

Video Object Coding and Rendering for Panoramic Video

Kin-Lok Chan and Shing-Chow Chan
Department of Electrical and Electronic Engineering,
The University of Hong Kong.
kinlok@graduate.hku.hk, scchan@eee.hku.hk

ABSTRACT

Panoramic video is a simple representation of a dynamic scene or static environment. It allows the users to change their viewpoints interactively in time or along a given path in space. For better users' experience, it is desirable to incorporate synthetic or real objects into a panoramic video. This paper studies the coding and rendering of these objects in panoramic video using the concept of video objects developed in the MPEG-4 standard. Experimental results showed that better user experience are achieved by incorporating these time varying objects into a static walkthrough application, only at the expenses of slightly more bandwidth and decoding complexity.

1. INTRODUCTION

With increasing demand for better user experience in interactive applications such as virtual walkthrough, computer games, and medical simulation, virtual reality techniques are becoming more and more important. Image based rendering (IBR) using the plenoptic function [4],[5] has recently emerged as a simple yet powerful photo-realistic representation of real world scenes. The basic principle is to render new views of a scene using rays that were previously captured in densely sampled pictures taken from the scene. One simple example of IBR is the 2D panorama. It can be constructed from a set of images taken by rotating the camera along a given axis. The images are then projected onto a cylinder or sphere and are stitched together to create a panoramic image. During rendering, part of the panoramic image is re-projected onto the screen to emulate the effect of panning and zooming. Other more sophisticated representations include the lightfield [9], lumigraph [8], and concentric mosaic [7]. Most of the IBR representations proposed focus on static environment or scenes. This is largely attributed to the logistic difficulties in capturing and transmitting these dynamic representations, which unavoidable involves huge amount of data. More recently, panoramic video (PV) which consists of a series of panoramas taken at regular time interval along a given path, was proposed as a representation for dynamic environment. They find useful applications in tele-presence and autonomous vehicles [13-15]. Another application of PV is to provide seamless walkthrough by constraining the camera location to a predefined path in a static scene. Much effort was placed on the construction and rendering of panoramic video [15,16]. The compression and transmission aspects of PV were also discussed in [3,12].

For virtual walkthrough applications such as virtual electronic shopping, it is sometimes desirable to offer better user's experience by incorporating synthetic or real objects into the panoramic video. This problem, which belongs to the general problem of augmented reality, requires the extraction, coding, processing and rendering of these objects. In principle, different views of these objects should be rendered from a 3D model or from other IBR representations, given their positions in the 3D space where the panorama is captured, before they can be rendering

together with the panorama. In addition, the panorama should also possess depth information to emulate the effect of occlusion. For best quality, lighting and shadowing effects of the object should also be taken into account. This motivates us to study the methods for incorporating these special effects and the necessary functionalities that we have to incorporate in the MPEG-2-based PV compression algorithm in [3,12]. For simplicity of discussion and practical advantage of fast rendering speed, we assume that we have limited number of views of the objects taken at different angles of viewing. For objects distance away from the viewing position, warping a single view of this object is usually sufficient. For example, in Fig. 3, we only have one view of a cartoon dinosaur to be rendered with the panoramic video frame. Each view of this object over time therefore constitutes as a video object as in the conventional MPEG-4 video compression standard [6]. The shadow, which depends on the lighting condition, can also be modelled by a series of video objects and rendered according to the given lighting condition and viewing angle. This is similar in concept to the view dependent texture in computer graphic. In this paper, detail coding and rendering of this video object in panoramic video will be given. Experimental results showed that better user experience are achieved by incorporating these time varying objects into a static walkthrough application, only at the expenses of slightly more bandwidth and decoding complexity.

The rest of the paper is organized as follows: in section 2, a brief description of panoramic video is given. The concept of video objects coding in panoramic video is then introduced in Section 3. Section 4 is devoted to the rendering of these video objects. Finally, conclusion is given in Section 5.

2. PANORAMIC VIDEO

A panoramic mosaic is a high-resolution image obtained by projecting a series of images, taken by rotating the camera along a given axis, onto a cylindrical or spherical surface. Because it is obtained by stitching several images together, the resolution is usually rather high (say for example 2048×768). Fig. 2 shows an example of a panorama of a synthetic scene, where the panorama has been remapped onto the six surfaces of a cube. On the other hand, a time-varying environmental map is obtained by taking panoramas along a trajectory in a static environment or at regular time interval either at a given location or along a certain path. If cubic projection is employed, we have 6 image sequences corresponding to the six surface of the cube. Such time-varying environmental map or panoramic video closely resembles a video sequence with very high resolution and can be compressed using a method similar to the MPEG-2 algorithm [3,12]. Since PV offers 360 degrees of viewing freedom, they very useful to provide virtual walkthrough application say in a static environment. For simplicity, we shall assume in the following that PV is stored using a cubic projection.

3. VIDEO OBJECT CODING IN PV

As mentioned earlier, the incorporation of synthetic or real objects into the panoramic video involves the extraction, coding, processing and rendering of these objects. First of all, we need to specify the trajectory of the object in the 3D space where the panorama is captured and the events or user interaction that they will be invoked. For example, a bird may start flying from east to west inside the PV “village” when the user reach some point in the path specified by the PV. After that, the bird should keep flying in that direction in the 3D space, according to some reasonable velocity, no matter where the viewer’s viewing angle is. In principle, we need different views of the bird during rendering because the user might be looking at different angles of the bird resulting from changing his/her viewpoint in the panorama. As mentioned earlier, it is assumed that the object is far enough from the view point that a limited number of views of the object taken at different angles of viewing are sufficient to emulate this special effect. Each view of this object over time therefore constitutes as a video object as in the conventional MPEG-4 video compression standard [6]. The shadow, which depends on the lighting condition, can also be modeled by a series of video objects and rendered according to the given lighting condition and viewing angle.

During rendering, the video object with angle best matching the current viewing angle of the user will be selected for rendering. According to the current position of the object, which is already defined in the trajectory of the object, the current video object plane (VOP) in the VO will be selected and re-projected onto the same cube as the panorama. Each pixel of the panorama is assumed to have a depth value measured from the centre of the cube (i.e. the viewing position). The region of the panorama that overlaps with the projection of the current VOPs will be rendered using the painter’s algorithm. That is, the one with the largest depth value will be rendered first followed by the next largest value and so on. Therefore, occlusion among the VOPs and the panorama can be animated. For synthetic scene, the depth value of each pixel in the panorama can be easy obtained during the generation of the panorama. It can then be recorded in stored in the PV for later use. For real scene, special devices have to be used to record the approximate depth values or they can be calculated offline from the panorama. For simplicity, we assume in this paper that the scene is synthetic and their depth values are already available.

Let’s consider a simple example in Fig. 3 where part of a panoramic video frame (3 faces of the cube) of the PV “village” is shown. The images used to generate the environment map or PV were rendered using 3D Studio Max®, and each panoramic video frame is stored by cubic projection (Fig. 2). Suppose that we want to place an object, say a cartoon dinosaur in Fig. 3, into this panoramic video frame. This object can either be extracted from an existing video or generated by some other means. The compression of this variable size object by the MPEG4 standard will be described later. Since the 6 images of the cube are just projection of the panorama, the video object cannot be pasted directly onto it. Instead, the video object is assumed to be a planar image in the 3D space having a certain depth and it has to be projected onto the same cube. In this particular example, the dinosaur will be projected onto the three images shown, Fig. 4. It is assumed that the dinosaur is closer to the viewing position than the

background pixels in the panorama. Therefore, the background pixels are occluded as illustrated in Fig. 4. In general, the video objects will be rendered using the Painter’s algorithm in descending order of their depth. For simplicity, we do not consider the effects of lighting and the shadow, although they can also be animated using more video objects or even real-time rendering of the object using 3D models. We now described the coding of the video objects using the MPEG-4 coding standard.

While former MPEG standards, that is, MPEG-1 and MPEG-2, mostly concerned with compression of videos, MPEG-4 provides additional functionalities such as content-based representation, bit-rate scalability, intellectual property management and protection, etc. In order to enable the content based interactive functionalities, the MPEG-4 Video Verification Model (VM) introduces the concept of Video Object Planes (VOP’s). In each frame of a video sequence, it is assumed that the frame is a combination of several segmented images region. VOPs are regions of image or video contents, obtained say through segmentation, which can have a different shape and location from frame to frame. Successive VOP’s belonging to the same physical object in a scene are treated as Video Objects (VO’s), which is a sequence of VOP’s of possibly arbitrary shape and location. During transmission, all the shape, texture, motion information of the video objects will be encoded and transmitted or maybe separately coded into several VOL (Video Object Layer). The compression algorithm used in MPEG-4 VM is based on the successful block-based hybrid DPCM/Transform coding technique taking into account the variable size nature of the VOPs. The first VOP will be encoded in Inter-Frame VOP coding mode (I-VOP). Subsequent frames can be coded using Inter-frame VOP prediction (P-VOP’s) or bidirectionally predicted VOP’s (B-VOP’s). The compression and decoding processes, DCT, Motion estimation, motion compensation, are similar to former MPEG standards.

In our testing system, Microsoft developed MPEG-4 module was used to compress the video objects, while the panoramic video is coded using the modified MPEG-2 algorithm in [3-12]. All video objects and the panoramic video are separately coded and decoded during the projection process. Table1 shows the coding results of the VO’s: “Green Bird”, “Baby Dinosaur” and “Fish”, used in subsequent simulation.

	Compression ratio (to raw)	S/N ratio
Green Bird	100:0.77	26.2 dB
Baby Dinosaur	100:3.43	25.3 dB
Fish	100:0.78	35.8 dB

Table1. Coding results of the VO’s used in the simulation.

4. RENDERING VIDEO OBJECTS IN PV

Given a VOP and its depth from the view point, the function of the rendering is to project it onto the panoramic mosaic after cubic projection. Fig. 1 illustrates this projection process for a mosaic image plane. For cubic projection, the GOP can in general be projected onto three different mosaic images as shown in 3. Therefore, it is required to determine the area of the mosaic image where the VOP will be projected. After that, the VOP will be projected onto the appropriate mosaic image through wrapping the appropriate proportion of the VOP.

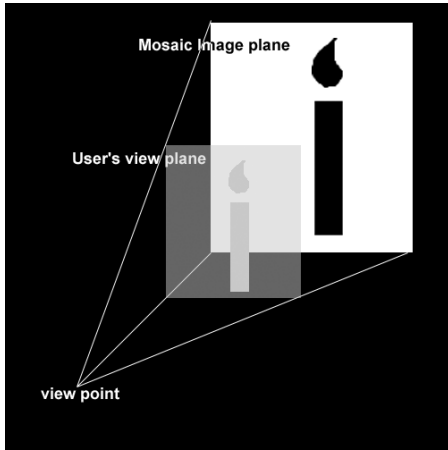


Figure 1. A candle was projected onto Mosaic Image. While viewing, the candle was mapped back to the view plane, so user may sense that the video objects is part of the mosaic image. (This is similar to the concept of imposter and sprites in 3D-model rendering).

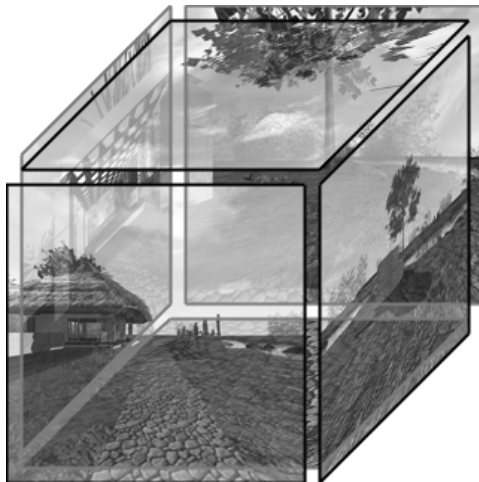


Figure 2. A panoramic video frame of the PV "village" using cubic projection.

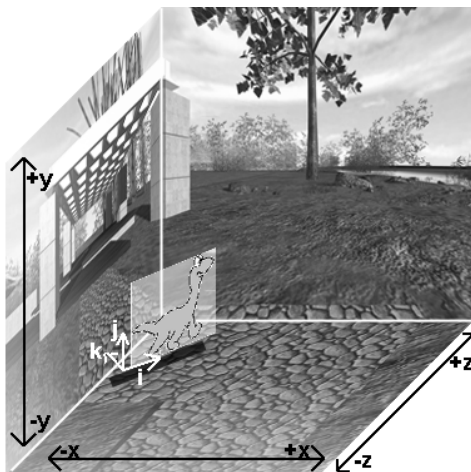


Figure 3. Rendering of a cartoon dinosaur inside the virtual space.

4.1. Area of projection

The virtual cubic space is represented in cartesian coordinates (x,y,z) (Figure 3).

The area of projection is determined as follows:

Step 1

Projects the four corners of the video objects onto the mosaic image. Boundary boxes are drawn to include all projected corners in each tile. If the projections of the four corners do not lie on the same mosaic image, step 2 is used to determine the individual areas in the mosaic image.

Step 2

In this step, 12 boundary lines along the cubic area will be traced. We will then record the minimum x , minimum y , minimum z , maximum x , maximum y , and maximum z coordinates that came across with video objects. Combining these results with first step's result, projection area on six tiles could be found, Fig. 4.

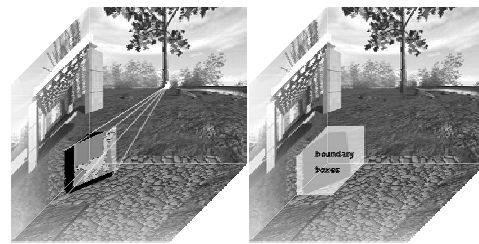


Figure 4. Projection Area and the boundary boxes.

4.2. Projection Process

After the area of projections has been determined, all the pixels inside the VOP will be mapped to the corresponding mosaic. A ray will be shoot from each pixel in the projection area towards the center. The cross point of this ray with the VOP is the pixel to be projected onto the original mosaic pixel, Fig. 5.

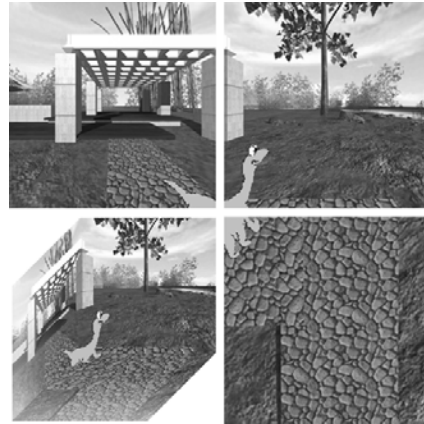


Figure 4. The mapping result of previous example

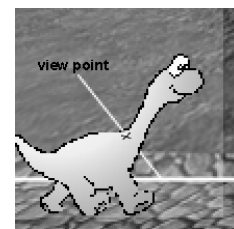


Figure 5. The cross point (marked with a cross) of a ray in the projection area and an VOP.

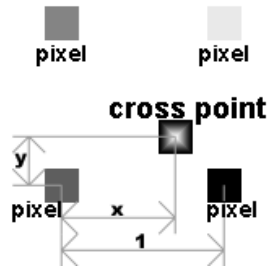


Figure 6. Cross points lies between pixels.



Figure 7. Rendered view of VO "dinosaur".

Since the cross point might not lie on an integer grid, 2-D interpolation [2] is used to generate the color information of the cross point. With the addition of the alpha channel information, the color information came out from the 2-D interpolation will be apply to the ray-emitting pixel. For some cases, the ray may pass through more than one VOP at the same time. At this time, the rendering process will be done using the painter's algorithm, starting from the farthest to the nearest VOPs. The mosaic images are therefore ready to be rendered by standard hardware or software. Fig. 7 shows an example of rendered views when the dinosaur is moving in from left to right. Fig. 8 shows another rendered views of the VO "bird" in the sky of the PV "village".



Figure 8. Rendered view of VO "bird".

5. CONCLUSION

This paper presents methods for coding and rendering of real-world or synthetic objects in panoramic video using the concept of video objects developed in the MPEG-4 standard. Experimental results showed that better user experience are achieved by incorporating these time varying objects into a static walkthrough application, only at the expenses of slightly more bandwidth and decoding complexity.

REFERENCES

- [1] Alan Watt. 3D Computer Graphics Third Edition. 2000
- [2] G. Wolberg. Digital Image Warping. 1990.
- [3] K. T. Ng, S. C. Chan and H. Y. Shum, "The Compression issues of Panoramic Video," in *Proc. Int'l Symp. on Intelligent Multimedia, Video and Speech Processing 2001*, pp. 36-39, May 2-4, 2001.
- [4] E. H. Adelson and J. Bergen, "The plenoptic function and the elements of early vision," in *Computational Models of Visual Processing*, pages 3-20. MIT Press, Cambridge, MA, 1991.
- [5] L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system," in *Computer Graphics (SIGGRAPH'95)*, pp.39-46.
- [6] J. Ostermann, E. Jang, J.S. Shin, and T. Chen, "Coding of Arbitrarily Shaped Video Object in MPEG-4." in *Proc. IEEE ICIP '97*.
- [7] H. Y. Shum, K. T. Ng, and S. C. Chan, "Virtual reality using the concentric mosaic: construction, rendering and data compression," in *Proc. IEEE ICIP2000*, Vol. 3, pp. 644-647, Sept. 10-13, 2000.
- [8] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph," in *Computer Graphics (SIGGRAPH'96)*, pp. 43-54.
- [9] M. Levoy and P. Hanrahan, "Light field rendering," in *Computer Graphics (SIGGRAPH'96)*, pp. 31-42, August 1996.
- [10] L. McMillan and G. Bishop, "Plenoptic modelling, "An image-based rendering system," in *Computer Graphics (SIGGRAPH'95)*, pp.39-46.
- [11] S. Peleg and J. Herman, "Panoramic mosaics by manifold projection" in *Proc. IEEE CVPR'97*, pp. 338-343, June 1997.
- [12] King-To Ng, Shing-Chow Chan, Heung-Yeung Shum, and Sing-Bing Kang, "On the data compression and transmission aspects of panoramic video", in *Proc. IEEE ICIP'2001*, Vol. 2, pp. 105-108, Oct. 7-10, 2001.
- [13] Jonathan Foote and Don Kimber, "FlyCam: Practical Panoramic Video and Automatic Camera Control", in *Proc. IEEE International Conference on Multimedia and Expo*, 2000, vol. 3., pp. 1419-1422.
- [14] J. Baldwin, A. Basu and H. Zhang, "Panoramic video with predictive windows for telepresence applications", in *IEEE International Conference on Robotics and Automation*, 1999, vol.3 pp. 1922-1927.
- [15] R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, 16(2):22-30, March 1996.
- [16] R. Szeliski and H.-Y. Shum, "Creating full view panoramic image mosaics and texture-mapped models," in *Computer Graphics (SIGGRAPH'97)*, pp. 251-258, August 1997.
- [17] S. E. Chen, "QuickTime VR - an image-based approach to virtual environment navigation," in *Computer Graphics (SIGGRAPH'95)*, pp. 29-38, August 1995.