Online Video Synthesis for Removing Occluding Objects Using Multiple Uncalibrated Cameras via Plane Sweep Algorithm

Takahide Hosokawa, Songkran Jarusirisawad and Hideo Saito
Keio University
3–14–1, Hiyoshi, Kohoku-ku, Yokohama-shi 223–8522, Japan
Email: {hosokawa, songkran, saito}@hvrl.ics.keio.ac.jp

***Abstract*—We present an online rendering system which removes occluding objects in front of the objective scene from an input video using multiple videos taken with multiple cameras. To obtain geometrical relations between all cameras, we use projective grid space (PGS) defined by epipolar geometry between two basis cameras. Then we apply plane-sweep algorithm for generating depth image in the input camera. By excluding the area of occluding objects from the volume of the sweeping planes, we can generate the depthmap without the occluding objects. Using this depthmap, we can render the image without obstacles from all the multiple camera videos. Since we use graphics processing unit (GPU) for computation, we can achieve real-time online rendering using a normal spec PC and multiple USB cameras.**

I. Introduction

Diminished reality is a technology for generating images in which some real objects are erased using images that captures the objective scene. We also can replace it with background scene so it is seen as if there were no obstacle in the space. Researches on diminished reality are divided into two groups. One is methods that use video sequences and the other is methods that use still images.

Mann and Fun [1] proposed the method in which plane objects are removed and replaced with another texture. Wang and Adelson [2] divided video sequence into several layers and removed one. Lepetit and Berger [3] detected occlusion persuing borders drawn by users and removed objects in the scene. These methods use temporal sequence of video so procedures are complicated and time-consuming.

Zokai et al [4] did not use temporal sequence of video but used three projective model of still images taken from different locations. This method enables to remove occluding objects using scene behind the object but it requires to appoint regions of obstacles and it does not suit for dynamic scenes where objects are moving.

As above many researches on diminished reality have been done but these always have some restrictions like occluding objects have to be fixed or objective scene should not move. Besides these procedures are limited to offline so users were not able to change the angle freely.

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 cameras whereas in the previous works cameras are strongly calibrated.

II. Method

In this section, we describe weak camera calibration framework for our plane-sweep method. We need to project 3D points into image frame of each camera including the virtual one to implement the plane-sweep algorithm.

Projective Grid Space (PGS) [5] allows to define that 3D space. It finds the projection without knowing intrinsic parameters of cameras or Euclidean information of scene.

A. Projective Grid Space

Projective Grid Space (PGS) is a 3D space defined by image coordinate of two arbitrarily selected cameras called basis camera 1 and basis camera 2. We call each 3D volume voxel and each line grid.

Figure 1 shows the relationship between Euclidian Grid Space and Projective Grid Space. Euclidian Grid Space is the 3D space which is defined by voxels all of which have the same volume no matter where cameras are. On the other hand, PGS has voxels which have diverse volumes. The farther it is from a camera, the larger the volume of the voxel is.

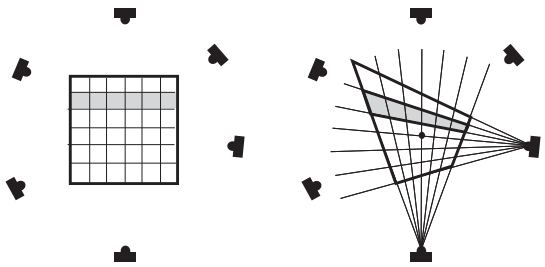


Fig. 1. Euclidian Grid Space and PGS

The definition of PGS is as follows. First we choose two cameras from several cameras. We call them basis camera 1 and basis camera 2. Then P and Q axes are defined as

978-1-4244-4620-9/09/$25.00

projections of X and Y axes in basis camera 1. R axis is also defined as a projection of X axis in basis camera 2. We denote the coordinate system in PGS by P-Q-R axis to distinguish this 3D space from the Euclidean one.

PGS is the 3D space defined by these three axes. Coordinates in PGS are defined by lines which connect a pixel in each image with point of view. P and Q axes in PGS correspond to X and Y axes in basis camera 1. R axis in PGS correspond to X axis in basis camera 2. These are shown in figure 2. We use this coordinate system for all other cameras.

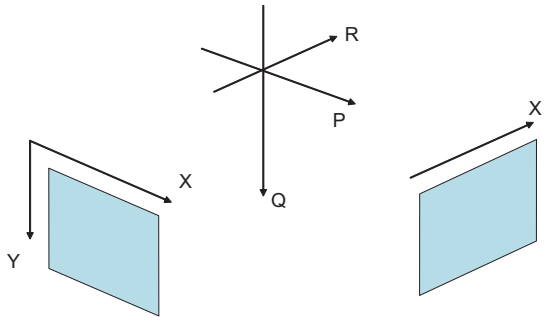


Fig. 2. Definition of PGS

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

B. Plane Sweep Algorithm in the Euclidean Space

The plane-sweep algorithm creates new views of a scene from several input images. This section explains the general idea of plane-sweep algorithm in the Euclidean Grid Space of calibrated cameras as in conventional methods first. After that, we shows the modification of plane-sweep algorithm where we use PGS.

First we consider scene where objects are exclusively Lambertian surfaces. we place the virtual camera cam_x somewhere around real video cameras. Then we define $near$ plane and far plane. Every object of the scene has to lie between these two planes. The space between $near$ and far planes is divided into several parallel planes π_k as depicted in figure 3.

Plane-sweep algorithm is based on the following assumption: a point lying on plane π_k provides projections on every input camera that are similar colors. These colors potentially correspond to the color of surface of an object. Considering a visible object of the scene lying on one of these planes π_k at a point P, it is seen by every camera with the same color, i.e. the object color.

Now we consider another point P_0 lying on a plane but not on a surface of an object. This point is not necessarily seen as the same color by other cameras. Figure 3 illustrates

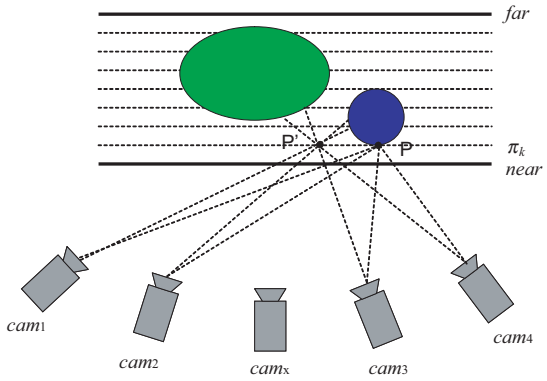


Fig. 3. Plane-sweep in the Euclidean space

this principal idea of the plane-sweep algorithm.

In the process of creating a new view, every plane π_k is computed in a back to front order. Each point P of plane π_k is projected onto the input images. Both score and representative color are computed according to the matching of the colors. Good score means that every camera captures similar colors. The computed scores and colors are projected onto the virtual camera cam_x. The pixel color in cam_x is updated only when the projected point P provides better score than the current one. Then the next plane π_{k+1} is computed. The final new view image is obtained when all the planes have been computed. This method is detailed on [6].

C. Plane Sweep Algorithm in PGS

Next we apply plane-sweep algorithm to PGS. We need to define a position of virtual camera to perform plane-sweep in PGS. In this section we describe the detail of each step.

We can define any arbitrary $near$ plane and far plane in PGS to perform plane-sweep. In our method we define planes along the R axis (x image coordinate of basis camera 2) as shown in figure 4.

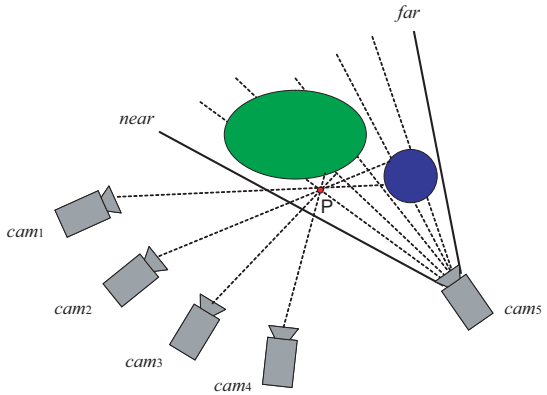


Fig. 4. Plane-sweep in PGS

In figure 4, cam_5 is basis camera 2 that defines every plane. In this approach, $near$ plane and far plane are easy to see since we can visualize them directly from basis camera 2 (cam_5). This is impossible in the case of the normal plane-sweep algorithm in the Euclidean space where full calibration is used. In that case actual depth of scene has to be measured so that $near$ plane and far plane cover all volume of interest.

In our approach, basis camera 2 is not used for color consistency testing during performing plane-sweep because every planes would be projected as a line in this image. So the basis camera 2 is required only for weak calibration. After using it we disable this camera to save CPU time.

D. Computing New View Images

In this section, we explain how we implement the plane-sweep algorithm after defining planes in PGS. If pixel p in a virtual camera is back projected to 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 5, the projection of 3D point P lying on π_k on the input image i can be performed by homography $\mathbf{H}_i$. Thus the projection p_i of a 3D point P on the camera i is computed by

$$x_i = \mathbf{H_i}\mathbf{H_x^{-1}}x \tag{1}$$

where x and x_i are positions of the pixel p and p_i respectively.

Homography $\mathbf{H_i}$, where i is a camera number, can be estimated from correspondences of at least four points. In our situation we select four points defined as the image corners of the basis camera 1 as shown in figure 5. Then, we project these points onto every real cameras 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 point one by one for the purpose of making computation time short.

Following algorithm summarizes our plane-sweep algorithm in PGS. First, we 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 input images excluding basis camera 2.
    · compute average color: $color_p = \frac{1}{n}\sum_{j=1\ldots n} c_j$ where c_j is the color from this projection on the j-th camera
    · compute color consistency score from variance: $score_p = \sum_{j=1\ldots n}(c_j - color_p)^2$
    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

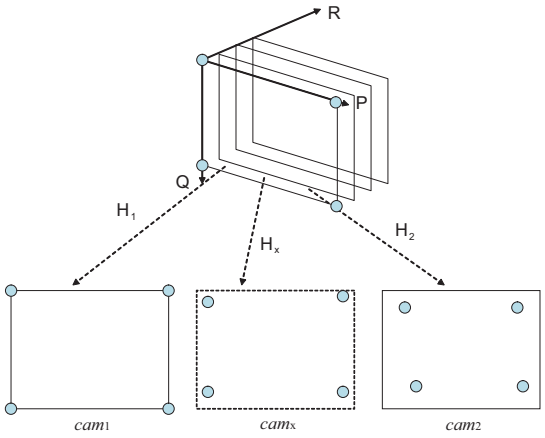


Fig. 5. Estimating Homography Matrices for Plane-sweep

In this algorithm, we use the score function proposed in [6].

E. Plane Sweep for Diminished Reality

In this paper, we present the method that makes it possible not only to create a new view image but to remove occluding objects in front of the objective scene. To achieve this, we exclude occluding objects from the planes that are defined by basis camera 2. Figure 6 shows this process.

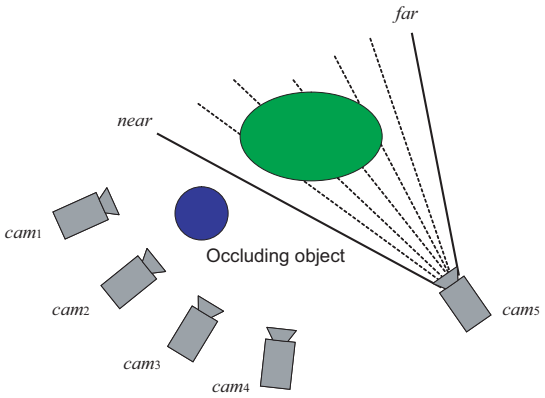


Fig. 6. Excluding Objects from PGS

If we simply exclude occluding objects from the planes, color information of that object must affect the result. In this section we describe how to deal with that information.

After computing the color consistency, the color of the pixel of the plane in each cameras are converted from RGB into HSV. Then we sort them by value of H using bubble sort as in figure 7.

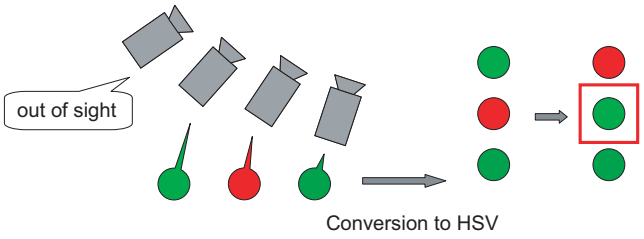


Fig. 7. Conversion and Sort

Our method is based on the assumption that the number of cameras that see each voxel on the objective scene (occluded object) is greater than the number of the occluded cameras. From this, outlier must not be the median of the order after sorting since H is defined by hue of that color. Therefore we adopt the median and update it as the color for that pixel.

We do not have to specify which camera gets inappropriate color since an outlier is removed automatically. This process removes occluding objects from the new image no matter which camera captures it.

Occluding object in a scene can be removed by defining the near and far planes so that occluding object does not lie in these planes. Different alignment of planes gives the different results. To illustrate the effect of defining planes, we show the example results that are generated from the different planes defining. Five cameras are used to capture input images. Camera 5 is selected as camera for defining planes. Figure 8 shows input images from camera 1 to 4.

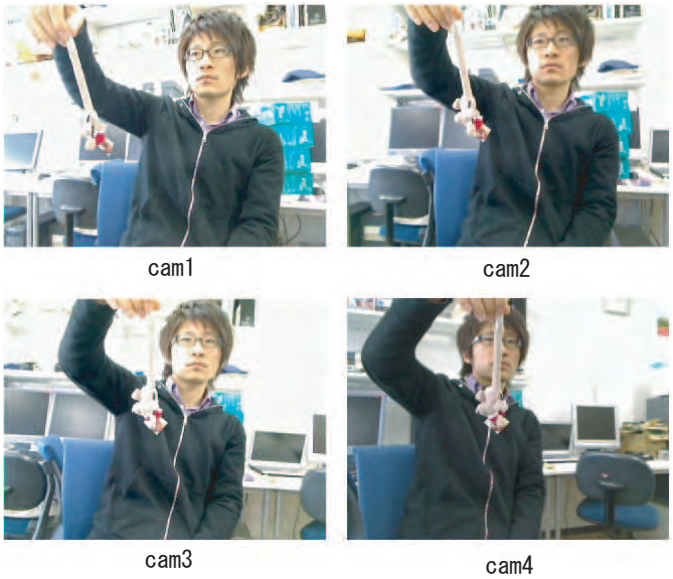
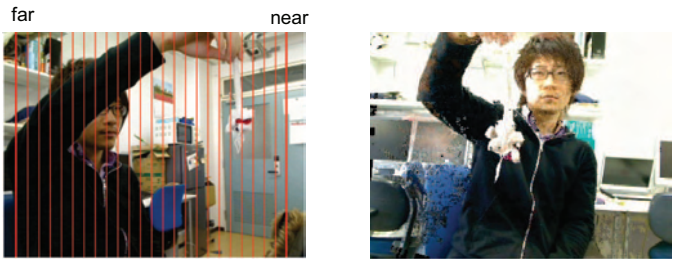


Fig. 8. Images from Camera 1 to 4

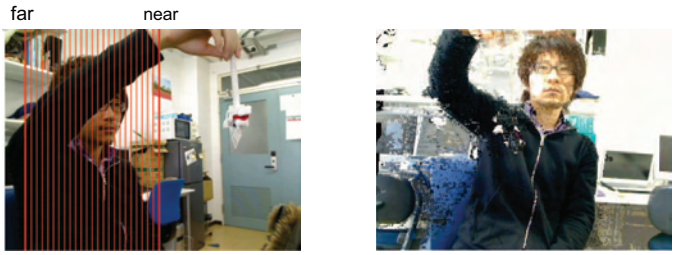
Figure 9 and 10 illustrate this effect. In Figure 9, planes are defined to include the whole scene while in Figure 10, planes are define to exclude occluding object. We can see that defining planes to exclude some objects can remove those objects from the rendered image using our proposed method.



cam5
(define planes)

rendered image
(the same view as cam 3)

Fig. 9. Defining Planes to Cover All Objects in The Scene.



cam5
(define planes)

rendered image
(the same view as cam 3)

Fig. 10. Defining Planes to Cover Only Occluded Object.

F. Combining with Real Image

The result image of plane-sweep is not so clear as the input image regardless of whether there is an object or not. If the purpose is to remove object from an input image, cam_x is not necessarily a new view image. Therefore, if cam_x is set in the same location as that of another camera (e.g. cam_2), we can use the input image from that camera to make the result clear. We employ following procedure to obtain more visually satisfactory ouput image.

We assume that cam_x is located in the completely same position as another camera (e.g. cam_2). We call this another camera user view camera.

1. In the first frame of user view camera, neither occluding objects nor objective scene should be in the image. This image is called background image.

2. For each pixel in each frame, we get a subtraction image using background image. Then we threshold it.

3. In the user view camera, pixels where subtraction is over threshold are replaced by the output image.

This procedure makes the region where there is no occluding object clear. As long as the user view camera is located in the same position as that of any input camera, it is better to go through this procedure.

Figure 11 show comparisons of output images without combining procedure and those with combining procedure. It is obvious that this combining procedure makes output image clearer especially the region where no other camera could capture colors so output pixel is black.

G. Implementing Real-time Plane-Sweep on GPU

We implement our plane-sweep algorithm in PGS on GPU to achieve real-time computation. Since GPU has a massive parallel processing, using GPU gives much more computation power compared to CPU. This section gives some details about our implementation.

We use OpenGL for the rendering part. Input images used for color consistency check are transfer to GPU as multi-textures. In drawing function we loop through each plane in PGS from $near$ to far plane. Homographies for projecting points on a virtual camera to 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 homography. Then we compute color consistency score.

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 make OpenGL select the best scores with the z-test. Finally it updates the color by choosing median at the best score. When rendering is finished for all planes, we get new view in the frame buffer.

III. Experimental Results

In this section, we show both qualitative and quantitative results of our proposed method. Figure 12 shows experimental setup.

We used the following hardware in the experiment.

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

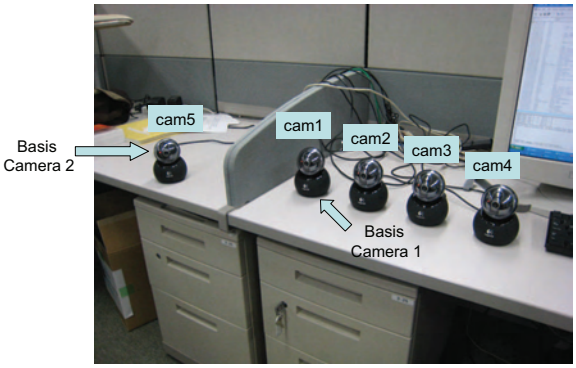


Fig. 12. Experimental Setup

2D-2D correspondences for estimating relationship among cameras can be selected from feature points in a scene. This time however we wave marker around a scene and track features for 2D-2D correspondence automatically. Thus we do not have a problem of calibrating even in the scene with only a few natural features. This process is shown in figure 13.

Fig. 13. Weak Calibration

A. Running time

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 more 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 I shows the running time for rendering output images using different number of cameras and 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.

	Number of planes				
	20	40	60	80	100
Using 4 Cameras	23.29	15.02	10.00	7.51	6.00
Using 5 Cameras	20.80	10.01	6.67	5.00	4.28

TABLE I
Frame rates (frame/sec.) of our method.

B. Qualitative Evaluation

In this section we show the result images comparing with the input images. We used an umbrealla as an occluding object. The purpose is to remove this object from the output image and reveal the occluded scene. We used five webcams and the resolution of each webcam is 320x240. Output view was selected to be the same view as camera 3. Figure 14 shows input images and output images from our method.

Since we set cam_x in the same place as the user view image, we used combining procedure to make output images clearer. Our method allows both occluding object and occluded object to move in any direction. From the results, even not all part of

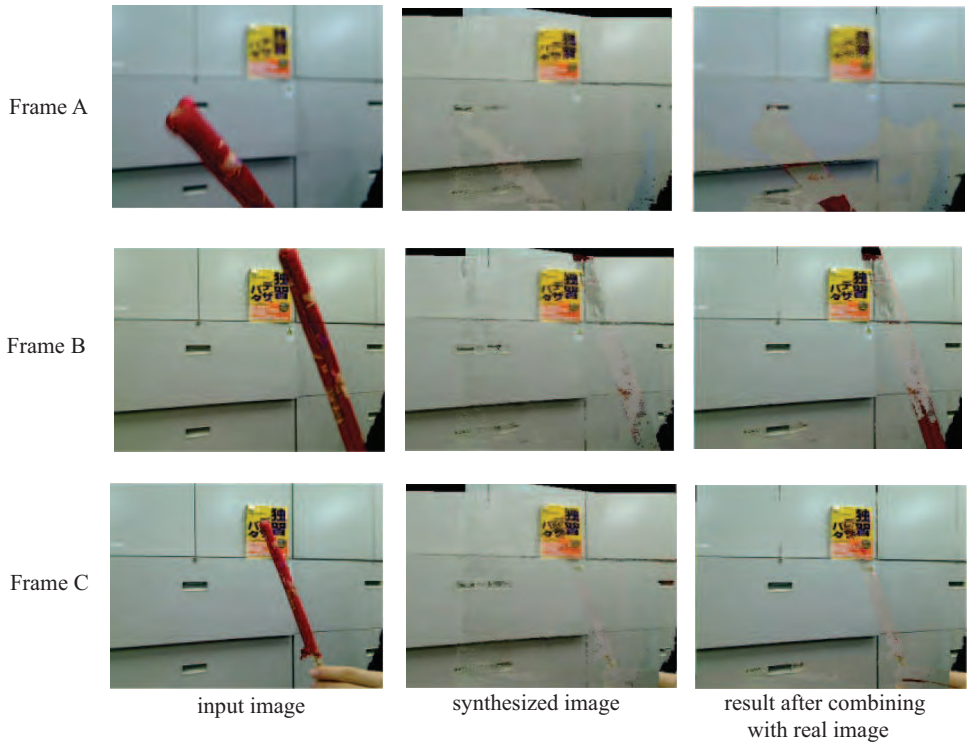


Fig. 11. Combining Synthesized Image with The Real Input Image.

occluding object was removed, it can be said that 3D object behind the occluding object is correctly reconstructed. In the experiment, we used 90 planes for doing plane-sweep. The average processing speed was 4.8 fps.

C. 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. We used our method to remove this occluding object from the input images. By using static background, we can have ground-truth references to measure the accuracy of results.

Figure 15 shows the different of the results when using different number of planes to reconstruct and render output images in which occluding object is removed.

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 200 consecutive frames. Figures 16 shows PSNR of our result images using the different number of planes for scene reconstruction. Table II shows the average PSNR over 200 frames.

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.

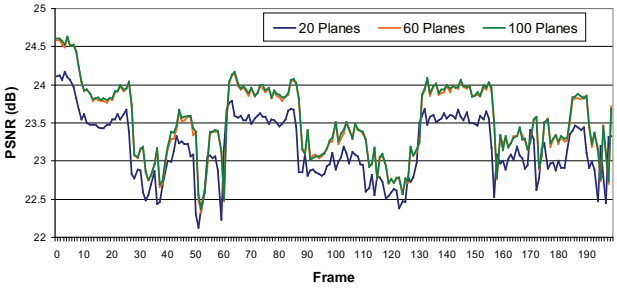


Fig. 16. PSNR Error of Synthesized Images.

Number of planes	Average PSNR(dB)
20 planes	23.19
60 planes	23.51
100 planes	23.53

TABLE II
FRAME RATES (FRAME/SEC.) OF OUR METHOD.

IV. Conclusion

In this paper we present a new online method for diminished reality using plane sweep algorithm. Our method needs neither marker nor calibration. $Near$ and far planes in PGS for doing plane sweep are easily defined since they are visualized from basis camera 2. This is impossible for the case of the normal plane-sweep algorithm in the Euclidean space in which full

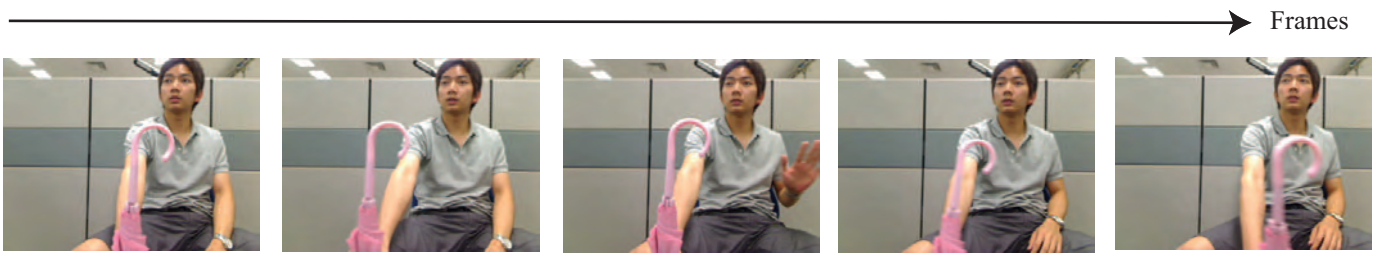


Fig. 14. Result of Removing Occluding Object from Input Camera 3

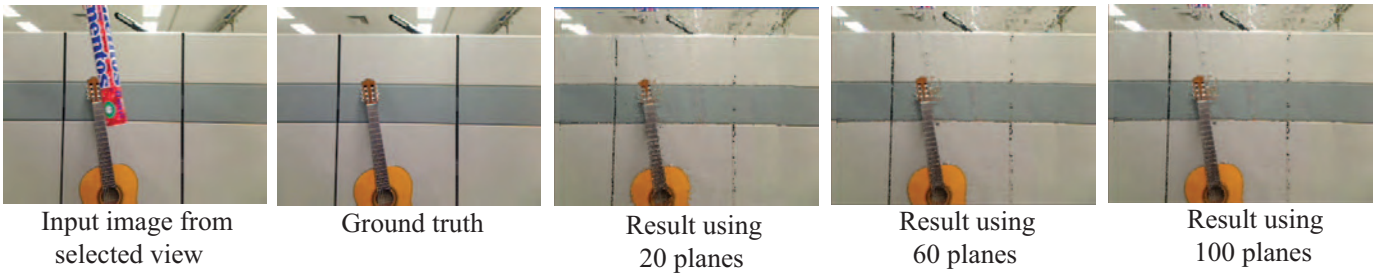


Fig. 15. Comparison of Using the Different Number of Planes to Render Output Images

calibration is used. Also our method can deal with 3D objects. Both occluding objects and objective scene can move in any direction. As long as we exclude occludings object from the planes, our plane-sweep algorithm automatically ignores the outlier by accepting median color. Combining process creates clear new images.

ACKNOWLEDGEMENT

This work is supported in part by a Grant-in-Aid for the Global Center of Excellence for High-Level Global Cooperation for Leading-Edge Platform on Access Spaces from the Ministry of Education, Culture, Sport, Science, and Technology in Japan.

REFERENCES

[1] S.Mann and J.Fung, "Videoorbits on eye tap devices for deliberately diminished reality or altering the isual perception of rigid planner patches of a real world scene," in *International Symposium on Mixed Reality(ISMR 2001)*.

[2] Adelson and Wang, "Representing moving images with layers," in *IEEE Transactions on Image Processing Special Issue: Image Sequence Compression*, September 1994.

[3] V.Lepetit and M.-O. Berger, "A semi-automatic method for resolving occlusion in augumented reality," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition(CVPR f00)*, June 2000.

[4] 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)*.

[5] H. Saito and T. Kanade, "Shape reconstruction in projective grid space from large number of images," in *Proceedings of the IEEE Com-*

puter Society Conference on Computer Vision and Pattern Recognition (CVPR'99), vol. 2, June 1999, pp. 49–54.

[6] V. Nozick and H. Saito, "On-line free-viewpoint video : From single to multiple view rendering," in *International Journal of Automation and Computing (IJAC), vol. 5, no. 3, pp. 257?267*, 2008.