IEICE TRANS. INF. & SYST., VOL.E86-D, NO.1 JANUARY 2003



PAPER

# Three-Dimensional Scene Walkthrough System Using Multiple Acentric Panorama View (APV) Technique

**Ping-Hsien LIN**[†] *and* **Tong-Yee LEE**[†], *Nonmembers*

**SUMMARY** In this paper, we propose a novel 2D plenoptic function called "acentric panorama view (APV)." This novel 2D plenoptic function samples the panorama scenes without over-sampling or undersampling. A single APV can be accelerated by view culling and list-priority rendering algorithm. Multiple APVs with special fields of view, 45°, 60°, 90°, and 120°, can be integrated into a larger configuration called augmented APVs, which can augment the walking area in a planar walkthrough environment to form a 4D plenoptic function. In augmented APVs, the acceleration schemes of a single APV can still be applied successfully.

***key words:*** *image-based rendering, acentric panorama view (APV), augmented APVs, view culling, list-priority rendering algorithm*

## 1. Introduction

In traditional rendering approaches, the rendering speed is inversely proportional to the complexity of geometric models within the current field of view (FOV). As the complexity of the geometric models grows, the rendering speed drops. In order to keep a good rendering quality, the speed of traditional rendering approach usually cannot satisfy the demand for interaction in a virtual environment. Additionally, this kind of rendering scheme has the nature: it cannot guarantee a constant frame rate so as to preserve the visual smoothness.

To conquer the drawbacks of traditional rendering approaches, image-based rendering (IBR) arose few years ago. The basic idea of IBR is to generate the novel-view image by several nearby reference-view images. Some researchers present the details of their IBR techniques [2]–[4], [6], [11]; others [5], [7]–[10], [12], [13] further provide the frameworks about how to construct the plenoptic function [1], which describes all the radiant energy perceived from the observer's viewpoint. As the taxonomy in [12], we classify the plenoptic functions into six catalogs by the dimension of the plenoptic function as listed in Table 1.

The original plenoptic function is a seven dimensional function, three for the viewing space $(x, y, z)$, two for the viewing direction (azimuth angle $\theta$ and elevation angle $\phi$. This 2D space is called panorama view or environmental map), one for wavelengths $(\lambda)$, and the residual dimension is time $(t)$. Eliminating the two dimensions of wavelengths and time, McMillan and Bishop [5] introduced the 5D plenoptic function to computer graphics society for IBR. They gave the definition for IBR as: "given a set of discrete samples (complete or incomplete) from the plenoptic function, the goal of IBR is to generate a continuous representation of that function." For the general terrain or city walkthrough (not flythrough, this means that the viewing space is a curved surface (2D space)) applications, the plenoptic function reduces to 4D. On the other hand Lightfield [7] and Lumigraph [8] construct the plenoptic function by restraining the viewing space on a cube's surface (2D). Therefore, they are also 4D plenoptic functions. In IBR, there is a fundamental limitation: a reference view should not be used to generate a novel view far away from it for the sake of sampling density. In [12], they propose a 3D plenoptic function called "concentric mosaics." Concentric mosaic is constructed by taking a series of slit images along a circle. Combining multiple concentric mosaics with different radii and sharing a common center, we can have the "concentric mosaics." Though a single concentric mosaic is not a perfect plenoptic function (since all slit images is taken along a circle instead of a fixed point), it plays the same role as a plenoptic function at a specific location (i.e., the center). Thus the radius is the 1D viewing space of that plenoptic function. Due to the sampling limitation, the radius should be small and therefore the observer is restricted in a small circle. Finally, if we further restrain the viewing space in a fixed point, the plenoptic function reduce to 2D panorama view [3], [9].

In this paper, we present a novel 2D plenoptic function called "acentric panorama view (APV)." In Sect. 2.1, we depict how the idea of APV emerges. Like traditional panorama view, the APV doesn't suffer the problems of over-sampling and under-sampling, either. A single APV can be accelerated by view culling and list-priority rendering algorithm. Thus, we can

**Table 1** The taxonomy of plenoptic functions.

| Dim. | Viewing space | Name | Year |
|---|---|---|---|
| 7 | free | plenoptic function [1] | 1991 |
| 5 | free | plenoptic modeling [5] | 1995 |
| 4 | on a cube's | lightfield [7],lumigraph [8] | 1996 |
| 4 | on a plane | augmented APVs | 2002 |
| 3 | inside a small plane | concentric mosaics [12] | 1999 |
| 2 | at a fixed point | panorama | 1994 |

Manuscript received March 27, 2002.
Manuscript revised July 22, 2002.

[†]The authors are with the Computer Science and Information Engineering, National Cheng-Kung University. Tainan, Taiwan, R.O.C.

quickly decide the portion of APV needed to be rendered and then render them without $z$-buffer testing. Multiple APVs with special fields of view (FOV), 45°, 60°, 90°, or 120°, can be integrated into a larger configuration, thus augment the walking area in an environment to form a 4D plenoptic function and the acceleration schemes of a single APV can still be applied successfully.

The remainder of this paper is organized as follows. In Sect. 2, we introduce the concept of APV. The view culling and list-priority rendering algorithm in a single APV is shown in Sect. 3. In Sect. 4, we discuss the configurations of multiple APVs (augmented APVs). Experimental results are given in Sect. 5. Conclusion and future work are given in Sect. 6.

## 2. Acentric Panorama View (APV)

Sampling analysis of the pinhole camera model is indeed a three-dimensional problem. For the reason of simplicity and without loss of generality, we only consider the two-dimension case in a plane (2D) walkthrough environment. In this section, we introduce how the acentric panorama view (APV) arises, the representation of APV, and the sampling of APV. In the following, when we mention the word: "camera," it also means the *depth* image associated with that camera. If we arrange reference cameras arbitrarily, the sampling of these reference cameras would tend to over-sampling (oblique zone in Fig. 1) or under-sampling (zigzag zone in Fig. 1). Over-sampling leads to waste of storage, whereas under-sampling leads to lack of scene information. To avoid over-sampling, we should arrange the reference cameras clockwise or counterclockwise such that the abutting boundaries of FOV of two adjacent cameras are collinear, as shown in Fig. 2. But after arranging a circular series of cameras, the over-sampling and under-sampling still arise.

From another point of view, there are two important applications in image-based rendering: *panorama view* and *object view*. We use Fig. 3 to demonstrate the relation and difference between these two configurations. Both panorama view and object view arise based on the concept of a "circle." In object view, all the COPs of cameras are distributed on a circle, and the viewing direction of each camera points to the center of the circle (Fig. 3 (f)). In panorama view, all the COPs are superposed at the same point, and all the viewing directions point outward from that point (Fig. 3 (a)). But, if we move each camera along its viewing direction with a fixed distance, the COPs of the cameras are also distributed on a circle (Fig. 3 (b)). Comparing Fig. 3 (b) with Fig. 3 (f), we find that the only difference between these two configurations is the viewing direction of each camera. The former points outward and the latter points inward. Between these two configurations, we can rotate each camera with the same angle from panorama view or object view, as shown in Figs. 3 (c), (d), (e).

In traditional panorama view, there is no problem of oversampling or undersampling. But in above figure of Fig. 3 (b), the undersampling arises. If we add additional cameras uniformly to reduce the undersampling area (as shown in the below of Fig. 3 (b)), the oversampling will arise. Here we don't consider the sampling condition inside the circle. This area can be thought as an empty space (i.e., no scene geometry inside this circle). From Fig. 3, we can find that only the traditional panorama view (Fig. 3 (a)) and Fig. 3 (d) have no oversampling or undersampling conditions. The sampling configuration in Fig. 3 (d) satisfies the statement we describe above (the reference cameras are arranged clockwise or counterclockwise such that the abutting boundaries of FOV of two adjacent cameras are collinear, as shown in Fig. 2) and without problem of oversampling or undersampling. We call this sampling configuration (Fig. 3 (d)) the "Acentric Panorama View (APV)."

To express an APV, we need four parameters (Fig. 4). The first parameter is the center of the APV:

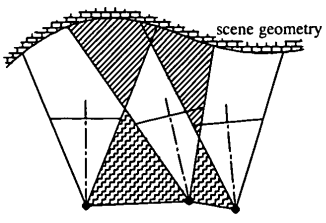


**Fig. 1** Over-sampling (oblique zone) and under-sampling (zigzag zone).

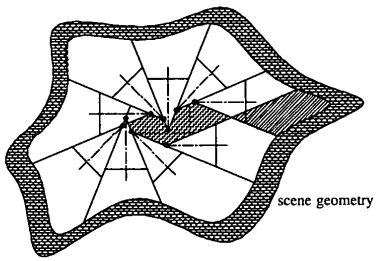


**Fig. 2** After arranging a circular series of cameras, the over-sampling and under-sampling still arise.

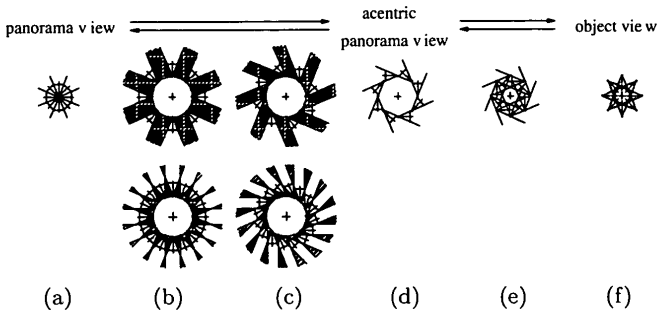


**Fig. 3** Configurations between panorama view and object view.

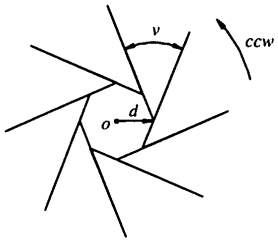


**Fig. 4** An example of counterclockwise APV.

$o$. The second parameter is the offset vector: $d$. The offset vector indicates not only the distance of camera's COP from $o$, but also the starting position of the first camera's COP. The third parameter is the FOV of camera: $v$ ($v$ must divide 360). In other words, the APV is composed of $\frac{360}{v}$ cameras. The last parameter: $c$ indicates whether the configuration is clockwise or counterclockwise. Therefore, an APV is denoted by:

$$APV(o, d, v, c) \tag{1}$$

Additionally, a traditional panorama view can be thought as an APV with zero offset ($d = 0$).

## 3. Acceleration of APV Rendering

Despite of rendering techniques (traditional rendering or image-based rendering), the view culling schemes can accelerate rendering speed dramatically, since it doesn't render the scene primitives we can not see. For example, in Fig. 5, we just need to render the depth image $A$, $B$, $C$, $D$, and $E$. On the other hand, list-priority algorithms can render the scene primitives with correct visibility but without $z$-buffer testing, therefore can accelerate the rendering speed, too. The list-priority algorithm for a single depth image has been introduced by McMillan [4]. These two rendering acceleration techniques are independent and both can be utilized simultaneously under our APV configuration as described in Sects. 3.1 and 3.2 respectively.

### 3.1 View Culling in APV

In this section, we show how to perform view culling inside a counterclockwise APV (i.e., the eye position must be inside the polygon boundaries formed by the camera's COPs of an APV). The view culling in a clockwise APV can be deduced similarly. First, we find the first depth image that can be seen in a counterclockwise order (image $A$ in Fig. 5). For this purpose, we calculate $\vec{r}_{eye} \times \vec{r}_*$ and $\vec{r}_{eye} \times \vec{l}_*$ (* denotes one of $A$ through $I$) for each camera. There is only one camera ($A$) satisfying the rule that $\vec{r}_{eye} \times \vec{r}_* < 0$ and $\vec{r}_{eye} \times \vec{l}_* > 0$. Then, keeping on the counterclockwise order, we find the first pair of two adjacent cameras that locate on each side of the left border of eye ($E$ and $F$). The former ($E$) is in the seen portion and the latter ($F$) is in the unseen portion. Thus, in Fig. 5, only the counterclockwise series

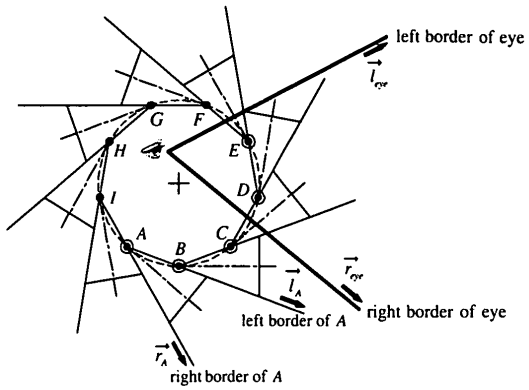


**Fig. 5** View culling of a counterclockwise APV.

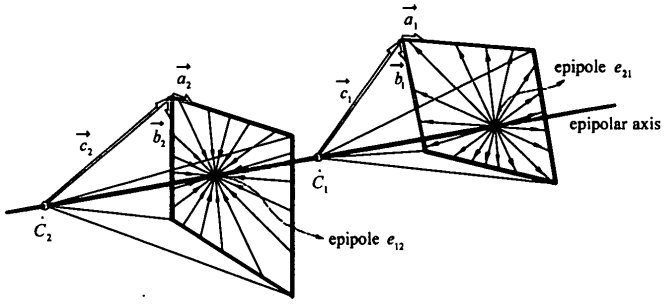


**Fig. 6** Vectors $\vec{a}_i$, $\vec{b}_i$, and $\vec{c}_i$ define an idea pinhole camera $i$ with COP located on $C_i$. The two cameras are reference view and novel view reciprocally. If the camera is the reference one, its image should be warped in the order indicated by the arrows.

from camera $A$ through $E$ are within view frustum.

### 3.2 List-Priority Rendering Algorithm in APV

McMillan's list-priority rendering algorithm [4] comes from the epipolar geometry of two pinhole cameras in computer vision. Figure 6 illustrates the basic idea of this algorithm. If the epipole of reference image is negative (for example, the epipole $e_{21}$ in Fig. 6 is negative), the rendering order should be from epipole toward image borders for saving $z$-buffer testing; otherwise, the rendering order is from image borders toward epipole.

McMillan's list-priority rendering algorithm [4] cannot be applied successfully to multiple depth images acquired from different positions. Though the depth images are taken from different positions in a single APV, we can apply the list-priority rendering algorithm to each depth image within current FOV in a clockwise (for clockwise APV) or counterclockwise (for counterclockwise APV) order, and still keep the visibility correct. We take the counterclockwise APV for example (Fig. 5). It should be emphasized that the viewer must be constrained inside the polygon boundaries formed by the camera's COPs of an APV. Since the APV has no over-sampling, it partitions the space into several disjoint regions. Additionally, in a counterclockwise APV, the right of two adjacent depth images within viewer's FOV is always farther than the left. Therefore, in the

case of Fig. 5, rendering the depth images in the order: *A-B-C-D-E* (i.e., a counterclockwise order), and applying the list-priority rendering algorithm to each depth image, we can still have correct visibility. Similarly, in a clockwise APV, the visible depth images should be rendered in a clockwise order.

## 4. Augmented APVs

Multiple APVs with special FOV (45°, 60°, 90°, or 120°) can be integrated to cover a larger walking area. We call this configuration augmented APVs. There are seven possible configurations of augmented APVs as shown in Fig. 7. In the configurations (b), (d), (e), (f), and (g), we just need to place the counterclockwise APVs (or just clockwise APVs), the clockwise (or counterclockwise) APVs are provided automatically by the adjacent APVs. In the following, we don't discuss the configurations (d) and (f), because they are too sparse to construct the plenoptic functions everywhere.

Now we just consider the configurations (b), (e), and (g). The configurations (e) and (g) are equivalent to place the traditional panorama views in a square-tessellated form and triangle-tessellated form respectively (the configuration (g) is equivalent to (b)). The multiple APVs offer another choice: configuration (b).

Under configuration (b) in Fig. 7, we can accelerate rendering speed by using the view culling and list-priority rendering algorithm depicted in Sect. 3. In configuration (b), there are two kinds of regions in the walking area: the hexagonal region and the triangular region. When the viewer is inside the hexagonal region and the center is left to the viewer as shown in Fig. 8 (a), using the counterclockwise APV to generate the view is better than using the clockwise APV. Because the sampling angle and position of the counterclockwise APV are more close to that viewer than that of the clockwise APV. To the contrary, if the center is on the right side of the viewer as shown in Fig. 8 (b), we should adopt the clockwise APV.

If the viewer is inside the triangular region, we classify the six cameras (*A* through *F* in Fig. 9 (a)) into interior cameras (*D*, *E*, and *F*) and exterior cameras (*A*, *B*, and *C*). We first find out the interior camera (*E*) whose viewing direction is the closest to that of the viewer. In Fig. 9 (c), the interior camera *E* is always chosen when the viewing direction of the viewer is within the marked 120 degrees). Then we calculate $\vec{r}_{eye} \times \vec{r}_E$, and $\vec{l}_{eye} \times \vec{l}_E$. If $\vec{r}_{eye} \times \vec{r}_E < 0$ and $\vec{l}_{eye} \times \vec{l}_E > 0$, it must be the case as Fig. 9(d). Here we assume that the FOV of the viewer is less than 60 degrees. This assumption is reasonable in most of the cases. Thus we just need to render the depth image *E*. Otherwise, if $\vec{r}_{eye} \times \vec{r}_E > 0$ and $\vec{l}_{eye} \times \vec{l}_E > 0$, it must be the case as Fig. 9 (e) or (f). If the COP of camera *A* is inside the FOV of the viewer (Fig. 9 (e)), we need to render the depths images *A*, *E*, and *F*. In this configuration, the

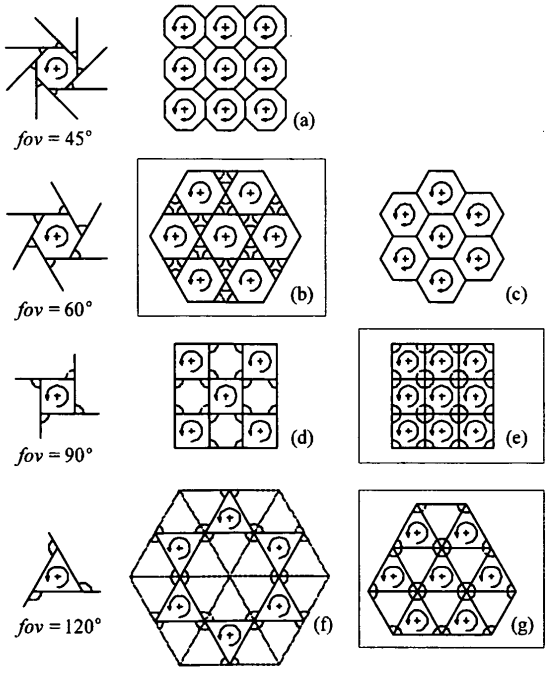


**Fig. 7** Use simplified symbol to express the seven kinds of augmented APVs. The large FOVs in the third row (90°) and the fourth row (120°) can be constructed by placing more than one camera in each FOV. The boundaries of FOVs in subfigure (a) and (c) are omitted for concise expression.

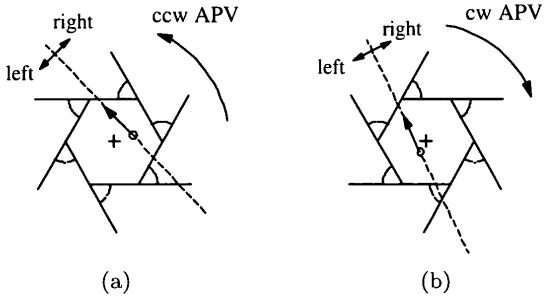


**Fig. 8** The rendering order in the hexagonal region.

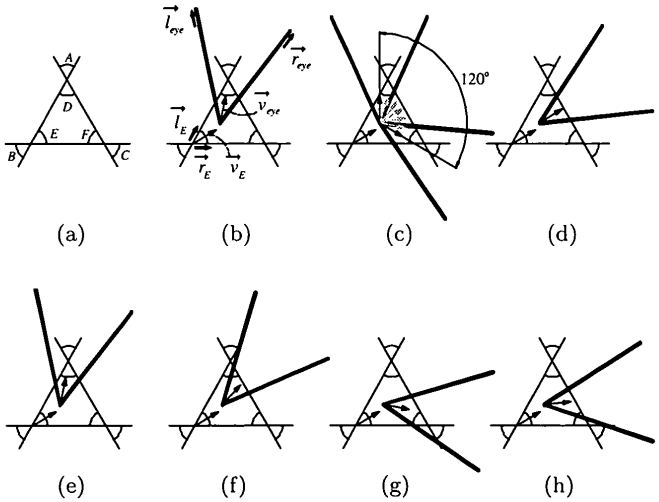


**Fig. 9** The rendering order in the triangular region. The oblique region stands for the scenes culled out.

```
Premise: the FOV of viewer is less than 60°
         the triangular region is empty
         (each depth image is rendered using [4])
         (refer to Fig. 9 for the camera indexes)
find out the interior camera whose viewing direction is closest
to that of the viewer.
(assume camera E is chosen in this step as the case in Fig. 9.)
if r_eye × r_E < 0 and l_eye × l_E > 0
   render depth image E
else if r_eye × r_E > 0 and l_eye × l_E > 0
   if the COP of camera A within FOV of the viewer
      render depth images in order of A-E-F (or A-F-E)
   else
      render depth images in order of A-E
else
   if the COP of camera C within FOV of the viewer
      render depth images in order of C-D-E (or C-E-D)
   else
      render depth images in order of C-E
```

**Fig. 10** Pseudocode of view culling in the triangular region.

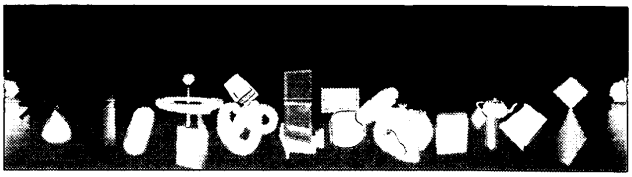

**Fig. 11** An example of counterclockwise APV.

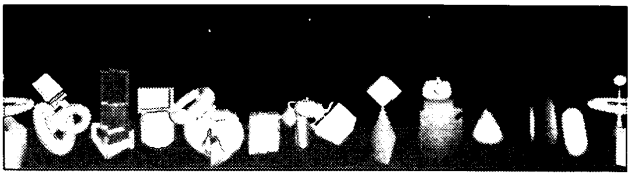

**Fig. 12** An example of clockwise APV.

depth image $A$ is farther from the viewer than $E$ and $F$. Moreover, the projections of the depth images $E$ and $F$ on the viewer's image plane have no union, hence the rendering order of depth images $E$ and $F$ doesn't make any difference to the viewer. Therefore, the rendering order in this case should be $A$-$E$-$F$ or $A$-$F$-$E$. Similarly, if in case of Fig. 9 (f), we should render the depth images $A$ and $E$ in order of $A$-$E$. Otherwise ($\vec{r}_{eye} \times \vec{r}_E < 0$ and $\vec{l}_{eye} \times \vec{l}_E < 0$), it must be the case as Fig. 9(g) or (h). If the COP of camera $C$ is inside the FOV of the viewer (Fig. 9 (g)), we should render the depth images $C$, $D$, and $E$ in order of $C$-$D$-$E$ or $C$-$E$-$D$. In the case as Fig. 9 (h), we should render depth images $C$ and $E$ in order of $C$-$E$. The list-priority rendering algorithm [4] can be applied to each depth image in the above rendering order to avoid the $z$-buffer testing. The pseudocode of this algorithm is given below (Fig. 10). When camera $D$ or $F$ is chosen in the first step, the view culling procedure can be deduced analogically.

In general, all regions in the walking area shouldn't be empty. In this case, we can first render the current region's APV, then render the current region's geometries using geometry-based rendering technique. Since the current region is the closet region to the viewer, it has significant parallax effect. Therefore, in our design, we render these local geometries using standard pipeline instead of the proposed method.

## 5. Experimental Results

Figures 11 and 12 are two examples of counterclockwise APV and clockwise APV, respectively, of Fig. 13. The discontinuity of an APV image is due to the different sampling position.

Figure 13 shows the views generated by polygon-based rendering, traditional panorama view, acentric panorama view, and the combination of traditional and acentric panorama view. The acentric panorama view in this figure is just formed by one clockwise APV and one counterclockwise APV (the star symbol in the

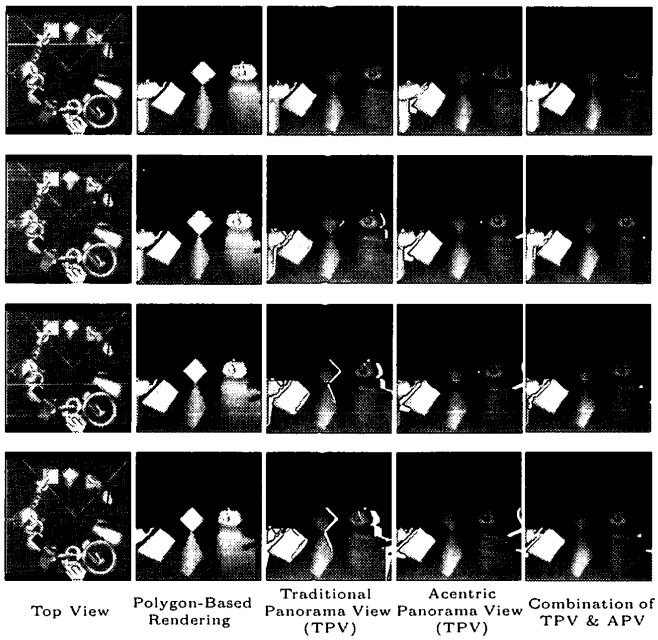


**Fig. 13** Demonstration of visibility in different framworks. The image sequence in each column is captured from the center to the left gradually as shown in "Top View" column. The star symbol at the center on the ground is the combined configuration of a clockwise APV and a counterclockwise APV.

"Top View" column in Fig. 13). To better understand the sampling condition, we don't use any holes filling technique in Fig. 13. Inspecting columns 3 (traditional panorama view) and 4 (acentric panorama view), we can know each configuration has its sampling superiority over the other. In this example, it is obvious that the quality of the views generated by the combination of traditional and acentric panorama view is better (i.e., with the least number of holes and gaps) than that generated by each alone. In Fig. 13, there are still few holes and gaps in the 5th column. This is because we just use one clockwise APV and one counterclockwise APV. To solve this sampling deficiency problem, we can improve the visibility of each depth image using the structure of LDI [11] to improve the sampling of APV.

We should note that it is not easy to see actual

rendering quality from the still images like Fig. 13. We experimentally tested the proposed scheme to walkthrough a large virtual environment that consists of multiple APVS. Please visit our web site at http://couger.csie.ncku.edu.tw/~vr/MAPV_demo.html and there is an animation about this experiment.

Finally, we should comment the complexity of the proposed method. In general, each region (including hexagonal and triangular region) in the walking area consists of many polygons. When a viewer is in a specific region, we will render the background scenes by the proposed method. Then, we render these polygons using the standard graphics pipeline. Therefore, the complexity of the proposed method is bounded by the number of polygons in this region and the number of depth images rendered by the proposed method. The number of depth images is always less than three in the proposed method (Figs. 8 and 9). To limit the number of polygons rendered in this specific region, practically, we divide the whole scene into many APVs with requiring the number of polygons less than a selected number. Therefore, we can control the fame rate for rendering a virtual environment in this way.

## 6. Conclusion and Future Work

In this paper, we propose a novel 2D plenoptic function: acentric panorama view (APV). A single APV has no problem with over-sampling and under-sampling. Its rendering speed can be accelerated by view culling and list-priority rendering algorithm. Additionally, it has the potential to be constructed into a 4D plenoptic function (augmented APVs). Our experimental results show the proposed technique can be useful to walk through a virtual scene. Our approach is a hybrid technique to render 3D scenes. When a viewer is located in a region of APV, the polygons in this region are rendered by the standard geometry-based rendering. For other polygons, we use multiple APV rendering. Currently, there is a problem in the proposed method. For a large scene, we need to divide the whole scene into many APV configurations. This requires pre-rendering many images and therefore, we need a large database. We plan to analyze the redundancy of database shared among APVs and try to reduce the amount of storage. In addition, to generate a virtual environment from real scenes is another hot topic. In near future, we plan to apply this technique to the real scene.

## Acknowledgement

The authors would like to thank the National Science Council, R.O.C. under Grant No. NSC 91-2213-E-006-077 for the partially support of this work.

## References

[1] E.H. Adelson and J.R. Bergen, The Plenoptic Function and the Elements of Early Vision, chapter 1, MIT Press, 1991.
[2] S.E. Chen and L. Williams, "View interpolation for image synthesis," Proc. SIGGRAPH '93, pp.270–288, 1993.
[3] S.E. Chen, "QuickTime VR — An image-based approach to virtual environment navigation," Proc. SIGGRAPH '95, pp.29–39, 1995.
[4] L. McMillan, "A list-priority rendering algorithm for redisplaying projected surfaces," Technical Report, 95-005, University of North Carolina, 1995.
[5] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," Proc. SIGGRAPH '95, pp.39–46, 1995.
[6] S.M. Seitz and C.R. Dyer, "View morphing," Proc. SIGGRAPH '96, pp.21–30, 1996.
[7] M. Levoy and P. Hanrahan, "Light field rendering," Proc. SIGGRAPH '96, pp.31–42, 1996.
[8] S.J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen, "The lumigraph," Proc. SIGGRAPH '96, pp.43–54, 1996.
[9] R. Szeliski and H. Shum, "Creating full view panoramic image mosaics and environment maps," Proc. SIGGRAPH '97, p.251–258, Aug. 1997.
[10] P. Rademacher and G. Bishop, "Multiple-center-of-projection images," Proc. SIGGRAPH '98, p.199–206, July 1998.
[11] J.Shade, S.Gortler, L.He, and R.Szeliski, "Layered depth images," Proc. SIGGRAPH '98, 231–242, 1998.
[12] H. Shum and L.-W. He, "Rendering with concentric mosaics," Proc. SIGGRAPH '99, pp.299–306, 1999.
[13] M. Oliveira and G. Bishop, "Image-based objects," Proc. 1999 ACM Symposium on Interactive 3D Graphics, pp.191–198, 1999.

**Ping-Hsien Lin** received the B.S in mechanical engineering and M.S in computer engineering from National Cheng-Kung University, Taiwan, R.O.C., in 1993 and 1998, respectively. Now, he is working toward his Ph.D degree at Department of Computer Science and Information Engineering, National Cheng-Kung University. Mr. Lin research interests include computer graphics, image processing, virtual reality, visualization and image-based rendering.

**Tong-Yee Lee** received his B.S in computer engineering from Tatung Institute of Technology in Taipei, Taiwan, in 1988, his M. S. in computer engineering from National Taiwan University in 1990, and his Ph.D in computer engineering from Washington State University, Pullman, in May 1995. Now, he is an Associate Professor in the department of Computer Science and Information Engineering at National Cheng-Kung University in Tainan, Taiwan, R.O.C. His research interests include computer graphics, visualization, and virtual reality.