Interactive Dynamic Environments Using Image-Based Modeling and Rendering

Timothy A. Davis and Stephen P. Ficklin

Department of Computer Science, Clemson University, Clemson, SC, U.S.A.

Abstract

This paper presents a method for rendering a dynamic virtual environment using IBMR at interactive rates. Most IBMR methods are designed for use with static scenes and are awkward when extended temporally due to excessive memory requirements or user intervention. In our method, spatial and temporal redundancy are removed from sets of input images to reduce the time and space needed to render an interactive dynamic scene. A pre-processing step is initially applied to remove the redundancy; rendering is performed any time afterward by the explorer. The rendering process is most similar to Layered Depth Images (LDI), while image storage is most similar to video encoding technology. The benefits of our method include the preservation of pixels sampled at higher rates without requiring additional data structures, and the freedom of movement for the user throughout the scene.

Categories and Subject Descriptors [according to ACM CCS]: I.3.3 [Computer Graphics]: Viewing Algorithms

1. Introduction

Providing walk-through capabilities within a temporally dynamic virtual environment, or animation, is an important topic in computer graphics. Most noticeably, this capability is available in the entertainment industry through virtual reality rooms and computer games. However, these applications are often limited in detail due to the difficulty of modeling complex scenes. Thus, when high levels of detail are provided, the dynamic portions of a scene are not. Many times freedom of movement is also limited or non-existent. Great benefits can therefore be gained by combining realistic detail with complete freedom of movement within an interactive dynamic 3D virtual environment.

Image-Based Modeling and Rendering (IBMR) techniques provide a means for creating environments with high levels of detail that are otherwise extremely time-consuming or even impossible to render or create by hand. In this paper, we propose an algorithm for providing interactive navigation through high-quality animations by extending existing IBMR methods. The original motivation for this work stemmed from a desire to walk through a high-quality animation of a molecular crystallization process that could not be rendered in real-time. We were especially interested in preserving the most highly sampled pixels in each image to facilitate zooming in on areas of interest while maintaining the scientific accuracy of the scene.

Our method uses a sequence of images obtained from a set of cameras and renders the dynamic environment in interactive time. A user can view the action of the scene from any vantage point. The level of detail is high, since the primitives used for display are images. The algorithm takes advantage of spatial and temporal redundancy within the set of images to improve render rates and lessen storage requirements. Render quality is highest since pixels sampled at higher rates are preserved in each image

2. Previous Work

IBMR methods, techniques and areas of research can be classified according to the following:

- warping methods
- scene reconstruction methods
- algebraic methods
- plenoptic methods
- point methods
- dynamic methods

These groups, while not a formally recognized set of classification, provide a convenient means of grouping past techniques and placing our work in proper context.

Warping methods attempt to create novel views or transitions between views from a set of reference images

by translating pixels from the source image plane to locations on the target image plane. A significant portion of early literature in IBMR can be classified in this category including: texture mapping[5,15] QuickTime VR[8], morphing[4,37,38], view morphing[31] and view interpolation[7,30]. These methods increase the detail of a scene or provide novel views at relatively quick speeds but severely limit the user's freedom of movement. Also, some of these methods require user-directed correspondence matching which prevents the algorithms from extending temporally since the time required to mark the many input images is often excessive.

Scene reconstruction methods recreate the 3D geometry of the scene by estimating the geometry of underlying scene objects represented in the input images. Some examples include: the 8-Point Method[24], automatic 3D reconstruction of the scene[13], the Michelangelo Project[22], volumetric methods[9], meshes[16], and hybrid techniques[11], among others. Despite the high level of detail that reconstruction methods provide, they are cumbersome when extended temporally since the underlying geometries need to reflect temporal changes in the scene. Also, rendering speed is dependent on the complexity of the model, which may not guarantee an interactive rendering rate.

Algebraic methods seek to create valid novel views of a 3D scene from a set of reference images by using the geometric or algebraic relationships between the cameras, the view planes of each reference image, and the virtual camera. Some examples include methods introduced by Laveau and Fargus[18], Shashua[34] and Avidan and Shashua[3]. Algebraic methods are limited primarily in their extendibility. Particularly, they require user intervention for correspondence marking or are otherwise limited by camera placement or the number of input images that can be used.

Point methods seek to reconstruct a novel view of a scene where points are the only object primitive available. These points are then rendered as is without conversion to geometric surfaces or correspondence matching. Levoy[20] initially proposed point rendering, and techniques that borrow from it include volume rendering[36], delta trees[10], LDIs[32,28], LDI trees[6], and post-rendering 3D warping[26]. Our research is most closely related to the point methods, in particular LDIs and LDI trees, since we take advantage of the fast rendering speed that point methods offer. However, these methods suffer from various disadvantages. LDIs in particular lose the highly sampled pixels in the input images. LDI trees attempt to overcome this deficiency but require the construction of an octree, which becomes cumbersome when extended temporally.

Plenoptic methods are based on the plenoptic function, formally introduced by Adelson and Bergen[1]. The Plenoptic Function describes all visible light within a scene and is represented, at most, by seven parameters.

- Image Collection Step *(performed only once)*
  - build a highly detailed synthetic scene
  - capture various animations of the scene from different viewpoints
    - obtain pixel color & depth information
    - obtain surface normals per pixel (not necessary but desired)
- Pre-Processing Step (*performed only once)*
  - estimate normals (if not provided)
  - remove temporal redundancy
    - separate pixels into background images
    - separate pixels into foreground images
  - remove spatial redundancy of foreground and background images
    - register the images with one another
    - remove redundant pixels with lower sampling rates
  - efficiently store the foreground and background images
- Rendering Step *(performed as often as desired)*
  - load the pre-processed images
  - register and reconstruct the novel views at interactive rates using point method rendering

Figure 1: *Algorithm for image-based rendering in a dynamic environment.*

These methods include plenoptic modeling[27], light-field rendering[21,17], lumigraph[14], multiple-center-of-projection images[29], concentric mosaics[33], and plenoptic stitching[2]. These methods allow a wide range of freedom of movement but are difficult to extend temporally, and require extreme amounts of memory.

Some recent dynamic methods include light fields for dynamic rendering[23], dynamic view morphing[25], and spatio-temporal view interpolation[35]. While these methods relate in their attempts to use IBMR to render dynamic scenes, they differ in their approaches. The extension of the light field and view morphing do not entirely overcome the limitations of their static ancestors. Spatio-temporal view interpolation differs from this research in that it seeks to interpolate the dynamics of the scene over time and is not concerned with rendering at interactive rates.

3. Algorithm Summary

Our algorithm consists of three principle stages, as shown in Figure 1: image collection, pre-processing and rendering. In the first stage, image collection, we use images of synthetic scenes accompanied with depth and normal information easily obtained from rendering engines. Using images taken of a real scene is possible, but

requires further processing to determine per-pixel depth, such as algebraic and normal estimation methods. Such techniques for determining pixel depths and normals from real images are restrictive, however, and do not always provide accurate results. Fortunately, we can compute these values directly since our target application is for high-quality synthetic images of scenes that are impossible to render traditionally at non-interactive rates.

The second stage is a pre-processing step that greatly improves the final rendering time. Many of the input images contain redundant pixels that represent the same point in space. We need not render redundant pixels and consequently, we gain significant improvement in render time by ignoring them. The pre-processing stage removes these redundant pixels from the input images, thus reducing the memory requirements and the image size. Pixels with higher sampling rates are kept while redundant pixels with lower sampling rates are discarded. The images are efficiently stored to take advantage of their smaller size.

Finally, the rendering stage, performed as often as desired, generates novel views of the scene from any vantage point the user desires. The algorithm renders quickly and is capable of sustaining interactive frame rates. We discuss each of these steps further in the following subsections.

3.1 Image Collection

In the first step, image collection, we collect the reference images used to create the environment. During scene generation, we place any number of cameras in static locations throughout a synthetic scene to capture the animation from multiple viewpoints. Each camera produces its own image sequence, which will be used to create the virtual environment. Figure 2 shows an example of a setup that could be used to create image sequences for our algorithm.

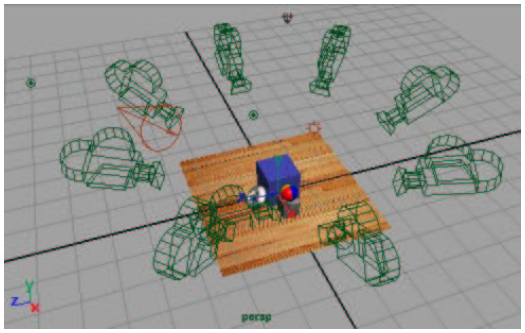

Figure 2: *A scene containing 9 cameras each viewing the same scene from different vantage points. The image sequences obtained from each are used to create the virtual environment.*

3.2 Pre-Processing

The pre-processing step consists of normal estimation (if required), the removal of temporal and spatial redundancy, and efficient image storage.

First, normals are an integral component of the pre-processing calculations and must be provided or estimated for each input image. If we use computer-generated images as our input primitives, we can easily compute normals during the standard rendering process, save them as accompanying data to the image, and use these exact normals during the pre-processing stage. If, however, normals are not provided with the input images, normal estimation can be performed with any normal estimation technique, such as those most commonly used with volume rendering.

Second, exploiting redundant information across an image set allows us to extract repetitive information, thus reducing the render time as well as the memory footprint. Redundancy occurs when pixels in two or more different images represent the same location in world coordinate space. Two pixels can be spatially redundant (they map to the same location in space) or temporally redundant (they remain constant over a period of time).

Removing spatially redundant pixels from an image set is not new. With LDIs, spatial redundancy is removed from an image set when two source pixels from different source images map to the same destination pixel and depth. In this case, one of the pixels is discarded. However, LDIs do not preserve sampling density of more highly sampled pixels since such pixels map to the same locations and are discarded. These pixels appear to be spatially redundant when, in fact, they are not. LDI trees attempt to preserve these aliased pixels but require the construction of an octree, which becomes cumbersome when extended to dynamic scenes.

Removing spatial redundancy requires the registration of all images into a common reference frame where pixels are compared to determine if they represent the same point on a surface in the scene. If two pixels are determined to be spatially redundant, one of the pixels is discarded. We perform this registration by mapping every image in the same time frame to every other image in that time frame and keeping those pixels that have the highest sampling rate. Figure 3 demonstrates how two different cameras can sample the same area of a surface at different rates.

We use the following equation to indicate the camera that samples a pixel at a higher rate:

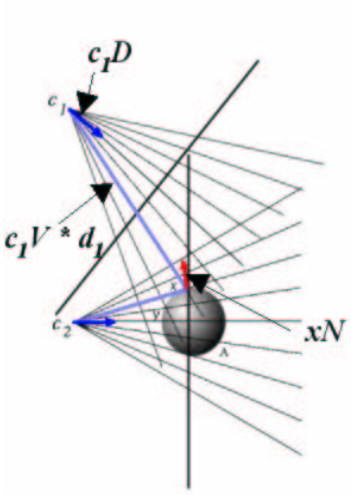

Figure 3: *The surface of object A is better sampled by camera c_2 near point y, and by the camera c_1 at point x.*

$$r = \left[\frac{(c_1V*d_1)\bullet c_1D}{(c_2V*d_2)\bullet c_2D}\right] * \left[\frac{\cos(xN\bullet -c_2V)}{\cos(xN\bullet -c_1V)}\right]$$

where

c_1V, c_2V	normalized view vector for c_1 and c_2
c_1D, c_2D	normalized look vector for c_1 and c_2
d_1 ,d_2	the depth of the pixel from c_1 and c_2
xN	the surface normal vector of point x

if $r = 1$	both pixels are sampled equally
if $r < 1$	c_1 samples the pixel more highly
if $r > 1$	c_2 samples the pixel more highly

Exploiting temporal redundancy has also been explored previously. Other branches of computer graphics, such as ray tracing and radiosity, use temporal redundancy (coherence) to decrease render times across frames[12]. Additionally, the MPEG encoding standard exploits redundancy to decrease the bit rate of its streaming video[19].

Once a pixel is determined to be temporally redundant across a set of frames, it is separated from the images and preserved in a separate set of background images. Once all pixels have been processed, those remaining in the source images constitute the dynamic portion of the scene, or in other words, the foreground. These foreground and background image sets are not representative of the actual background and foreground of the underlying scene (although usually the static portion of the scene consists of the actual background); they represent two-dimensional image sets where the background consists of pixels that remain constant over time and the foreground consists of pixels that are different from one frame to the next. This process is shown in Figure 4.

Finally, efficient image storage is the last stage of the pre-processing stage. To reduce the size of the images, we tile each one and ignore tiles where all pixels were redundant and removed. We use simple run-length encoding to reduce the image size further and store the images in a customized IFF file format.

3.3 Rendering

The rendering stage consists of two steps: loading the pre-processed images from memory, and re-projecting the pixels in the images of the current time step to the view plane of the novel view. We use the following equations to perform the projection of the pixel from 2D image coordinates to world coordinates:

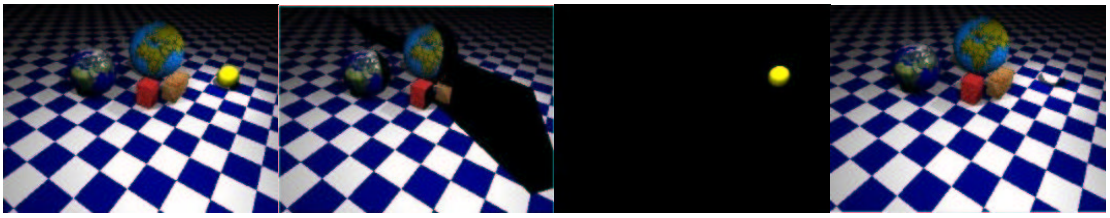

Figure 4: *The first image (left) is an original image from one camera in the scene where the yellow ball is the only dynamic object. The second image shows pixels that have been removed due to spatial redundancy with pixels from a different camera in the same time frame, which were sampled more highly in that image. Significantly more pixels will be removed once comparisons with all other images in the time frame have been performed. The third image represents the foreground image created from extracting the dynamic portions of the scene. The final image is a background image created by extracting the static portions of the scene. The algorithm creates one background image per camera, and N foreground images per camera, where N is the number of frames in the animation.*

$$x = \frac{\left(\frac{w * u}{rw} - \frac{w}{2}\right)}{\left(\frac{fl}{a}\right)} * d \qquad y = \frac{\left(\frac{h * v}{rh} - \frac{h}{2}\right)}{fl} * d$$

$z = d$

$\mathbf{P} = (x, y, z, 1)^T$

where

u, v	row column index of the current pixel
w, h	width and height of the image plane
rw, rh	resolution of the image in pixels
fl	focal length
a	aspect ratio: w / h
d	pixel's depth value
x, y, z	3D eye space coordinates
$\mathbf{P}$	3D eye space coordinate vector

Once in world coordinates, the pixels are projected onto the novel view plane using standard matrix multiplication. We perform a coarse blending within a 4-pixel region to reduce the appearance of holes. These calculations can be performed incrementally to speed render time.

We render the background images once and "paint" the foreground over top for each consecutive time step (frame). Only when the user moves vantage points do we re-render the background to reflect the changes in viewpoint. Since a significant number of pixels are removed from the foreground images, we have substantially fewer pixels to render, thus allowing for significant speedup.

4. Results

Testing was conducted on a Gateway AMD Athlon 1.1 GHz machine with 640Mb of RAM, a 7500 RPM IDE Maxtor hard drive, and an NVIDIA GeForce4 Ti4400 AGP graphics card, running Red Hat Linux 7.3.

Our first test case, shown in the top row of Figure 5, consisted of an image set from nine cameras viewing a dynamic scene. At least one of the cameras was placed to obtain a highly sampled view of a scene object. Each camera produced 60 anti-aliased frame images. We used Alias|Wavefront Maya 4.5 for modeling and rendering. Surface normals of the scene objects were not provided.

Our second test case consisted of an image set from four cameras viewing a dynamic scene and is shown in the bottom row of Figure 5. One of the cameras was placed to obtain a highly sampled view. Each camera produced 300 frames. Normals were provided with each image.

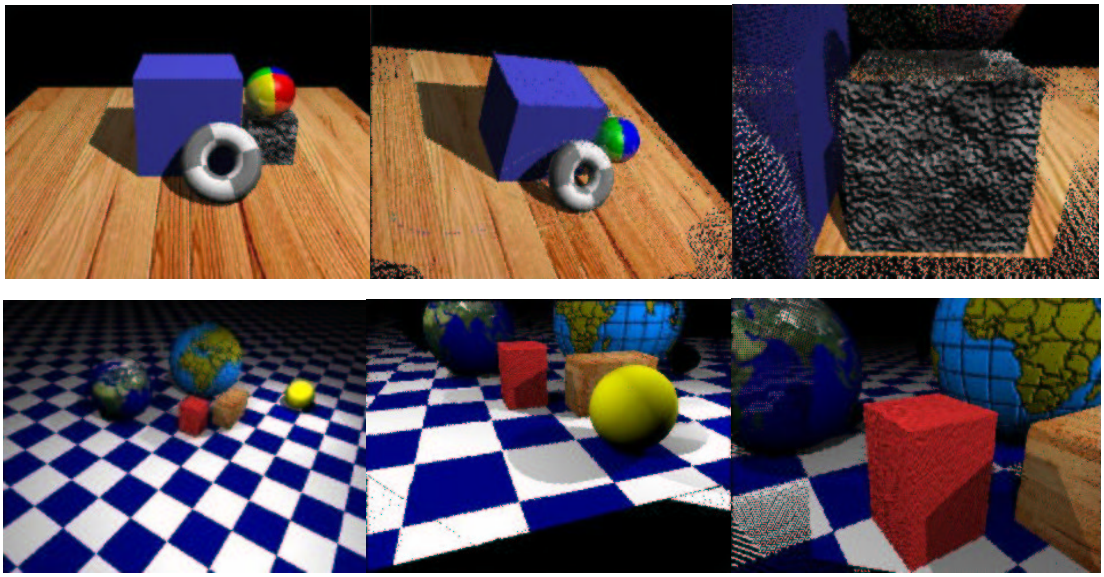

Figure 5: *The top and bottom left-most images represent original views of two dynamic scenes. The top scene was constructed using Maya and the bottom using a customized ray tracer. The center images represent a novel view of each scene from a vantage point not available in any of the source images. The right-most images represent novel views and show that highly sampled pixels are not lost. Note the high-quality detail of pixels in regions of original high sampling, while surrounding regions of lower sampling produce poorer image quality. Overall, image quality of the top set of novel views is not as good as that of the bottom images since the normals were estimated and the input images were anti-aliased. Normals computed for the bottom images, which were not anti-aliased, provided more accurate reconstruction.*

In each test case, users were free to explore the environment from any vantage point. No data structures, other than the foreground and background images themselves, were needed to aid in rendering. As a result, we achieved an 85% reduction in memory requirements, compared to that of storing the original images, and on average, an interactive render rate of 19 frames per second. The render rate slows, however, when the user changes vantage points since the background must be repainted. (The background generally contains most of the pixels in the scene and takes a bit longer to render.)

Note that we also do not adequately fill all holes that appear in the scene. In general, the rendering remains high in quality as long as the user's sampling rate is lower than that represented in the input images.

Overall, these results are quite promising, and although performance is related to the amount of spatial and temporal redundancy inherent in the scene, this technique will work across a broad range of animations.

5. Conclusion

This research provides two primary contributions. First, it presents an IBMR method that provides a walk-through environment of a high-quality dynamic scene in interactive time, where the user is not limited in freedom of movement. Second, it uses a unique method for storing pixels from a set of images in a medium conducive to speed and low memory, while preserving more highly sampled pixels without the creation of any new data structures.

This research is useful in a wide variety of applications. One example is scientific visualization, where scientists could observe changes in their experiments with high detail from any vantage point. Another example is virtual showcases of real estate properties where the viewer, unlike with currently available applications, could look at *any* portion of a piece of property. Virtual reality and possibly gaming applications could also benefit from the level of detail obtained from such a system.

While the algorithm has several benefits, some limitations exist as well. Performance is dependent on the number of pixels rendered, and the more cameras viewing the scene, the more time spent rendering. The amount of redundancy in a scene also affects the number of pixels. While no limitations exist within the algorithm as to the placement of cameras, placement does affect the performance of the final rendering since it contributes to the amount of spatial redundancy available. Cameras whose viewing frustums do not overlap cannot benefit from the removal of spatial redundancy and temporal redundancy can also be similarly limited.

Another limitation, common to many IBMR methods, is that the novel view lacks view-dependent lighting. Reflective and translucent materials cannot be properly displayed since the algorithm knows nothing of the underlying scene—it merely projects pixels and does not alter pixel colors for proper view-dependent lighting. The scene, therefore, is required to consist primarily of diffuse objects.

This work seems to be part of the natural step that IBMR is taking: from that of creating virtual static scenes to virtual dynamic ones. We hope this work can provide a building block in future research using IBMR for virtual dynamic scenes.

References

1. Adelson, E. H. and Bergen, J.R., "The Plenoptic Function and the Elements of Early Vision," *Computational Models of Visual Processing*, 1991, pp. 3-20.

2. Aliaga, D. G. and Carlbom, I., "Plenoptic Stitching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs," *SIGGRAPH '01 Computer Graphics Proceedings, Annual Conference*, ACM SIGGRAPH, August 2001, pp. 443-450.

3. Avidan, S. and Shashua, A., "Novel View Synthesis in Tensor Space," *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, June 1997, pp. 1034-1040.

4. Beier, T. and Neely, S., "Feature-Based Image Metamorphosis," *SIGGRAPH '92 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1992, pp. 35-42.

5. Blinn, J.F. and Newell, M. E., "Texture and Reflection in Computer Generated Images," *CACM*, Vol. 19, No. 10, October 1976, pp. 542-547.

6. Chang, C., Bishop G. and Lastra A., "LDI Tree: A Hierarchical Representation for Image-Based Rendering," *SIGGRAPH '99 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1999, pp. 291-298.

7. Chen, S. and Williams, L., "View Interpolation for Image Synthesis," *SIGGRAPH '93 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1993, pp. 279-288.

8. Chen, S., "QuickTime VR—An Image-Based Approach to Virtual Environment Navigation," *SIGGRAPH '95 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1995, pp. 29-38.

9. Curless B. and Levoy M., "A Volumetric Method for Building Complex Models forom Range Data." *SIGGRAPH'96 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1996, pp. 303-312.

10. Dally, W.J., McMillan L., Bishop, G. and Fuchs, H., "The Delta Tree: An Object-Centered Approach to Image-

Based Rendering," MIT AI Lab Technical Memo 1604, May 1996.

11. Darsa, L., Silva, B. C. and Varshney, A., "Navigating Static Environments using Image-Space Simplification and Morphing," *Proc 1997 ACM Symposium on Interactive 3D Graphics*, April 1997, pp. 25-34.

12. Davis, T. A. "Generating Computer Animations with Frame Coherence in a Distributed Computing Environment," *ACM Southeast Conference*, April 1998.

13. Dorai C., Wang G., Jain, A.K. and Mercer, C., "From Images to Models: Automatic 3D Object Model Construction from Multiple Views," *Proc. of the 13th IAPR International Conference on Pattern Recognition*, 1996, pp. 770-775.

14. Gortler, S.J., Grzeszczuk, R., Szeliski, R. and Cohen, M. F., "The Lumigraph," *SIGGRAPH '96 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1996, pp. 43-54.

15. Greene, N. and Kass, M., "Approximating Visibility with Environment Maps," IEEE *Computer Graphics and Applications*, Vol. 6, No. 11, November 1986, pp. 21-29.

16. Hoppe, H., "Progressive Meshes." *SIGGRAPH '96 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1996, pp. 99-108.

17. Isaksen, A., McMillian, L. and Gortler, S., "Dynamically Reparameterized Light Fields," *SIGGRAPH '00 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 2000, pp. 297-306.

18. Laveau, S. and Faugeras O., "3-D Scene Representation as a Colletion of Images," *Twelfth International Conference on Pattern Recognition* (ICPR '94), Vol. A, October 1994, pp. 689-691.

19. Le Gall, D., "MPEG: A Video Compression Standard for Multimedia Applications", *Communcations of the ACM*, Vol 34:4, April 1991, pp. 47-58.

20. Levoy, M. and Whitted, T., "The Use of Points As a Display Primitive," UNC-CS Technical Report TR85-022, University of North Carolina, 1985.

21. Levoy, M. and Hanrahan P., "Light Field Rendering," *SIGGRAPH '96 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1996, pp. 31-42.

22. Levoy, M., Pulli, K., Curless, B. et al., "The Digital Michelangelo Project: 3D Scanning of Large Statues." *SIGGRAPH'00 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, July 2000, pp. 131-144.

23. Li, W., Ke, Qi., Huang, X., Zheng, N., "Light Field Rendering of Dynamic Scenes." *Machine Graphics and Vision*, Vol 7, No 3, 1998.

24. Longuet-Higgins, H.C., "A Computer Algorithm for Reconstructing a Scene from Two Projections." *Nature*, Vol. 293, September 1981, pp. 133-135.

25. Manning, R.A., Dyer, C.R, "Interpolating view and scene motion by dynamic view morphing." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 1999, pp. 388-394.

26. Mark, W.R., McMillan, L., Bishop, G., "Post-Rendering 3D Warping," *Proc 1997 ACM Symposium on Interactive 3D Graphics*, April 1997, pp. 7-16.

27. McMillan, L. and Bishop G., "Plenoptic Modeling: An Image-Based Rendering System," *SIGGRAPH'95 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1995, pp. 39-46.

28. Oliveira, M. M. and Bishop, G., "Image-Based Objects," Proc. *of ACM Symposium on Interactive 3D Graphics*, April 1999, pp. 191-198.

29. Rademacher P. and Bishop G., "Multiple-Center-of-Projection Images," SIGGRAPH '98. *Computer Graphics* Proceedings, Annual Conference Series, July 1998, ACM SIGGRAPH, pp. 199-206.

30. Seitz, S. M. and Dyer, C. R., "Physically-Valid View Synthesis by Image Interpolation," *Proc. IEEE Worksop on the Representations of Visual Scenes*, 1995, pp. 18-25.

31. Seitz, S. M. and Dyer, C. R., "View Morphing," *SIGGRAPH '96 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1996, pp. 21-30.

32. Shade, J., Gortler, S., He, L. and Szeliski, R., "Layered Depth Images," *SIGGRAPH '98 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, July 1998, pp. 231-242.

33. Shum, H. and He, L., "Rendering with Concentric Mosaics," *SIGGRAPH '99 Computer Graphics Proceedings, Annual Conference Series*, ACM SIGGRAPH, August 1999, pp. 299-306.

34. Shashua, A., "Algebraic Functions for Recognition", IEEE *Transactions on Pattern Analysis & Machine Intelligence, V*ol. 17, No. 8, August 1995, pp. 779-789.

35. Vedula, S., Baker, S., "Spatio-Temporal View Interpolation." *Proceedings of the 13th ACM Eurographics Workshop on Rendering*, June 2002.

36. Watt A., Watt M., *Advanced Animation and Rendering Techniques Theory and Practice*, Addison-Wesley, 1992.

37. Wolberg, G., "Digital Image Warping," IEEE Computer Society Press, 1990.

38. Wolberg, G., "Image Morphing: A Survey," The *Visual Computer*, Vol. 14, 1998, pp. 360-372.