Depth+Texture Representation for Image Based Rendering

P. J. Narayanan, Sashi Kumar P and Sireesh Reddy K
Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad 500019

Abstract

Image Based Rendering holds a lot of promise for navigating through a real world scene without modeling it manually. Different representations have been proposed for IBR in the literature. In this paper, we argue that a representation using depth maps and texture images from a number of viewpoints is a rich and viable representation for IBR. We discuss different aspects of this representation including capture, representation, compression and rendering. We show several results to show how the representation can be used to model and render complex scenes.

1. Introduction

Image Based Modeling and Rendering (IBMR) has attracted much attention in the past decade. The potential to produce new views of a real scene with the realism impossible to achieve by other means makes it very appealing. IBMR aims to capture an environment using a number of (carefully placed) cameras. Any view of the environment can be generated from these views subsequently. Different internal representations are used by different IBR methods. These can be divided into two broad categories: representations without any geometric model and representations with geometric model of some kind.

Early IBR efforts produced new views of scenes given two or more images of it [4, 18]. They used point-to-point correspondence, which contained all the structural information about the scene. Many later techniques followed this philosophy of generating new views purely from images. These include methods that represent the scene as a collection of rays, which in the most general case produced the plenoptic function [1]. A new view is generated by picking an appropriate subset of such rays [13, 10, 7, 19]. They require a large number of input views – often running into thousands – for modeling a scene satisfactorily. This makes them practically unusable other than for static scenes. The representation was also bulky and needs sophisticated compression schemes.

The use of approximate geometry for view generation was a significant contribution of Lumigraph rendering [7]. The availability of even approximate geometry can reduce the requirements on number of views drastically. View-dependent texture mapping [6] used known geometry and selects textures relevant to the view being generated to model architectural monuments. Unstructured Lumigraph [3] extend this idea to rendering using an unstructured collection of views and approximate models. The Virtualized Reality system captured dynamic scenes and modeled them for subsequent rendering using a studio with a few dozens of cameras [15]. Many similar systems have been built in recent years for modeling, immersion, videoconferencing, etc. [20, 2]. Recently, a layered representation with full geometry recovery for modeling and rendering dynamic scenes has been reported by Zitnick et al. [22].

In this paper, we discuss a representation consisting of depth and texture from different locations for image based modeling and rendering. We call this the Depth+Texture Representation (D+TR). Similar representations have been used in the past [12, 14]. McMillan called it depth image and discussed how they could be warped to new views. Mark generated depth images of complex models on the fly using graphics and used a post-rendering warping scheme for faster rendering of the models [11].

The D+TR is useful for many reasons. One, the state-of-the-art in 3D structure recovery using multicamera stereo has made it possible to capture aligned depth and texture of dynamic events from many vantage points satisfactorily. The number of cameras required is reasonable and they can be calibrated effectively. Two, the models generated from D+TR are compatible with standard graphics tools; hence rendering can be done effectively using standard graphics hardware. Third, the visibility-limited aspect of the representation provides several locality properties. A new view will be affected only by depths and textures in its vicinity. This increases the fidelity of the generated views even when the geometric model recovered is inexact.

We describe the depth+texture representation, algorithms to render using it, and the important aspects of this representation in this paper. We also show results of applying it on synthetic scenes and real scenes involving complex viewpoints.

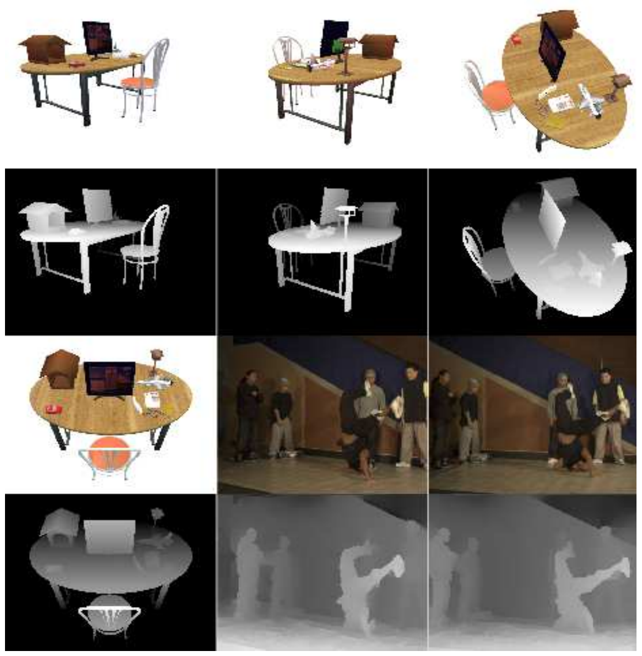
Figure 1: Texture and depth map from four viewpoints of a synthetic scene and two views of a real scene. Depth map is shown immediately below the corresponding image.

2. Depth+Texture Representation

The basic representation consists of an image and a depth map aligned with it, along with the camera calibration parameters. The depth is a two-dimensional array of real values, with location (i, j) storing the depth or normal distance to the point that projects to pixel (i, j) in the image. Computer vision provides several methods to compute such structure of visible points, called the 2-$\frac{1}{2}$D structure, using different clues from images. Motion, shading, focus, interreflections, etc., have been used to this end, but stereo has been most popular. Traditional stereo tries to locate points in multiple views that are projections of the same world point. Triangulation gives the 3D structure of the point after identifying it in more than one view. Volumetric methods turn this approach round and map each world volume cell or voxel to the views in which it is visible [9]. Visual consistency across these cameras establishes the voxel as part of a visible, opaque surface. Recovering such geometric structure of the scene from multiple cameras can be done reliably today using stereo [17]. Range scanners using lasers, structured lighting, etc., can also been used to detect structure. Figure 1 gives images and depth maps for synthetic and real scenes from different viewpoints. Closer points are shown brighter.

The depth map gives the Z-coordinates for a regularly sampled X-Y grid coinciding with pixel grid of the camera. Combined with camera calibration parameters, this represents the 3D structure of all points visible from a 3D location as a point-cloud. Grouping of points into higher level structures such as polygons and objects is not available and will have to be inferred. The D+TR from one viewpoint represents local, partial structure of the scene, i.e., parts visible from a point in space with a limited view volume. The entire scene can be represented using multiple, distributed D+TRs which together capture all of the scene space. It is possible to merge these partial models into a single global structure using mesh stitching [21], volumetric merging [5], etc.

We now discuss some of the issues relating to the depth+texture representation.

Construction: The D+TR can be created using a suitable 3D structure recovery method, including stereo, range sensors, shape-from-shading, etc. Multicamera stereo remains the most viable option as cameras are inexpensive and non-intrusive. Depth and texture needs to be captured only from a few points of view since geometry can be interpolated. A calibrated, instrumented setup consisting of a dozen or so cameras can capture static or dynamic events as they happen. Depth map can be computed for each camera using other cameras in its neighbourhood and a suitable stereo program. The camera image and calibration matrix complete one D+TR. This is repeated for all cameras resulting in the D+TR representation of the scene.

Representation: Depth and texture can be stored essentially as images in memory and disks. The depth map contains real numbers whose range depends on the resolution of the structure recovery algorithm. Images with 16 bits per pixel can represent depths upto 65 meters using integer millimeter values. The depth images need to be handled differently as they do not carry photometric information. Each D+TR needs 12 numbers to represent the 3×4 calibration matrix.

Compression: The image representation of the depth map may not lend themselves nicely to standard image compression techniques, which are psychovisually motivated. The scene representation using multiple D+TRs contains redundant descriptions of common parts and can be compressed together. Multiview compression of texture images can be performed by exploiting the constraints between views such as disparity, epipolar constraint, multilinear tensors, etc. Multiview compression of the depth maps is an area that merits serious attention.

Rendering: The depth map gives a visibility-limited model of the scene and can be rendered easily using graphics techniques. Texture mapping ensures that photorealism can be brought into it. Rendering the scene using multiple depth maps, however, require new algorithms. We discuss this issue in detail in the next section.

3. D+TR Rendering

We first describe ways in which a single D+TR can be rendered, followed by a discussion on how the scene can be rendered seamlessly using multiple D+TRs.

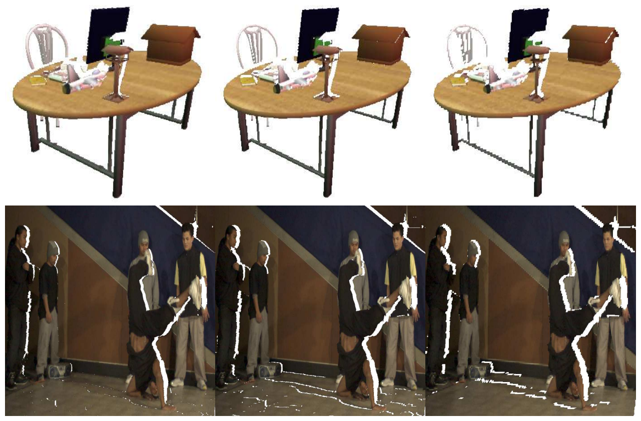
Figure 2: Rendered views of the synthetic scene (top) and real scene (bottom) from the same viewpoint. Left column uses splatting for rendering, middle column uses implicit triangulation. Last column subsamples the D+TR by a factor of 2 and uses triangulation for rendering.

3.1 Rendering one D+TR

A depth map represents a cloud of 3D points and can be rendered using one of two ways using a standard graphics system. In each case, the underlying model is visibility limited since the cloud of points is.

Splatting: The point cloud can be splatted or rendered as point-features. Splatting techniques broaden the individual 3D points to fill the space between points. The colour of the splatted point is obtained from the corresponding image pixel. Splatting has been used as the method for fast rendering, as point features are quick to render [16]. The disadvantage of splatting is that holes can show up where data is missing if we zoom in much. The left column of Figure 2 shows the results of rendering the D+TR from a viewpoint using single pixel splatting. The holes due to the lack of information can be seen as “shadows”, for example of the vertical bar on the table top.

Implied Triangulation: The pixel grid of D+TR provides a regular, dense, left-to-right and top-to-bottom ordering in X and Y directions of the point cloud. These points are sufficiently close to each other in 3D except where depth discontinuities are involved. A simple triangulation can be imposed on the point cloud as follows: Convert every 2×2 section of the depth map into 2 triangles by drawing one of the diagonals. The depth discontinuities are handled by breaking all edges with large difference in the z-coordinate between its end points and removing the corresponding triangles from the model. Triangulation results in the interpolation of the interior points of the triangles, filling holes created due to the lack of resolution. The interpolation can produce low quality images if there is considerable gap in the resolutions of the captured and rendered views, such as when zooming in. This is a fundamental problem in image based rendering. Images of the middle column of Figure 2 have been rendered using this approach. Holes due to shift in viewpoint can be seen on the computer screen and on the people at the back.

3.2 Subsampling the D+TR

The regular grid of the D+TR makes it easy to reduce the model complexity. Subsampling the grid will reduce the number of primitives to be drawn. The reduction in detail is blind and not geometry driven. A hierarchy of representations is possible with the maps subsampled by different factors. Subsampling can have a serious impact when splatting as no interpolation is performed. Splatting involves less rendering resources and the need for subsampling may be felt less. The right column of Figure 2 shows the same viewpoints rendered after subsampling the D+TR by a factor of 2 respectively in the X and Y directions. The overall quality is passable, but small features like the chair back has been affected badly.

3.3 Rendering Multiple D+TRs

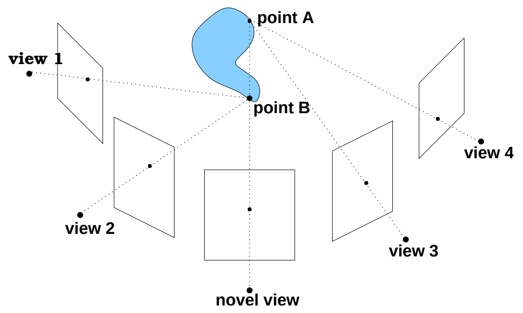

Figure 3: Different cases for view generation. See text for details.

Each D+TR can be mapped to a new view using splatting or implied triangulation. The generated view will have holes or gaps corresponding to the part of the occluded scene being exposed in the new view position. These holes can be filled using another D+TR that sees those regions. Parts of the scene could be visible to multiple cameras. The views generated by multiple D+TRs have to be blended in such cases. The general situation is shown in Figure 3. Both hole filling and blending can be considered together as the merging of multiple views. We discuss merging next.

3.4 Merging Multiple D+TRs

The colour and the depth values of each pixel of the new view are available from each D+TR. The first task is to fill the holes in one view using the others. Each pixel of the new view could contain colour and z values from multiple D+TRs. For example, point A and point B of Figure 3

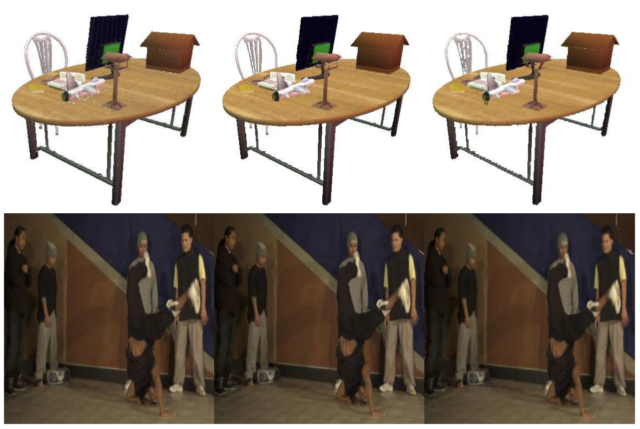
Figure 4: Hole-free, merged versions of views shown in Figure 2

map to the same pixel. The closest point is the correct point and should be chosen to provide colour. In general, when n D+TRs map to a pixel, they should be merged based on the closest z value in the new view. The conventional z-buffering algorithm can be used for this and can take advantage of hardware acceleration. When a portion of the scene is part of multiple D+TRs, the z-buffer values will be close, as for point B using views 1 and 2. The colour value from any view can be given to the pixel. Blending the colour values will provide better results as the viewpoint shifts. Buehler et al. present a detailed discussion on blending different colour estimates at each pixel due to different views in their paper on Unstructured Lumigraph Rendering [3]. The weights assigned to each view should reflect its proximity with the view being generated. We present a discussion on blending next. Figure 4 shows the blended versions of new views given in Fig. 2.

Pixel Blending: Consider each input D+TR that contributes a colour to a pixel of the novel view. Its contribution should be weighted using a proximity measure of the imaging ray of that D+TR to the imaging ray in the new view. Several proximity measures could be used for this. We use one derived from angular distance between the two imaging rays as shown in Figure 5. The blend function is applied independently on each pixel. The blend function should result in smooth changes in the generated view as the viewpoint changes. Thus, the views that are close to the new view should get emphasized and views that are away from it should be deemphasized.

Blend Function: As shown in Figure 5, t_1 and t_2 are the angular distances at a pixel for D+TRs from C1 and C2. Several proximity measures can be defined using these angles. Exponential blending computes weights as $w_i = e^{-ct_i}$ where i is the view index, t_i is the angular distance of view i, and w_i is the weight for the view at that pixel. The constant c controls the fall off as the angular distance increases. Input views for which $t_i \geq \pi/2$ are eliminated as they

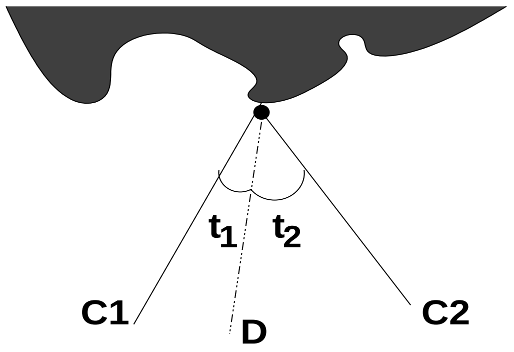

Figure 5: Blending is based on angular distances

view the scene from the other side. In practice, $c = 1$ or $c = 2$ have been found to work well. Cosine weighting uses $w_i = \cos^r t_i$ for a suitable $r > 2$.

The w_i values are sorted and the top k values are chosen for blending as long as the weight is above a threshold. These k weights are normalized at each pixel by dividing by their sum to yield w'_i values. The colour assigned to a pixel is $\sum_i w'_i c_i$ where c_i is the colour at the pixel due to the view i and w'_i is the normalized weight for it. We found that $k = 3$ or $k = 4$ work quite well.

The complete rendering algorithm is given below.

```
for each D+TR $d$ that is on the same side do
   1. Generate the new view using $d$'s depth and texture.
   2. Read back image and $z$ buffers.
End for
for each pixel $\mathbf{p}$ in the new view do
   3. Compare the $z$ values for it across all views.
   4. Keep all views with the nearest $z$ value within a threshold.
   5. Estimate the angles for each D+TR to 3D point x$p$
   6. Compute the weights of each D+TR model.
   7. Assign the weighted colour to pixel $\mathbf{p}$.
End for
```

Figure 6 shows the relative contributions of different D+TRs for 4 points marked on the new view. The D+TRs that are close to the new viewpoint are emphasized in general, but the effect of occlusions can be perceived in the selection of the cameras.

4. Results

We demonstrate the representation and rendering algorithm for D+TR models of synthetic and real scenes in this section. The synthetic scene contains a table, chair, a flat-panel and a few objects on the table, etc. Twenty D+TRs were used for it in two rows and separated by 36 degrees. Another version enclosed this model in a room with distinct textures for each wall. Images were saved as PPM files and the depth maps as raw files containing floating point depth

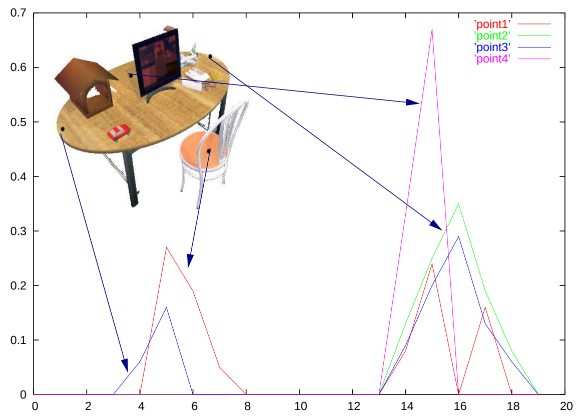

Figure 6: Weights of different cameras at 4 points shown. X axis gives D+TR number and Y axis its weight

values. The depth and image values were obtained after rendering the 3D model and reading back both the image and the Z-buffer.

We also tested the scheme on real data of an instant of the breakdance sequence from the Interactive Visual Media group of Microsoft Research. This data set has 8 D+TRs from viewpoints that are shifted horizontally. The depth maps were computed from a stereo program and were not perfect [22]. Since the coverage of the scene is sparse, the flexibility in choosing the new view without holes is limited.

Figure 7 shows 3 random views of the synthetic indoor scenes. They were rendered using implicit triangulation method and blending top 6 views at each pixel. The figure also shows the breakdance instant from three new viewpoints. The quality of generated viewpoints is quite good, especially for the synthetic scene. The D+TR models of the breakdance sequence has problems as there are a few regions that are not visible to any of the 8 cameras. Exponential weighting was used, which provides for smooth new views. The rendered quality is good for synthetic views but effects of poor depth maps can be found on the real images, especially on the floor.

Figure 8 highlights one important aspect of capturing a real world scene using cameras. The image captured by a camera could have different gains and offsets due to the difference in settings of the camera as well as the digital capture system. When images from two such cameras are merged to form a new view, the mismatch in the images can appear as artificial edges in the generated image. We can see this on the table top in Figure 8.

5. Discussions and Conclusions

In this paper, we analyzed the depth+texture representation for image based rendering. We presented the advantages of the representation and showed results of rendering for synthetic and real scenes. This representation holds promise as structure recovery using cameras has become practical. It

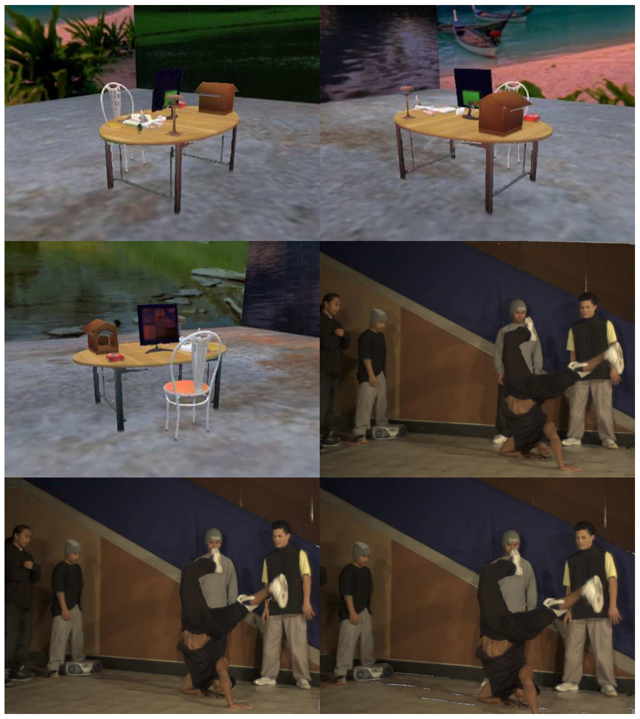

Figure 7: A few new views of the synthetic and real scenes.

is possible to merge the multiple depth maps into a single global model which is analogous to a conventional graphics model [5]. The following properties make D+TR particularly suitable for IBR.

Locality: A scene rendered using the D+TR is correct if the new view direction is at the origin of the local model or very close to it, even if the depths are inaccurate. Thus, using a combination of D+TR models can create faithful views of the scene even if the underlying depth maps are not accurate. Merging the D+TR models to a global model computes a consensus model. Such a model will contain distortions which will be visible from all points of view. Locality is advantageous to IBR. The structure recovered from a far-

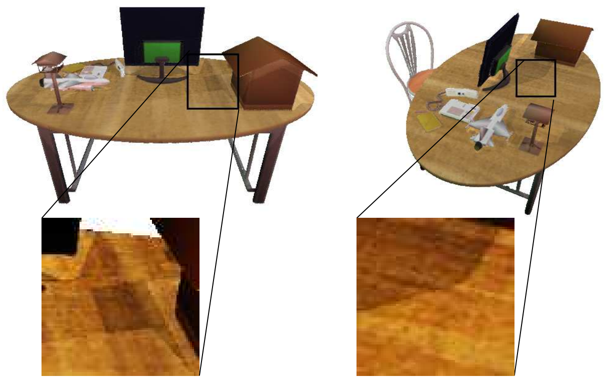

Figure 8: Mismatch in colour between two views results in artificial edges and other artifacts in the generated view.

off viewpoint and direction should have minimal impact on a generated view, if any. This is easily achieved using D+TR models as each represents a local structure of the scene. A small number of them from the same neighbourhood can be used to make up for holes when the viewpoint shifts.

Model Size: View frustum culling is among the first steps to be performed while rendering large models. The D+TR models contain only visible parts of the scene and need no culling. The locality of D+TRs ensure that the amount of geometry to be rendered is low. Thus, the computational resource requirement using D+TR rendering could be a lower. The representational complexity of the model could be lower for a merged, global model. The D+TR could be bulky as adjacent views share a lot of information. It should be possible to compress the D+TR models by taking advantage of the redundancy.

Compression: Compression is an aspect of this representation that need special attention. The representation lends itself to compression easily since the scene is described redundantly in multiple views. The compression of the depth maps and texture images have to be done differently as each represents qualitatively different signals. Compression of the depth maps as images produces poor results as image compression is tuned to remove psychovisual redundancy [8]. We are currently working on a compression scheme that takes into account the properties of depth maps and the constraints of the multiview structure.

Distribution of D+TRs: How many D+TRs are required to represent a given scene adequately? An understanding of this issue helps in planning the scene capture. Each D+TR provides a sample of the 3D world from its location. Obviously, the sampling has to be dense near the areas of fine scene structure. The quality of rendered views is likely to suffer when using depth maps that are quite far. This is an issue that requires careful study.

These properties make the D+TR representation suitable for IBR. The representation could be used even for synthetic models as the rendering requirements for a particular view could be lower. We are currently exploring the following aspects of the representation. Blending functions should be defined so that the influence of every view tapers off smoothly. This will eliminate the artificial "edges" in rendered view when the captured images differ in colour or brightness. We are studying the compression of the depth maps and the texture images together, taking advantage of the properties and constraints of the geometry of the input views. We have not paid sufficient attention to real-time rendering of the D+TR models. This is critical for its widespread use.

Acknowledgments: We thank Richard Szeliski and Sing Bing Kang for various discussions on the topic. We also thank the Interactive Visual Media group of Microsoft Research for the data used in the real image example.

References

[1] E. H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational Models of Visual Processing*. 1991.

[2] H. Baker, D. Tanguay, I. Sobel, M. E. G. Dan Gelb, W. B. Culbertson, and T. Malzbender. The Coliseum Immersive Teleconferencing System. In *International Workshop on Immersive Telepresence (ITP2002),* 2002.

[3] C. Buehler, M. Bosse, L. McMillan, S. J. Gortler, and M. F. Cohen. Unstructured Lumigraph Rendering. In *SIGGRAPH*, 2001.

[4] S. Chen and L. Williams. View Interpolation for Image Synthesis. In *SIGGRAPH*, 1993.

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

[6] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and Rendering Architecture from Photographs: A Hybrid Geometry and Image-Based Approach. In *SIGGRAPH*, 1996.

[7] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The Lumigraph. In *SIGGRAPH*, 1996.

[8] R. Krishnamurthy, B.-B. Chai, H. Tao, and S. Sethuraman. Compression and Transmission of Depth Maps for Image-Based Rendering. In *ICIP*, 2001.

[9] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. IJCV, 2000.

[10] M. Levoy and P. Hanrahan. Light Field Rendering. In *SIGGRAPH*, 1996.

[11] W. R. Mark. *Post-Rendering 3D Image Warping: Visibility, Reconstruction, and Performance for Depth-Image Warping*. PhD thesis. University of North Carolina, 1999.

[12] L. McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics*. PhD thesis. UNC, 1997.

[13] L. McMillan and G. Bishop. Plenoptic Modelling: An Image-Based Rendering Algorithm. In *SIGGRAPH*, 1995.

[14] P. J. Narayanan. Visible Space Models: $2\frac{1}{2}$-D Representations for Large Virtual Environments. In *International Conference on Visual Computing (ICVC99)*, 1999.

[15] P. J. Narayanan, P. W. Rander, and T. Kanade. Constructing Virtual Worlds Using Dense Stereo. In *Proc of the International Conference on Computer Vision*, Jan 1998.

[16] S. Rusinkiewicz and M. Levoy. QSplat: A multiresolution point rendering system for large meshes. SIGGRAPH, 2000.

[17] D. Scharstein, R. Szeliski, and R. Zabih. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002.

[18] S. M. Seitz and C. R. Dyer. View Morphing. In *SIGGRAPH*, 1996.

[19] H.-Y. Shum and L.-W. He. Rendering with concentric mosaics. In *SIGGRAPH*, 1999.

[20] H. Towles, W.-C. Chen, R. Yang, S.-U. Kam, and H. Fuchs. 3D Tele-Collaboration Over Internet2. In *International Workshop on Immersive Telepresence (ITP2002)*, 2002.

[21] G. Turk and M. Levoy. Zippered Polygon Meshes from Range Images. In *SIGGRAPH*, pages 311 – 318, 1994.

[22] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. In *SIGGRAPH*, 2004.