Real-Time View Synthesis System Using Active Illumination and Adaptive Space-Time Cost-Volume Filtering

Sho Nishimura, Shunsuke Yamada, Keita Takahashi and Toshiaki Fujii
Nagoya University, Aichi, Japan
E-mail: s.yamada@fujii.nuee.nagoya-u.ac.jp Tel/Fax: +81-052-789-3163

***Abstract*—Our target is development of a view synthesis system that includes the entire process from capturing of multi-view videos to synthesize virtual view in real-time. Depth estimation of the target scene is indispensable for view synthesis from multi-view videos. In this paper, we improved the depth estimation method we had developed in a previous work, where an active illumination technique was combined with an efficient layer based algorithm. More specifically, we proposed an adaptive space-time filtering for the cost volumes constructed for depth estimation. The adaptive space-time filtering adapts its shape for each pixel automatically according for depth estimation, resulting in higher quality of the depth estimation especially in dynamic scenes. Our method was tested on a system consisting of 16 video cameras and a Digital Light Processing (DLP) projector to show its effectiveness. We achieved higher quality depth estimation, resulting in higher quality virtual view synthesis with a nearly real-time frame rate.**

I. Introduction

Virtual view synthesis from multi-view videos has been an attractive research area during recent years. We aim at development of a view synthesis system that can accomplish the entire process from image capturing to view synthesis in real-time and generate high quality virtual views. Our target applications include 3-D live broadcasting and 3-D teleconferencing systems where real-time processing from input to output is an essential requirement.

View synthesis from multi-view images requires depth estimation of the target scene. Not only accuracy but also speed and efficiency are required for the depth estimation if dynamic scenes should be processed in real time. As a promising approach, view-dependent depth estimation, where depth values are estimated directly for a virtual view to synthesize, has been introduced into view synthesis systems [1–8]. This approach estimates only the necessary information for the virtual view to synthesize, leading to high efficiency. Since the obtained depth map is unique to each viewpoint, depth estimation is repeated for each viewpoint. We adopted a view-dependent approach with depth layers [2], [7], [8] because it is suitable for our aim.

As for the accuracy of depth estimation, much has remained to be improved. When we use a passive method (using only cameras), the estimated depth values are prone to be unstable especially in the regions including texture-less objects, resulting in low quality virtual views. Meanwhile, active illumination can help depth estimation because with projected random patterns texture-less regions are made to be texture rich, resulting in improved depth estimation. Moreover, by using time varying illuminating patterns, the scene is characterized also along time. It has been proven that using several temporal frames simultaneously for stereo matching (space-time filtering on cost volumes) leads to more stable depth estimation [9], [10].

Mori et al. [8] adopted active illumination and space-time matching for real-time view synthesis. However, their space-time matching was limited to two sequential time frames, which resulted in limited depth accuracy. In this paper, we extend their idea to more than two temporal frames in order to further improve the accuracy of depth estimation. Moreover, we propose an adaptive space-time filtering where the filter's shape is adapted for each pixel automatically according to the motion of the pixel. We implemented our method on a view synthesis system consisting of 16 cameras and a DLP projector, where the entire processing from image capturing to view synthesis can be performed in real time.

II. Proposed Method

A. Overview

We assume that the system consists of multiple video cameras and a DLP projector. The cameras are arranged on a 2-D grid that roughly lie on a plane, and all the cameras are calibrated and synchronized. The projector does not need to be calibrated against the cameras because it is used only for attaching pattens to the scene objects. The projection range should almost overlap the field of views of the cameras.

The projector casts spatially randomized 2-D binary patterns generated by M-sequence random numbers, and the patterns vary over time. These patterns are helpful for stabilizing the depth estimation but should be eliminated from the synthesized view. Therefore, we insert a blank pattern with a uniform luminance alternatively along time as shown in Fig. 1. We use only the videos with random patterns for depth estimation but only the videos with the blank pattern for view synthesis. Therefore, we treat two sequential random and blank patterns as a unit with the same time. This makes view synthesis intervals twice, but we have to accept it as long as we use active illumination with visible light.

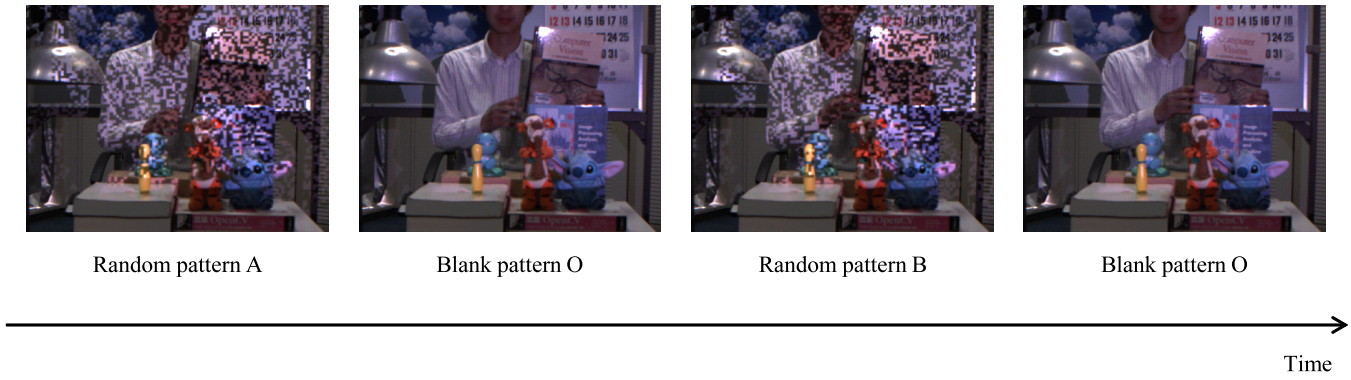

Fig. 1. Videos with active illumination. A and B are different random patterns and O is the blank pattern.

B. Depth estimation and view synthesis

The configuration is illustrated in Fig. 2, where we want to synthesize the virtual view from the input views. We need to obtain light rays that reach the virtual viewpoint. One of such light rays, referred to as a target light ray, is denoted as $r(\boldsymbol{x}, t)$, where $\boldsymbol{x}$ represents the position of the light ray in the virtual view and t is time. To obtain $r(\boldsymbol{x}, t)$, we divide the scene into several depth layers whose depths are represented as $z \in \mathcal{Z}$, and estimate the originating depth of $r(\boldsymbol{x}, t)$. To estimate the originating depth, we refer to the three input cameras close to the intersection of $r(\boldsymbol{x}, t)$ and camera array plane, which are represented as a set of cameras $V(\boldsymbol{x})$. The intersection between the target light ray $r(\boldsymbol{x}, t)$ and a depth layer with z is represented as $\boldsymbol{p}(\boldsymbol{x}, z, t)$. For each z, we evaluate the color consistency between the reference light rays $r_i(\boldsymbol{x}, z, t)$, which correspond to the back-projection of $\boldsymbol{p}(\boldsymbol{x}, z, t)$ to the i-th input camera with $i \in V(\boldsymbol{x})$. Each RGB component of $r_i(\boldsymbol{x}, z, t)$ takes a floating-point value between 0 and 1. The color-consistency cost is given by

$$C(\boldsymbol{x}, z, t) = \text{consistency}(r_i(\boldsymbol{x}, z, t)|_{i \in V(\boldsymbol{x})}) \tag{1}$$

We use the sum of variances for each RGB component as the consistency measure. At the same time, we obtain the color of the target light ray $r(\boldsymbol{x}, t)$ by blending the reference light rays with z.

$$I(\boldsymbol{x}, z, t) = \sum_{i \in V(\boldsymbol{x})} w_i(\boldsymbol{x}) \boldsymbol{r_i}(\boldsymbol{x}, z, t) \tag{2}$$

where $w_i(\boldsymbol{x})$ denotes the weight for the reference light ray $r_i(\boldsymbol{x}, z, t)$, which takes a floating-point value between 0 and 1 depending on the position of the reference cameras and the target light ray. With a given time t, $C(\boldsymbol{x}, z, t)$ and $I(\boldsymbol{x}, z, t)$ are obtained for all $\boldsymbol{x}$ and z. Then, we refer to $C(\boldsymbol{x}, z, t)$ and $I(\boldsymbol{x}, z, t)$ as cost volume and color volume, respectively. More precisely, the reference light rays to calculate $C(\boldsymbol{x}, z, t)$ are taken from the videos with random patterns, while $I(\boldsymbol{x}, z, t)$ is taken from the videos with the blank pattern.

Generally, the consistency cost in Eq. (1) is insufficient for stable depth estimation because it only contains one point

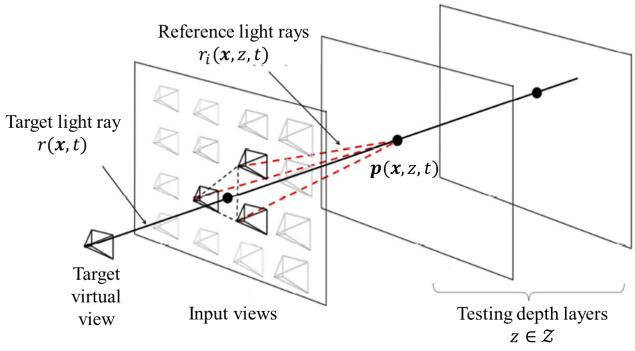

Fig. 2. Configuration for rendering a virtual view

consistency at one time. Therefore, this cost should be spatially and temporally filtered in each depth layer as

$$\tilde{C}(\boldsymbol{x}, z, t) = \sum_{t' \in \mathcal{T}(t)} \sum_{\boldsymbol{x'} \in \mathcal{S}(\boldsymbol{x})} T_{\boldsymbol{x}, z, t}(t - t') S_{\boldsymbol{x}, z, t}(\boldsymbol{x} - \boldsymbol{x'}) C(\boldsymbol{x'}, z, t') \tag{3}$$

where $\mathcal{T}(t) = \{t' | t - t' \geq 0\}$ denotes the set of times in the past of t. $\mathcal{S}(\boldsymbol{x}) = \{\boldsymbol{x'} | \ |\boldsymbol{x} - \boldsymbol{x'}| \leq W\}$ represents a set of points within a square window centered at $\boldsymbol{x}$. $T_{\boldsymbol{x}, z, t}(t')$ and $S_{\boldsymbol{x}, z, t}(\boldsymbol{x'})$ denote temporal and spatial filter kernels respectively, which can take different shapes for different $\boldsymbol{x}$, z, and t. The design of these filters will be addressed in section II-C.

Finally, the depth value that minimizes the cost function is selected for each target light ray $r(\boldsymbol{x}, t)$ as

$$z_{opt}(\boldsymbol{x}, t) = \arg \min_{z \in \mathcal{Z}} \tilde{C}(\boldsymbol{x}, z, t) \tag{4}$$

Given its originating depth $z_{opt}(\boldsymbol{x}, t)$, the color of the target light ray $r(\boldsymbol{x}, t)$ is selected as

$$r(\boldsymbol{x}, t) = I(\boldsymbol{x}, z_{opt}(\boldsymbol{x}, t), t) \tag{5}$$

By gathering $r(\boldsymbol{x}, t)$ for all $\boldsymbol{x}$, the virtual view at time t is finally synthesized.

Fig. 3. Our camera array and projector system

C. Space-time filtering

The optimal shapes for spatial and temporal filters depend on the motion and texture around each pixel. For example, a larger spatial window is preferred for weakly textured regions, but it causes errors (boundary fatting effect) around the large depth gaps. Meanwhile, keeping the spatial window size small and extending temporal window size will be a remedy for the above errors. However, this remedy is effective only for regions without motions. In this work, we employ a simple filter adaptation method based on the temporal variation on the color volume $I(\boldsymbol{x}, z, t)$, which can be efficiently implemented on GPU.

The color volume includes motion information. We estimate the static-ness of $(\boldsymbol{x}, z)$ from the two sequential temporal frames with the blank pattern.

$$\alpha(\boldsymbol{x}, z, t) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-k\left|I(\boldsymbol{x}, z, t) - I(\boldsymbol{x}, z, t-1)\right|^2}{2\sigma^2}\right) \tag{6}$$

where k and δ are positive constants. A smaller $\alpha(\boldsymbol{x}, z, t)$ indicates that $(\boldsymbol{x}, z)$ is more likely to belong to a dynamic region. According to the value of $\alpha(\boldsymbol{x}, z, t)$, the spatial filter kernel is determined as

$$\mathcal{S}_{\boldsymbol{x},z,t}(\boldsymbol{x'}) = 1 - \frac{\alpha(\boldsymbol{x}, z, t)|\boldsymbol{x'}|}{W} \tag{7}$$

Using this filter kernel, we first apply spatial filtering to the color-consistency cost as

$$C'(\boldsymbol{x}, z, t) = \sum_{\boldsymbol{x'} \in \mathcal{S}(\boldsymbol{x})} S_{\boldsymbol{x},z,t}(\boldsymbol{x} - \boldsymbol{x'}) C(\boldsymbol{x'}, z, t) \tag{8}$$

Then, we perform temporal filtering by

$$\tilde{C}(\boldsymbol{x}, z, t) = (1-\alpha(\boldsymbol{x}, z, t))C'(\boldsymbol{x}, z, t) + \alpha(\boldsymbol{x}, z, t)\tilde{C}(\boldsymbol{x}, z, t-1) \tag{9}$$

Note that Eq. (8) is implemented as a FIR filter (kernel convolution), while Eq. (9) is implemented as an IIR filter (recursive update). However, combination of Eqs. (8) and (9) is equivalent to Eq. (3). Note that we can virtually use many temporal frames while Mori's method [8] used only two temporal frames. Nevertheless, our implementation as an IIR filter keeps our method from being too much computationally expensive and memory hungry.

For an element $(\boldsymbol{x}, z, t)$ in a static region, the weights for spatial neighbors tend to be small but the weights for temporal neighbors tend to be large. It will reduce boundary fatting effect around the large depth gaps, while keeping the stability with larger temporal weights. In contrast, for an element in a dynamic region, the weights are large for spatial neighbors and small for temporal neighbors. It will prevent using the past information in moving regions, while keeping the stability with larger spatial weights. This adaptation is performed per element $(\boldsymbol{x}, z, t)$ because we use $\alpha(\boldsymbol{x}, z, t)$ to control the space-time filter.

TABLE I
COMPARED METHODS

	active illumination	space-time filtering
(i)	-	spatial only
(ii)	✓	spatial only
(iii)	✓	space-time ($\alpha(\boldsymbol{x}, z, t)$ = 0.5)
(iv)	✓	space-time ($\alpha(\boldsymbol{x}, z, t)$ = 0.9)
(v)	✓	space-time (adaptive)

III. EXPERIMENT

Our system consists of 16 video cameras (Point Grey Research Flea3), a DLP projector (NEC L50W), and a workstation (HP Z820). Each camera captures RGB color images in 640 × 480 pixels at 60 fps. The workstation has Intel 2.40 GHz dual processors, 16 GB main memory, and an NVIDIA GeForce GTX 660 graphics card with 2.0 GB video memory.

The appearance of the system is shown in Fig. 3. We arranged the cameras in a 4 × 4 array, whose intervals were about 50 mm both in horizontal and vertical directions. All cameras and a DLP projector were synchronized by sharing vertical synchronous signal. The cameras were calibrated with OpenCV and rectified with the method in [11]. Color calibration was not performed, but we set the same exposure and gain values for all cameras. As the target scene, we located several objects including a person in front of the system. We placed 30 depth layers to cover the distance 1100–4000 mm from the camera array.

We compared five methods shown in Table 1, which can be categorized in terms of (A) active illumination and (B) space-time filtering. The size of spatial filtering window was set to 21 × 21 for all methods. The weights were constant over the window for methods (i) and (ii), while they varied according to Eq. (7) for methods (iii)–(v). Method (i) is the baseline reference, where no active illumination was employed. In this case, we used only the videos with the blank pattern both for the depth estimation and view synthesis. Method (ii) used active illumination without temporal filtering. Methods (iii) and (iv) are with active illumination and space-time filtering with a fixed filter shape for all pixels; the value of $\alpha(\boldsymbol{x}, z, t)$ was fixed to 0.5 and 0.9, respectively, regardless of the motion of each pixel. Method (v) is our proposal, where we used

(a) method (i) (b) method (ii) (c) method (iii)

(d) method (iv) (e) method (v) (proposed method)

Fig. 4. Comparison (top: depth maps, bottom: virtual views)

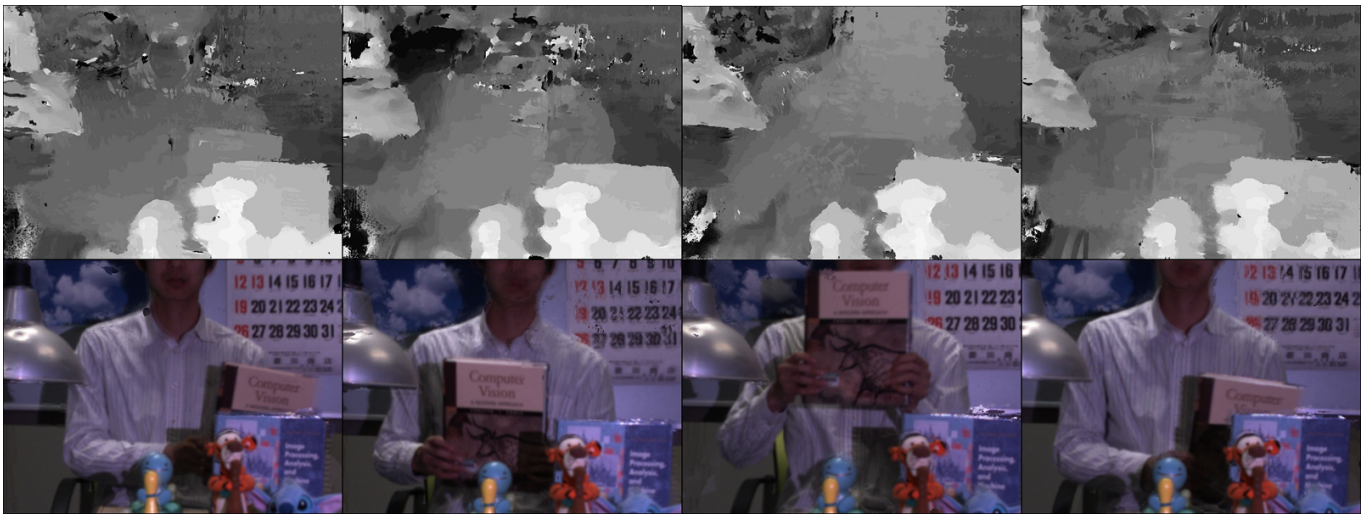

Fig. 5. Depth maps and virtual views for different viewpoints with the proposed method.

active illumination and adaptive space-time filtering. We set $k = 10$ and $\sigma = 0.44$ in Eq. (3). Figure 4 shows the estimated depth maps and generated virtual views by the five methods, which were generated from the same stored images. It can be observed from the results of methods (i) and (ii) that depth estimation was marginally improved with active illumination, but was not of satisfactory quality without temporal filtering. As shown in the results of methods (iii)–(v), space-time filtering greatly improved depth estimation quality, especially with adaptive filtering in method (v). The values of $\alpha(\boldsymbol{x}, z_{opt}(\boldsymbol{x}, t), t)$ are also visualized in Fig. 4(e), where a darker value indicates a larger motion. The value of $\alpha(\boldsymbol{x}, z, t)$ was about 0.9 at static pixels. Figure 5 shows depth maps and virtual views generated by the proposed method for different viewpoints. Both of the target scene and the viewpoint are moving. Please refer to the supplementary video for more detail.

We measured the processing time of the entire process including video capturing from the cameras. New views were synthesized in every 130 msec (7.6 fps) with our current GPGPU implementation. Our adaptive space-time filtering method was performed on live video inputs in a nearly real-time frame rate.

IV. Conclusion

In this paper, we proposed an adaptive space-time filtering for virtual view synthesis that can improve the quality of depth maps, and consequently, the virtual views generated from them. Our method determines the shape of space-time filtering for each pixel according to the motion. Our method was tested on a system consisting of 16 video cameras and a DLP projector to show its effectiveness. We achieved high quality virtual view synthesis with a nearly real-time frame rate. In the future work, we will explore better space-time filter shapes that can achieve more stable depth estimation and is suitable for real-time implementation as well.

References

[1] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, and L. McMilan, "Image-Based Visual Hulls," Proc. ACM SIGGRAPH 2000, pp. 369–374, 2000.
[2] R. Yang, G. Welch, and G. Bishop, "Real-Time Consesus-Based Scene Reconstruction Using Commodity Graphics Hardware," Proc. Pacific Graphics, pp. 225–235, 2002.
[3] G.G. Slabaugh, R.W. Schafer, and M.C. Hans, "Image Based Photo Hulls for Fast and Photo-Realistic New View Synthesis," Real-Time Imaging, Vol. 9, No. 5, pp. 347–360, 2003.
[4] I. Kitahara and Y. Ohta, "Scalable 3D Representation for 3D Video in a Large- Scale Space," Presence, Vol. 13, No. 2, pp.164–177, 2004.
[5] C. Zhang and T. Chen, "A Self-Reconfigurable Camera Array," Proc. Eurographics Simposium on Rendering, pp.243–254, 2004.
[6] M. Li, M. Magnor, and H.-P. Seidel, "Hardware-Accelerated Rendering of Photo Hulls," Proc. Eurographics 2004, Vol. 23, No. 3, pp. 635–642, 2004.
[7] Y. Taguchi, K. Takahashi, T. Naemura, "Real-Time All-in-Focus Video-based Rendering Using a Network Camera Array," 3DTV-Conference, pp. 241–244, May 2008.
[8] T. Mori, K. Takahashi, and T. Fujii, "Real-Time Free-Viewpoint Image Synthesis System Using Time Varying Projection," ITE Transactions on Media Technology and Applications Vol. 2, No. 4 pp. 370–377, 2014.
[9] J. Davis, D. Nehab, R. Ramamoorthi, S. Rusinkiewicz: "Spacetime Stereo: A Unifying Framework for Depth from Triangulation," IEEE TRAMI, Vol. 27, No. 2, pp. 296–302, 2005.
[10] L. Zhang, N. Snavely, B. Curless, S.M.Seitz: "Spacetime faces: High-resolution capture for modeling and animation," ACM Annual Conference on Computer Graphics, pp. 548–558, 2004.
[11] N. Fukushima, T. Yendo, T. Fujii, M. Tanimoto, "A Novel Rectification Method for Two-Dimensional Camera Array by Parallelizing Locus of Feature Points," Proc. IWAIT2008, Jan. 2008.