Regular Paper

Real-time free viewpoint rendering via viewport range driven polygon plane arrangement

Keisuke Nonaka[1,a)] Ryosuke Watanabe[1] Jun Chen[1] Sei Naito[1]

Abstract: Free viewpoint technologies that synthesize virtual viewpoint by using multiple actual videos would provide immersive experiences for users. To realize this concept, many conventional methods have been proposed. However, these methods require high computational cost to synthesize a virtual viewpoint because they have to calculate huge data to express three-dimensional information. To overcome this problem, we propose a simple and fast free viewpoint synthesis method based on a visual hull, which is a general concept in this field. We calculate the silhouette of an object along view-dependent planes in virtual space by simple projection from images to the 3D space, and express the whole shape of the object by integrating the planes. This scheme works very quickly while providing fine quality because it consists of standard functions in general GPU architecture. The experimental results show our method can generate a fine virtual view of an object by using multiple videos in real time.

Keywords: free-viewpoint, visual hull, voxel model, plane-based model, real-time processing

1. Introduction

In recent years, free-viewpoint navigation systems from a multi camera environment [1, 2] have been among the hottest topics in computer vision, and such systems are being reported with increasing frequency. In the navigation system, users can select their viewpoint freely (not limited to actual camera positions), and the selected scene is synthesized by using multi-camera videos and several additional items of information. This makes the free-viewpoint system very useful for improving the user's understanding of a scene, and creates an immersive and ultra-realistic user experience, especially when viewing sports.

A lot of methods [3–10] have been proposed to realize the free-viewpoint system and we can organize these method into three approaches. A "image based method [3, 4]" propose a method to construct light field based on real optical properties captured by multiple images. This approach has an advantage that can synthesis natural and fine virtual view, but it require so many cameras or specific equipment like light field camera. In addition, it has a limitation that the range of virtual viewpoint is very narrow and computational cost is high. Other approach called as "depth based method [5]" uses depth value obtained by some sensors to calculate the structure of target 3D space and it realizes wide range virtual viewpoint. However, even in this approach, there is a strict requirement regarding shooting environment, e.g. use of depth sensor, and the computational cost is also high.

In addition, the "model-based method [6–10]" is also a well-known approach for creating a free-viewpoint. This approach constructs a three dimensional computer graphics (3DCG) model that represents the actual target object shape and texture by using information from sparsely arranged cameras, using fewer cameras compared with other approaches [4, 5]. The selection of the viewpoint is more flexible compared to the selection in other methods. Considering these features, the approach is suitable for commercial use.

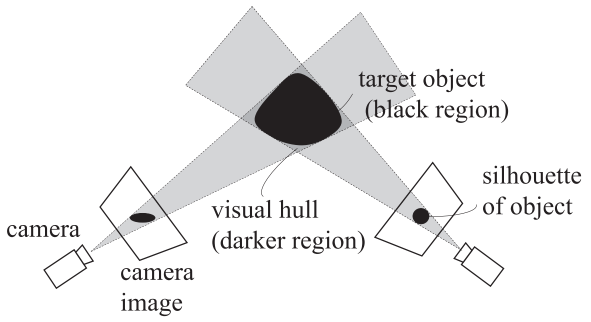

Fig. 1: Concept of visual hull.

A model-based method called "visual hull [6–10]" calculates the shape of an object by using the intersection of the silhouette obtained by multiple camera images (Fig. 1). Generally, a visual hull is represented as a set of many points by using voxels that are rectangular parallelepiped. However, to represent the fine detail of a complicated object (like a human), it is necessary to use many voxels and calculate which voxels belong to the object. This requires huge calculation cost for generating the data of a 3D model for free-viewpoint and it causes a time delay from input to synthesis. Considering the use of live streaming of free-viewpoint, the calculation time is a serious limitation, and we have to generate at least one frame data per 33.3 milliseconds.

To overcome this limitation, we propose a fast calculation approach to represent the shape of the object based on the visual hull

[1] KDDI Research, Inc., Saitama 356–8502, Japan
[a)] ki-nonaka@kddi-research.jp

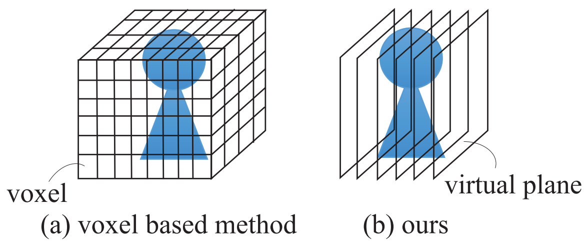

Fig. 2: Differences between two approaches.

concept. Differing from the conventional method, our method uses a virtual plane as a unit of 3D space and the visual hull can be calculated by using a simple projection scheme implemented in a graphical processing unit (GPU). This scheme leads to acceleration of the calculation. The experimental result shows that our method can generate almost same free-viewpoint synthesis compared with the conventional method and it works in real-time, less than 33.3 milliseconds per one frame.

2. Proposed method

2.1 Concept and Overview

Our method is based on the concept of a visual hull, which is a general scheme for obtaining the shape of a shooting object. A visual hull generates a 3DCG model of the target object by calculating the intersection of a projected silhouette cone of the object which is obtained by multiple camera images (Fig. 1).

The intersection is calculated by using voxels, rectangular parallelepiped, as units of 3D space (Fig. 2 (a)). On the other hand, changing the point of view of the visual hull construction, our method uses a virtual plane instead of voxels to express the shape of the object. Based on the plane, the intersection can be calculated by using the simple projection scheme proposed in this paper, and this feature contributes to accelerating the calculation of the visual hull. Fig. 2 shows an overview of the framework of our method.

Our method is composed of five parts, "Data acquisition," "Mask extraction," "Virtual plane setting,""Alpha projection" and "Texture projection." In the following section, we describe the five procedures in detail and how to implement them in GPU for synthesizing a virtual viewpoint.

2.2 Shooting Environment and Data Acquisition

In our method, we assume that the object is surrounded by K cameras, each camera position is fixed and the camera parameters are known. The time synchronization between all cameras is adjusted in advance. The camera image is represented as $S_i (i \in \{1, \ldots, K\})$. Throughout this paper, we discuss how to synthesize a visual hull from a virtual viewpoint in only one frame, but our method can easily be applied to all video sequences because the procedure is independent frame by frame.

2.3 Mask Extraction

Being similar to the conventional voxel method [10], we use a binary mask image of the target object to know the shape of the object in an image. Therefore, we generate silhouettes of the object by using a conventional background subtraction (BS) method [11]. The BS method outputs binary mask image corresponding to the silhouette of the object, executing the subtraction between the background model image and the current frame image. Let $M_i (i \in \{1, \ldots, K\})$ be the binary mask image generated by i-th camera image. The pixel value included in the object is 1, and the other is 0 in this case.

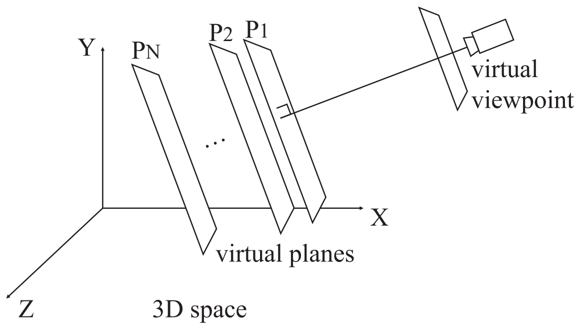

Fig. 3: Virtual plane setting.

2.4 Virtual Plane Setting

Then, we set a square virtual plane $P_n (n \in \{1, \ldots, N\})$ as a unit of the model, into 3DCG space. Where N is the number of the planes in the target space. By using a homography matrix H_i, which is given by the pre-calculated camera parameters, M_i can be easily projected onto the plane as follows,

$$P_n(x, y, z) \propto H_i M_i(u, v, 1), \tag{1}$$

where the (u, v) are the coordinates in M_i and the (x, y, z) are the coordinates in P_n. Considering the projection for all the mask images $M_1, M_2, \ldots, M_K$, the intersection of all the mask P_n^I can be calculated as,

$$P_n^I(x, y, z) = \prod_{\forall (u,v) \in M_i} P_n(x, y, z). \tag{2}$$

Since the P_n^I is the intersection of all the object silhouettes in the camera image, we consider it as a sectional view of the object along plane P_n. In other words, if we can stack the plane for a sufficient number, we can represent the actual shape of the object.

In addition, if we can use an infinite number of planes in the target 3D space, the planes can be set at arbitrary positions and angles. However, considering the usage of hardware memory, a finite number of planes should be set at the optimal position according to the virtual viewpoint keeping the quality of the synthesis. Our method sets the planes as if they face toward the virtual viewpoint perpendicularly (Fig. 3). This solution avoids an artifact caused by the distance between the planes to the extent.

2.5 Alpha Projection

In an actual case, we do not use the binary value of the mask image because the binary silhouette representation is too strict considering the noise of the mask image. Instead of the binary value, to retain the ambiguity of the shape representation, we set an alpha value A_i related to all the pixels in the mask image as follows,

$$A_i(u,v) = \begin{cases} 1 & M_i(u,v) = 1 \\ \alpha & otherwise. \end{cases} \tag{3}$$

α works to dilate the shape of the object and we set $\alpha = 0.01$ heuristically. This alpha value is projected onto planes P_n^I in the same way as Eq. 2.

The integrated alpha value on P_n^I is used as the actual opacity of the projected texture described in Section 2.6. Besides, by summing the value for all planes P_n along an optical axis of the virtual viewpoint, the virtual view can obtain the shape (opacity) of the target object.

2.6 Texture Projection

To render a virtual viewpoint, we have to know the color of the target object. For this purpose, we use texture mapping of the camera image similar to alpha projection. However, we use only Q number of camera images near the virtual viewpoint as follows,

$$P_n^T(x,y,z) = \prod_{\forall(u,v)\in S_q} \lambda_q P_n(x,y,z), \tag{4}$$

where index q represents the index corresponding to the near Q cameras and λ_q is the weight of texture alpha blending. $P_n^T(x,y,z)$ represents the bended texture on P_n by the Q cameras. By iterating this procedure for all planes, we can render the virtual viewpoint. λ_q is calculated according to distance $d(q)$ between the virtual viewpoint and each camera as,

$$\lambda_q = \frac{\sum_{x\in Q'\backslash q} d(x)}{(Q-1)\sum_{x\in Q'} d(x)}, \tag{5}$$

where, Q' returns a set of index of Q cameras, and we set $Q = 2$ to blend just two camera textures in this paper.

2.7 Implementation for Rendering

Below we briefly describe how to implement the above procedure. Basically, we use programmable shader, the general GPU function, for this implementation. Firstly, we calculate the array of all the corner points of P_n and enter them into a vertex shader. Then, the corresponding (u,v) in all the images and the virtual viewpoint can be calculated. After that, all the pixel values (including alpha) of the viewpoint can be easily obtained by the alpha and the texture projection which is executed in a fragment shader. That works very quickly because the pixel value is independent for the neighboring pixel values and the current GPU is optimized for the pixel-wise procedure.

3. Experimental results

To examine the performance of our method, we synthesized images by the conventional method [10] and our method, and the results are shown in Fig. 5. For both methods, we used 16 full HD cameras surrounding in a semicircular position (Fig. 4) and the results are also rendered in full HD. Only for the background subtraction, we resized the input image to 640×360 [pixels] to accelerate it, because in our implementation, the procedure is implemented for a CPU. In addition, we used 1.0 cm as the distance between each voxel in the conventional method and as the distance between each virtual plane in our method. This means the capability of expressing an object shape by both methods is almost the same, theoretically. Under these settings, the number of voxels to be calculated was approximately 1.5×10^8, and the number of virtual planes was 8.0×10^2. The background 3DCG model in this experiment (the baseball stadium) was constructed in advance, and the results shown in Fig. 5 were cropped to enhance the difference between the two methods.

Table 1: Objective evaluation of two methods.

	Voxel	Ours
SSIM (Virtual viewpoint 3)	**0.686**	0.684
SSIM (Average score of 10 viewpoints)	**0.670**	0.689
run time [milliseconds/frame]	1.96×10^2	$\mathbf{3.17 \times 10}$ (< **33.3**)

In Fig. 5, Virtual Viewpoint 1 is a viewpoint in the center of two cameras (the 4th and the 5th camera from the left in Fig. 4), and Virtual Viewpoint 2 is a viewpoint that is placed at a higher position far from all the cameras. Virtual Viewpoint 3 is a viewpoint with a camera position, but we did not use the corresponding camera for the result (we used 15 cameras). This result was for the objective evaluation described below.

From the qualitative viewpoint of Fig. 5, our method generated similar results compared with the conventional method. In some cases, the voxel based method had a jaggy noise along the boundary of the player because the extracted mask had an error. On the other hand, along the boundary, our method did not have a jaggy noise but had a burred texture caused by the layered plane and the alpha projection. To clarify this finding, we cropped a red rectangular region around the bat, and placed an enlarged one in the top right corner in Fig. 5 (c).

In addition, we examined the quality of the synthesized images by objective score. The most important feature of our method is its ability to preserve both the player's shape and structure in the synthesis process. To verify this feature, we used a well-known structure evaluation method related to human perception, Structural SIMilarity (SSIM) [12]. The input synthesized image for this evaluation is the same as that shown in Fig. 5 (c) and we calculate the SSIM for the luminance between the ground truth (a camera image) and each method. We chose 10 viewpoints which consists of virtual viewpoint 3 in Fig. 5 and randomly selected 9 camera positions as virtual viewpoints in this objective experiment. The camera images on the virtual viewpoints were not used for the synthesis but the viewpoints were the same. A higher value is better in SSIM, and the range is from 0 to 1. The results of SSIM are shown in Table 1 and the score shows our method can generate almost the same result, although the score is slightly degraded.

In addition, we evaluated the calculation time of the methods. Table 2 shows the specifications of the desktop PC used for this evaluation and we developed some viewers implemented by C++ with OpenGL [13] to measure the time from image input to rendering. The time includes the calculation time of all procedures, e.g. mask extraction.

From the third line of Table 1, the result shows that our method outperforms the conventional voxel method from the viewpoint of calculation time and can be adopted for the real-time streaming system in 30 [frames/second]. This acceleration is caused by

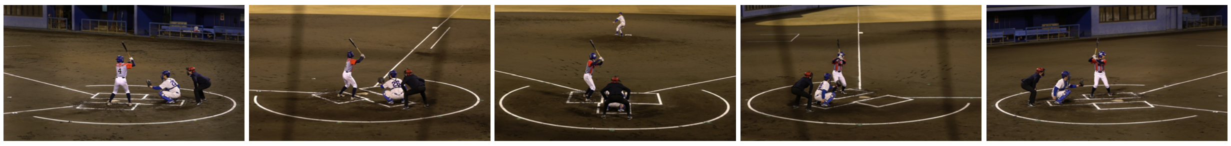
Fig. 4: List of input images (in this figure, we selected only 5 camera images from 16 cameras due to the limitation of space).

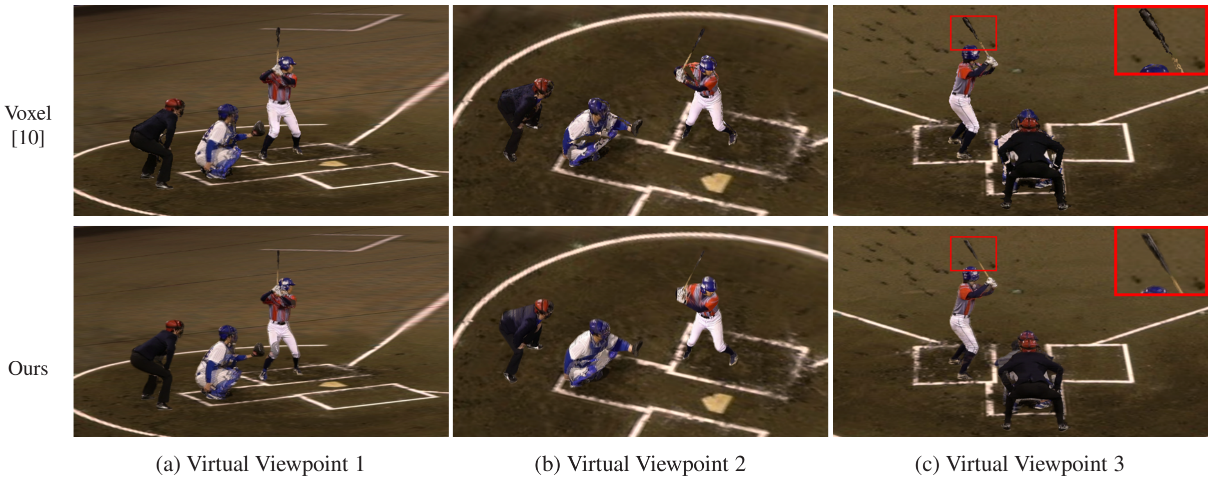

Fig. 5: Synthesized images by the conventional and our method.

Table 2: Specifications of devices in this experiment.

	desktop PC
OS	Windows 10 64bit
CPU	Intel Core i7-6700K CPU @4.00GHz
GPU	NVIDIA GeForce GTX 1080
RAM	32GB

the fact that our method does not calculate the 3D position of the silhouette intersection explicitly and is implemented by fundamental functions in a general computer graphic library such as OpenGL which is optimized for hardware.

Of course, the quality of both methods is significantly affected by the quality of mask extraction of the target object. For the sake of keeping the real-time rendering, currently we use a very simple BG scheme executed in less than 33.3 [milliseconds/frame]. This means it is not so easy to apply our method to complicated scene rendering.

4. Conclusion

We propose a fast plane-based free-viewpoint synthesis technology for real-time live streaming based on the concept of a visual hull. Differing from the conventional voxel based visual hull, our method sets a virtual plane as a unit of 3DCG space, and projects silhouette masks and textures of a target object given by multiple camera images on the plane. The projected silhouettes and the textures are easily utilized to obtain the intersection of the silhouettes and it becomes one of the expressions of the visual hull. In the synthesis scheme, a vertex shader and a fragment shader can execute the projection quickly within GPU hardware in parallel.

In our experiments, the results show that our method can generate a similar result compared with the conventional voxel method, even in terms of objective evaluation. In addition, we confirmed our method can synthesize the virtual viewpoint in over 30 fps from input to rendering. This means our method can be easily applied to real-time live streaming.

Currently, we assume that the background of an actual scene is not so complicated because we use a simple BS method to keep the real-time processing. In future, we will propose a fast BS method that can be applied to complicated scenes and apply our method to cases of actual use.

References

[1] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, “Free-viewpoint tv,” *IEEE Signal Processing Magazine*, vol. 28, no. 1, pp. 67–76, 2011.
[2] R. Suenaga, K. Suzuki, T. Tezuka, M. P. Tehrani, K. Takahashi, and T. Fujii, “A practical implementation of free viewpoint video system for soccer games,” in *Proceedings of SPIE Electronic Imaging, 3D Image Processing, Measurement and Applications*, 2015, p. 93930G.
[3] A. Ishikawa, M. P. Tehrani, S. Naito, S. Sakazawa, and A. Koike, “Free viewpoint video generation for walk-through experience using image-based rendering,” in *Proceedings of ACM International Conference on Multimedia*, 2008, pp. 1007–1008.
[4] C. Lipski, C. Linz, K. Berger, A. Sellent, and M. A. Magnor, “Virtual video camera: Image-based viewpoint navigation through space and time,” *Computer Graphics Forum*, vol. 29, no. 8, pp. 2555–2568, 2010.
[5] Z. Cui, J. Gu, B. Shi, P. Tan, and J. Kautz, “Polarimetric multi-view stereo,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 369–378.
[6] C. Buehler, W. Matusik, L. McMillan, and S. Gortler, “Creating and rendering image-based visual hulls,” Massachusetts Institute of Technology, Cambridge, MA, USA, Technical Report, 1999.
[7] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, “Image-based visual hulls,” in *Proceedings of Annual Conference on Computer Graphics and Interactive Techniques*, 2000, pp. 369–374.
[8] Y. Furukawa and J. Ponce, “Carved visual hulls for image-based modeling,” *International Journal of Computer Vision*, vol. 81, no. 1, pp. 53–67, 2009.
[9] V. Chari, A. Agrawal, Y. Taguchi, and S. Ramalingam, “Convex bricks: A new primitive for visual hull modeling and reconstruction,” in *Proceedings of IEEE International Conference on Robotics and Automation*, 2012, pp. 770–777.
[10] H. Sankoh, S. Naito, K. Nonaka, H. Sabirin, and J. Chen, “Robust billboard-based, free-viewpoint video synthesis algorithm to overcome occlusions under challenging outdoor sport scenes,” in *Proceedings of the 26th ACM International Conference on Multimedia*, 2018,

pp. 1724–1732.

[11] A. M. Elgammal, D. Harwood, and L. S. Davis, “Non-parametric model for background subtraction,” in *Proceedings of the 6th European Conference on Computer Vision-Part II*, 2000, pp. 751–767.

[12] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[13] (2018) OpenGL. [Online]. Available: https://www.opengl.org