HAL
open science

Real-Time Free Viewpoint Video from Uncalibrated Cameras Using Plan e-Sweep Algorithm

Songkran Jarusirisawad, Hideo Saito, Vincent Nozick

To cite this version:

Songkran Jarusirisawad, Hideo Saito, Vincent Nozick. Real-Time Free Viewpoint Video from Uncalibrated Cameras Using Plan e-Sweep Algorithm. 3DIM, 2009 IEEE 12th International Conference on Computer Vision Workshops, Oct 2009, Japan. pp.1740-1747. hal-00733381

HAL Id: hal-00733381

https://hal.science/hal-00733381

Submitted on 18 Sep 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Real-Time Free Viewpoint Video from Uncalibrated Cameras Using Plane-Sweep Algorithm

Songkran Jarusirisawad and Hideo Saito
Keio University,
3-14-1 Hiyoshi, Kohoku-ku,
Yokohama, 223-8522, Japan
songkran,saito@hvrl.ics.keio.ac.jp

Vicent Nozick
Université Paris-Est Marne-la-Vallée,
Cité DESCARTES, 5 boulevard Descartes,
77454 Marne-la-Vallée CEDEX 2, France
vnozick@univ-mlv.fr

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. We obtain geometrical relation among the cameras from projective grid space (PGS)[18] 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 next section we firstly review the previous works on 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 section 4 and 5, respectively. Section 6 presents the implementation detail in GPU. Finally, we show the experimental results and conclusion.

2. Previous Works

We may categorize video based rendering (VBR) techniques into off-line and online methods. The online VBRs are the ones that can recover 3D shapes and rendering new views from live or prerecorded input videos in real-time, while the off-line VBRs are the ones that cannot. Normally, the computation time of 3D reconstruction is longer than the rendering. Thus, some of the off-line VBRs can provide an interactive frames rate of the new views rendering, from the prerecorded videos, by doing 3D reconstruction before hand as a preprocessing step.

2.1. Off-line Video-Based Rendering

One of the earliest VBR method is the Virtualized Reality proposed by Kanade et al. [7]. In that research, 51 cameras are placed around hemispherical dome called 3D Room to transcribe a scene. 3D structure of a moving human is

2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops
978-1-4244-4441-0/09/$25.00 ©2009 IEEE

extracted using multi-baseline stereo [17]. Then free viewpoint video is synthesized from the recovered 3D model.

Immersive Video system proposed by Moezzi et al [13] 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 offline.

Carranza et al. [1] 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 [19] also recover a human model using silhouettes together with stereo correspondences and feature cues which are manually selected from the image.

Several methods of free viewpoint video from uncalibrated cameras have been proposed. Yaguchi and Saito [22] use projective grid space (PGS) [18], a weak cameras calibration framework, to define the geometrical relation between uncalibrated static cameras. The 3D model of the actor in PGS is reconstructed using shape from silhouettes for new views rendering.

Ito and Saito [5] extend the proposed method in [22] to the moving cameras case, where the 2D-2D correspondences for weak cameras calibration are tracked from the arbitrary placed markers in the scene. Songkran and Saito [6] improve the stability of weak calibration in PGS by using trifocal tensors for finding the projection of 3D points in PGS instead of fundamental matrices. They also proposed the method to dynamically calibrate the rotating and zooming cameras from the natural features in the scene without special markers.

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) [7, 14], manual preprocessing is needed [13, 19, 6], or they are focusing on the quality of the generated image rather than the processing time [1].

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 [9, 10, 21, 12]. Among all these visual hulls methods, the image-based visual hulls presented by Matusik et al.[12] 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. [23] 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 8 computers for 64 cameras and additional hardware.

Some plane-sweep implementations achieve online rendering using graphics processing unit (GPU). The plane-sweep algorithm introduced by Collins[2] was adapted to online rendering by Yang et al. [24]. They computed new views in real-time from five cameras using four computers. Geys et al.[3] 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 [15] introduced a plane-sweep implementation for moving camera where all the input cameras are calibrated in real-time using ARToolkit [8] markers.

Our method belongs to the online VBR group based on plane-sweep algorithm. The main difference is that in the previous works [2, 24, 3, 15] they assume that cameras are strongly calibrated. This paper present a new method for online video-based rendering from uncalibrated cameras using our proposed plane-sweep algorithm in projective grid space.

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 allow 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) [18] 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. Figure 1 shows the definition of PGS. x and y axis in the image of basis camera 1 corresponds to the P and Q axis, 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

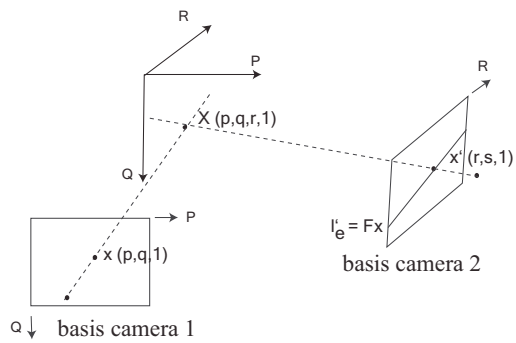

Figure 1. Definition of Projective Grid Space.

3D point, $\mathbf{x}'$ must lie on the epipolar line of $\mathbf{x}$. Thus, s coordinate of $\mathbf{x}'$ is determined from ${\mathbf{x}'}^T \mathrm{F}\mathbf{x} = \mathbf{0}$ 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. Either fundamental matrices or trifocal tensor between basis cameras and non-basis camera can be used for that task. 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 is transfered to a non-basis camera by either intersection of epipolar lines computed from fundamental matrices or point transfer by trifocal tensor (figure 2).

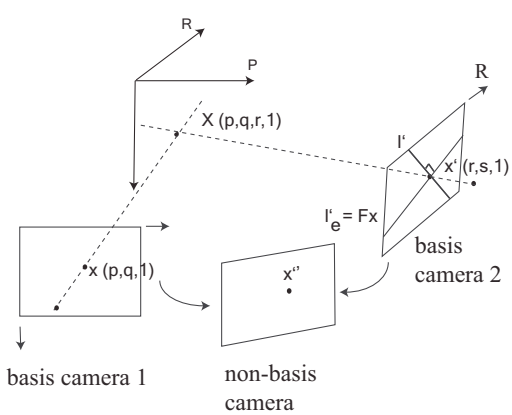

Figure 2. Weakly calibrating non-basis camera using trifocal tensor.

However point transfer using fundamental matrices gives less accuracy if a 3D point lies near the trifocal plane (plane that defined by three camera centers). For example, if three cameras are in the same horizontal line, 3D points in front of cameras will lie on trifocal plane. Even in the less severe case, transfered point will also become inaccurate for the points lying near this plane. Thus, trifocal tensors are used for weakly calibrating non-basis cameras in our implementation of PGS. For more detail of using fundamental matrices for point transfer please refer to [18].

3.1. Weakly calibrating non-basis camera using trifocal tensor

Trifocal tensor τ_i^{jk} is a homogeneous 3×3×3 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 7 point correspondences are necessary to estimate the trifocal tensor using the normalized linear algorithm [4].

Given point correspondence $\mathbf{x}$ and $\mathbf{x}'$, we can find corresponding point $\mathbf{x}''$ in the third camera by equation (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 figure 2, given a 3D point $\mathbf{X} = (p, q, r, 1)^T$ in PGS and tensor τ 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 ${\mathbf{x}'}^T \mathrm{F}\mathbf{x} = \mathbf{0}$.

2. Compute epipolar line $\mathbf{l}'_e = (l_1, l_2, l_3)^T$ of $\mathbf{x}$ on basis camera 2 from $\mathbf{l}'_e = \mathrm{F}\mathbf{x}$.

3. Compute line $\mathbf{l}'$ which pass $\mathbf{x}'$ and perpendicular to $\mathbf{l}'_e$ by $\mathbf{l}' = (l_2, -l_1, -rl_2 + sl_1)^T$.

4. The transfered 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 the 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 π_k as depicted in figure 3.

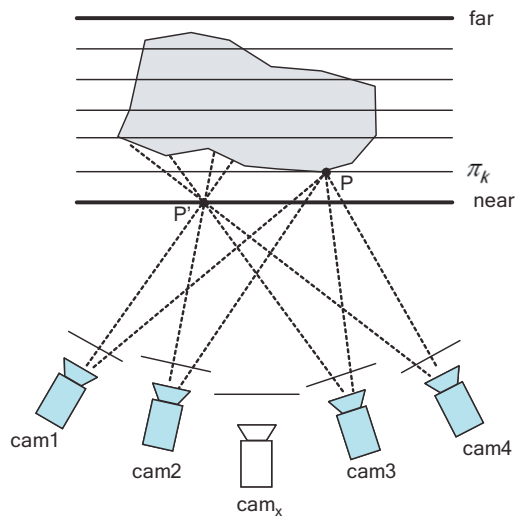

Figure 3. Plane-sweep algorithm in the Euclidean space.

Plane-sweep algorithm is based on the following assumption: a point lying on a plane π_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 π_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. Figure 3 illustrates this principal idea of the plane-sweep algorithm.

During the new view creation process, every plane π_k is computed in a back to front order. Each point P of a plane π_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 cameras 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 π_{k+1} is computed. The final new view image is obtained once every plane has been computed. This method is detailed on [16].

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 projected 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 allows 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. Figure 4 illustrate this definition. In figure 4, a ratio r equals to 0 (respectively 1) means the virtual camera has the same position as camera 1 (respectively camera 2).

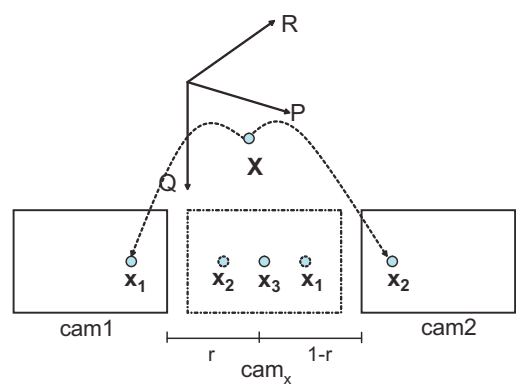

Figure 4. Defining virtual camera in Projective Grid Space.

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 camera 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 figure 4.

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

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 figure 5. 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 the color consistency testing in plane-sweep algorithm because every planes 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.

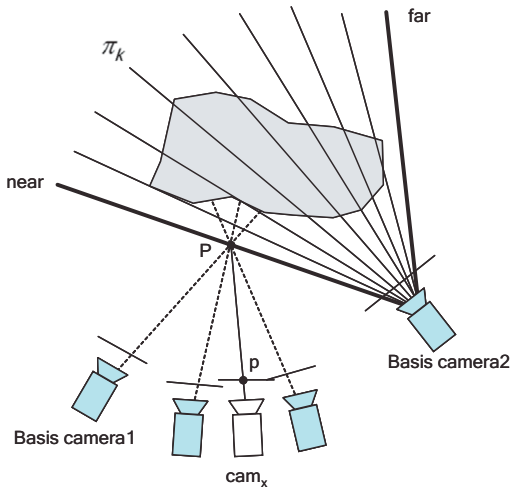

Figure 5. Defining planes for doing plane-sweep in Projective Grid Space.

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 π_k on a point P, we want to find the projection of P on every input image for the score computation step. As illustrated in Figure 6, the projection of 3D point P lying on π_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 figure 6. 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.

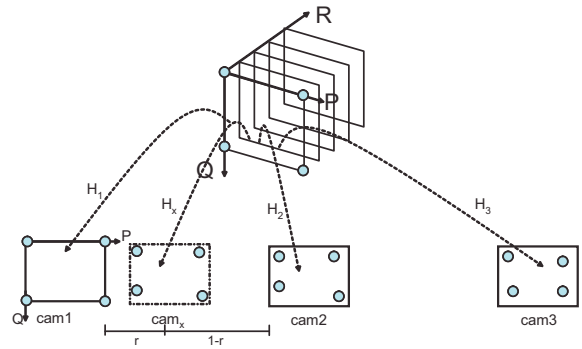

Figure 6. Estimating homography matrices for plane-sweep

Reset color consistency score of the virtual camera to the max value.
foreach *plane* π_k *in PGS* **do**
  foreach *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 : $color_p = \frac{1}{n}\sum_{j=1..n} c_j$
    • compute color consistency score from variance: $score_p = \sum_{j=1..n}(c_j - color_p)^2$
    if $score_p$ *is lower than current score of pixel p* **then**
      update score and color on virtual camera to $score_p$ and $color_p$.
    end
  end
end

Algorithm 1: Plane-sweep algorithm in Projective Grid Space.

In algorithm 1, we use the score function proposed in [16].

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 application 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 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 fragment shader, we apply the homography and compute the color consistency score as described in algorithm1. 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 $\mathrm{gl_FragDepth}$ while the average color is sent via $\mathrm{gl_FragColor}$. 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 2.00 GHz CPU with graphic card NVIDIA GeForce 8600M GT. Five Logitech fusion webcams with a resolution 320x240 are used to capture input videos. The camera setting is as figure 7. We select two cameras for defining Projective Grid Space as figure 7.

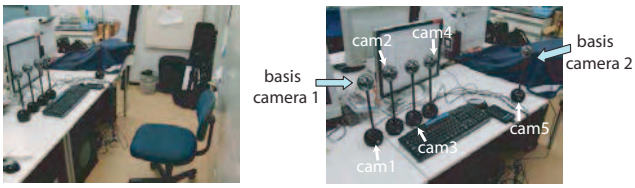

Figure 7. Camera configuration.

Fundamental matrix between camera 1 and 5, three trifocal tensors defined by camera 1,5,2, camera 1,5,3 and camera 1,5,4 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 [11].

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 number of 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 become satisfied.

Table 1 shows the number of planes and the running time for rendering new view images using 5 webcams implemented on CPU and GPU. Both implementations are tested on Intel(R) Core(TM) 2 Duo 2.00 GHz CPU with graphic card NVIDIA GeForce 8600M GT. 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 60 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.

	Number of planes					
	40	50	60	70	80	90
CPU	0.096	0.078	0.066	0.057	0.050	0.046
GPU	30.54	30.06	29.58	20.71	20.68	15.96

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

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 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.

Figure 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 20 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 give a good visual quality and fast enough for online VBR applications.

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.

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 [20] and PSNR (Peak Signal to Noise Ratio) 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 2 as a ground-truth reference. Camera 2 is leaved-out from plane-sweep

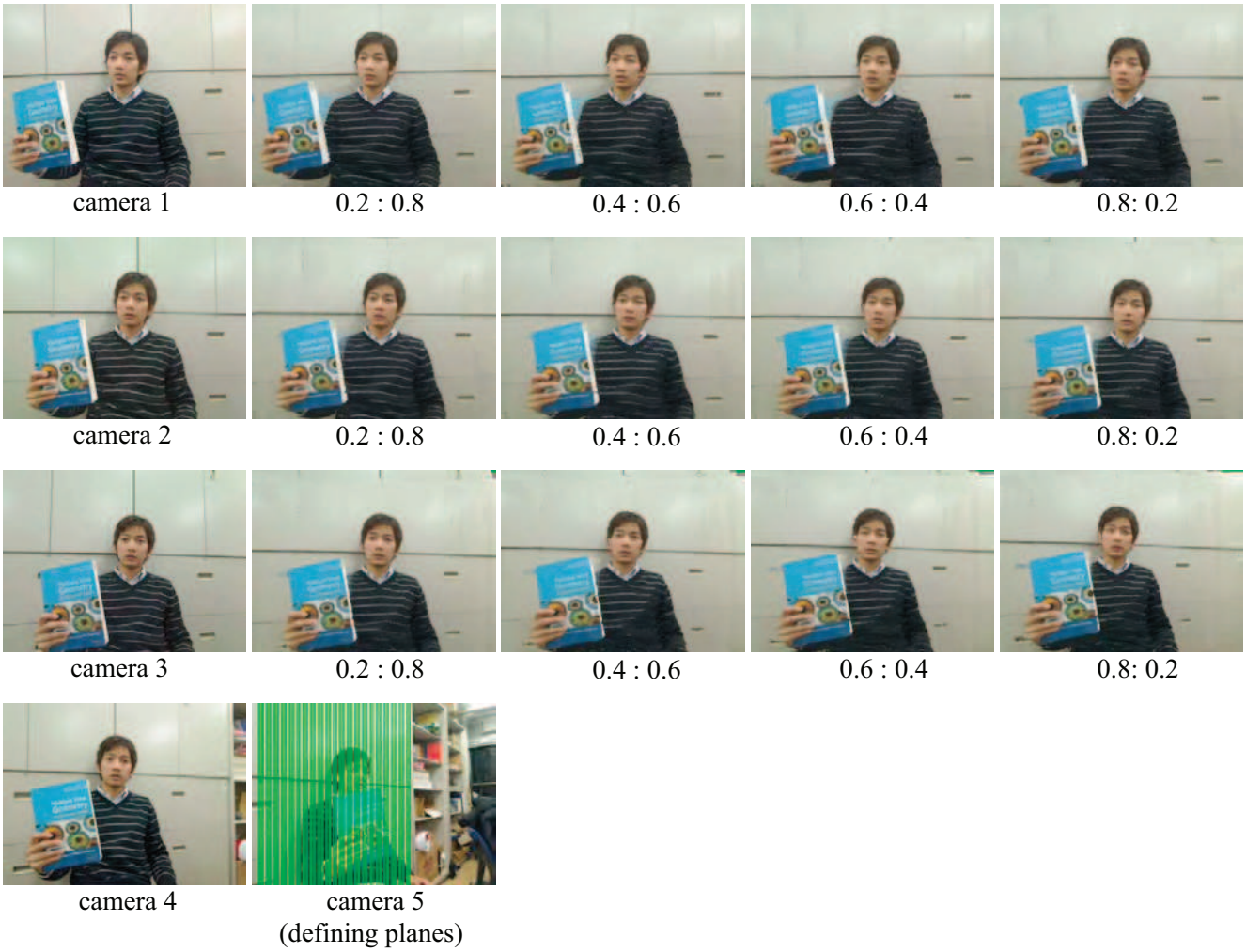

Figure 8. Result new view images from our proposed plane-sweep algorithm in projective grid space using 80 planes.

algorithm and views at that camera are synthesized. Figures 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.

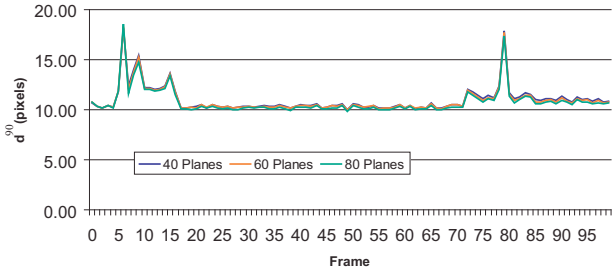

Figure 9. d^{90} registration error of new view images.

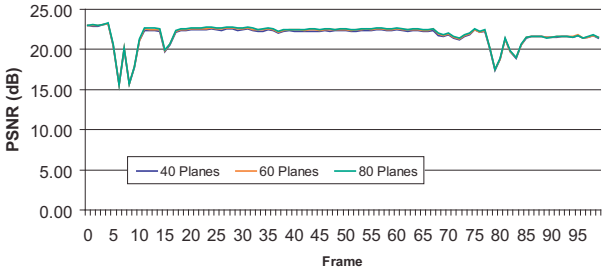

Figure 10. PSNR registration error of new view images.

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),

Number of planes	$d^{90}(pixels)$	$PSNR(dB)$
40 planes	11.000	21.738
60 planes	10.929	21.838
80 planes	10.788	21.909

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

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. This is impossible for the case of the normal plane-sweep algorithm in the Euclidean space in which strong calibration 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.

Acknowledgment

This work has been supported by "Foundation of Technology Supporting the Creation of Digital Media Contents" project (CREST, JST), Japan.

References

[1] J. Carranza, C. Theobalt, M. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. In *Proceedings of ACM SIGGRAPH'03*, pages 569–577, 2003.

[2] 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*, pages 358–363, 1996.

[3] I. Geys, S. Roeck, and L. Gool. The augmented auditorium: Fast interpolated and augmented view generation. In *Proceedings of the 2nd IEE European Conference on Visual Media Production*, pages 94–103, 2005.

[4] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[5] Y. Ito and H. Saito. Free-viewpoint image synthesis from multiple-view images taken with uncalibrated moving cameras. In *The IEEE International Conference on Image Processing (ICIP)*, pages 29–32, September 2005.

[6] S. Jarusirisawad and H. Saito. 3DTV view generation using uncalibrated pure rotating and zooming cameras. *Image Communication*, 24(1–2):17–30, January 2009.

[7] T. Kanade, P. W. Rander, and P. J. Narayanan. Virtualized reality: concepts and early results. In *IEEE Workshop on Representation of Visual Scenes*, pages 69–76, 1995.

[8] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, pages 85–94, 1999.

[9] M. Li, M. Magnor, and H. peter Seidel. Hardware-accelerated visual hull reconstruction and rendering. In *Proceedings of Graphics Interface 2003*, pages 65–71, 2003.

[10] M. Li, M. Magnor, and H. Seidel. Online accelerated rendering of visual hulls in real scenes. *Journal of WSCG*, 11(2):290–297, 2003.

[11] B. Matei and 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*, volume 2, pages 18–25, 2000.

[12] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *Proceedings of ACM SIGGRPAH'00*, pages 369–374, 2000.

[13] S. Moezzi, A. Katkere, D. Y. Kuramura, and R. Jain. Reality modeling and visualization from multiple video sequences. *IEEE Computer Graphics and Applications*, 16(6):58–63, 1996.

[14] S. Moezzi, L.C.Tai, and P.Gerard. Virtual view generation for 3D digital video. *IEEE Transaction on MultiMedia*, 4(1):18–26, 1997.

[15] V. Nozick and H. Saito. Real-time free viewpoint from multiple moving cameras. In *Advanced Concepts for Intelligent Vision Systems (ACIVS)*, pages 72–83, 2007.

[16] V. Nozick and H. Saito. On-line free-viewpoint video : From single to multiple view rendering. *International Journal of Automation and Computing (IJAC)*, 5(3):257–267, 2008.

[17] M. Okutomi and T. Kanade. A multiple-baseline stereo. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[18] H. Saito and 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)*, volume 2, pages 49–54, June 1999.

[19] J. Starck and A. Hilton. Model-based multiple view reconstruction of people. In *Proceedings of IEEE International Conference on Computer Vision (ICCV'03)*, pages 915–922, Washington, DC, USA, 2003. IEEE Computer Society.

[20] J. Starck, J. Kilner, and 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*, pages 225–228, May 2008.

[21] C. Theobalt, M. Li, M. Magnor, and H.-P. Seidel. A flexible and versatile studio for multi-view video recording. In *Proceedings of Vision, Video and Graphics 2003*, pages 9–16, Bath, UK, July 2003.

[22] S. Yaguchi and H. Saito. Arbitrary viewpoint video synthesis from multiple uncalibrated cameras. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34(1):430–439, 2004.

[23] J. Yang, M. Everett, C. Buehler, and L. McMillan. A real-time distributed light field camera. In *Proceedings of the 13th Eurographics Workshop on Rendering*, pages 77–86, 2002.

[24] R. Yang, G. Welch, and 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)*, page 225, Washington, DC, USA, 2002. IEEE Computer Society.