VIEW GENERATION FOR FTV IN CIRCULAR CAMERA ARRANGEMENT

Takeshi Uemori, Tomohiro Yendo, Toshiaki Fujii and Masayuki Tanimoto

Graduate School of Engineering Nagoya University

ABSTRACT

There are many researches about the methods to generate free viewpoint image in linear camera arrangement, but it is rare in circular camera arrangement. In this paper, we propose a novel method to generate free viewpoint image for FTV in circular camera arrangement. This method is based on the Ray-Space method, which is one of the Image Based Rendering (IBR) techniques. We can generate arbitrary views by interpolating ray data of the Ray-Space that is constructed by collecting all images of a sequence captured by cameras arranged circularly but not enough in number. Moreover, by analyzing the trajectory of a ray in the Ray-Space, our method can be applied to the case using perspective cameras or the case that the object is not at the center of the circle or it is not small enough compared to the circle.

Index Terms— FTV, free viewpoint image, Ray-Space, circular camera arrangement

1. INTRODUCTION

We have proposed the FTV (Free Viewpoint Television) [1]. FTV is a television that enables us to watch from any viewpoint. So, if FTV is realized, we can move in the space of the television freely. In this paper, we introduce a novel method to generate free viewpoint image for FTV.

There are mainly two different approaches to generate arbitrary viewpoint images. One is the Model Based Rendering (MBR) and this method consists of creating a 3D Computer Graphics (CG) model of the object inside the image. However, it is difficult to make a graphic model close to the real object. As a result, the created image appears to be artificial and unnatural. On the contrary, there is another method called Image Based Rendering (IBR) [2, 3, 4], which is not related to the geometrical characteristics of the object. The Ray-Space method is one of the IBR approaches and this method records the position and direction of rays that are transmitted in the space. We try to put FTV into practice by the Ray-Space method [1].

There are mainly two types of the capturing systems we propose for FTV: the linear camera arrangement and the circular camera arrangement. The circular camera arrangement is more freely to watch scenes than the linear camera arrangement. However, there are fewer researches about the circular camera arrangement because of the complexity of interpolation techniques. Once before, we have proposed the method to generate free viewpoint image in circular camera arrangement. In that method, we have regarded captured images as orthographic images to simplify the interpolation process though we have used perspective cameras. That method therefore can be applied to the only scene case that the object is at the center of the circle and it is small enough compared to the circle. With the new method we proposed in this paper, we can generate a free viewpoint image in every situation from perspective images.

Takeshi Uemori:e-mail uemori@tanimoto.nuee.nagoya-u.ac.jp

2. RAY-SPACE METHOD

2.1. Concept of Ray-Space method

The Ray-Space method is one of IBR methods to generate arbitrary views. In this method, we represent 3D space by using ray information transmitted in space without consideration of the object's geometry. Here, we regard a camera as a device not to take a photo but to acquire rays and an image as ray data. The block of ray data, we call it Ray-Space, can be constructed by collecting all images of a sequence (see Fig.1). If the Ray-Space is dense enough, we can generate arbitrary views only by loading ray data from the Ray-Space. For real-world multi-camera setup, however, dense sampling is very difficult and costly. A setup for large field would require an immense amount of cameras to sample rays according to these requirement. Therefore, it is necessary to render arbitrary views from the Ray-Space that is not dense enough using interpolation techniques.

2.2. Feature of Ray-Space

The Ray-Space has some interesting features. Fig.1 is a horizontal section image of the Ray-Space constructed of orthographic images in circular camera arrangement. As you can see sine-curves in Fig.1, a ray radiated from a 3D point draws a sine-curve by its trajectory on a horizontal section image. Therefore, in this case, we can find corresponding points between cameras by searching on one plane along a sine-curve. However, these ideal sine-curves are shown only when orthographic images are used. Feldmann et al. show in the liter-

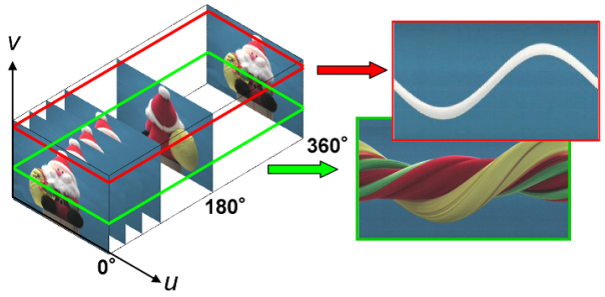


Fig. 1. Horizontal Section Image of Ray Space

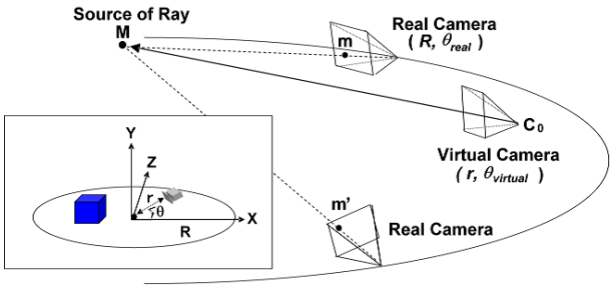


Fig. 2. Ray Tracing

ature [5] that if perspective cameras with focal length F are used, due to the depth dependency in projection, ideal sine-curves are slightly distorted. Additionally, the trajectory of a ray radiated from a 3D point is no longer restricted to lie on one horizontal plane in the Ray-Space.

3. CONVENTIONAL METHOD

One of conventional methods is to regard a circle as a regular polygon [6]. That is to say, in this method, an arc of inter camera is regarded as a straight line, and we can treat as the case of the linear camera arrangement. However, in case of the angular space between cameras is broad, the accuracy of interpolation is low. In such a case, it is necessary to interpolate ray data as a circle. There is another interpolation method in circular camera arrangement we have proposed once before. This method enables us to utilize the feature of a sine-curve on a horizontal section plane by regarding captured perspective images as orthographic images. If the scene condition is that the object is at the center of the circle and it is small enough compared to the circle, we can regard captured perspective images as orthographic images in circular camera arrangement. However, these scenes are very limited and this method is not efficient.

4. PROPOSED METHOD

The proposed method can be applied to the case using perspective cameras instead of orthographic cameras or the case that the object is not at the center of the circle or it is not small enough compared to the circle. This method focuses on how a ray radiated from a 3D point draws its trajectory in the Ray-Space constructed of perspective images.

Now, this process has the following three steps. First, we compute the trajectory of a ray in the Ray-Space by changing the depth, or the distance between a focal point of camera and a 3D point. Second, we determine which trajectories (that are representing depth values) are correct by the projective block matching and smoothing the depth. Third, we interpolate the pixel by blending corresponding points with consideration of occlusion. The details are described in the sections below.

4.1. Analysis of Trajectory

If we use perspective cameras, the trajectory of a ray in the Ray-Space is computed by following steps. Consider the case of Fig.2. We first project a focused pixel(u, v) on the image plane of a virtual camera located at $(r, \theta_{virtual})$ into the 3D space as a ray. Then, we re-project one point on the ray, $M = [X, Y, Z]$, into each image plane of real cameras located at (R, θ_{real}). M is given by next equation with arbitrary depth t:

$$M = C_0 + td \tag{1}$$

where C_0 is the coordinate of the virtual camera, and d is the direction vector of a ray given by

$$C_0 = \begin{bmatrix} r\cos\theta_{virtual} \\ h \\ r\sin\theta_{virtual} \end{bmatrix} \tag{2}$$

$$d = \begin{bmatrix} -\frac{F\cos\theta_{virtual} + u\sin\theta_{virtual}}{k} \\ -\frac{v}{k} \\ -\frac{F\sin\theta_{virtual} - u\cos\theta_{virtual}}{k} \end{bmatrix} \tag{3}$$

$$k = \sqrt{u^2 + v^2 + F^2} \tag{4}$$

where h is the Y-coordinate of the camera and F is the focal length of the camera.

Then, the re-projected point(u_{real}, v_{real}) on the image plane of a real camera is given by

$$\begin{cases} u_{real} = \frac{X\sin\theta_{real} - Z\cos\theta_{real}}{X\cos\theta_{real} + Z\sin\theta_{real} - R}F \\ v_{real} = \frac{Y - h}{X\cos\theta_{real} + Z\sin\theta_{real} - R}F \end{cases} \tag{5}$$

Equation (5) shows that the ideal sine-curve is slightly distorted and the trajectory of a ray is not lying on one horizontal plane in the Ray-Space when we use perspective cameras.

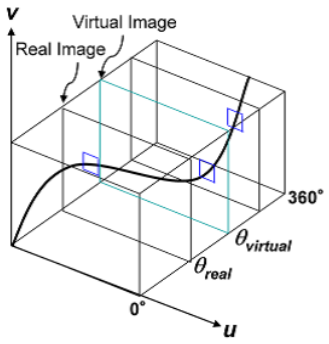


Fig. 3. Trajectory of Ray

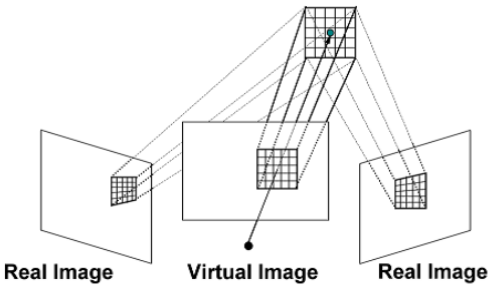


Fig. 4. Projective Block Matching

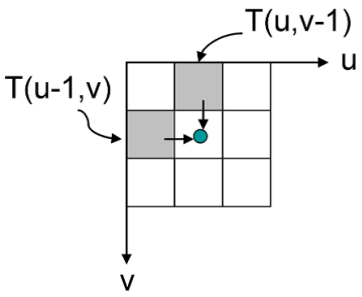


Fig. 5. Smoothing of Depth

4.2. Projective Block Matching

We can obtain some trajectories with arbitrary depth t from equation (5). Thus we determine which trajectories are correct by searching for the re-projected pixel-to-pixel correspondence between real images (see Fig.3). However, it is difficult to get correct correspondence by pixel matching due to noise and texture redundancy. Therefore, we need to use block matching technique. However, in circular camera arrangement, the projected shape of the object is variant according to the direction which the object is captured from. Moreover, if the object is not at the center of the circle, scales of the object inside images are variant. Then if we use commonly used block matching technique and put a same square window of block matching on two or more images, each of pixels contained in windows do not always correspond with each other even when each focused pixel is correspondent. Thus, in this paper, we use the projective block matching. The projective block matching is the method as described below (see Fig.4). First, we create a square window on the virtual image plane. Second, we project the window into the 3D space according to the depth t. Third, we re-project the projected window in the 3D space into the image planes of each real camera. At last, we calculate the SAD (Sum of Absolute Differences) between windows of each real camera, which is given by

$$E_{12} = \sum_{i=-N}^{N} \sum_{j=-M}^{M} \left| I_1(u_1+j, v_1+i) - I_2(u_2+j, v_2+i) \right| \quad (6)$$

where $(2N+1)$ x $(2M+1)$ is the window size, (u_n, v_n) is the focused pixel of the camera(n) and I_n is the value of the pixel.

Now, we can interpolate one view from at least two images of nearest cameras. However, in consideration of occlusion, four images of near cameras are needed. As we can tell from Fig.6, the corresponding points can be found certainly in one of three pairs, regardless of occluded regions. Therefore, we interpolate one view from four images of near cameras and define the error in the projective block matching as E_P, which is the sum of three pairs given by

$$E_P = E_{12} + E_{23} + E_{34} \quad (7)$$

E_{12}: Error function between camera1 and camera2
E_{23}: Error function between camera2 and camera3
E_{34}: Error function between camera3 and camera4

4.3. Smoothing of Depth

We can reduce some mismatching by smoothing the depth. In general, the surface of an object changes smoothly. That is, the depth, or the distance between the virtual camera and the 3D point on an object, changes smoothly.

Now, we smooth the depth value t from two directions of the left pixel$(u-1, v)$ and the upper pixel$(u, v-1)$ of the focused pixel(u, v) (see Fig.5). Therefore, we define the error of differences between each depth as E_D, which is given by

$$E_D = \frac{1}{2}\left[\left| T(u-1, v) - t(u, v) \right| + \left| T(u, v-1) - t(u, v) \right| \right] \quad (8)$$

where T is the depth value that has already been determined and t is determined as the result of smoothing.

To sum up the matter, we determine the correct depth value t which minimizes E_{TOTAL}, the sum of the projective block matching error E_P and the smoothing depth error E_D, given by

$$E_{TOTAL} = E_P + \lambda E_D \quad (9)$$

where λ is a weighting factor and it is determined in consideration of the balance between the projective block matching error E_p and the smoothing depth error E_D. If the value is too small, smoothing effect is unavailable and if it is too big, mismatching occurs in boundary regions.

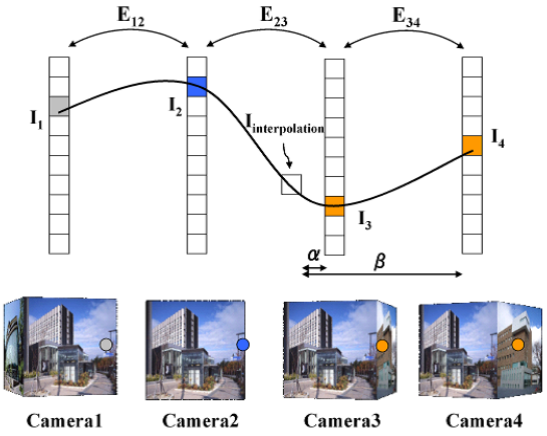
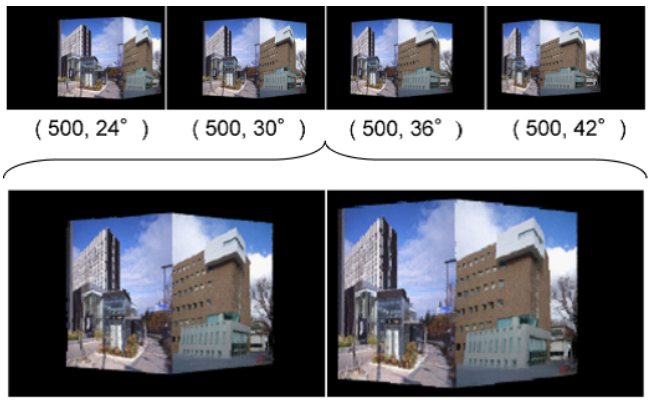


Fig. 6. Interpolation from four images



(a) Generated View$(500, 33°)$ (b) Generated View$(450, 33°)$

Fig. 7. Generated View Results

4.4. Interpolation considering occlusion

We described in section4.2, for handling of occlusion, it is necessary to interpolate the pixel from four images of near cameras.

Now therefore we summarize our interpolation algorithm here. First, we compute the trajectory of a ray in the Ray-Space by changing the depth. Then, the correct trajectory is determined by the projective block matching and smoothing the depth. At this time, the correct trajectory has the depth value t that minimizes the error E_{TOTAL} defined by equation (9). Finally, we determine the value of the pixel from the corresponding points along the following lines. We select a pair of the corresponding points with minimum projective block matching error among E_{12}, E_{23} and E_{34}. Then we determine the value of the pixel by blending the corresponding points according to the angular distance between the virtual camera and each real camera. For example, in case of Fig.6, we determine the value of the pixel as below.

$$I_{interpolation} = \frac{\beta I_3 + \alpha I_4}{\alpha + \beta} \tag{10}$$

In this way, we interpolate the pixel and generate a free viewpoint image.

5. EXPERIMENTAL RESULTS

We experimented to test the effectiveness of the proposed algorithm on synthetic image sequence captured under the following conditions: 60 cameras were arranged at every 6 degrees on the circle with a radius of 500 centimeters, and a cube, 60 centimeters on a side, pasted photographs on the surfaces was located at the point 150 centimeters on the X axis from the center of the circle. The window size for the projective block matching was set to 11 x 11 and the weighting factor for smoothing of depth λ was set to 20.

Fig.7(a) is the result of generated view when we set the virtual camera at the point$(500, 33°)$ and the PSNR comparison to the originally image captured by real camera at the same point shows 34.65dB. Fig.7(b) is the generated view when we moved the virtual camera to the point$(450, 33°)$ from the point of Fig.7(a) and the PSNR shows 32.37dB.

6. CONCLUSION AND FUTURE WORK

In this paper, we have presented the method of generating free viewpoint images in circular camera arrangement. By analyzing the trajectory of a ray in the Ray-Space, the proposed method can be applied to the case using perspective cameras. In our experimental result, we set the virtual camera at user's choice and generated the image from that viewpoint successfully. However, this method cannot render these images in real time at present. Therefore, to realize FTV system, we need to improve the rendering speed. Furthermore, as future works, we will conduct experiment on real image sequence instead of synthetic one and show the efficiency of the proposed method.

7. REFERENCES

[1] T. Fujii, M. Tanimoto: "Free-viewpoint Television based on the Ray-Space representation", SPIE IT-Com'02, pp.175-189, Aug.2002.

[2] S. E. Chen, L. Williams: "View Interpolation for Image Synthesis", Proc. ACM SIGGRAPH'93, pp. 279-288, 1993.

[3] M. Levoy, P. Hanrahan: "Light Field Rendering", Proc. ACM SIGGRAPH'96, pp. 31-42, 1996.

[4] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M.F. Cohen: "The Lumigraph", Proc. ACM SIGGRAPH'96, pp. 43-54, 1996.

[5] I. Feldmann, P. Eisert, and P. Kauff: "Extension of epipolar image analysis to circular camera movements", Proc. ICIP'03, pp. 697-700, Sept.2003.

[6] N. Fukushima, T. Yendo, T. Fujii, and M.Tanimoto: "Free Viewpoint Image Generation Synchronized With Free Listening-Point Audio For 3-D Real Space Navigation", Proc. 3DTV-Conderence'07, May.2007.