J. Vis. Commun. Image R. 21 (2010) 577–585

Contents lists available at ScienceDirect

J. Vis. Commun. Image R.

journal homepage: www.elsevier.com/locate/jvci

# Real-time video-based rendering from uncalibrated cameras using plane-sweep algorithm

Songkran Jarusirisawad [a,*], Vincent Nozick [b], Hideo Saito [a]

[a] Department of Information and Computer Science, Keio University 3-14-1 Hiyoshi, Kohoku-ku, Yokohama 223-8522, Japan
[b] Institut Gaspard Monge, Université Paris-Est Marne-la-Vallée, Cité DESCARTES, 5 Boulevard Descartes, 77454 Marne-la-Vallée Cedex 2, France

ARTICLE INFO

*Article history:*
Received 1 August 2009
Accepted 12 January 2010
Available online 25 January 2010

*Keywords:*
Video-based rendering
Free viewpoint video
Plane-sweep
Projective grid space
Real-time
Graphics processing unit (GPU)
View interpolation
Uncalibrated cameras

ABSTRACT

In this paper, we present a new online video-based rendering (VBR) method that creates new views of a scene from uncalibrated cameras. Our method does not require information about the cameras intrinsic parameters. For obtaining a geometrical relation among the cameras, we use projective grid space (PGS) which is 3D space defined by epipolar geometry between two basis cameras. The other cameras are registered to the same 3D space by trifocal tensors between these basis cameras. We simultaneously reconstruct and render novel view using our proposed plane-sweep algorithm in PGS. To achieve real-time performance, we implemented the proposed algorithm in graphics processing unit (GPU). We succeed to create novel view images in real-time from uncalibrated cameras and the results show the efficiency of our proposed method.



## 1. Introduction

Conventional 2D video provides a fixed viewpoint of the recorded event that viewers can only see a video playback passively. Viewpoint of a video playback is always the same as how the scene was recorded. In contrast, free viewpoint video is a system for viewing a video of a real-world event, allowing the user to control the viewpoint and generate new views of a dynamic scene from the desired 3D position. This means that each viewer of the same content may be observing from a unique viewpoint.

Most of the proposed video-based rendering (VBR) methods for creating free viewpoint video usually assume that cameras are strongly calibrated, i.e. cameras' internal parameters such as optical axis, focal length are assumed to be known.

In this paper, we present a new online VBR method that creates new views of the scene from uncalibrated cameras. Our contribution is the use of uncalibrated cameras with the plane-sweep algorithm. We obtain geometrical relation among the cameras from projective grid space (PGS) [1] framework. Our proposed plane-sweep algorithm in PGS, which is implemented on graphics processing unit (GPU), can create new views in real-time. In the conventional plane-sweep algorithm for strongly calibrated cameras, the near and far planes that bound the reconstructed volume are measured and defined from the actual 3D positions of a scene. The advantage of our proposed plane-sweep algorithm in PGS is that these planes are easily defined and can be visualized from the image of basis camera 2.

In the following sections, we firstly give a survey of previous works of VBR methods. Then projective grid space framework that we use for weak camera calibration is explained in Section 3. We describe conventional plane-sweep algorithm in the Euclidean space and our proposed plane-sweep algorithm in PGS in Sections 4 and 5, respectively. Section 6 presents the implementation detail in GPU. Finally, we show the experimental results and conclusion.

## 2. Previous works

Image-based rendering (IBR) is the process for creating a new view from several input images of a static scene. VBR can be thought as an extension of IBR to the dynamic scene. Compared to IBR techniques, methods for VBR involve challenging problems to reach a real-time rendering while a number of input images need to be processed. Hence, time consuming algorithms which seem to be practical for IBR will be not suitable for VBR.

We may categorize VBR techniques into off-line and online methods. In the off-line ones, input videos are recorded before hand, rendering begins only after the scene information has been extracted from the input videos. This processing time can be long which increases the time between acquisition and rendering. Off-line techniques can handle a large amount of data and use

* Corresponding author.
*E-mail addresses:* songkran@hvrl.ics.keio.ac.jp, kpriony@hotmail.com (S. Jarusirisawad).

1047-3203/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jvcir.2010.01.005

sophisticated algorithms because only the rendering part should be done in real-time since the prior processing can be computed off-line. In the online VBR methods, 3D data extraction should be performed in real-time. Since running time is a critical criterion for online VBR methods, sophisticate or time consuming scene reconstruction algorithms are usually not applicable.

### 2.1. Off-line video-based rendering

One of the earliest VBR method is the Virtualized Reality proposed by Kanade et al. [2]. In that research, 51 cameras are placed around hemispherical dome called 3D room to transcribe a scene. 3D structure of a moving human is extracted using multi-baseline stereo [3]. Then free viewpoint video is synthesized from the recovered 3D model.

*Immersive video* system proposed by Moezzi et al. [4] use three to six synchronized cameras to capture different viewpoints of a scene. The static portion of the scene (background) is first manually built. Dynamic objects are extracted as voxel representations using volume intersection technique. All models construction in this system is done off-line.

Zitnick et al. [5] generated high quality new view images in real-time from eight cameras. Color segmentation-based stereo algorithm is used to generate photo consistent correspondences. Mattes for areas near depth discontinuities are automatically extracted to reduce artifacts, then rendering is performed with a layered image representation. Their proposed stereo algorithm, while very effective, is not fast enough for the entire system to operate in real-time.

Carranza et al. [6] recover human motion at off-line process by fitting a human shaped model to multiple view silhouettes. Multi-view texturing is employed during rendering and it can run at real-time frame rates using conventional graphics hardware. Starck and Hilton [7] also recover a human model using silhouettes together with stereo correspondences and feature cues which are manually selected from the image.

Jarusirisawad and Saito [8] proposed free viewpoint video from uncalibrated pure rotating and zooming cameras. Using rotating and zooming cameras allow cameraman to capture the area of interest at higher resolutions. Their contribution is the method to weakly calibrate these non-static cameras from natural features in a scene. They synthesize free viewpoint video of a foreground moving object from conventional visual hull. The limitation of this method is that the background must be approximated as several planes which is not applicable to all scenes and user has to segment these planes manually at the initial frame. Long computation time of finding corresponding features and 3D reconstruction makes the system not fast enough to run in real-time.

Proposed systems in off-line VBR category cannot get a real-time processing for the whole process mainly because they are dealing with a large number of cameras (ranging from tens to hundred) [2,9], manual preprocessing is needed [4,7,8], or they are focusing on the quality of the generated image rather than the processing time [6].

### 2.2. Online video-based rendering

Only a few VBR methods reach online rendering. Complex algorithms used in off-line methods are simply too slow for real-time implementation. Therefore, the generated new view images from online methods might have less accuracy comparing to the off-line ones.

One of the popular online VBR methods is the visual hulls algorithm. The 3D shape of the object is approximated by the intersection of the projected silhouettes. There are some online implementations of the visual hulls algorithm [10–13]. Among all these visual hulls methods, the image-based visual hulls presented by Matusik et al. [13] is a VBR method from uncalibrated cameras. This method reconstruct visual hull of the object using epipolar geometry in an image space instead of the 3D space. The main drawbacks of all visual hulls methods are the impossibilities to reconstruct concave objects and handle the background of the scene.

Yang et al. [14] use a distributed light field for online rendering from 64-camera device based on a client-server scheme. The cameras are clustered into groups controlled by several computers. These computers are connected to a main server and transfer only the image fragments needed to compute the requested new view. This method provides real-time rendering but requires at least eight computers for 64 cameras and additional hardware.

Schirmacher et al. [15] presented a system for reconstructing arbitrary views from multiple images with depth using a generalized Lumigraph data structure and a warping-based rendering algorithm. With their technique, it is possible to render arbitrary views of dynamic, non-diffuse scenes at interactive frame rates.

Some plane-sweep implementations achieve online rendering using graphics processing unit (GPU). The plane-sweep algorithm introduced by Collins [16] was adapted to online rendering by Yang et al. [17]. They computed new views in real-time from five cameras using four computers. Geys et al. [18] also used a plane-sweep approach to recover the scene geometry and rendered new views in real-time from three cameras and one computer. Nozick and Saito [19] introduced a plane-sweep implementation for moving camera where all the input cameras are calibrated in real-time using ARToolkit [20] markers.

Our method belongs to the online VBR group based on plane-sweep algorithm. The main difference is that in the previous works [16–19] they assume that cameras are strongly calibrated. Our contribution is the use of plane-sweep algorithm with projective grid space to achieve free viewpoint video system from uncalibrated cameras. This paper, as an extension of [21], presents a new method for online video-based rendering from uncalibrated cameras using plane-sweep algorithm in projective grid space.

Our method in this paper is also based on the use of projective grid space as in [8]. However, our contribution is different from [8]. In this paper, we focus on the 3D reconstruction and rendering algorithm for uncalibrated cameras that can run in real-time and do not need the assumption about the scene (planar background, single moving object) as in [8]. Even plane-sweep algorithms has been proposed in the literatures, to our best knowledge, they were applied to calibrated cameras. Using plane-sweep algorithm in projective grid space also has several advantages over conventional plane-sweep in the Euclidean space. By using PGS, our method create new view image in without information about intrinsic parameters. Near and far planes in PGS for doing plane-sweep are easily defined and visualized from basis camera 2. These planes must be specified in 3D world coordinates if the plane-sweep algorithm in the Euclidean space (calibrated cameras) is used.

## 3. Projective grid space

This section describes weak cameras calibration framework for our plane-sweep method. To implement the plane-sweep algorithm, we need to project 3D points into image frame of each camera including the virtual one. Projective grid space allows us to define that 3D space and find the projection without knowing cameras intrinsic parameters or Euclidean information of a scene.

Projective grid space (PGS) [1] is a 3D space defined by image coordinate of two arbitrarily selected cameras called basis camera 1 and basis camera 2. To distinguish this 3D space from the Euclidean one, we denote the coordinate system in PGS by $P$–$Q$–$R$-axis. Fig. 1 shows the definition of PGS. The x and $y$-axes in the image

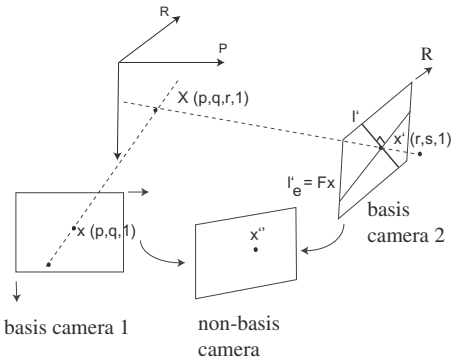


**Fig. 1.** Definition of projective grid space.

of basis camera 1 corresponds to the *P*- and *Q*-axes, while *x*-axis of the basis camera 2 corresponds to the *R*-axis in PGS.

Homogeneous coordinate $\mathbf{X} = (p, q, r, 1)^T$ in PGS is projected on image coordinate $\mathbf{x} = (p, q, 1)$ and $\mathbf{x}' = (r, s, 1)$ of the basis camera 1 and the basis camera 2, respectively. Because $\mathbf{x}$ and $\mathbf{x}'$ are the projection of the same 3D point, $\mathbf{x}'$ must lie on the epipolar line of $\mathbf{x}$. Thus, $s$ coordinate of $\mathbf{x}'$ is determined from $\mathbf{x}'^T F\mathbf{x} = \mathbf{0}$ where $F$ is the fundamental matrix from basis camera 1 to basis camera 2.

Other cameras (non-basis cameras) are said to be weakly calibrated once we can find the projection of 3D point from the same PGS to those cameras. The key idea is that 3D points in PGS will be projected onto both two basis cameras first to make 2D–2D point correspondence. Then, this correspondence can be transferred to a non-basis camera.

In the original work of projective grid space [1], fundamental matrices were used to transfer these correspondences. However, point transfer using fundamental matrices will become numerical instable if a 3D point lies near the trifocal plane. To overcome this problem, we extend the proposed method by using trifocal tensor instead.

Trifocal tensor $\tau_i^{jk}$ is a homogeneous $3 \times 3 \times 3$ array (27 elements) that satisfies

$$l_i = l'_j l''_k \tau_i^{jk} \tag{1}$$

where $l_i, l'_j$ and $l''_k$ are corresponding lines in the first, second and third image, respectively.

Trifocal tensor can be estimate from point correspondences or line correspondences between three images. In case of using only points correspondences, at least seven point correspondences are necessary to estimate the trifocal tensor using the normalized linear algorithm [22].

Given point correspondence $\mathbf{x}$ and $\mathbf{x}'$, we can find corresponding point $\mathbf{x}''$ in the third camera by Eq. (2).

$$x''^k = x^i l'_j \tau_i^{jk} \tag{2}$$

where $\mathbf{l}'$ is the line in the second camera which pass though point $\mathbf{x}'$.

We can choose any line $\mathbf{l}'$ which pass point $\mathbf{x}'$ except the epipolar line corresponding to $\mathbf{x}$. If $\mathbf{l}'$ is selected as the epipolar line corresponding to $\mathbf{x}$, then point $\mathbf{x}''$ is undefined because $x^i l'_j \tau_i^{jk} = 0^k$. A convenient choice for selecting the line $\mathbf{l}'$ is to choose the line perpendicular to epipolar line of $\mathbf{x}$.

To summarize, considering Fig. 1, given a 3D point $\mathbf{X} = (p, q, r, 1)^T$ in PGS and tensor $\tau$ defined by basis camera 1, basis camera 2 and non-basis camera we can project point $\mathbf{X}$ to non-basis camera as the following:

1. Project $\mathbf{X} = (p, q, r, 1)^T$ to $\mathbf{x} = (p, q, 1)^T$ and $\mathbf{x}' = (r, s, 1)^T$ on basis camera 1 and basis camera 2, respectively. $s$ is found by solving $\mathbf{x}'^T F\mathbf{x} = \mathbf{0}$.
2. Compute epipolar line $\mathbf{l}'_e = (l_1, l_2, l_3)^T$ of $\mathbf{x}$ on basis camera 2 from $\mathbf{l}'_e = F\mathbf{x}$.

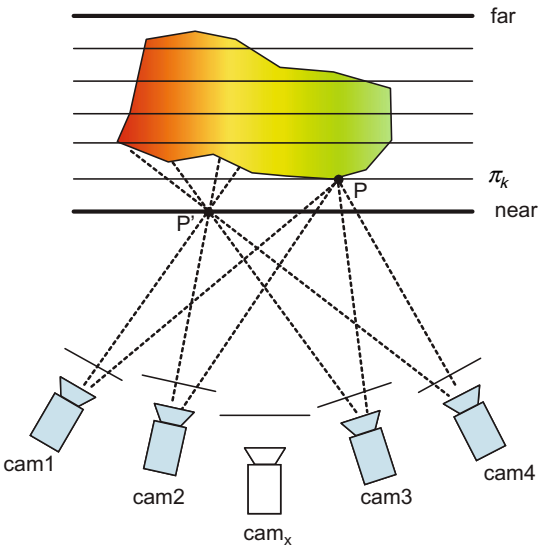


**Fig. 2.** Plane-sweep algorithm in the Euclidean space.

3. Compute the line $\mathbf{l}'$ which passes $\mathbf{x}'$ and perpendicular to $\mathbf{l}'_e$ by $\mathbf{l}' = (l_2, -l_1, -rl_2 + sl_1)^T$.
4. The transferred point in non-basis camera is $x''^k = x^i l'_j \tau_i^{jk}$.

## 4. Plane-sweep in the Euclidean space

This section explains the general idea of conventional plane-sweep algorithms in the Euclidean space of the calibrated cameras. Then, we present our proposed one for using with projective grid space in Section 5.

The plane-sweep algorithm creates novel views of a scene from several input images. Considering a scene where the objects are exclusively Lambertian surfaces, the viewer should place the virtual camera $cam_x$ somewhere around the real video cameras and define a *near* plane and a *far* plane such that every object of the scene lies between these two planes. Then, the space between *near* and *far* planes is divided into several parallel planes $\pi_k$ as depicted in Fig. 2.

Plane-sweep algorithm is based on the following assumption: a point lying on a plane $\pi_k$ whose projection on every input camera provides a similar color will potentially correspond to the surface of an object. Considering a visible object of the scene lying on one of these plane $\pi_k$ at a point $P$, this point will be seen by every camera with the same color, i.e. the object color. Now consider another point $P'$ lying on a plane but not on the surface of the visible object, this point will probably not be seen by the capturing cameras with the same color.[1] Fig. 2 illustrates this principal idea of the plane-sweep algorithm.

During the new view creation process, every plane $\pi_k$ is computed in a back to front order. Each point $P$ of a plane $\pi_k$ is projected onto the input images. A score and a representative color are the computed according to the matching of the colors found. A good score means every camera see a similar color. The computed scores and colors are projected onto the virtual camera $cam_x$. The pixel color in the virtual view will be updated only if the projected point $p$ provides a better score than the current one. Then the next plane $\pi_{k+1}$ is computed. The final new view image is obtained once every plane has been computed. This method is detailed on [23].

[1] For interpretation of the references to color in Fig. 2, the reader is referred to the web version of this paper.

## 5. Plane-sweep in projective grid space

To do plane-sweep in PGS, we need to define a position of virtual camera, define planes in 3D space and then compute a new view image from the defined planes. In this section, we describe the detail of each step.

### 5.1. Defining virtual camera position

To perform plane-sweep algorithm, 3D point on a plane must be able to project to a virtual camera. In the calibrated cameras cases, projection matrix of a virtual camera can be defined from camera's pose (extrinsic parameters) because intrinsic camera parameters are known. This allowing virtual camera to be moved anywhere around a scene.

In our case where PGS is used, intrinsic parameters of any camera are unknown. Method for defining virtual camera in calibrated case is not applicable to our case. In our method, the position of the virtual camera is limited to only between two real reference cameras. A ratio $r$ from 0 to 1 is used for defining distance between these reference cameras. Fig. 3 illustrate this definition. In Fig. 3, a ratio $r$ equals to 0 (respectively 1) means the virtual camera has the same position as camera 1 (respectively camera 2).

To find the projection of 3D point $\mathbf{X}$ in PGS on a virtual camera, 3D point $\mathbf{X}$ is projected onto both real reference cameras first. The position of the same 3D point in the virtual camera is calculated using linear interpolation. If the projected points in the real reference cameras 1 and 2 are $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively, the projected point $\mathbf{x}_3$ in a virtual camera is calculated from (3) as in Fig. 3.

$$\mathbf{x}_3 = (1-r)\mathbf{x}_1 + r\mathbf{x}_2 \tag{3}$$

### 5.2. Defining planes in PGS

Any arbitrary *near* and *far* planes in PGS can be defined for doing plane-sweep. In our method, we define the planes along the $R$-axis ($x$ image coordinate of basis camera 2) as shown in Fig. 4. This approach makes the 3D *near* and *far* planes adjustment become easy since we can visualize them directly from the image of basis camera 2. This is impossible for the case of the normal plane-sweep algorithm in the Euclidean space in which full calibration is used. In that case, actual depth of a scene has to be measured so that *near* and *far* planes cover all volume of interest.

In our approach, basis camera 2 will not be used for color consistency testing during perform plane-sweep because every plane would be projected as a line in this image. So the basis camera 2 is needed only for weakly calibrated cameras to PGS, after that we can disable it to save CPU time.

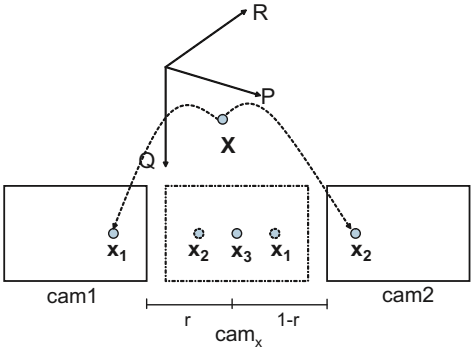


**Fig. 3.** Defining virtual camera in projective grid space.

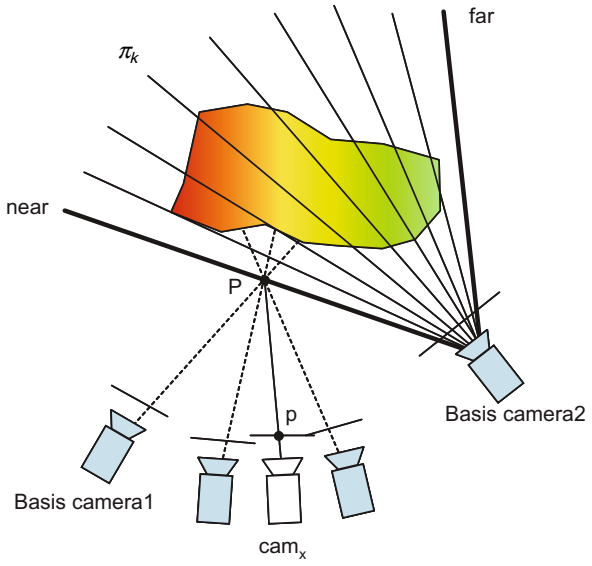


**Fig. 4.** Defining planes for doing plane-sweep in projective grid space.

### 5.3. Computing new view images

In this section, we explain how we implemented the plane-sweep algorithm after defining the virtual camera's position and planes in PGS. If pixel $p$ in a virtual camera is back projected to a plane $\pi_k$ on a point $P$, we want to find the projection of $P$ on every input image for the score computation step. As illustrated on Fig. 5, the projection of 3D point $P$ lying on $\pi_k$ on the input image $i$ can be performed by a homography $\mathbf{H}_i$. Thus, the projection $p_i$ of a 3D point $P$ on the camera $i$ is calculated from

$$\mathbf{x}_i = \mathbf{H}_i\mathbf{H}_x^{-1}\mathbf{x} \tag{4}$$

where $\mathbf{x}$ and $\mathbf{x}_i$ are the position of the pixel $p$ and $p_i$, respectively.

Homography $H_i$, where $i$ is a camera number, can be estimated from at least four point correspondences. In our situation, we select four points defined as the image corners of the basis camera 1 as shown in Fig. 5. Then, we project these points onto every real cameras as described in Section 3 for making 2D–2D point correspondences. Then, all homographies used for the plane-sweep method can be estimated from these correspondences. During the score computation, we estimate these homographies instead of projecting every 3D points one by one for computation time purpose.

Algorithm 1 summarizes our plane-sweep algorithm in PGS.

Reset color consistency score of the virtual camera to the max value.

**Algorithm 1.** Plane-sweep algorithm in projective grid space.

**for each** *plane* $\pi_k$ *in PGS* **do**
  **for each** *pixel* $p$ *in* $cam_x$ **do**
- project pixel $p$ to n input images excluding basis camera 2. $c_j$ is the color from this projection on the $j$th camera
- compute average color: $color_p = \frac{1}{n}\sum_{j=1\ldots n} c_j$
- compute color consistency score from variance: $score_p = \sum_{j=1\ldots n}(c_j - color_p)^2$

    **If** $score_p$ *is low than current score of pixel* $p$ **then**
      update score and color on virtual camera to $score_p$ and $color_p$
    **end**

  **end**
**end**

In Algorithm 1, we use the score function proposed in [23].

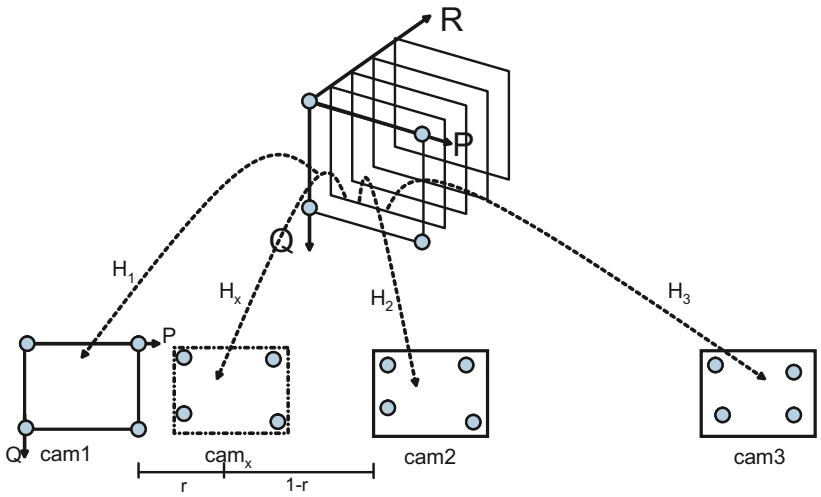


**Fig. 5.** Estimating homography matrices for plane-sweep

## 6. Implementing real-time plane-sweep on GPU

To achieve real-time computation, we implement our plane-sweep algorithm in projective grid space on GPU. Because GPU has a massive parallel processing, using GPU can give much more computation power in many applications comparing to CPU. This section gives some details about our implementation. We use OpenGL for the rendering part. Input images that will be used for color consistency checking are transfer to GPU as multi-textures. In the drawing function, we loop though each plane in PGS from near to far plane. Homographies for warping points on virtual camera to the other cameras are sent to GPU as texture matrices.

We use orthographic projection and draw square to cover the whole image of virtual camera. In the fragment shader, we apply the homography and compute the color consistency score as described in Algorithm 1. Fragment color is assigned to be an average color from all views. The score of fragment is sent to the next rendering pipeline (frame buffer operation) via gl_FragDepth while the average color is sent via gl_FragColor. Then we let OpenGL select the best scores with the *z*-test and update the color in the frame buffer. When rendering is done for all planes, we get novel view in the frame buffer.

## 7. Experimental results

We tested our proposed method on PC Intel(R) Core(TM) 2 Duo 3.00 GHz CPU with graphic card NVIDIA Quadro FX 570. Six Logitech webcams with a resolution $320 \times 240$ are used to capture input videos. The camera setting is depicted by Fig. 6. We select two cameras for defining projective grid space, as illustrated by Fig. 6. Our result is available online at http://www.hvrl.ics.keio.ac.jp/~songkran/jvci/index.html.

Fundamental matrix between cameras 1 and 6, four trifocal tensors defined by camera 1,6,2, camera 1,6,3, camera 1,6,4 and camera 1,6,5 are estimated for weakly calibrating cameras to PGS. 2D–2D correspondences for estimating fundamental matrix and trifocal tensors can be automatically extracted from natural feature points in a scene. In our experiment, we wave marker around a scene and track features for accurate 2D–2D correspondence. We use the code for estimating trifocal tensors from [24].

### 7.1. Running time

Running time and quality of new view image rendering depend on complexity of a scene and the number of planes used in plane-sweep algorithm. The appropriate number of planes varies depending on the complexity of a scene. Using more planes makes processing time become longer but usually gives a better result. In our experiment, it is shown that using 40 planes or more makes the visual result becomes satisfied.

Table 1 shows the number of planes and the running time for rendering new view images using six webcams implemented on CPU and GPU. Both implementations are tested on the same PC. Implementation of our proposed plane-sweep algorithm on GPU is significantly faster than on CPU. Our system gives the same frame rates as the input webcams (30 fps) when using 25 planes or less for scene reconstruction. When implementing plane-sweep algorithm on GPU, most of the computation is done by the graphic card, hence the CPU is free for the video stream acquisition and the virtual camera control.

### 7.2. Qualitative evaluation

We do our plane-sweep algorithm in PGS as described in Section 5. In our experiment, planes are defined from *x*-axis of basis

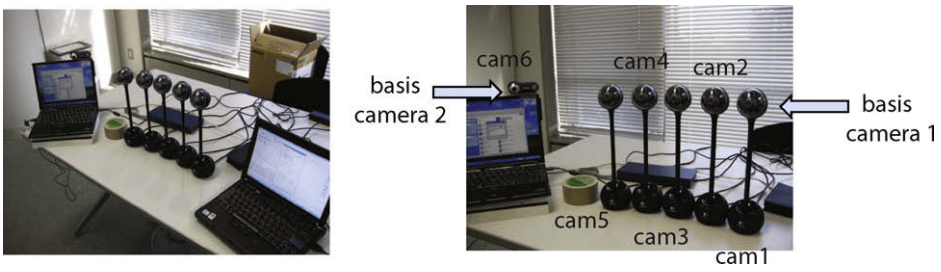


**Fig. 6.** Camera configuration.

**Table 1**
Frame rates (frame/s) of our plane-sweep algorithm implemented on CPU and GPU.

| | Number of planes | | | |
|---|---|---|---|---|
| | 25 | 40 | 60 | 80 |
| CPU | 0.199 | 0.130 | 0.089 | 0.068 |
| GPU | 30.04 | 15.03 | 12.00 | 8.58 |

camera 2 (corresponds to *R*-axis in PGS). *near* and *far* planes are adjusted so that all objects in the other cameras lie between these planes. Fig. 7 shows new view images synthesized from our proposed method using the different number of planes for scene reconstruction.

Some artifacts in the rendered view come from planes discretization. The object that lies between two planes is sometimes reconstructed at the plane that is far from the actual one, so this object will be noticed as artifacts in the rendered view. One possible solution to reduce this errors is to increase the number of planes used in plane-sweep algorithm.

Fig. 8 shows the result new view video at several view point from the selected one input frames. We use 80 planes for reconstructing the scene and our implementation can reach about 9 fps using this configuration. The ratio written under each figure is a virtual camera position between two real reference cameras as described in Section 5.1. The result shows that our method gives a good visual quality and fast enough for online VBR applications.

### 7.3. Quantitative evaluation

This section gives objective quality measurements of our result. One camera is selected as a ground-truth reference and excluded from the plane-sweep algorithm. View at ground-truth camera is then synthesized to measure visual errors. Two metrics $d^{90}$ proposed in [25] and peak signal-to-noise ratio (PSNR) are computed to measure the errors in the synthesized images. $d^{90}$ tells us about the overall distance of misaligned pixels between synthesized image and ground-truth reference. The lower the value of $d^{90}$, the better the quality of the output in new view images.

If the rendered image is much different from the ground-truth, then there will likely be visual artifacts or blurred textures in the synthesized image. We measure these values for 100 consecutive input frames using camera 3 as a ground-truth reference. Camera 3 is leaved-out from plane-sweep algorithm and views at that camera are synthesized. Figs. 9 and 10 show each error metric of our new view images using the different number of planes for scene reconstruction. Table 2 shows the average $d^{90}$ *and* PSNR values over 100 frames.

### 7.4. Effect of the base-line between cameras

The quality of new views depends on several factors, such as complexity of a scene, the number of planes and the base-line between cameras. This section experiments about the effect of the base-line between cameras with our method.

To compare the result between small and wide base-line, we use the same input images as Fig. 8 and in crease the base-line by exclude camera 2 and camera 4 from plane-sweep algorithm. Fig. 11 shows the result free viewpoint images.

Fig. 12 shows the comparison of new views generated from wide and small base-line cameras configuration. From the results

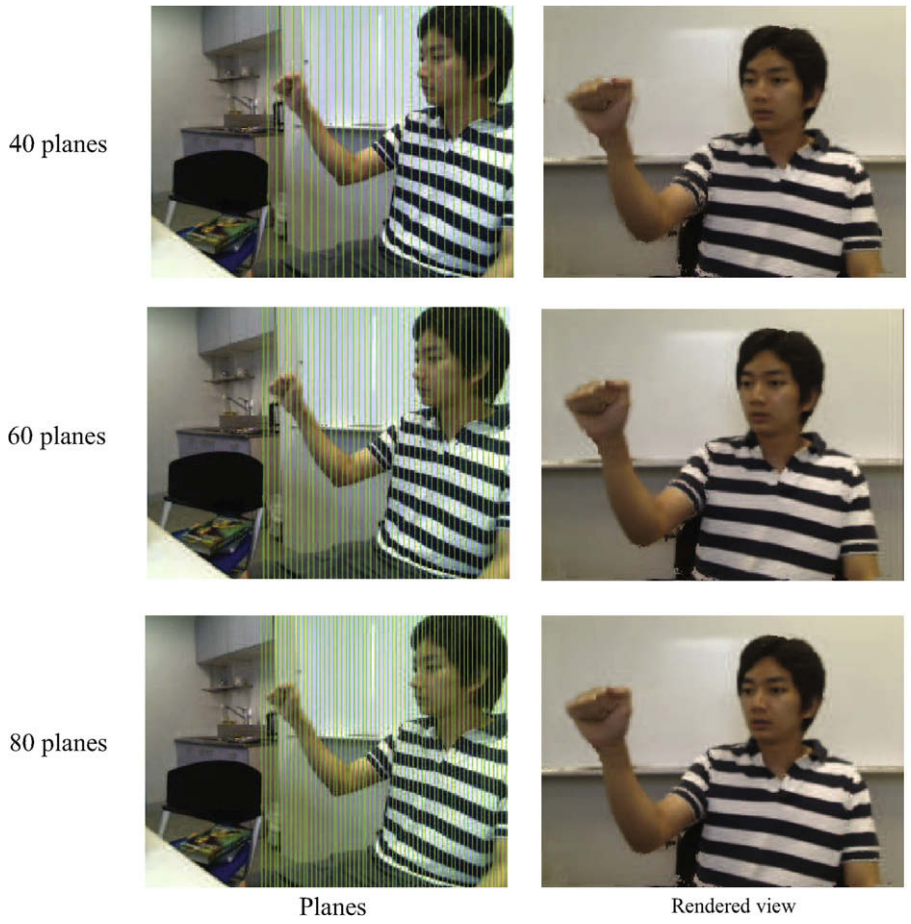


**Fig. 7.** Result new view images using the different number of planes in plane-sweep algorithm.


**Fig. 8.** New views produced by our proposed plane-sweep algorithm in projective grid space using 80 planes.

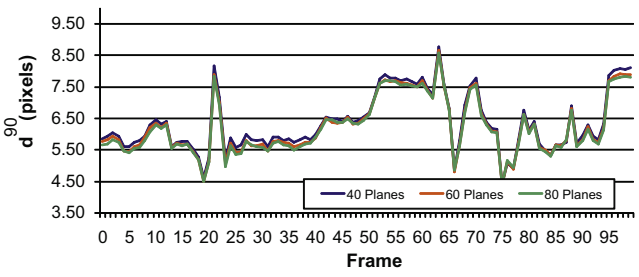


**Fig. 9.** $d^{90}$ Registration error of new view images.

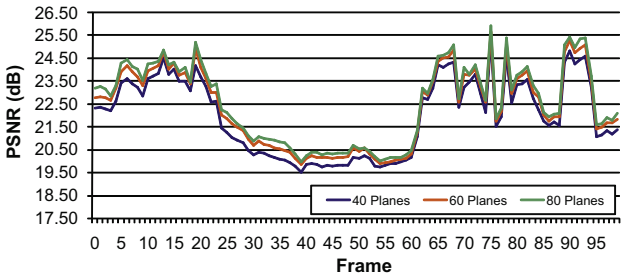


**Fig. 10.** PSNR registration error of new view images.

we can see that artifacts increase with the base-line of cameras. Normally plane-sweep algorithms both for calibrated cases and uncalibrated case (this paper) give a good result when base-line between cameras is small.

**Table 2**
Error measurements for the resulting new view images (average of 100 frames).

| Number of planes | $d^{90}$ (pixels) | PSNR (dB) |
|---|---|---|
| 40 | 5.970 | 21.670 |
| 60 | 5.857 | 22.092 |
| 80 | 5.810 | 22.314 |

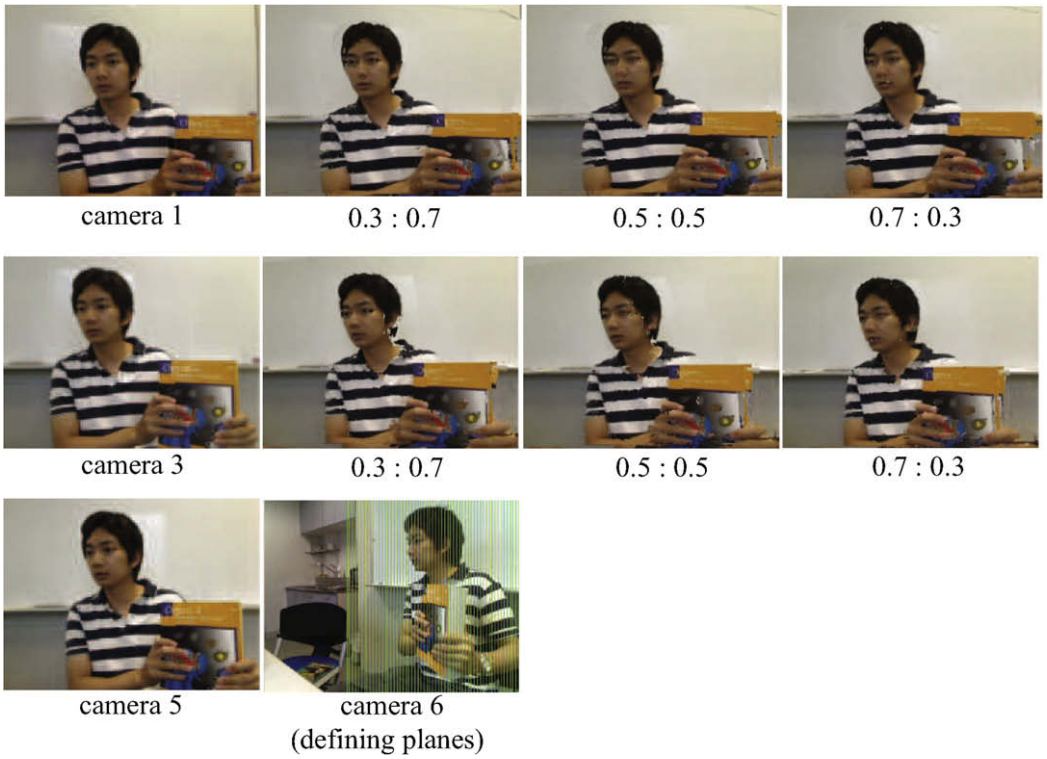


**Fig. 11.** New views produced by our proposed plane-sweep algorithm in projective grid space using 80 planes.

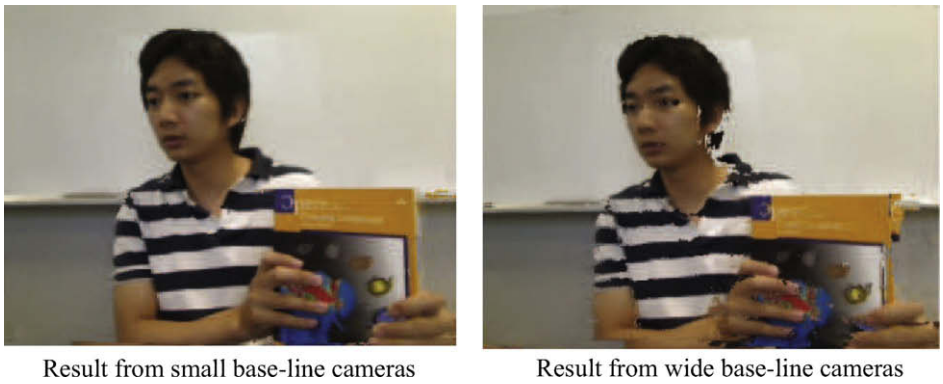


**Fig. 12.** Comparison of new views from small base-line and wide base-line cameras configuration.

## 8. Conclusion

In this paper, we present a new online VBR method that using uncalibrated cameras to creates new views of the scene. Most of previous methods usually assume that cameras are calibrated. By using projective grid space (PGS), our method create new view image in without information about intrinsic parameters. Near and far planes in PGS for doing plane-sweep are easily defined and visualized from basis camera 2. These planes must be specified in 3D world coordinates if the plane-sweep algorithm in the Euclidean space (calibrated cameras) is used. We simultaneously reconstruct and render novel view using plane-sweep algorithm in PGS. Our experiment shows convincing results and achieves real-time performances by implementing our plane-sweep algorithm on GPU.

## References

[1] H. Saito, T. Kanade, Shape reconstruction in projective grid space from large number of images, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'99), vol. 2, 1999, pp. 49–54.
[2] T. Kanade, P.W. Rander, P.J. Narayanan, Virtualized reality: concepts and early results, in: Proceedings of IEEE Workshop on Representation of Visual Scenes, 1995, pp. 69–76.
[3] M. Okutomi, T. Kanade, A multiple-baseline stereo, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (4) (1993) 353–363.
[4] S. Moezzi, A. Katkere, D.Y. Kuramura, R. Jain, Reality modeling and visualization from multiple video sequences, IEEE Computer Graphics and Applications 16 (6) (1996) 58–63.
[5] C. Zitnick, S. Kang, M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, ACM Transactions on Graphics 23 (3) (2004) 600–608.
[6] J. Carranza, C. Theobalt, M. Magnor, H.-P. Seidel, Free-viewpoint video of human actors, ACM Transactions on Graphics 22 (3) (2003) 569–577.
[7] J. Starck, A. Hilton, Model-based multiple view reconstruction of people, in: Proceedings of IEEE International Conference on Computer Vision (ICCV'03), IEEE Computer Society, Washington, DC, 2003, pp. 915–922.
[8] S. Jarusirisawad, H. Saito, 3DTV view generation using uncalibrated pure rotating and zooming cameras, Image Communication 24 (1–2) (2009) 17–30.
[9] S. Moezzi, L.C. Tai, P. Gerard, Virtual view generation for 3D digital video, IEEE Transactions on MultiMedia 4 (1) (1997) 18–26.
[10] M. Li, M. Magnor, H.-P. Seidel, Hardware-accelerated visual hull reconstruction and rendering, in: Proceedings of Graphics Interface 2003, 2003, pp. 65–71.
[11] M. Li, M. Magnor, H.-P. Seidel, Online accelerated rendering of visual hulls in real scenes, Journal of WSCG 11 (2) (2003) 290–297.

[12] C. Theobalt, M. Li, M. Magnor, H.-P. Seidel, A flexible and versatile studio for multi-view video recording, in: Proceedings of Vision, Video and Graphics 2003, Bath, UK, 2003, pp. 9–16.
[13] W. Matusik, C. Buehler, R. Raskar, S.J. Gortler, L. McMillan., Image-based visual hulls, in: Proceedings of ACM SIGGRPAH'00, 2000, pp. 369–374.
[14] J. Yang, M. Everett, C. Buehler, L. McMillan, A real-time distributed light field camera, in: Proceedings of the 13th Eurographics Workshop on Rendering, 2002, pp. 77–86.
[15] H. Schirmacher, M. Li, H. Peter Seidel, On-the-fly processing of generalized lumigraphs, in: Proceedings of Eurographics 2001, 2001, pp. 165–173.
[16] R.T. Collins, A space-sweep approach to true multi-image matching, in: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1996, pp. 358–363.
[17] R. Yang, G. Welch, G. Bishop, Real-time consensus-based scene reconstruction using commodity graphics hardware, in: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications (PG 2002), IEEE Computer Society, Washington, DC, 2002, p. 225.
[18] I. Geys, S. Roeck, L. Gool, The augmented auditorium: Fast interpolated and augmented view generation, in: Proceedings of the Second IEE European Conference on Visual Media Production, 2005, pp. 94–103.
[19] V. Nozick, H. Saito, Real-time free viewpoint from multiple moving cameras, in: Advanced Concepts for Intelligent Vision Systems (ACIVS), 2007, pp. 72–83.
[20] H. Kato, M. Billinghurst, Marker tracking and hmd calibration for a video-based augmented reality conferencing system, in: Proceedings of the Second IEEE and ACM International Workshop on Augmented Reality, 1999, pp. 85–94.
[21] S. Jarusirisawad, H. Saito, V. Nozick, Real-time free viewpoint video from uncalibrated cameras using plane-sweep algorithm, in: Proceedings of the IEEE International Workshop on 3-D Digital Imaging and Modeling (3DIM), 2009, pp. 1740–1747.
[22] R.I. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, second ed., Cambridge University Press, Cambridge, 2004, ISBN 0521540518.
[23] V. Nozick, H. Saito, On-line free-viewpoint video: from single to multiple view rendering, International Journal of Automation and Computing (IJAC) 5 (3) (2008) 257–267.
[24] B. Matei, P. Meer, A general method for errors-in-variables problems in computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, 2000, pp. 18–25.
[25] J. Starck, J. Kilner, A. Hilton, Objective quality assessment in free-viewpoint video production, in: Proceedings of the 3DTV Conference: The True Vision – Capture, Transmission and Display of 3D Video, 2008, pp. 225–228.