Real-time Rendering System of Moving Objects

Yutaka Kunita Masahiko Inami Taro Maeda Susumu Tachi

Department of Mathematical Engineering and Information Physics
Graduate Shool of Engineering, The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
{kuni, minami, maeda, tachi}@star.t.u-tokyo.ac.jp

Abstract

We have been developing a system named "mutual tele-existence" which allows for face-to-face communication between remote users [14]. Although image-based rendering (IBR) is suitable for rendering human figures with a complex geometry, conventional IBR techniques cannot readily be applied to our system. Because most IBR techniques include time-consuming processes, they cannot capture the source images and simultaneously render the destination images. In this paper, we propose a novel method focusing on real-time rendering. Moreover, we introduce equivalent depth of field to measure the fidelity of the synthesized image. If the object is in this range, accurate rendering is guaranteed. Then, we introduce a prototype machine that has 12 synchronized cameras on the linear actuator. Finally, we present some experimental results of our prototype machine.

1 Introduction

We have been developing a system which allows for face-to-face communication between remote users[14]. In this system, users who are far apart in the real world share computer-generated three-dimensional space. Figure 1 shows the concept behind our system. In this system, users communicate with one another by each entering a booth which has a cylindrical structure surrounding the user. This structure works both as the display device and the image-capturing device. It displays stereoscopic images of the computer-generated three-dimensional space around the user just like CAVE[3], and simultaneously captures a set of images from all sides of the user.

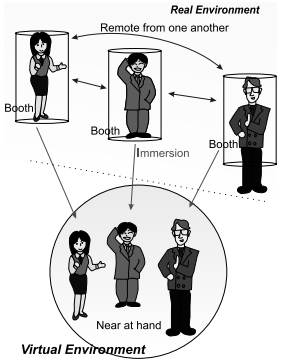

Figure 1: The concept of our tele-communication system.

In most conventional tele-conferencing systems, users exchange images from a fixed viewpoint only. However, we believe that our system should display the images from the users' eye position in the computer-generated three-dimensional space, in order to provide the users with the feeling that they are all in the same place, i.e. "presence."

However, the set of images captured by the booth is the one from the fixed orbit around the booth, and the images to be displayed are the ones from the

users' viewpoints in the computer-generated space. Therefore, generating images from arbitrary viewpoints using a set of images from the fixed orbit would be a technological key to realizing our system.

A considerable amount of research is available about this kind of rendering technique [2], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13]; however, all these studies present the same problem in their application to tele-conferencing systems, namely, that take too much time.

For example, some techniques take the approach of making use of geometric information [6],[10],[11],[12]. One typical piece of geometric information is a "depth map," the depth values which corresponds to each pixel of the image. Once the depth map is acquired, it is easy to synthesize the images from arbitrary viewpoints. However, the process of acquiring this depth map is very time-consuming and lacks robustness.

In another approach, geometric information is not used. Instead, a subset of the "plenoptic function"[1] as a representation of the data stage is used[5],[8]. The plenoptic function is a 5-dimensional function which describes the ray's intensity in all positions and at all angles. If the plenoptic function could be acquired, it would also be possible to acquire images from arbitrary viewpoints. However, this is actually impossible. So, the dimension of the prenoptic function is usually reduced from 5 to 4 by assuming a free space and transparency of the route of the ray. But, since the amount of data of this 4-dimensional function is still too large, it is acquired in an off-line process. The technique using the representation of EPI (epipolar-plane images) [7] can also be categorized here.

To overcome the difficulties outlined above, we here propose a novel method focusing on real-time rendering of moving objects in the real world. This method has the following features:

1. Only one parameter of the object's distance is needed. Thus, geometric information such as a "depth map" is not required.

2. Fast rendering is possible by making use of graphic hardware acceleration of texture mapping.

3. The data processed by the rendering computer are reduced in advance at the stage of capturing the source images.

This paper is organized as follows.

In Section2, we describe the basic theory of our rendering method. Although our method does not need explicit geometric information (Feature 1), the distance of one typical point of the object is required. This is analogous to the way in which we take photos with a single camera: we adjust the focal distance to one point of the object. In this analogy, we define "equivalent focal distance" and "equivalent depth of field," which represent the supposed distance from cameras to objects and the range of distance within which objects are guaranteed correct image synthesis, respectively. Fast-rendering algorithms are also included using texture-mapping (Feature 2) in this section.

In Section 3, we introduce our prototype machine, which has 12 synchronized cameras. These cameras, which are located in a row on the linear actuator, will in future acquire images during a reciprocating motion. However, the cameras are still at present. In this prototype system, the rendering machine does not capture the image signals from all 12 cameras but only the one signal that has to be rendered at the time. Because of this, this system uses only one video-capture device, and the amount of image data required to deal with it is extremely reduced (Feature 3).

Next, in Section 4, we show some experimental results using this prototype system. We succeed in real-time rendering of moving objects in the real world.

Finally, in Section 5, we present our conclusions and project out future work.

2 Image Synthesis Method

2.1 Plenoptic Function

The plenoptic function[1] is a 5-dimensional function which describes the ray's intensity in all positions and at all angles. If the plenoptic function could be achieved, images from arbitrary viewpoints could also be obtained. Since this is actually impossible in

the real world, there is an approach which reduces the dimension of the plenoptic function from 5 to 4 by assuming a free space and the transparency of the ray's route [5],[8]. Although this 4-dimensional function is easier to achieve than the plenoptic function, the amount of data is still considerably large. Moreover, the rendering method using such a function requires the ray to have relatively high-sampling density. Because of these characteristics, the ray-sampling process of this kind of technique is very time-consuming. Thus, the objects are assumed to be static. In the following subsection, we describe the method we employed for our real-time rendering system. Although this method is based on the concept of the plenoptic function, the required sampling densities are relatively low and the rendering process is extremely simple.

2.2 Basic Theory

Hereafter, we explain our theory in the two-dimensional space for the sake of simplicity. However, it can be easily extended to the three-dimensional space without loss of generality.

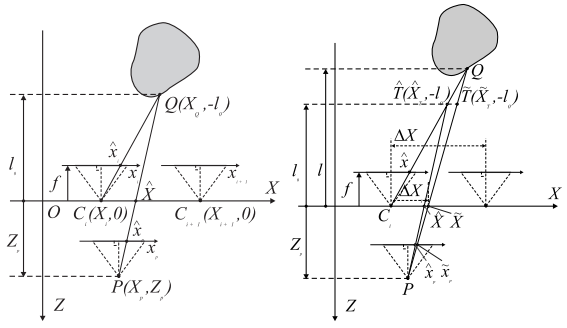

Figure 2: **Left**: Image synthesis at the viewposition P using a set of cameras C_i which are aligned on the X-axis. **Right**: Error between the desired position $\tilde{x_P}$ and the synthesized position $\hat{x_P}$ of Q on the image plane, when Q is not at the equivalent focal distance l_0.

In The left of Figure 2, plural cameras (focal length: f) are aligned so that their centers of projection $C_i(X_i, 0)$ are located on the X-axis. These cameras capture the images of the object in region $Z < 0$. Each camera has an image plane at length f from the center of projection C_i, which has a coordinate x_i. Let $P(X_P, Z_P)$ denote the virtual camera's viewpoint ($Z_P > 0$) and suppose this virtual camera also has an image plane: x_P. Our purpose is to describe the method used for synthesizing the image from P: $g_P(x_P)$ using a set of images of camera C_i: $g_i(x_i)$.

The ray which makes the image where $x_P = \hat{x_P}$ intersects the X-axis at point $X = \hat{X}$. If there were a camera at point $\hat{X}$, the image of the ray sampled by the camera would be the same as that of P, assuming there is no occluder (free space) and the route has transparency. However, a case in which there is no camera at point $\hat{X}$ could happen at low sampling densities, i.e. when the intervals of the cameras are long.

However, the ray which makes the image at $x_P = \hat{x_P}$ comes from $Q(X_Q, -l_0)$, and the ray which makes the image at $x_i = \hat{x_i}$ comes from the same point as well. Assuming that the surface of the object is diffusive, it can be said that

$$g_P(x_P) = g_i(x_i) \tag{1}$$

where the following equation is established between $\hat{x_p}$, $\hat{x_i}$.

$$X_i + l_0 \cdot \frac{\hat{x_i}}{f} = X_P + (l_0 + Z_P) \cdot \frac{\hat{x_P}}{f} = X_Q \tag{2}$$

In the equation above, l_0 is an approximate distance to the object, which we call "equivalent focal distance." Note that this value is not an array of distances like a depth map but, rather, a length to a representative point of the object's surface, exactly like a camera's focal distance.

Of course, not every point on the surface of the object exists at all times on length l_0, except for objects which have a planer surface $Z = -l_0$. In such cases, the image of Q is synthesized at the wrong position by the method mentioned above.

For example, in the right of Figure 2, the ray from Q makes the image at $x_i = \hat{x_i}$ of camera C_i. However, using the equivalent focal distance l_0, which differs

from l (the correct distance to Q), the image of Q is synthesized at position $\hat{x_p}$, whereas the image of Q should be synthesized at position $\tilde{x_P}$. Now, we define the error of the positions between the synthesized image and the correct image as:

$$\Delta x_P \equiv \hat{x_P} - \tilde{x_P} \tag{3}$$

Then, the following equations are established providing $(\tilde{X_t}, -l_0)$ denotes the point of intersection of $Z = -l_0$ and PQ, and $(\hat{X_t}, -l_0)$ that of $Z = -l_0$ and C_i, respectively.

$$\Delta x_P = \frac{f}{l_0 + z_P}(\hat{X_t} - \tilde{X_t}) \tag{4}$$

$$\hat{X_t} - \tilde{X_t} = \frac{l - l_0}{l}(X_i - \tilde{X}) \tag{5}$$

By substituting Equation(5) into Equation(4), we obtain

$$\Delta x_P = \frac{l - l_0}{l} \cdot \frac{f}{l_0 + z_P} \cdot (X_i - \tilde{X}) \tag{6}$$

Besides,

$$\begin{aligned} X_i - \tilde{X} &= (\hat{X} - \tilde{X}) + (X_i - \hat{X}) \\ &= \frac{z_P}{f} \cdot \Delta x_P + (X_i - \hat{X}) \end{aligned} \tag{7}$$

Finally, by substituting equation (7) into equation (6), we obtain the following equation:

$$\Delta x_P = \frac{f(l - l_0)}{l_0(l + Z_P)} \cdot (\hat{X} - X_i) \tag{8}$$

2.3 Equivalent Depth of Field

Now, let δ denote the maximum error allowed. After this, we determine the range of l, where $|\Delta x_P| \leq \delta$ for all $\hat{X}$.

As we can see in Equation(8), the absolute error of the image position $|\Delta x_P|$ is in proportion to $|\hat{X} - X_i|$. Therefore, we choose camera C_i among several cameras so that $|\hat{X} - X_i|$ gets the minimum value. When the cameras are aligned at the intervals ΔX, we must choose camera C_i so that $|\hat{X} - X_i| \leq \frac{\Delta X}{2}$. Hence, from Equation(8) it can be said that

$$|\Delta x_P| \leq \frac{f|l - l_0|}{l_0(l + Z_P)} \cdot \frac{\Delta X}{2} \tag{9}$$

By solving (*the right side of Equation(9) = δ*) for l, we obtain two solutions $l = l_1, l_2 (l_1 \leq l_0 \leq l_2)$, as follows:

$$l_1 = l_0 \cdot \frac{1}{1 + \frac{l_0 + Z_p}{\frac{f}{\delta}\frac{\Delta X}{2} + Z_P}} \tag{10}$$

$$l_2 = l_0 \cdot \frac{1}{1 - \frac{l_0 + Z_p}{\frac{f}{\delta}\frac{\Delta X}{2} + Z_P}} \tag{11}$$

Then, $|\Delta x_P| \leq \delta$ is established for all l which satisfies $l_1 \leq l \leq l_2$.

If we let δ be the length of one pixel on the image plane, the points on the objects at the distance within $l_1 \leq l \leq l_2$ are guaranteed to be synthesized at actually correct positions whose errors are within one pixel. We call this range of distance $\Delta l \equiv l_2 - l_1$ the "*equivalent depth of field.*" Δl is a function of ΔX and monotone decreasing.

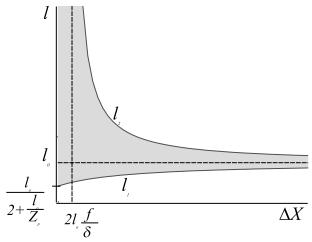

Figure 3: The graph of the equivalent depth of field. The equivalent depth of field is expressed by the length between l_1 and l_2.

Figure 3 shows a graph plotting l_1 and l_2 versus ΔX. In this figure, the equivalent depth of field is expressed by the length between l_1 and l_2. Note that $l_1 = \infty$ when $\Delta X = 2l_0 \cdot \frac{\delta}{f}$. If we place the cameras at smaller intervals than $2l_0 \cdot \frac{\delta}{f}$, all the points on the objects at length $l \geq l_1$ can be correctly synthesized.

2.4 Rendering Algorithm

In this subsection, we will describe two different algorithms which implement the method mentioned above.

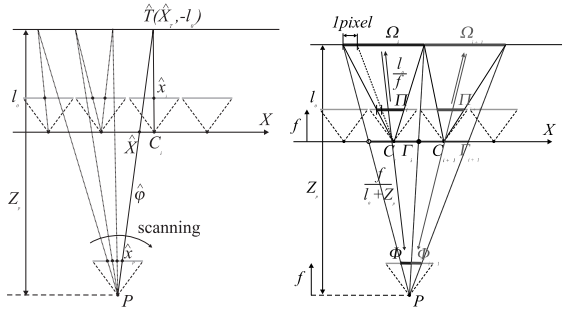

Figure 4: **Left**:Rendering by ray-tracing algorithm. **Right**:Rendering by multi-texture-mapping algorithm.

2.4.1 Ray-Tracing Algorithm

In this algorithm, the image is acquired from the virtual viewpoint P by scanning the rays which intersect P. See the left of Figure4.

The whole process is as follows:

For each $x_P = \hat{x_P}$,

1. Let $\hat{\varphi}$ denote the ray which makes the image at $x_P = \hat{x_P}$; $\hat{\varphi}$ intersects $Z = 0, -l_0$ at $X = \hat{X}, \hat{X_T}$ respectively, where $\hat{X} = X_P + Z_P \frac{\hat{x_P}}{f}$ and $\hat{X_T} = X_P + (Z_P + l_0)\frac{\hat{x_P}}{f}$.

2. Choose camera C_i, whose $X-$coordinate X_i is nearest to $\hat{X}$.

3. Obtain $\hat{x_i}$, which is the intersection of the line from $(\hat{X_T}, -l_0)$ to C_i and the image plane of the camera C_i. This process corresponds with solving Equation(2) for $\hat{x_i}$.

4. Copy the image of camera C_i at $x_i = \hat{x_i}$ to that of the virtual viewpoint P at $x_P = \hat{x_P}$, that is to say: $g_P(\hat{x_P}) = g_i(\hat{x_i})$.

The algorithm mentioned above includes many repetitions, and the calculation processes which find $\hat{x_Q}$ corresponding to $\hat{x_P}$ are rather complicated. Therefore, this algorithm is not suitable for real-time rendering

2.4.2 Multi-texture Mapping Method

Unlike the algorithm explained above, the one below is very straightforward and designed for real-time rendering.

See the right of Figure 4. First, we define Γ_i which is the region on $Z = 0$ as:

$$\Gamma_i \equiv \left(\frac{X_{i-1} + X_i}{2}, \frac{X_i + X_{i+1}}{2} \right]. \tag{12}$$

Suppose the rays go through P, whose X-intercepts are within Γ_i i.e., $\hat{X} \in \Gamma_i$. The closest camera to these intercepts is surely C_i. If the images from P corresponding to these rays were rendered with the previous ray-tracing method, the source images within Π_i would be copied to the destination images within Φ_i. Besides, the rays would intersect $Z = -l_0$ within the region Ω_i. From a simple similarity, we can obtain:

$$\text{len}\,\Omega_i = \frac{l_0}{f} \cdot \text{len}\,\Pi_i \tag{13}$$

$$\text{len}\,\Phi_i = \frac{f}{l_0 + Z_P} \cdot \text{len}\,\Omega_i \tag{14}$$

where "len" expresses the length of the region. From Equation(13)(14),

$$\text{len}\,\Phi_i = \frac{l_0}{l_0 + Z_P} \cdot \text{len}\,\Pi_i \tag{15}$$

Because the correspondence between $\hat{x_P} \in \Phi_i$ and $\hat{x_i} \in \Pi_i$ is linear, we can copy the source image $g_i(x_i \in \Pi)$ to the destination image $g_P(x_P \in \Phi_i)$ with the scaling ratio $l_0/(l_0 + Z_P)$ all at once. By performing such a procedure for each i, we can obtain the desired image from a virtual viewpoint P without scanning each ray through P.

Furthermore, this scaling procedure can be done automactically with a general 3-D graphic library as follows:

1. Construct a scene consisting of a viewpoint P and a plane T.

2. For each i, paste the corresponding image of camera C_i: $g_i(x_i \in \Pi_i)$ to region Ω_i on plane T as a texture-mapping image. By taking the

ratio [1 pixel of Π_i]: [1 pixel of Ω_i] as [len Π_i]: [len Ω_i] ($= f : l_0$), this process is equal to simple memory copy without scaling from the image captured by the camera to the texture-mapping image.

3. Render the perspective image from P.

Note that none of the processes above includes explicit scaling procedures, but we can obtain the same effect. Each image of camera C_i: $g_i(x_i \in \Pi_i)$ is copied to Ω_i on plane T at scaling ratio l_0/f. Then, these images of region Ω_i are rendered in the image plane of P at scaling ratio $f/(l_0+Z_P)$. Thus, each $g_i(x_i \in \Pi_i)$ is rendered on the image plane of P at scaling ratio $l_0/(l_0 + Z_P)$ as a result.

This rendering algorithm combines simplicity and speed by making use of hardware acceleration in texture mapping such as scaling and smoothing. By employing this method, our prototype system, which we describe in the next section, realized real-time rendering.

3 Prototype System

3.1 Overall

The left picture of Figure 5 shows our prototype system, and that on the right is its block diagram. This system consists of a camera unit, a rendering PC, and a control PC.

In the camera unit, 12 small color CCD cameras (Toshiba IK-C40, the length of one pixel $\delta \sim 0.01mm$) are aligned horizontally in a row on the linear actuator at intervals of 54mm. Very wide-angle lens (focal distance $f = 3.5mm$) are attached to the cameras. In future, the actuator will reciprocate while the cameras acquire the image. At present, the actuator remains still. These cameras are located on the gantry with rotating 90° around their optical axes. Thus, the direction of their scanning lines is vertical. Moreover, all these cameras are synchronized by the same genlock signal.

The rendering PC continuously captures a video signal from the camera unit by one video-capturing board. Note that if we intended to capture all 12 video signals, 12 video-capturing boards would be required, which would not be practical. Consequently, we installed a video switch between the camera unit and the rendering PC, allowing only one scanning line among 12 scanning lines to be selectively captured by one video-capturing board. The timing of switching channels is synchronized with the cameras' genlock signal.

The control PC indicates the channel of this switch and controls the motion of the linear actuator. This control PC and the rendering PC exchange data fast by using a shared memory.

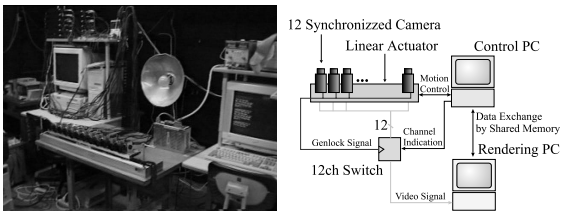

Figure 5: **Left**:the overview of our prototype system. **Right**:the block diagram of our prototype system.

3.2 Implementation of Rendering

In this subsection, we describe the implementation of real-time rendering of our prototype system. We employ the multi-texture-mapping algorithm mentioned in the previous section.

First, with a virtual viewpoint given by a user, the rendering PC calcurates the regions of each camera's image that are used for multi-texture-mapping. Because the cameras are aligned horizontally, each region is made by dividing the whole image of the camera horizonally. Thus, its shape is a vertically long column.

Next, the information of the region is send to the control PC and used as an indication of the video switch. The timing of switching is synchronous with the camerass' scanning, and the direction of the cameras' scanning line is vertical. In this way, the rendering PC can selectively capture the necessary column image from the cameras.

Then, the captured images are texture-mapped on a plane at the distance of the equivalent focal distance in the virtual three-dimensional space. This process is equal to the simple memory copy mentioned previously.

Finally, the rendering PC renders the scene by using a 3D graphic library.

One cycle of all these processes is finished within video rate (1/30[sec]), so we can say our system realizes real-time rendering.

Some conventional IBR systems capture all the images from the cameras and construct bulky data as a subset of the plenoptic funcion, whereas our system reduces the image data to deal with before capturing them at the analog signal level. On the other hand, the scaling process is rapidly carried out by digital graphic hardware. This combination of analog and digital technologies is the key to real-time rendering.

4 Experimental Result

4.1 Relation between Sampling Density and Equivalent Depth of Field

In this experiment, we synthesize the images of a static object with with different camera intervals ΔX.

The upper left of Figure 6 shows the positions of the cameras, the virtual viewpoint, and the two objects. The cameras are aligned on $Z = 0$. The virtual viewpoint is at $Z = 50[mm]$. The two objects are on $Z = -350$ and $-70[mm]$, respectively.

The upper right shows a graph plotting the equivalent depth of field on condition that the cameras' parameters $f = 3.5[mm], \delta = 0.01[mm]$, and the equivalent focal distance $l_0 = 350[mm]$, i.e., focused on the front object.

When $\Delta X = 4[mm]$, the equivalent depth of field is $522[mm]$ and the lower left picture is synthesized. As we can see, both the image of the back object and that of the front object are synthesized accurately.

On the other hand, when $\Delta X = 30[mm]$, the equivalent depth of field is $53mm$, which is not long enough to synthesize the image of the back object.

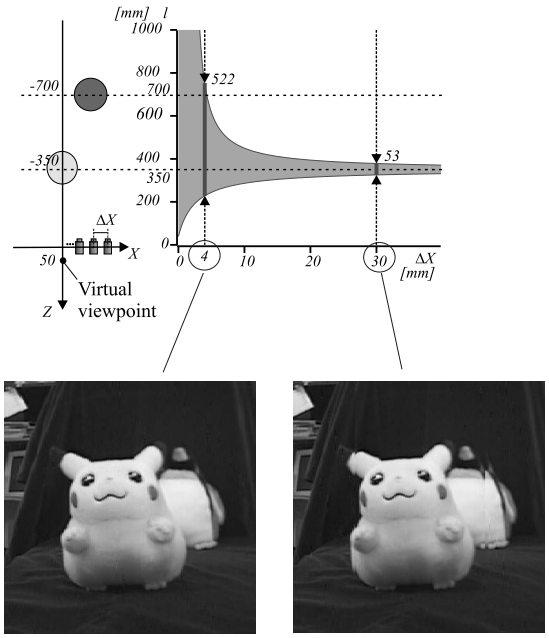

Figure 6: Image synthesis of static objects with different camera intervals ΔX.

The lower right picture shows the discontinuity on the back object.

4.2 Correct Expression of Occlusion

In this experiment, we compare the multi-texture mapping algorithm introduced previously with conventional texture mapping using an image from a single camera.

The upper part of Figure 7 shows a scene, where the cameras are aligned on $Z = 0$ with intervals of $54mm$, the moving object is near $(0, -1000)$, and the virtual viewpoint P is at $(200, 500)$. The object shades his face with his hand against the virtual viewpoint P. Therefore, the object's face should not be seen from P.

The lower left of Figure 7 shows the image synthesized by texture-mapping the image in camera C_0 on plane $T(Z = -1000)$. We can see the object's face, and the correct occlusion is not expressed. On the Contraty, the lower right image which is synthesized

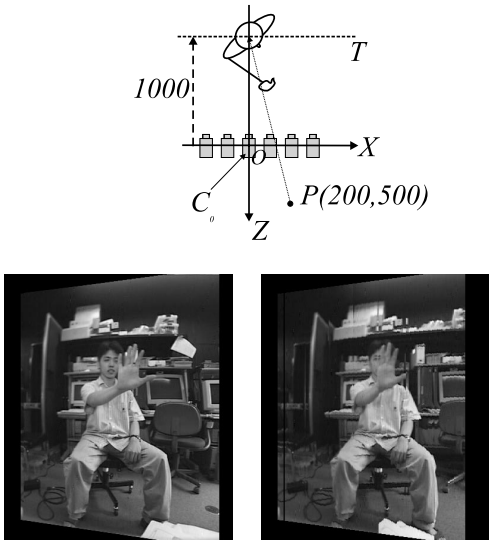

Figure 7: Comparison between the images synthesized by using a single texture (left) and multiple textures (right). While the one on the left can not express correct occlusion, the one on the right can.

by the multi-texture mapping method, succeeds in expressing the correct occlusion.

5 Conclusions

We introduced a novel image synthesis method for our tele-communication system focusing on real-time rendering. Next, we mentioned "*equivalent depth of field*" to measure the fidelity of the synthesized images. Then, we showed a prototype machine which realized real-time rendering and its results. A topic of future work will be to synthesize images during reciprocating cameras on a linear actuator. By moving cameras, our prototype system can synthesize images at higher spatial sampling densities. Then, extending an image synthesis method proposed in this paper, we will investigte a method where cameras are arranged around the object.

References

[1] E. H. Adelson and J. R. Bergen. *The Plenoptic Function and the Elements of Early Vision*, chapter 1. The MIT Press, 1991.

[2] S. E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH'93 Conference Proceedings*, pages 279–288, 1993.

[3] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and imprementation of the cave. In *SIGGRAPH'93 Conference Proceedings*, pages 135–142, 1993.

[4] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geomtry- and image-based approach. In *SIGGRAPH'96 Conference Proceedings*, pages 11–20, 1996.

[5] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH'96 Conference Proceedings*, pages 43–54, 1996.

[6] T. Kanade, P. J. Narayanan, and P. Rander. Virtualized reality: Being mobile in a visual scene. In *Proceedings of the 5th International Conference on Artificial Reality and Tele-Existence (ICAT 95)*, pages 133–142, 1995.

[7] A. Katayama, K. Tanaka, T. Oshino, and H. Tamura. A viewpoint dependent stereoscopic display using interpolation of multi-viewpoint images. In *Stereoscopic Displays and Virtual Reality Systems II*, volume 2409, pages 11–20. SPIE, 1995.

[8] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH'96 Conference Proceedings*, pages 31–41, 1996.

[9] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH'95 Conference Proceedings*, pages 39–46, 1995.

[10] S. Moezzi, A. Katkere, D. Y. Kuramura, and R. Jain. Immersive video. In *Proceedings of VRAIS '96*, pages 17–24. IEEE, 1996.

[11] P. Rademacher and G. Bishop. Multiple-center-of-projection images. In *SIGGRAPH'98 Conference Proceedings*, pages 199–206, 1998.

[12] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH'98 Conference Proceedings*, pages 179–188, 1998.

[13] J. Shade, S. Gortler, L. wei He, and R. Szeliski. Layered depth images. In *SIGGRAPH'98 Conference Proceedings*, pages 231–242, 1998.

[14] S. Tachi, T. Maeda, Y. Yanagida, M. Koyanagi, and H. Yokoyama. A method of mutual tele-existence in a virtual environment. In *Proceedings of the 6th International Conference on Artificial Reality and Tele-Existence (ICAT 96)*, pages 9–18, 1996.