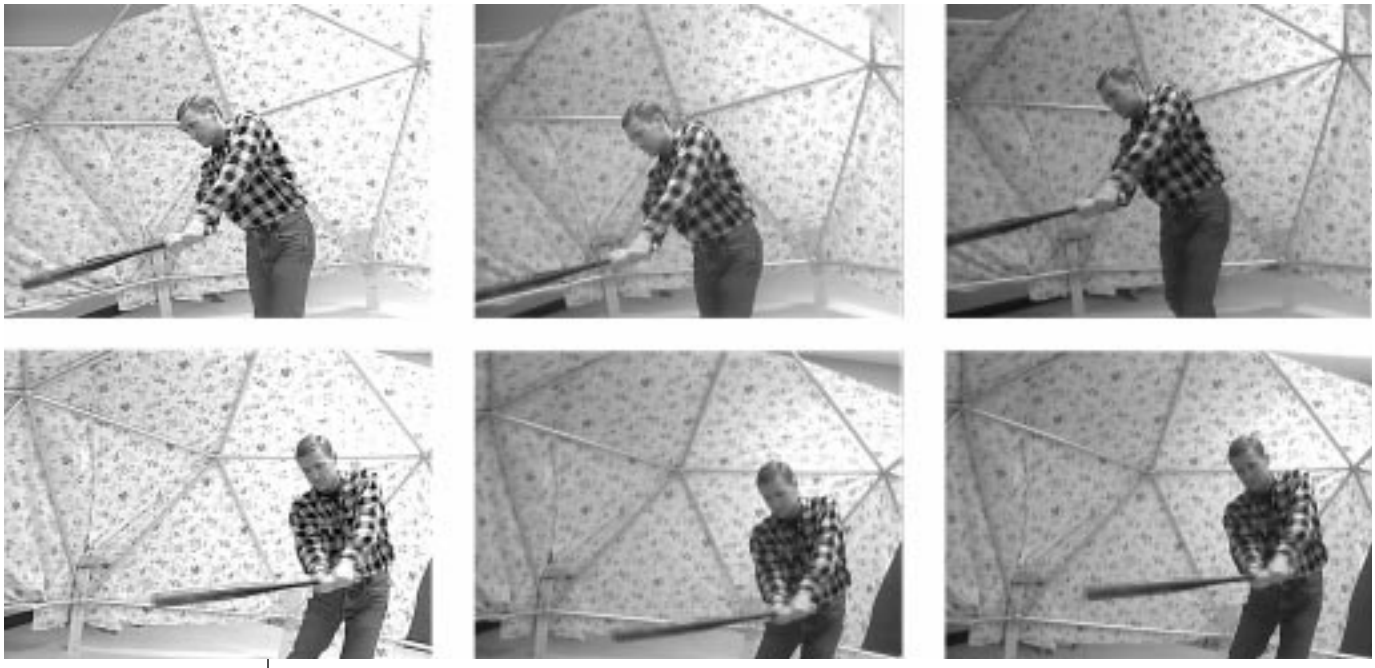


Figure 2. Six captured images to be used to compute scene descriptions.

horizontal field of view) of the dome. Color cameras can provide more realistic virtualization, but at higher costs.

Synchronous multi-camera image acquisition

Synchronous digital acquisition of many video streams is a difficult task because even a single monochrome camera providing 30 images per second, each image captured digitally to contain 512 by 512 8-bit pixels, represents a bandwidth of 7.5 Mbytes per second. The sustained bandwidth to store even a few video streams onto a secondary storage device exceeds the capabilities of typical image capture and digital storage systems, even with the best lossless compression technology available today. For example, our current system—a Sun Sparc 20 workstation with a standard 1-Gbyte hard drive and with a K^2T V300 digitizer—can capture and store only about 750 Kbytes per second. Specialized hardware can improve the throughput, but at a cost level unsuitable for replication.

To overcome these limitations, we perform digital acquisition in two steps: real-time recording and off-line digitization. The recording system uses standard CCD cameras and consumer-grade VCRs. The cameras are electronically synchronized to a common sync signal. Every field of each camera's video output is time stamped with a common Vertical Interval Time Code (VITC) before being recorded onto video tape. The hardware for this part of the system runs in real time, allowing continuous capture of long motion sequences. The tapes are digitized individually off line.

A computer program interprets the VITC time code of each field in real time as the tape plays. Given a list of frames to capture, the digitizing program captures as many from the list as it can. When the tape goes past the last required frame, the computer rewinds it to the starting frame, replays the tape, and repeats the process. The real-time VITC interpretation helps to identify frames missed in previous passes, guaranteeing capture of unique frames on each repetition.

In our experience, four passes suffice to capture every frame, as the V300 can transfer every fourth frame of a video stream to the computer memory. The VITC time code in each field of the captured image also serves to synchronize fields and frames from different cameras. Figure 2 shows a still frame as seen by six cameras of the virtualizing studio digitized using the setup described above. A separate report[15] gives more details on the synchronous multi-camera recording and digitizing setup.

Computing scene descriptions

Scene descriptions are analogous to the 2.5D sketch of the Marr paradigm, which encodes the geometric and photometric structure of all surfaces visible from a specific location. We use a stereo technique to compute the geometric structure, so this location coincides with the location of the reference camera used in stereo computa-

tion. Since the images themselves have limited field of view, the geometric extent of the scene description is limited to the viewing frustum of the reference camera. The transcription angle, then, must include orientation as well as position. Clearly, the distribution of the transcription angles is important to the quality of the virtualized reconstruction of the event.

Distribution of transcription angles

Our rendering strategy uses a scene description—more precisely, a textured triangle mesh model derived from this description—as the fundamental unit of structure. Each transcription angle specifies the location and orientation of one scene description. Using the model from one scene description, we can synthesize the appearance of the surfaces from arbitrary viewing positions. As the virtual camera moves away from the corresponding transcription angle, the quality of the reconstruction decreases because of occlusion and errors in the recovered structure. By minimizing the distance between any desired viewpoint and a real camera, we maximize the quality of the synthesized images.

This behavior implies that transcription angle density should be uniform and high to maximize the output quality. In fact, a high enough density eliminates the need for correspondences, allowing direct interpolation of the images using algorithms like the lumigraph[13] or light field rendering.[12] The problem with such techniques is that the density of camera views can easily become extremely high. For example, Levoy and Hanrahan use as many as 8,000 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. Either way, a trade-off must be made between the number of transcription angles and the quality of reproduction.

The number of cameras sets an upper limit on the number of transcription angles because each angle requires a real view of the scene. If all cameras are used to provide transcription angles, then the camera distribution should satisfy the same constraints as the transcription angle distribution. Another factor in determining the camera distribution is that stereo usually makes fewer mistakes as the cameras are moved closer together. Because we expected to use all cameras to obtain transcription angles, we selected the number of cameras (51) to meet these criteria. The initial experiments described in this article use all 51 of the transcription angles in the view synthesis process; we intend to investigate the economy of representation in the future.

Camera clusters

We compute stereo with each of the 51 cameras as reference and with several other cameras providing the baselines required for multi-baseline stereo (MBS). Using many cameras improves the extent and accuracy of stereo as long as features in the reference camera can be successfully matched to the features in the other cameras. The matching gets increasingly difficult the more the viewing angle differs. Arbitrarily adding more cameras provides little improvement in stereo accuracy while increasing the computational cost. We therefore use only the immediate neighbors—three to six depending on the specific camera in our setup—of the reference camera for stereo computations. These neighbors radiate out from the reference camera in all available directions. We modified the original formulation of MBS to handle verged cameras. We are thus free to place cameras in any orientation.

Camera calibration

Stereo programs require fully calibrated cameras to extract structure in metric units. We calibrate for the intrinsic and extrinsic camera parameters using Tsai's[16] approach modified and implemented by Reg Willson. We do this in two steps because our arrangement of cameras does not adapt very well to a simultaneous calibration procedure involving all cameras and all calibration parameters. 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, image center (two parameters), aspect ratio, and one radial lens distortion parameter. We then place the cameras in their positions on the dome and calibrate the six extrinsic parameters—rotation and translation relative to a world coordinate frame—using a set of dots on the floor visible to all cameras. The world coordinate frame is rooted on the floor in our case.

Multi-baseline stereo

We adapted the MBS algorithm developed by Okutomi and Kanade[1] for a general camera configuration (that is, nonparallel cameras) by incorporating the Tsai camera model. Two factors

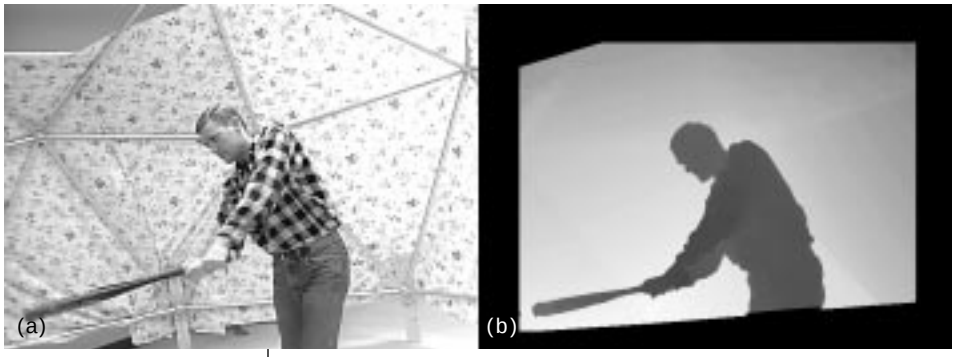


Figure 3. Scene description of one image. (a) Intensity image. (b) Aligned depth map.

primarily motivate the choice of MBS. 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.

To understand the MBS algorithm, we begin by considering two-camera stereo, with two parallel cameras with the same focal length F. The perpendicular distance z to a scene point is related to the disparity d (the difference in the image locations of the scene point) by

$$d = BF\frac{1}{z} \quad \textbf{(1)}$$

where B is the baseline, or distance between the two camera centers. The precision of the estimated distance increases as the baseline between the cameras increases. Increasing the baseline, however, also increases the likelihood of matching points incorrectly. The correctness of correspondence depends also on the image; the aperture problem makes it difficult to localize features parallel to the camera baseline. MBS attempts to improve matching by computing correspondences between multiple pairs of images, each with a different baseline. Since disparities are meaningful only for a pair of images, we reformulate Equation 1 to relate correspondences from multiple image pairs as

$$\frac{d}{BF} = \frac{1}{z} = \varsigma \quad \textbf{(2)}$$

The search for correspondences can now be performed with respect to the inverse depth ς, which has the same meaning for all image pairs with the same reference camera, independent of disparities and baselines. The resulting correspondence search combines the correct correspondence of narrow baselines with the precision of wider baselines. The effect of the image features' orientation with respect to the baselines can be mitigated by using baselines oriented in multiple directions.

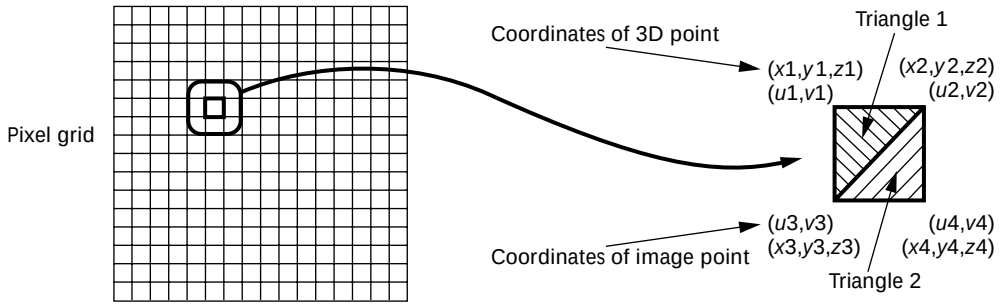
A popular method to compute correspondences between a pair of images compares a small window of pixels from one image to corresponding windows in the other image. The correct position of the window in the second image is constrained by the camera geometry to lie along the epipolar line of the position in the first image. The matching process for a pair of images involves shifting the window along this line as a function of ς, computing the match error—using normalized correlation or sum of squared differences—over the window at each position. MBS combines the error estimates for each camera pair by computing their sum before finding the minimum error. The inverse depth $\hat{\varsigma}$ for the point is the ς at this minimum.

Window-based correspondence searches do not localize matches well in regions of low image texture and along depth discontinuities, so we have incorporated an interactive depth map editor into our process of structure extraction. This step can roughly be compared to movie editing but is not as critical to the virtualizing process. We are currently exploring improvements to the stereo algorithm to make this step unnecessary. Figure 3b shows the edited depth map computed by MBS, aligned with the reference image of Figure 3a, from the images shown in Figure 2. The farther points in the depth map appear more white.

View generation

We *virtualize* an event using a number of scene descriptions. The other part of Virtualized Reality deals with synthesizing the scene from a viewer-specified angle using one or more scene descriptions. We first translate the scene description into a textured triangle mesh. We then synthesize new views by rendering this mesh to create novel views of the scene. This process takes advantage of the capabilities of graphics workstations, which have specialized hardware to render and texture map polygon meshes. We use Silicon Graphics workstations such as the Onyx/RE2 that can render about 1 million texture mapped triangles per second, though any rendering platform would suffice.

We first describe how new views are synthesized from a single scene description and how the view generated from a single scene description will be lower in quality as the virtual viewpoint moves



Triangle 1:
Vertex 1: ($x1$,$y1$,$z1$) Texture coord.: (u_1/m,v_1/n)
Vertex 2: ($x2$,$y2$,$z2$) Texture coord.: (u_2/m,v_2/n)
Vertex 3: ($x3$,$y3$,$z3$) Texture coord.: (u_3/m,v_3/n)

Triangle 2:
Vertex 1: ($x2$,$y2$,$z2$) Texture coord.: (u_2/m,v_2/n)
Vertex 2: ($x3$,$y3$,$z3$) Texture coord.: (u_3/m,v_3/n)
Vertex 3: ($x4$,$y4$,$z4$) Texture coord.: (u_4/m,v_4/n)

Figure 4. Triangle mesh and texture coordinate definition.

away from the transcription angle corresponding to that scene description. We then describe how other scene descriptions can be used to improve the quality of the synthesized image. Finally, we present a method of traversing the virtualized space in all directions using all the transcription angles.

Synthesizing views from one scene description

There are two aspects to view generation using one scene description: object definition and occlusion handling. Object definition is the process of converting a scene description into a textured triangle mesh model. A simple method of object definition is to construct one large surface that passes through all of the points in a depth map. Since real scenes frequently contain independent objects which may occlude each other, we introduce a method to handle these occlusions.

Object definition. There are two steps in defining the object from a scene description: converting depth into (x, y, z) coordinates and constructing a triangle mesh from these coordinates to describe the surfaces visible from the transcription angle. Converting depth into 3D coordinates is a simple transformation, since the depth values give the z-coordinates (in the camera-centered coordinate system) of the surface points at each pixel. To compute the (x, y) coordinates, we use the camera's intrinsic parameters to determine the viewing ray (the ray emanating from the camera center and passing through a given pixel) and intersect that ray with the plane at the given distance z read from the depth map. We then apply the camera's extrinsic parameters to convert the camera-centered (x, y, z) position into a position in the common-world coordinate frame.

We must now convert the 3D points into a surface representation. We treat the 3D points as vertices of a triangle mesh, which changes the task to selecting the 3D connectivity of the points. In general, this problem will have many solutions, but in our case we know more than just the 3D locations of the vertices: We also know how they project into the image plane. That is, we already have a local connectivity of 3D points because of their corresponding projections into the image plane. The task therefore reduces to selecting local connectivity in the image plane, a simpler problem than the general 3D one.

In addition, we can exploit the regular pixel grid to guide the connectivity process, because all the image projections of the 3D points lie exactly on the pixel grid. Using this information, we apply a simple pixel connectivity rule: Convert every 2×2 pixel area into two triangles by adding a diagonal, as shown in Figure 4. The only remaining question with this approach is which diagonal to select, a decision that affects the local surface properties of the mesh. Because our mesh is sampled extremely finely, however, the visible impact of the decision is small and we therefore always select the same diagonal. The texture to apply to each triangle is easily obtained by using the pixel coordinates of the triangle vertices.

This simple strategy for object definition has several important features. First, the 3D coordinates are quickly and easily extracted from the depth map using the calibration parameters. Second, the resulting "model" is little more than surface interpolation between 3D data points and therefore faithfully reproduces the exact structure inherent in the depth map. This fact implies that although we use a 3D model, we are essentially using the specialized rendering hardware as an efficient way to perform geometrically correct view synthesis, fundamentally eliminating the

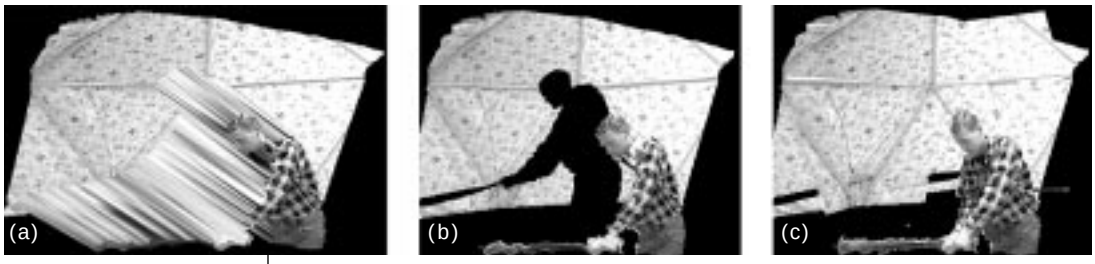


Figure 5. (a) Phantom surfaces at depth discontinuity. (b) Holes appear when phantom surfaces are removed. (c) Merged view using another scene description to fill holes.

geometric distortions that may exist in purely correspondence-based view interpolation.[3-5]

The major drawback of this method is that it generates $O(N^2)$ triangles for a depth map of size $N \times N$. The resulting object is simple and regular, though, making it suitable for efficient rendering on graphics workstations. In addition, the resulting mesh can be simplified easily by applying standard mesh decimation algorithms. According to our experiments, a reduction by a factor of 10 or 20 can be achieved without significant loss in quality.

Occlusion handling. A scene description encodes the 3D structure of the scene visible from the transcription angle. The object definition method just discussed implicitly assumes that only one continuous surface is visible from the transcription angle. If this assumption is violated, then the model will contain additional "phantom" surfaces that bridge depth discontinuities between real surfaces, as seen in Figure 5a. Even if the model is perfect everywhere else, the phantom surfaces cause the rendered view to become visually unrealistic as the viewing angle moves farther from the transcription angle.

We avoid phantom surfaces by eliminating triangles with large depth discontinuities along any edge. This results in "holes," as seen in Figure 5b. The holes correspond to areas not visible from the transcription angle. This behavior is desirable, since the scene description encodes information only about the visible 3D scene structure.

Merging multiple scene descriptions

Holes in the views synthesized using a single scene description correspond physically to the regions occluded from the transcription angle. In many cases, these regions are visible in the scene descriptions from other transcription angles. These scene descriptions can be used to fill the holes and therefore improve the visual realism of the synthetic views. Our rendering strategy uses two additional scene descriptions for this purpose. Our complete approach, then, uses three scene descriptions to construct each synthetic image. The *reference* scene description corresponds to the transcription angle "closest" to the desired synthetic viewpoint. The two scene descriptions used for hole filling are called *supporting* scene descriptions.

We can develop a number of methods to "merge" multiple scene descriptions. For example, we could merge these different 3D views into a single 3D model. Because our goal is only to construct a synthetic image, we developed a simpler merging strategy that works only with renderings of the objects defined by the scene descriptions. This strategy requires only 2D merging, making it faster and simpler than 3D merging. We first synthesize the desired view using the reference scene description. To identify "hole" pixels, we render separately the triangles that were eliminated because of large depth variation. The synthetic view is then re-rendered using the supporting scene descriptions, with the rendering limited to the hole pixels.

For example, Figure 5c shows the results of filling the holes of Figure 5b using one supporting view. Notice that the background and the right shoulder of the person have been filled properly. The holes remaining in the image correspond to the portion of the scene occluded from both the reference and supporting transcription angles.

It is insightful to compare this algorithm to a simpler one that identifies hole pixels by finding pixels not touched after rendering the objects from the reference scene description. This strategy is appealing because of its simplicity and because the eliminated triangles do not need to be rendered.

However, under certain conditions this strategy avoids filling regions that are obviously incorrect. In Figure 5b, for example, the reference scene description did not see all of the person's right shoulder, so the synthesized view fills in this region with pixels from the background. Using this simpler merging strategy would not allow the supporting view to correct these pixels, resulting in the obvious error in the synthesized output.

Selecting reference transcription angle. Our view generation strategy exhibits a highly desirable property: The synthesized view is exact and

hole-free when the desired view coincides with a transcription angle, even with arbitrary errors in calibration and depth estimation. In addition, for a general scene, the quality of the view synthesized from a single scene description deteriorates as the viewing angle moves away from its transcription angle. This can happen even with ideal models because of foreshortening.

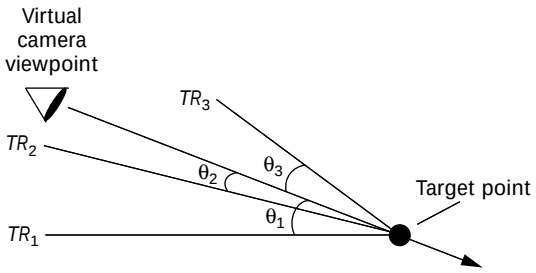


Figure 6. Selecting the reference transcription angle. θ_i is the angle between the virtual camera's line of sight and the line joining the target point with the position of the transcription angle TR_i. We select the transcription angle TR_i with the smallest angle θ_i as the reference transcription angle.

Consider, for example, a scene containing a single receding plane as viewed from a transcription angle. Because of foreshortening, the image of the scene has less texture resolution for far parts of the plane than for near parts. Even if the structure of this scene is recovered perfectly, the texture provided by a single scene description will be forced to expand as the virtual camera moves to view the plane frontally.

These properties suggest that the transcription angles used to synthesize a virtual image should be as "close" as possible to the desired viewpoint. Despite its seeming simplicity, finding a good definition of "close" is, in general, a highly complex problem because of the possibility of occlusion. Intuitively, we would expect the usefulness of a transcription angle to increase as the virtual camera moves closer (in physical space) to it.

Because the transcription angle has a limited field of view, however, this intuitive metric is insufficient. Using direction of gaze alone also has clear problems. Potentially serious problems due to occlusion arise even if we combine these two metrics. For example, even if the virtual camera is close to a real transcription angle and oriented in the same direction, the desired viewpoint can lie beyond the visible surfaces in the scene description—that is, the desired viewpoint can lie in the region occluded from the transcription angle. As a result, this transcription angle would be a poor choice for synthesizing this image.

The metric we use to evaluate the "closeness" of a virtual viewpoint to a transcription angle is based on several assumptions about the distribution (3D placement) and orientation (field of view) of the transcription angles and about the general regions of interest in a typical scene. The viewing direction of the virtual camera is specified in terms of 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. We measure closeness as the angle between this line of sight and the line connecting the target point to the 3D position of the transcription angle, as shown in Figure 6. This measure works well when both the virtual viewpoint and all the transcription angles point at the same general region of space. In our system, the assumption about transcription angle is true by design, which tends to focus the user's attention on this same region.

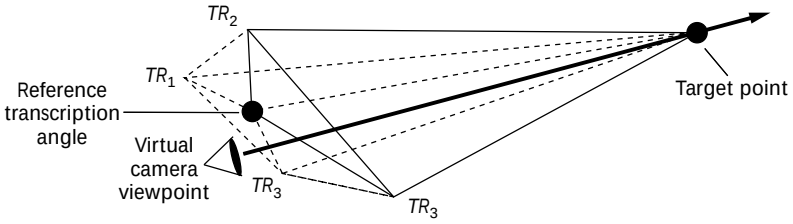


Figure 7. Selection of supporting transcription angles. Given a reference transcription angle, we determine the supporting angles by first enumerating the neighboring angles (TR_i) of the reference and then forming the triangles containing the reference transcription angle and two of the neighbors. The triangle pierced by the line of sight of the virtual camera determines the supporting angles.

Selecting supporting transcription angles. We use supporting angles to compensate for the occlusion in the reference description. The general problem of covering all occlusions with a few transcription angles relates to the convex covering problem and has no general solutions. That is, for any given configuration of transcription angles, it is possible to construct a scene in which certain surfaces are occluded from all the transcription angles. However, it is frequently possible to obtain adequate coverage of occluded regions by concentrating only on the neighbors of the reference transcription angle, especially when the transcription angles are densely distributed.

Consider the triangles formed by the locations of the reference transcription angle and all adjacent pairs of its neighbors, as in Figure 7. If the transcription angle has *n* neighbors, there are *n*

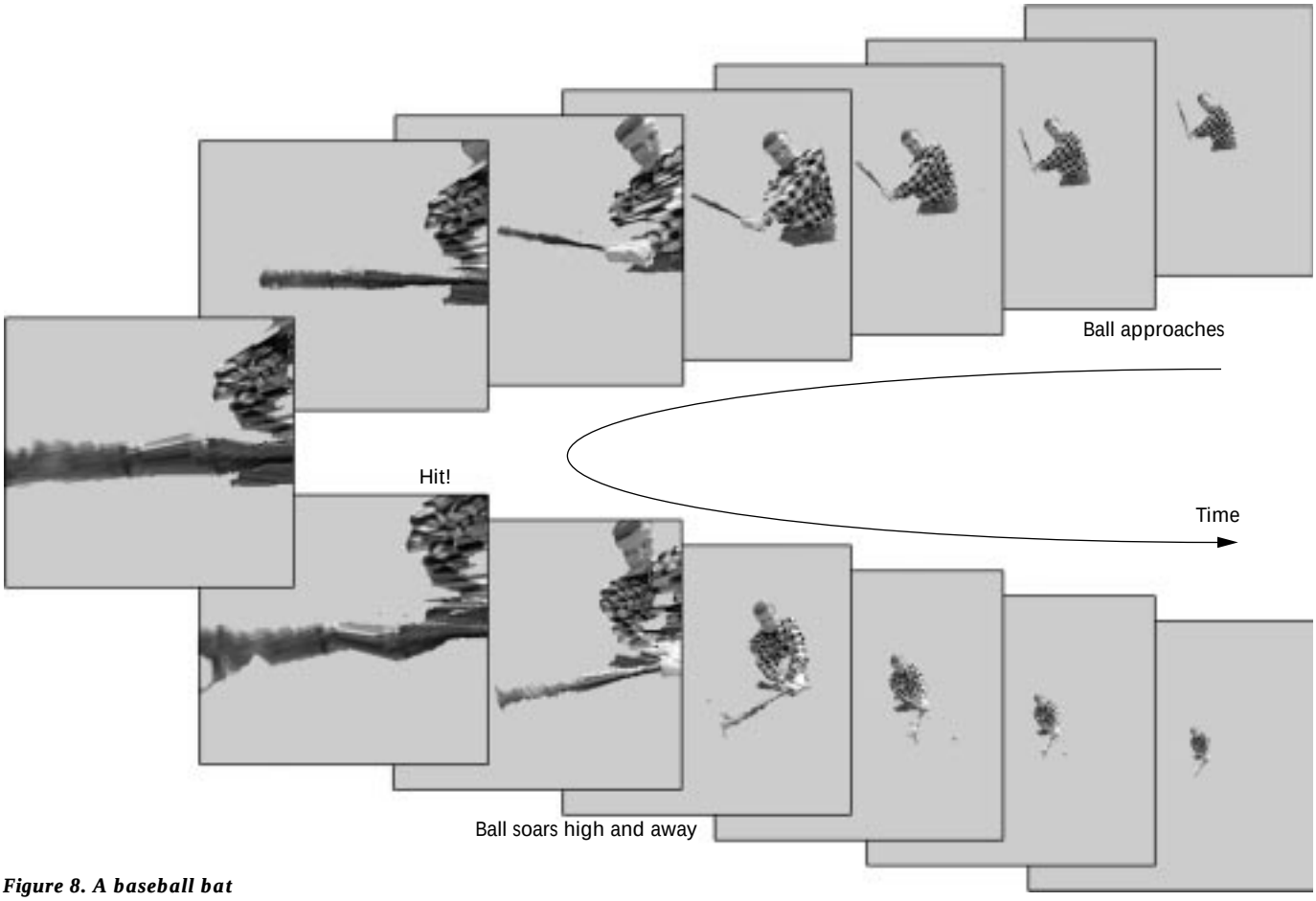


Figure 8. A baseball bat swing from the baseball's point of view.

such triangles. We determine which of these triangles is pierced by the line of sight of the virtual camera. The non-reference transcription angles that form this triangle are selected as the supporting transcription angles. Using this approach, the reference and supporting views "surround" the desired viewpoint, essentially reducing the view synthesis process to interpolation of an intermediate viewpoint from three real views.

Experimental results

We now present a few examples using real data, including both static and dynamic scenes. While the earlier sections on computing scene descriptions and generating new views only discussed static scene analysis, the extension to dynamic scenes is straightforward. Recall that video is simply a sequence of images sampled and displayed fast enough to give the illusion of continuous motion. Exploiting the same phenomenon in our 3D environment, we apply static scene analysis at each sample of a 3D time-varying event to create the illusion not only of continuous scene motion but also of a continuous user-controlled viewpoint. (Note that examples of this nature are best seen as a movie. You can find a few movies at the project's Web page, http://www.cs.cmu.edu/Groups/VirtualizedR/.)

Ball's eye view

Because the virtual camera can move through the virtualized environment without interfering with the scene, it is possible to maneuver the camera through paths that would be impossible in the real world. To demonstrate this capability, we reconstructed the flight of a virtual baseball thrown at and hit by a batter. The real images (the same images used throughout this paper to explain the various components of Virtualized Reality) come from a dynamic sequence of a person swinging a baseball bat.

We collected the image sequences using our initial hardware configuration of 10 cameras (recall that six camera views for one time instant appear in Figure 2). We distributed these 10 cameras into two clusters of five cameras and computed stereo

Figure 9. Comparing generated virtual-camera views with real-camera views, located near the middle of the two transcription angles. Views from a real camera (top row). Views generated for the same positions using Virtualized Reality (bottom row).

for a single camera in each cluster, resulting in only two transcription angles. The virtual images (in Figure 8) show the trajectory of a ball pitched to the batter, hit by the bat, and knocked high after being hit (in baseball terms, a popup). The transcription angles were about 2,750 mm and 3,000 mm away from the baseball bat at the point of the virtual camera's closest approach (the image labeled "Hit!" in Figure 8), while the virtual camera was approximately 250 to 300 mm away from the bat. As the images demonstrate, the synthetic output is qualitatively correct even for this large change in depth.

The distortions present in this example result from three sources. Fundamentally, the real images limit the texture resolution, so as the virtual camera moves into the scene, the texture mapping may zoom in beyond this available resolution. Second, scene structure is limited by the quality of camera calibration and correspondence estimation and by the available image resolution (which affects stereo). Therefore, the movement of the camera "deep" into the scene also highlights any inaccuracies in the computed scene structure. Finally, the system has only two transcription angles from which to work, and these viewpoints are physically separated by almost two meters, representing a rotation of nearly 60 degrees around the person. Having only two views leaves more holes in the synthetic output, while wide separation of those views requires more accurate surface reconstruction to maintain consistency of structure between the transcription angles.

Virtual camera versus real camera

Virtualized Reality seeks only to provide "believable" views of a scene at times, which requires some internal consistency but not necessarily consistency with the real world. This example explores the more difficult case requiring faithful, accurate reconstruction. Here we compare the virtual view generated using Virtualized Reality with what a real camera placed at that location sees for three frames of a golf swing. The scene was again captured from two transcription angles (four cameras in a cluster), located at (821, –74, –2,502) and (–968, –35, –2,334) in the world coordinate frame (units are millimeters). The person was approximately 2,500 mm away from the transcription angles.

Figure 9 shows the virtual and real camera views located at (–228, –353, –2,580), near the middle of the two transcription angles. Figure 10 does the

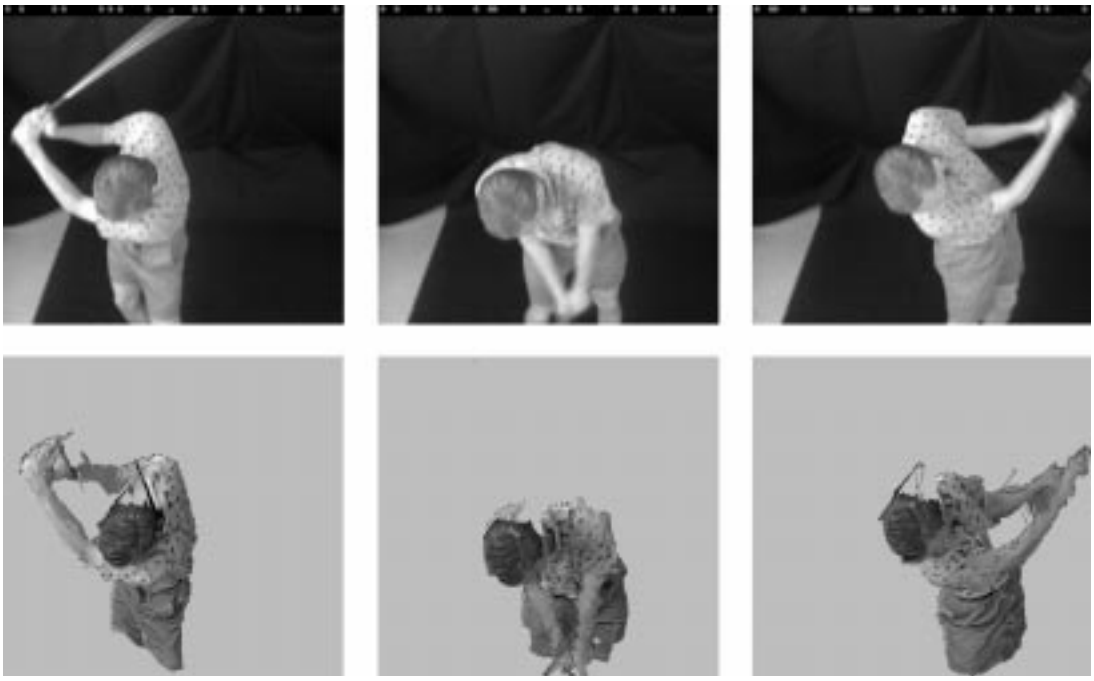


Figure 10. Comparing generated virtual-camera views with real-camera views, located far from the two transcription angles. Views from a real camera (top row). Views generated for the same positions using Virtualized Reality (bottom row).

Figure 11. Some views from a spiraling path in a static scene.

same for the virtual and real camera position (181, –1,050, –1,500), significantly far from the transcription angles in x, y, and z directions. The gap at the back of the person's neck corresponds to the region occluded from both transcription angles. The virtual camera view is quite faithful to the real camera view, though the errors in recovering the structure show up more prominently when the virtual camera is far from the transcription angles. Note that this scene was processed using full-image frames, each of which contains two image fields taken at different times—clearly visible in the real images of the golf club, for example. Even without compensation for this effect, the recovered structure was sufficient to produce realistic output.

All-around view

As noted, the two examples above used an early setup for studying Virtualized Reality that consisted of two transcription angles. We have subsequently expanded the setup to 51 transcription angles to virtualize the event from all directions. We expected the increase in the number of transcription angles to improve the overall synthetic image quality in three ways. First, with more than two views, we expected that hole filling would work better since we could always select three transcription angles from which to render the scene. Second, the cameras were more closely spaced than before, reducing the space over which each transcription angle was required to assist with the rendering. Finally, because the cameras surrounded the scene, the viewer could move nearly anywhere in the scene and still see a reasonable reconstruction.

We demonstrate this updated system using a simple static scene of a person sitting on a table. Figure 11 shows a few views of this scene from a virtual camera winding through the dome. As expected, the number of unfilled pixels dropped significantly. Almost all of the remaining holes

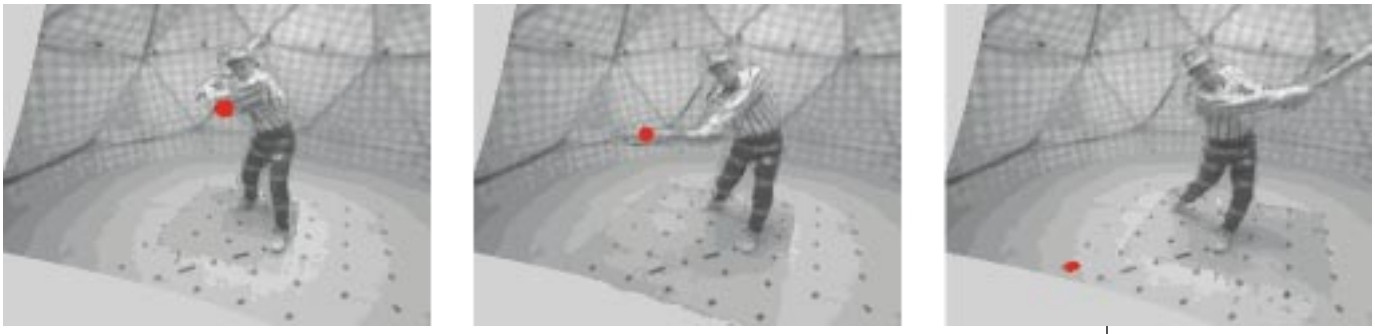


Figure 12. Introducing a virtual ball into a virtualized scene.

occur because the virtual camera has moved below all of the real transcription angles, which requires extrapolation rather than interpolation of the scene descriptions at the reference and supporting transcription angles.

Two significant 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 scene description. It results from inconsistencies in the video gain and offset levels of each camera. If a "bright" camera is used as a supporting angle of a "dark" camera, then even correctly matched regions can exhibit significant differences in apparent brightness. This problem is compounded because the absolute intensity received from a real surface usually varies with the viewing angle, but our system does not currently model this effect.

The second error, not really discernible in a set of static renderings, is the inaccuracies in reproducing motion parallax. That is, as the virtual camera moves around a static scene, the viewer expects to see continuous motion in the virtual image sequence, but the actual virtual image motion exhibits occasional discontinuities. This "jerkiness" occurs when the reference transcription angle switches, which may reveal the inconsistencies between the structures recovered from the two angles.

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 ball, for example, is introduced into a virtualized baseball scene as shown in Figure 12. 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. We can use this approach to extend chromakeying, 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.[17] In fact, we can even simulate shadows of the virtual object on the virtualized scene, and vice versa, further improving the output image realism.

Conclusions and future work

Our examples prove that useful and interesting applications of Virtualized Reality are just around the corner. We intend to pursue such possibilities in the coming months. We also plan to integrate a stereoscopic display with head tracking to give viewers the feeling of true immersion in the scene.

An open issue concerns the combined effect of imperfect calibration, correspondence finding, and mesh generation. The experiments presented here reveal a clear need to improve the rendering system to be more resilient to these inaccuracies. This would affect the filling of holes and the motion parallax of the synthesized output—a dynamic property that has received almost no attention from the view synthesis community in the past. In addition, it is clear that systems with many cameras require synthesis algorithms capable of handling images of varying brightness to prevent "ghosts" in the holes present in the image synthesized from the reference transcription angle.

A research effort currently under way involves creating object-centered models of the scene by merging the scene descriptions. We use a voxel-based approach for the merging, with each scene description voting for the presence or absence of surfaces in the 3D world based on the depth map from that transcription angle. The contribution of each depth map is only one of many, providing a significant degree of robustness in the presence of noise in the depth maps. The final model is recovered by triangulating an isosurface extracted from the merged voxel space, providing an object-centered model for the scene such as the one in

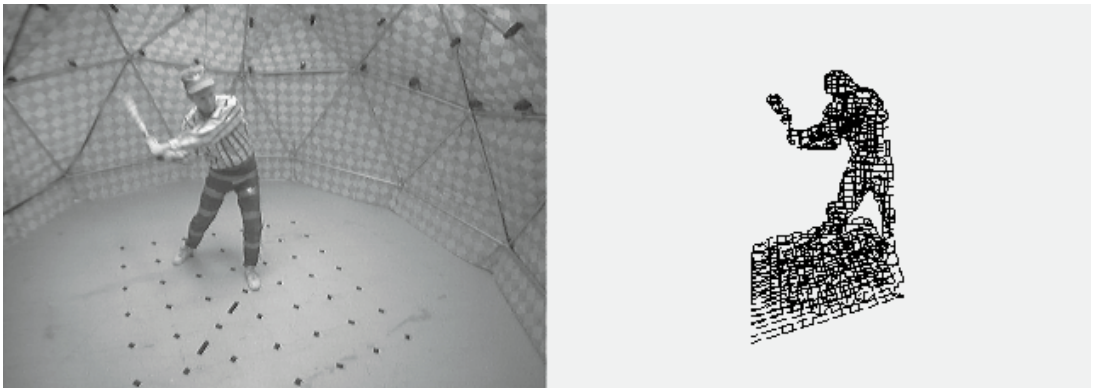


Figure 13. An object-centered model (right) computed by merging 51 depth maps and shown aligned with an actual camera view (left).

Figure 13. Using the recovered structure, it is also possible to accumulate texture from the 51 intensity images.

The virtualizing facility itself opens up possibilities for interesting research in other areas as well. For example, other view generation strategies, image-based rendering or view interpolation in particular, can be studied and compared with the depth map-based view generation. We could also extend the capabilities of Virtualized Reality to let the viewer interact with the virtualized world. For instance, modeling a scene as a collection of distinct objects lets the viewer manipulate the position and appearance of each independently. We intend to pursue these directions in the near future. MM

References

1. M. Okutomi and T. Kanade, "A Multiple-Baseline Stereo," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 4, 1993, pp. 353-363.
2. B.L. Tseng and D. Anastassiou, "Multiviewpoint Video Coding with MPEG-2 Compatibility," *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 6, No. 4, Aug. 1996, pp. 414-419.
3. E. Chen and L. Williams, "View Interpolation for Image Synthesis," *Proc. Siggraph 93*, ACM Press, New York, 1993, pp. 279-288.
4. T. Werner, R.D. Hersch, and V. Hlavac, "Rendering Real-World Objects Using View Interpolation," *Proc. IEEE Int'l Conf. on Computer Vision*, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 957-962.
5. S. M. Seitz and C.R. Dyer, "Physically Valid View Synthesis by Image Interpolation," *IEEE Workshop on the Representation of Visual Scenes*, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 18-25.
6. S. Laveau and O. Faugeras, "3D Scene Representation as a Collection of Images," *Proc. 12th Int'l Conf. on Pattern Recognition*, IEEE CS Press, Los Alamitos, Calif., 1994, Vol. 1, pp. 689-691.
7. L. McMillan and G. Bishop, "Plenoptic Modeling: An Image-Based Rendering System," *Proc. Siggraph 95*, ACM Press, New York, 1995, pp. 39-46.
8. S.B. Kang and R. Szeliski, "3D Scene Data Recovery Using Omnidirectional Multibaseline Stereo," *Proc. IEEE Computer Vision and Pattern Recognition (CVPR 96)*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 364-370. Also to appear in *Int'l J. of Computer Vision*, 1997.
9. R. Jain and K. Wakimoto, "Multiple Perspective Interactive Video," *Proc. IEEE Conf. on Multimedia Computing and Systems*, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 202-211.
10. A. Katayama et al., "A Viewpoint Dependent Stereoscopic Display Using Interpolation of Multiviewpoint Images," *SPIE Proc. Vol. 2409: Stereoscopic Displays and Virtual Reality Systems II*, SPIE, Bellingham, Wash., 1995, pp. 11-20.
11. K. Satoh, I. Kitahara, and Y. Ohta, "3D Image Display with Motion Parallax by Camera Matrix Stereo," *Proc. IEEE Conf. on Multimedia Computing and Systems*, IEEE CS Press, Los Alamitos, Calif., 1996, pp. 349-357.
12. M. Levoy and P. Hanrahan, "Light Field Rendering," *Proc. Siggraph 96*, ACM Press, New York, 1996, pp. 31-42.
13. S.J. Gortler et al., "The Lumigraph," *Proc. Siggraph 96*, ACM Press, New York, 1996, pp. 43-54.
14. T. Kanade, P.J. Narayanan, and P.W. Rander, "Virtualized Reality: Concept and Early Results," *Proc. IEEE Workshop on the Representation of Visual Scenes*, IEEE CS Press, Los Alamitos, Calif., 1995, pp. 69-76.
15. P.J. Narayanan, P. Rander, and T. Kanade, *Synchronizing and Capturing Every Frame from Multiple Cameras*, Tech. Report CMU-RI-TR-95-25, Robotics Institute, Carnegie Mellon Univ., 1995.
16. R. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE J. Robotics and Automation*, Vol. RA-3, No. 4, Aug. 1987, pp. 323-344.

17. T. Kanade et al., "A Stereo Machine for Video-Rate Dense Depth Mapping and its New Applications," *Proc. IEEE Computer Vision and Pattern Recognition* (CVPR 96), IEEE CS Press, Los Alamitos, Calif., 1996, pp. 196-202.

Takeo Kanade is U.A. and Helen Whitaker Professor of Computer Science and Robotics and Director of the Robotics Institute at Carnegie Mellon University. He received his PhD in electrical engineering from Kyoto University, Japan, in 1974. He is a Fellow of IEEE, a Founding Fellow of the American Association of Artificial Intelligence, and the founding editor of the *International Journal of Computer Vision*. He received the Marr Prize in 1990 and the Joseph Engelberger Award in 1995 and has served on many government, industry, and university advisory committees including the Aeronautics and Space Engineering Board (ASEB) of the National Research Council.

Peter Rander is a PhD candidate in electrical and computer engineering at Carnegie Mellon University. He received a BEE from University of Detroit in 1991 and an MS from Carnegie Mellon in 1993. His research interests include computer vision, computer graphics, and virtual reality. He received the Yokogawa Award for best applications paper at the 1996 International Conference on Multisensor Fusion and Integration for Intelligent Systems.

P.J. Narayanan is a scientist at the Centre for Artificial Intelligence and Robotics (CAIR) in Bangalore, India. He received a BS from IIT, Kharagpur, in 1984 and a PhD in computer science from University of Maryland in 1992. He was a faculty member at the Robotics Institute of Carnegie Mellon from 1992 to 1996. His research interests include computer vision, virtual reality, computer graphics, and parallel processing.

Contact Rander at the Robotics Institute, Carnegie Mellon University, Smith Hall, Pittsburgh, PA 15213-3890, e-mail rander@cs.cmu.edu.