Special issue: 3D image and video technology

Research Paper

Diminished reality using plane-sweep algorithm with weakly-calibrated cameras

Songkran JARUSIRISAWAD[1], Takahide HOSOKAWA[2], and Hideo SAITO[3]
[1,2,3]*Department of Infomation and Computer Science, Keio University*

ABSTRACT

We present a plane-sweep algorithm for removing occluding objects in front of the objective scene from multiple weakly-calibrated cameras. Projective grid space (PGS), a weak cameras calibration framework, is used to obtain geometrical relations between cameras. Plane-sweep algorithm works by implicitly reconstructing the depth maps of the targeted view. By excluding the occluding objects from the volume of the sweeping planes, we can generate new views without the occluding objects. The results show the effectiveness of the proposed method and it is fast enough to run in several frames per second on a consumer PC by implementing the proposed plane-sweep algorithm in graphics processing unit (GPU).

KEYWORDS

Diminished reality, plane-sweep, real-time, video-based rendering, Graphics processing unit (GPU)

1 Introduction

Augmented reality (AR) [1], [2] is a term of the technology that composites the computer generated image to the real image viewed by the user. AR supplements the computer generated image to the real scene, rather than completely replacing it. Ideally, the rendered virtual object should coexist in the scene realistically so that the viewer cannot distinguish between the real and the virtual ones.

In contrast, the goal of diminished reality research (so-called mediated reality [3]) is to generate the images in which some real objects are deliberately removed. This means that whatever is behind the object should be rendered when the occluding objects are removed. The information about occluded area normally comes from either the other camera views or from the same camera view but at the different frames in which the occluded area is seen.

This paper proposed a new method of diminished reality from multiple weakly-calibrated cameras using plane-sweep algorithm. Plane-sweep algorithms have been proposed in the literatures [4]–[7] to generate free viewpoint videos. In this paper, we adapted plane-sweep algorithm to use for diminished reality applications with weakly-calibrated cameras. Fig. 1 illustrates what our proposed system does. From several cameras, user can create new view images between any cameras while also remove the occluding object to reveal the

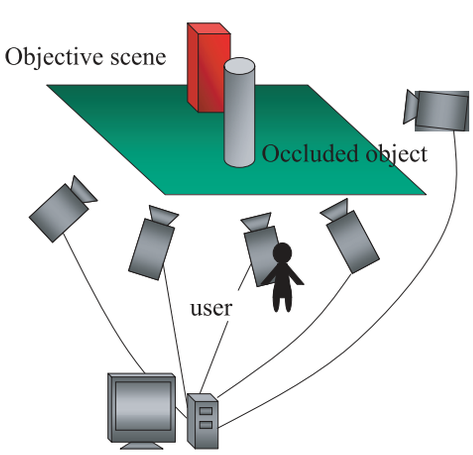


Fig. 1 Overview of the proposed system.

Received September 30, 2009; Revised December 15, 2009; Accepted January 4, 2010.
[1)] *songkran@hvrl.ics.keio.ac.jp*, [2)] *hosokawa@hvrl.ics.keio.ac.jp*, [3)] *saito@hvrl.ics.keio.ac.jp*
DOI: 10.2201/NiiPi.2010.7.3

hidden area behind.

We utilize projective grid space (PGS) [8], a weak cameras calibration framework, to obtain the geometrical relation between multiple cameras. Thus, our method does not need the information about cameras' intrinsic parameters or the Euclidean information of a scene. The proposed plane-sweep algorithm in PGS can render image in which the occluding object is removed at any viewpoint between two reference views. We also implemented our method on GPU to reduce the computation time.

The rest of this paper is organized as follows. We firstly survey about previous works of diminished reality. Then projective grid space framework that we use for weak cameras calibration is explained in section 3. We describe conventional plane-sweep algorithm in the Euclidean space that was used for rendering free viewpoint video in section 4. Then we explain our proposed plane-sweep algorithm on PGS for diminished reality application in section 5. Section 6 presents the implementation detail on GPU. Finally, we show the experimental results and conclusion.

2 Related works

Several researchers have used "Diminished Reality" term in the past. Mann and Fung [9] proposed the method in which planar objects are removed from a video sequence and replaced with another texture. Wang and Adelson [10] divided video sequence from a single camera into several layers using motion analysis. The occluding layer then can be simply removed and replaced with the layers below. Lepetit and Berger [11] proposed a semi-interactive method for outlining the occluding objects over a video sequence for diminished reality applications. The user has to outline the occluding object in some key-frames, then the algorithm can predict the contours in other frames. Hill et al. implemented diminished reality platform that runs in real-time using graphics processing unit (GPU) [12].

In the mention works, single camera is used to capture the scene. Information that is used to fill the area hidden by the occluding object comes from the other frames of the same camera where the occluded area is seen.

There are also several works that proposed method for diminished reality from multiple cameras. Zokai et al. [13] proposed a paraperspective projection model to generate the background behind the occluding object from multiple calibrated views. However, it requires the user to select the regions of obstacles which is not suitable for dynamic scenes.

Jarusirisawad et al. [14] proposed a diminish reality system that can remove the occluding object while also render free viewpoint images from pure rotation and zoom cameras. The limitation is that the static background must be approximated and segmented to several planes manually at the first frame. Silhouette of occluding object must also be labeled by the user.

The mentioned diminished reality researches have some restrictions in each work. For example, they assume that cameras must be previously calibrated, occluding objects or the objective scene must not move or manual operation is necessary.

In this paper, we present a new method for online diminished reality. It allows both occluding objects and objective scenes to move. Moreover, we do not need to know intrinsic parameters of the cameras. This is different from most of the previous works which assume that cameras are strongly calibrated.

This paper is the extension of our previous work in [15]. In [15], the rendered views are limited to the viewpoint of the capturing cameras and the background image must be captured in advance. In this paper, our method does not need to capture the background image beforehand and the rendered images can be synthesized at the virtual viewpoint between real cameras.

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 onto 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 the Euclidean information of a scene.

Projective grid space (PGS) [8] 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. 2 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

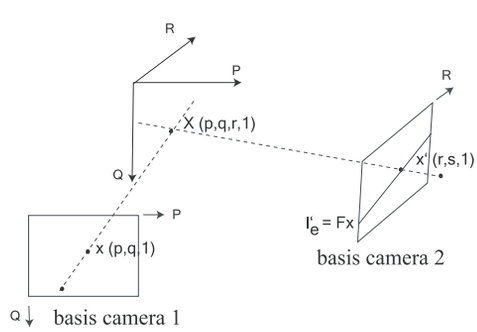


Fig. 2 Definition of Projective Grid Space.

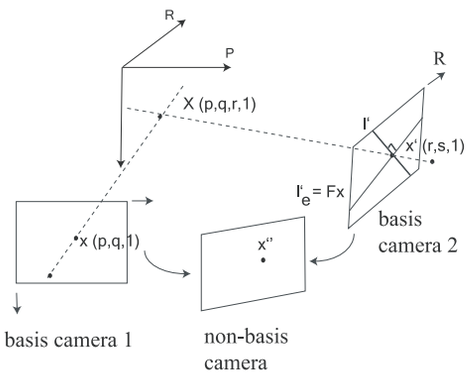


Fig. 3 Weakly calibrating non-basis camera using trifocal tensor.

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 $\mathbf{x}'^T\mathrm{F}\mathbf{x} = \mathbf{0}$ 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 a 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 transferred to a non-basis camera by either intersection of epipolar lines computed from fundamental matrices or point transfer by trifocal tensor (Fig. 3).

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 [8].

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. For more information about the tensor notation, please refer to Appendix.

Trifocal tensor can be estimated 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 [16].

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 Fig. 3, 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

Before explaining about our contribution of the plane-sweep algorithm in projective grid space for diminished reality in Section 5, this section gives a general idea about conventional plane-sweep algorithms in the Euclidean space of the calibrated cameras.

The plane-sweep algorithm was previously proposed for creating novel views of a scene from several input images [4], [6], [7], [17]. 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 Fig. 4.

Plane-sweep algorithm is based on the following as-

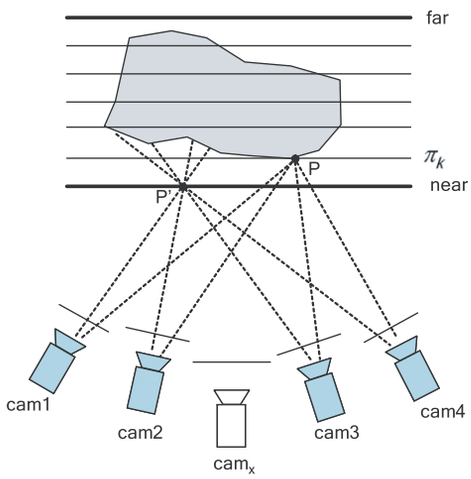


Fig. 4 Plane-sweep algorithm in the Euclidean space.

sumption: 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 an object in a scene lying on one of these planes π_k at point P, this point will be seen by every camera as the same color, i.e., the object color. Now consider another point P' lying on a plane but not on the surface of the object, this point will probably not be seen by the capturing cameras as the same color. Fig. 4 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 on a plane π_k is projected onto input images. A score and a representative color are 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 π_{k+1} is computed. The final new view image is obtained once every plane has been computed. This method is detailed in [4], [6], [7] with some variations in score functions and constraints.

5 Plane-sweep in projective grid space for diminished reality

In this paper, the goal is not only to generate novel view images but also to remove unwanted objects from the output images. We propose a method that apply plane-sweep algorithm to use with projective grid space (weakly-calibrated cameras) for diminished reality application. From the user point of view, they can easily remove occluding object from the rendered image by defining near and far planes to exclude unwanted objects from this volume. We proposed a score computation of color consistency that take into account the outlier colors of occluding objects.

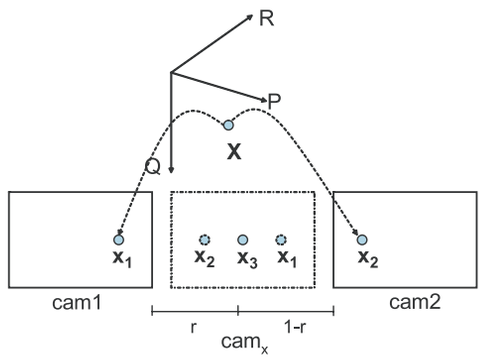


Fig. 5 Defining virtual camera in Projective Grid Space.

To do a plane-sweep in PGS, we need to define a position of a virtual camera, define the planes in 3D space and then compute new view images. These steps are different from conventional plane-sweep algorithm in the Euclidean space because in our case, cameras' intrinsic parameters are unknown. In this section, we describe the detail of each step.

5.1 Defining virtual camera position

To perform a plane-sweep algorithm, 3D point on a plane must be 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. Fig. 5 illustrate this definition. In Fig. 5, 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 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 Fig. 5.

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

Note that we use view interpolation [18] that does not

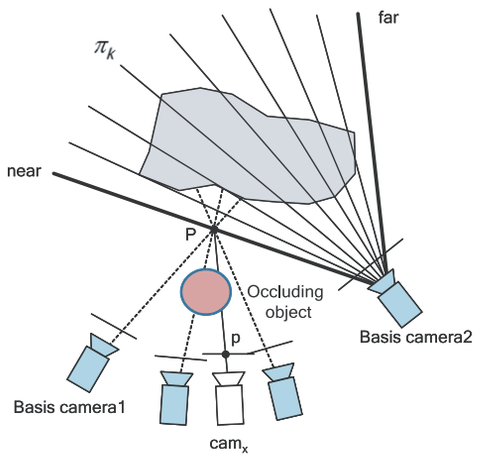


Fig. 6 Defining planes for doing plane-sweep in Projective Grid Space.

guarantee the physical validity of the scene. To get a physically valid rendered view, a prewarp step is necessary [19], [20]. However, cameras in our system are set to be horizontal, and be parallel to each other. Thus the rendered image does not have much distortion even without such a prewarp step.

5.2 Defining planes in PGS

We define the *near* and *far* planes along the R axis (x image coordinate of basis camera 2) as shown in Fig. 6. 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. These planes are adjusted so that the occluding object is excluded from the volume defined by these planes.

This is different from the case of the normal plane-sweep algorithm in the Euclidean space in which full calibration is used. In that case, the actual depth of a scene has to be measured so that *near* and *far* planes cover only 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.

5.3 Computing new view images and removing occluding objects

If pixel p in a virtual camera is back projected to a plane π_k at a point P, we want to find the projection of P on every input image to decide the color of pixel$p$ based on the color observed in the other cameras. As illustrated in Fig. 7, the projection of 3D point P lying on π_k on input image i can be performed by a homography matrix $\mathbf{H}_i$. Thus, the projection p_i of a 3D point

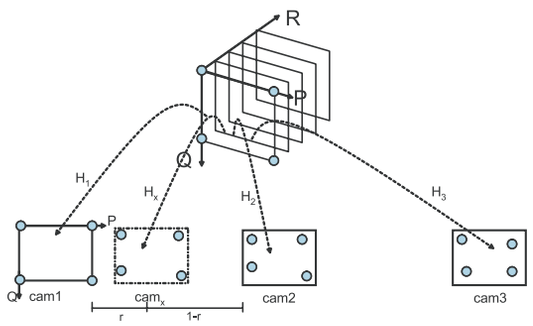


Fig. 7 Estimating homography matrices for plane-sweep.

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 image corners of the basis camera 1 as shown in Fig. 7. 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 to reduce the computation time.

Considering Fig. 6, the pixel color of the projection of point P the lies on the object's surface may not be the color of point P due to occlusion. Virtual view rendering algorithm must determine and exclude the color of the occluding object when rendering new view image. Our score function that we use is motivated by [6].

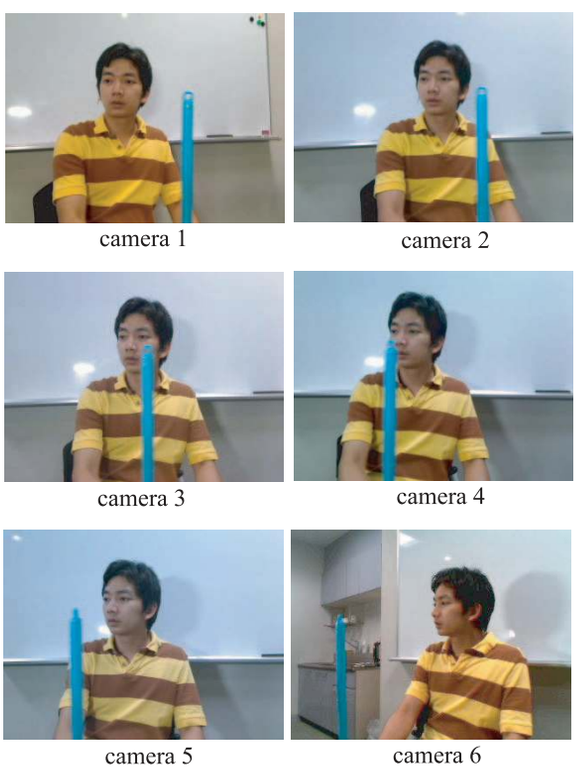
Algorithm 1 summarizes the score function that we used to compute new views while remove unwanted object from the scene. The algorithm iterates by computing color consistency score and average color. At the end of iteration, the view that has the color farthest from the average color is removed (this view probably see the occluding object's color). Then the algorithm starts to compute color consistency score and average color again from the current views set. The algorithm terminates when it is confident with the current result (color consistency score is less than some threshold) or when there are two views or less in the current iteration.

Our color consistency score is defined by a variance of colors from current views set plus the constance k times the number of excluded views. The latter term is added to bias the score so that with an equal variance, the color from the set that has a higher cardinal is preferred.

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

forech *pixel p in* cam_x **do**
- $\bullet\ score_p = \infty$
- $\bullet$ project pixel p to n input images excluding basis camera 2 . c_j is the color from this projection on the j-th camera
- **forech** *plane* π_k *in PGS* **do**
  - $\bullet\ S = \{1, 2, .., n\}$
  - **repeat**
    - $\bullet$ compute the average color : $color_S = \frac{1}{|S|} \sum_{j \in S} c_j$
    - $\bullet$ compute the color consistency score : $score_S = \sum_{j \in S} (c_j - color_S)^2 + k(n - |S|)$
    - **if** $score_S < score_p$ **then**
      - $\bullet\ score_p = score_S$
      - $\bullet\ color_p = color_S$
    - **end**
    - $\bullet\ S = S - \{c_f\}$ where c_f is the farthest color from $color_S$
  - **until** $score_p < Threshold \ || \ |S| \leq 2$
- **end**

end



camera 1 camera 2

camera 3 camera 4

camera 5 camera 6

Fig. 8 Images from camera 1 to 6.

Camera 6 (defining planes) Rendered image (the same view as camera 3)

Fig. 9 Result when defining planes to cover all objects in the scene.

Camera 6 (defining planes) Rendered image (the same view as camera 3)

Fig. 10 Result when defining planes to cover only occluded object.

Occluding object in a scene can be removed by defining near and far planes so that occluding object does not lie in these planes. Different alignment of planes gives the different rendering results. To illustrate this effect, we show example results that are generated from the different planes defining. Six cameras are used to capture input images. Camera 6 is selected as a camera for defining planes. Fig. 8 shows input images from all cameras.

Figs. 9 and 10 show the different results when define planes differently. In Fig. 9, planes are defined to include the whole scene while in Fig. 10, planes are define to exclude the occluding object. We can see that the objects lying outside the planes are removed from the rendered image using our proposed method.

6 Implementing plane-sweep algorithm on GPU

To achieve a fast computation (several frames per second), we implemented 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.

The algorithm 1 is a pseudo code that is easy to read when think of a processing as a single thread. However, when this algorithm is implemented on GPU, this algorithm must be converted to suite the predefined ren-

dering pipeline of the GPU. There are several ways to convert this abstract algorithm to a shader program on GPU so we explain our actual implementation in this section.

We use OpenGL and GLSL in our implementation. Input images are transfered to GPU as multi-textures. In the drawing function of OpenGL, we do N-pass rendering from near to far plane where N is the number of planes. We use Orthographic projection and draw square to cover the whole image of virtual camera. In fragment program of GLSL, we can think that it is a processing of a pixel p with the k-th plane as in algorithm 1. In fragment program, we have to find the projection of this pixel p on all cameras. To do that, we compute homographies as Equation 4 for each rendering pass in OpenGL and sent them to GPU as texture matrices.

We apply these homographies in the fragment program to compute the projection on all cameras and compute the color consistency score as described in algorithm1. Fragment color is assigned to be an average color from the best views set. 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 which the occluding object is also removed in the frame buffer.

7 Experimental results

In this section, we show both qualitative and quantitative results of our proposed method. Fig. 11 shows experimental setup.

We used the following hardware in the experiment.

- CPU: Intel Core2 DUO 3.0 GHz
- GPU: NVIDIA Quadro FX 570
- Webcams: Logicool Qcam Orbit QVR-1

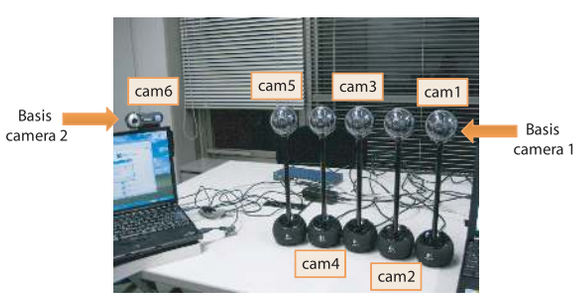


Fig. 11 Experimental setup.

2D-2D correspondences for estimating relationship among cameras can be selected from feature points in a scene. In our experiment we wave a marker around a scene and track the features for 2D-2D correspondence before start rendering. In future work, we plan to find these correspondences from natural features in a scene in real-time so that the cameras can also be moved freely during rendering.

In our experiment, we did not use hardware synchronized cameras but just consumer web cameras. Because the object did not move so fast, even without synchronization our results still seem acceptable. In case that the object move fast so that the images captured by the multiple cameras become clearly unsynchronized, more artifacts will appear in the result videos. In this case, using the cameras with synchronization mechanisms would solve the problem.

7.1 Running time

The computation time for rendering output image depends on the number of cameras and planes that are used for plane-sweep algorithm. The appropriate number of planes varies depending on the complexity of a scene. Using higher number of planes makes processing time become longer but usually gives a better result. In our experiment, it is shown that using 60 planes or more makes the visual result become satisfied.

Table 1 shows the running time for rendering output images using different number of planes. 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 other processing.

7.2 Qualitative evaluation

In this experiment, the objective is to remove a moving stick from the output image and reveal the occluded scene. We used six webcams with resolutions 320×240 pixels. Output view was selected to be the same view as camera 3 to compare with the input images. Fig. 12 shows input and output images from our method.

From the results, even not all part of occluding object was perfectly removed, it can be said that 3D object behind the occluding object is correctly reconstructed.

Table 1 Frame rates (frame/sec.) when using six cameras.

	Number of planes				
	20	40	60	80	100
Frame rates	10.00	6.67	3.49	2.47	1.76

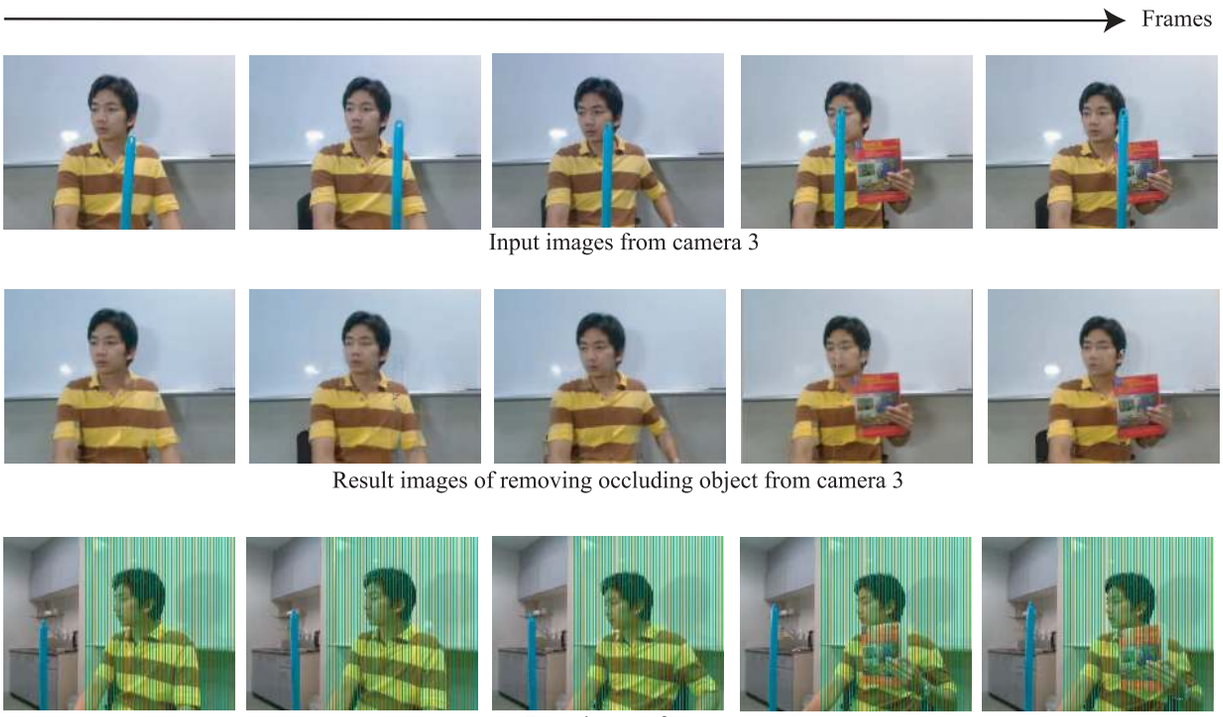


Fig. 12 Result of removing occluding object from input camera 3.

Fig. 13 Rendered images at different views from the input cameras. Six input images used for rendering these results are the same as depicted in Fig. 8.

In the experiment, we used 80 planes for doing plane-sweep. The average processing time was 2.5 fps.

Our proposed method can also remove occluding object from the different views other than on the input views as describe in Section 5. Fig. 13 shows the rendering results at different viewpoint from the input cameras.

7.3 Quantitative evaluation

This section gives quantitative quality measurements of our result. We used the scene that consists of occluding object moving in front of a static background. By using static background, we can have ground-truth references to measure the accuracy of results. We used our method to remove the occluding object from the input images and compare with the ground-truth.

Views at one selected camera was rendered and compare with ground-truth. PSNR (Peak Signal to Noise Ratio) are computed to measure the errors in the rendered images for 100 consecutive frames. Fig. 15 shows the PSNR of our results respect to the number of planes used in scene reconstruction. Table 2 shows the average PSNR over 100 frames.

Fig. 14 shows the different of the results when using different number of planes to reconstruct and render output images in which occluding object is removed.

From the results, it is shown that increasing the number of planes in reconstruction gives a better result. However, when enough planes has already been used, increasing the number of planes would not give a significant improvement.

Fig. 14 Comparison of using the different number of planes to render output images.

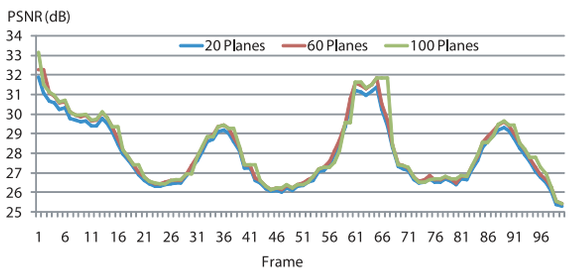


Fig. 15 PSNR error of the rendered images.

Table 2 PSNR error of the rendered images using different number of planes.

Number of planes	Average PSNR(dB)
20 planes	28.22
60 planes	28.43
100 planes	28.49

8 Conclusion

In this paper we present a new online rendering method for diminished reality using plane sweep algorithm. By utilizing projective grid space (PGS), our method has mainly two advantages over doing plane-sweep in the Euclidean space. Firstly, our method does not need information about cameras' intrinsic parameters. Secondly, *Near* and *far* planes in PGS are easily defined since they are visualized from the image of basis camera 2. Plane-sweep algorithm in the Euclidean space has to define these planes in the real 3D coordinates which is sometimes difficult to measure or visualize. Our system can render images at several frames per second thanks to the implementation of plane-sweep algorithm on GPU. The results show the effectiveness of our proposed method.

References

[1] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators and Virtual Environments*, vol.6, no.4, pp.355–385, 1997.

[2] R. T. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, "Recent advances in augmented reality," *IEEE Computer Graphics and Applications*, vol.21, no.6, pp.34–47, 2001.

[3] S. Mann, 'mediated reality'. TR 260, M.I.T. Media Lab Perceptual Computing Section, Cambridge, Massachusetts, 1994.

[4] 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*, pp.358–363, 1996.

[5] S. Jarusirisawad, H. Saito, and 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'09)*, Kyoto, Japan, October 2009.

[6] V. Nozick, S. Michelin, and D. Arques, "Real-time plane-sweep with local strategy," *Journal of WSCG*, vol.14, no.1–3, pp.121–128, 2006.

[7] 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)*, p. 225, Washington, DC, USA, 2002. IEEE Computer Society.

[8] 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)*, vol.2, pp.49–54, June 1999.

[9] S. Mann and J. Fung, "Videoorbits on eye tap devices for deliberately diminished reality or altering the visual perception of rigid planar patches of a real world scene," In *International Symposium on Mixed Reality (ISMR2001)*, March 14–15 2001.

[10] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," In *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, September 1994.

[11] V. Lepetit and M.-O. Berger, "A semi-interactive and intuitive tool for outlining objects in video sequences with application to augmented and diminished reality,"

In *Proceedings of International Symposium on Mixed Reality*, Yokohama, Japan, March 2001.

[12] R. Hill, J. Fung, and S. Mann, "A parallel mediated reality platform," In *ICIP*, pp.2865–2868, 2004.

[13] Y. G. Saivash Zokai, Julien Esteve, and N. Navab, "Multiview paraperspective projection model for diminished reality," In *Proceedings of the Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003)*.

[14] S. Jarusirisawad and H. Saito, "Diminished reality via multiple hand-held cameras," In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2007)*, pp.251–258, Vienna, Austria, September 2007.

[15] T. Hosokawa, S. Jarusirisawad, and H. Saito, "Online video synthesis for removing occluding objects using multiple uncalibrated cameras via plane sweep algorithm," In *Proceedings of ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC 2009)*, Como, Italy, August 2009.

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

[17] 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*, pp.94–103, 2005.

[18] S. Chen and L. Williams, "View interpolation for image synthesis," In *Proceedings of ACM SIGGRAPH'93*, pp.279–288, 1993.

[19] S. Seitz and C. Dyer, "View morphing," In *SIGGRAPH 96*, pp.21–30, 1996.

[20] S. M. Seitz and C. R. Dyer, "Physically-valid view synthesis by image interpolation," In *Proceedings of the IEEE Workshop on Representations of Visual Scenes*, pp.18–25, 1995.

Appendix
Tensor notation

This appendix gives an introduction to the tensors for the reader who is unfamiliar with tensor notation. For more details, please refer to [16].

A tensor is a multidimensional array that extends the notion of scalar, vector and matrix. A tensor is written using an alphabet with contravariant (upper) and covariant (lower) indexes. For example, the trifocal tensor τ_i^{jk} has two contravariant indexes and one covariant index.

Considering a representation of vector and matrix using tensor notation, entry at row i and column j of matrix $\mathbf{A}$ is written using tensor notation as a_j^i, index i being contravariant (row) index and j being contravariant (column) index. An image point represented by the homogeneous column vector $\mathbf{x} = (x^1, x^2, x^3)^T$ is written using tensor notation as x^i, while a line represented using the row vector $\mathbf{l} = (l_1, l_2, l_3)$ is written as l_i.

Writing two tensors together means doing a contraction operation. The contraction of two tensors produce a new tensor where each element is calculated from a sum of product over the repeated index. For example consider a matrix multiplication $\hat{\mathbf{x}} = \mathbf{Ax}$, this can be written using tensor notation as $\hat{x}^i = a_j^i x^j$. This notation imply a summation over the repeated index j as $\hat{x}^i = \sum_j a_j^i x^j$.

Songkran JARUSIRISAWAD

Songkran JARUSIRISAWAD received the B.E. (First class honors) degree in computer engineering from Chulalongkorn University, Bangkok, Thailand, in 2005. He earned the M.E. and Ph.D. degrees in information and computer science from Keio University, Yokohama, Japan, in 2007 and 2009, respectively. Currently, he is working at Freewill FX Co., Ltd., Bangkok, Thailand. His research interests include computer vision, image based modeling and rendering.

Takahide HOSOKAWA

Takahide HOSOKAWA received B.E. degree in Information and Computer Science from Keio University, Japan, in 2009. He is now a graduate student of Graduate School of Science and Technology, Keio University, Japan. His research interests include video analysis and processing.

Hideo SAITO

Hideo SAITO received B.E., M.E., and Ph.D. degrees in Electrical Engineering from Keio University, Japan, in 1987, 1989, and 1992, respectively. He has been on the faculty of Department of Electrical Engineering, Keio University, since 1992. In 1997 to 1999, he stayed in the Robotics Institute, Carnegie Mellon University as a visiting researcher. Since 2006, he has been a Professor of Department of Information and Computer Science, Keio University. He is currently the leader of the research project "Technology to Display 3D Contents into Free Space", supported by CREST, JST, Japan. His research interests include computer vision, mixed reality, virtual reality, and 3D video analysis and synthesis. He is a senior member of IEEE, and IEICE, Japan.