Image-Based Rendering by Virtual 1D Cameras

Naoyuki Ichimura

National Institute of Advanced Industrial Science and Technology (AIST)
1-1-1, Umezono, Tsukuba, Ibaraki 305-8568, Japan
nic@ni.aist.go.jp,
http://staff.aist.go.jp/naoyuki.ichimura/

Abstract. Image-based rendering (IBR) has been used to synthesize images corresponding to a new view point from stored images. Rendering methods based on a three-dimensional plenoptic function are attractive due to the simplicity of image capture. Only a few specific discussions, however, have been done for the scaling problem to correct aspect ratio distortion, which heavily affects the quality of a synthesized image. This paper presents a rendering algorithm with a scaling scheme, which is general in that it can handle arbitrary camera paths. We model a virtual camera by a set of one-dimensional (1D) cameras. The ray representation of the 1D camera enables us to devise a rendering algorithm for the cases where the camera paths to create ray databases are arbitrary curves. We conclude with experimental results that demonstrate the usefulness of the proposed algorithm.

1 Introduction

Image-based rendering (IBR) has been used to synthesize images corresponding to a new view point from stored images [1]. Figure 1 shows an example of IBR. A scene is captured by multiple cameras set on a linear camera path as shown in Fig. 1 (a). A set of images captured by the cameras is called a ray database, because storing the images is equivalent to storing the rays associated with the cameras. A new view is generated by properly extracting the rays in the database, which correspond to the rays of a virtual camera. Figure 1 (b) shows a new view obtained by placing a virtual camera at the back of the camera path.

Making a ray database is interpreted as sampling the plenoptic function [2] shown in Fig. 2 (a). Due to the high dimensionality of the plenoptic function $P(\mathrm{x}, \mathrm{y}, \mathrm{z}, \theta, \phi, \lambda, t)$, which has 7 dimensions of positions, directions, wavelength and time, sampling the function by arranging cameras in space is extremely difficult. Practical IBR algorithms have been developed using a 4- or 3-dimensional (4D or 3D) plenoptic function by providing constraints for the arrangement of cameras, wavelength and time. Typical algorithms using a 4D function $P(\mathrm{x}, \mathrm{y}, \theta, \phi)$ are light field rendering [3] and lumigraph [4], in which cameras are arranged on the vertical plane defined by the $\boldsymbol{X}_{\mathrm{w}}$ and $\boldsymbol{Y}_{\mathrm{w}}$ axes in Fig. 2 (a). Cameras are arranged on the horizontal plane defined by the $\boldsymbol{X}_{\mathrm{w}}$ and $\boldsymbol{Z}_{\mathrm{w}}$ axes in concentric mosaics [5], bi-centric camera [6,7] and cross-slits projection [8,9,10]; this enables us to use a 3D function $P(x, z, \phi)$.

T. Wada, F. Huang, and S. Lin (Eds.): PSIVT 2009, LNCS 5414, pp. 423–435, 2009.
© Springer-Verlag Berlin Heidelberg 2009

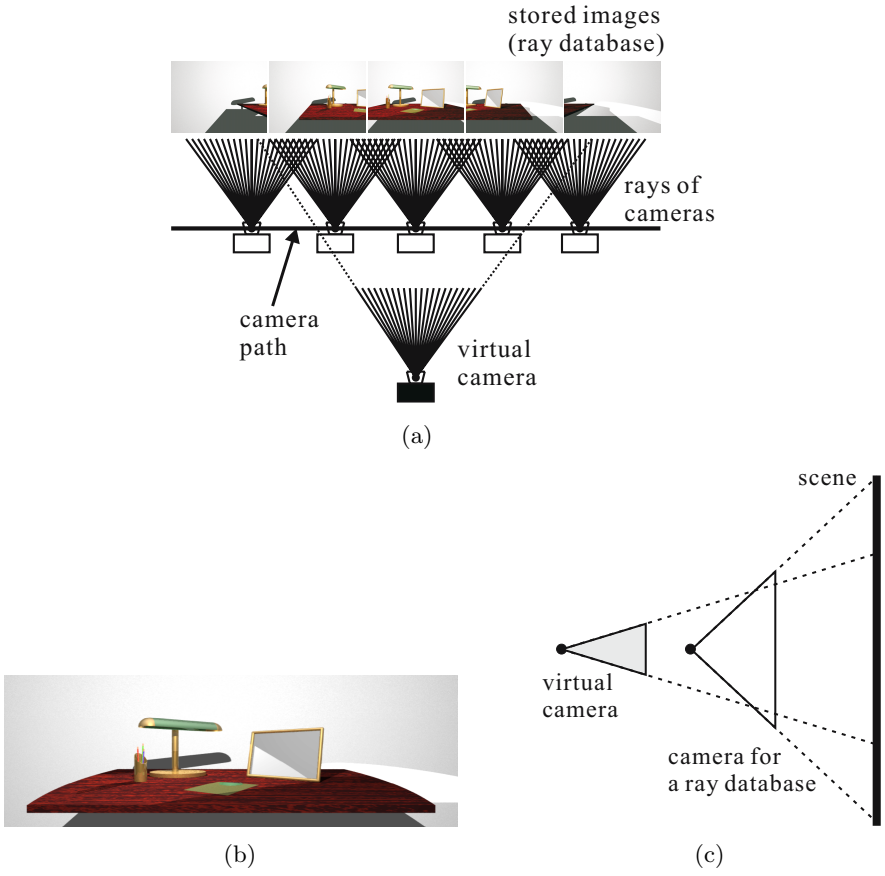


Fig. 1. An example of image-based rendering (IBR). (a) A ray database and a virtual camera. (b) A new view from the virtual camera. Note that the image shows an entire scene. (c) The difference between the vertical fields of views of a camera used for a ray database and a virtual camera. The difference is the source of aspect ratio distortion which would appear in a new view.

Since no cameras are required for the vertical direction, there are two advantages of IBR with a 3D plenoptic function. The first advantage is ease of image capture. The second one is that new view generation can be simply performed by mosaicing, where columns (or vertical strips) of pixels of images in a ray database are concatenated [5,6,7,8,9,10]. On the other hand, there is a serious drawback as well; the aspect ratio of a new view is changed depending on the difference between the vertical fields of views of cameras used for a ray database and the vertical field of view of a virtual camera. Figure 1 (c) shows an example of the difference. Since we can determine the position and focal length of a virtual camera arbitrary, the extent of the scene captured by a virtual camera could be smaller or larger than that captured by cameras for a ray database. If a new view is generated by concatenating columns of pixels of images in a ray

database without taking the difference of the fields of views into account, the aspect ratio of the new view is changed because we only have images captured from the positions of cameras for a ray database. Thus the distortion due to the change in the aspect ratio would appear in the new view. This distortion is called aspect ratio distortion or vertical distortion [5,8], and badly affects the quality of a synthesized image, especially when forward/backward motions of a virtual camera are simulated.

In order to remove this distortion, we need to find an appropriate factor for scaling columns of pixels used in mosaicing to compensate for the difference between the vertical fields of views. Without any scaling, a virtual camera has to have the same vertical field of view as cameras for a ray database have. This is unacceptable for practical new view generation, because the position and focal length of a virtual camera are strictly restricted. The derivation of the scaling factor has been discussed only for the cases where the camera paths for ray databases are linear and circular [5,6,7,8,9,10]. The limitation on the camera paths should be removed to take full advantage of a 3D plenoptic function. The unstructured Lumigraph rendering [11] which allows an arbitrary configuration of a set of cameras for a 4D plenoptic function may be used with a 3D plenoptic function. No explanation for distortion correction, however, has been presented. To the best of our knowledge, no complete consideration of the scaling factor exist for the cases where the camera paths are arbitrary curves.

This paper presents a rendering algorithm with a scaling scheme, which is general in that it can handle arbitrary camera paths. First, we model a virtual camera by a set of one-dimensional (1D) cameras. Then, we present a rendering algorithm for the cases where the camera paths to create ray databases are arbitrary curves. We conclude with experimental results that demonstrate the usefulness of the proposed algorithm.

2 Modeling Virtual Camera

In this section, we first explain how the rays of a virtual camera are represented by view planes of 1D cameras. Then, we derive the equation for the view plane.

2.1 Representing Rays of Virtual Camera

In IBR based on a 3D plenoptic function, a new view is generated by concatenating columns of pixels corresponding to the rays of a virtual camera [5,6,7,8,9,10] (Fig. 2 (b)). Since a single column of pixels serves as the building block for a new view, we can bundle the rays of each column to represent a virtual camera by a set of 1D cameras corresponding to the columns of pixels. Using 1D cameras, we can model a virtual camera in which all the rays pass through a view point, i.e., a central camera, as shown in Fig. 2 (b).

We call the plane containing the rays of a 1D camera a *view plane*. The position and direction of a view plane determine which a column of pixels needs to be extracted from a ray database. We derive the equation representing the position and direction of a view plane in the next section.

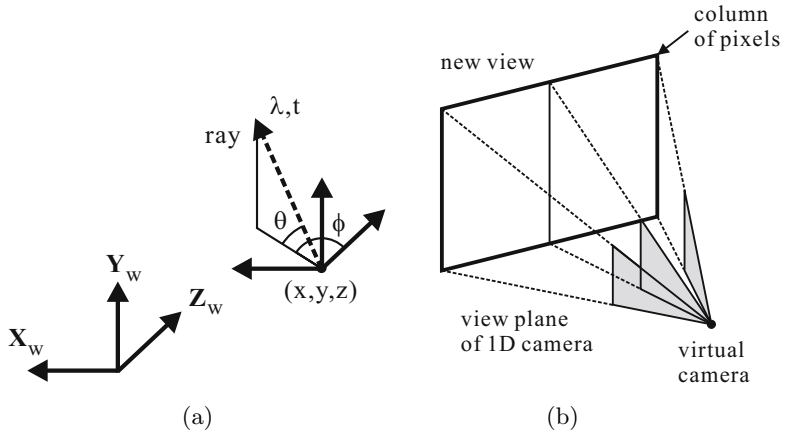


Fig. 2. The plenoptic function and a virtual camera. (a) The plenoptic function in the world coordinate system defined by the $\boldsymbol{X}_{\mathrm{w}}$, $\boldsymbol{Y}_{\mathrm{w}}$, and $\boldsymbol{Z}_{\mathrm{w}}$ axes. Rays are represented by the 7 dimensional plenoptic function with positions $(\mathrm{x}, \mathrm{y}, \mathrm{z})$, directions (θ, ϕ), wavelength λ and time t. To generate new views by extracting rays sampled as a ray database, we represent a virtual camera by a set of 1D cameras. (b) The 1D cameras emulate a central camera in which all rays pass through a view point.

2.2 Deriving View Plane Equation

Figure 2.2 depicts the imaging geometry of a 1D camera. The world coordinate system is denoted by $\boldsymbol{X}_{\mathrm{w}}$, $\boldsymbol{Y}_{\mathrm{w}}$ and $\boldsymbol{Z}_{\mathrm{w}}$. The index of the position of a camera is represented by k. The motion of the k-th camera is represented by the rotation matrix $\boldsymbol{R}_k$ and translation vector $\boldsymbol{t}_k$, which define the camera coordinate system given by $\boldsymbol{X}_k$, $\boldsymbol{Y}_k$ and $\boldsymbol{Z}_k$. The 1D detector of the camera lies on the $\boldsymbol{Y}_k$–$\boldsymbol{Z}_k$ plane and it produces a 1D image.

We denote the world and camera coordinates of a 3D point P as $\boldsymbol{p}_w = (\mathrm{x}_w, \mathrm{y}_w, \mathrm{z}_w)^t$ and $\boldsymbol{p}_k = (\mathrm{x}_k, \mathrm{y}_k, \mathrm{z}_k)^t$, respectively. We know that the camera and world coordinates are related to each other as follows:

$$\boldsymbol{p}_k = \left(\boldsymbol{R}_k^t \mid -\boldsymbol{R}_k^t \boldsymbol{t}_k\right) \begin{pmatrix} \boldsymbol{p}_w \\ 1 \end{pmatrix} , \tag{1}$$

where,

$$\boldsymbol{R}_k^t = (\boldsymbol{i}_k, \boldsymbol{j}_k, \boldsymbol{k}_k)^t \ , \ \boldsymbol{t}_k = (t_{xk}, t_{yk}, t_{zk})^t \ . \tag{2}$$

The rows of the rotation matrix, $\boldsymbol{i}_k$, $\boldsymbol{j}_k$ and $\boldsymbol{k}_k$, define the directions of the axes of the camera coordinate system, $\boldsymbol{X}_k$, $\boldsymbol{Y}_k$ and $\boldsymbol{Z}_k$.

We can express the camera coordinate x_k by expanding Eq. (1) as follows:

$$\mathrm{x}_k = \boldsymbol{i}_k \cdot \boldsymbol{p}_w - \boldsymbol{i}_k \cdot \boldsymbol{t}_k \, . \tag{3}$$

Note that, since the view plane (1D detector) lies on the $\boldsymbol{Y}_k$–$\boldsymbol{Z}_k$ plane, we have $\mathrm{x}_k = 0$. Therefore, we have:

$$\boldsymbol{i}_k \cdot \boldsymbol{p}_w = \boldsymbol{i}_k \cdot \boldsymbol{t}_k \, . \tag{4}$$

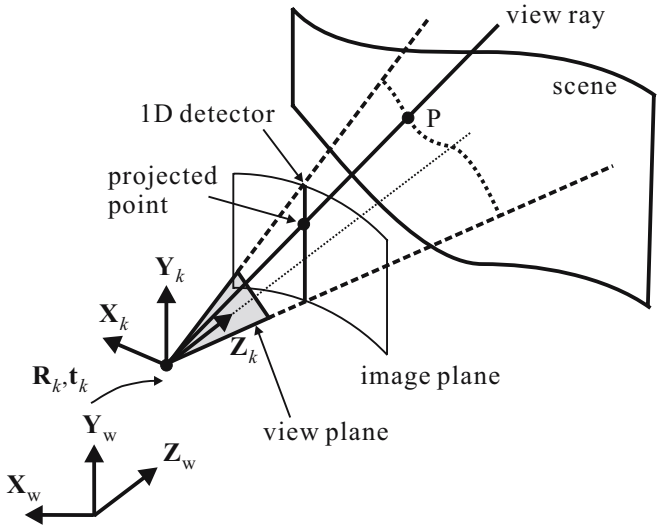


Fig. 3. Imaging geometry of a 1D camera. The world coordinate system is denoted by the $\boldsymbol{X}_\mathrm{w}$, $\boldsymbol{Y}_\mathrm{w}$ and $\boldsymbol{Z}_\mathrm{w}$ axes. The camera coordinate system for the k-th camera is denoted by the $\boldsymbol{X}_k$, $\boldsymbol{Y}_k$ and $\boldsymbol{Z}_k$ axes.The relationship between the world and camera coordinate systems is given by the rotation matrix $\boldsymbol{R}_k$ and translation vector $\boldsymbol{t}_k$. The world and camera coordinates of a 3D point P are $\boldsymbol{p}_w = (\mathrm{x}_w, \mathrm{y}_w, \mathrm{z}_w)^t$ and $\boldsymbol{p}_k = (\mathrm{x}_k, \mathrm{y}_k, \mathrm{z}_k)^t$, respectively. The 1D detector (and hence the view plane) lies on the $\boldsymbol{Y}_k$–$\boldsymbol{Z}_k$ plane.

The above expression, which represents the view plane of a 1D camera with the position $\boldsymbol{t}_k$ and direction $\boldsymbol{R}_k$ passing through the 3D point, is called the *v*iew plane equation.

The rotation of a 1D camera is only along the horizontal direction, i.e., around the $\boldsymbol{Y}_k$ axis, for a 3D plenoptic function $P(x, z, \phi)$. The vertical component of the translation is zero because the camera is on the $\boldsymbol{X}_\mathrm{w}$–$\boldsymbol{Z}_\mathrm{w}$ plane. These facts lead to the following camera motion:

$$\boldsymbol{R}_k^t = \begin{pmatrix} \cos\phi_k & 0 & -\sin\phi_k \\ 0 & 1 & 0 \\ \sin\phi_k & 0 & \cos\phi_k \end{pmatrix}, \quad \boldsymbol{t}_k = (t_{xk}, 0, t_{zk})^t \ . \tag{5}$$

where, ϕ_k is the rotation angle around the $\boldsymbol{Y}_k$ axis of the k-th camera. Using the camera motion, we have the following equations from Eq. (1):

$$\mathrm{x}_k = \mathrm{x}_w \cos\phi_k - \mathrm{z}_w \sin\phi_k - t_{xk}\cos\phi_k + t_{zk}\sin\phi_k \ , \tag{6}$$

$$\mathrm{y}_k = \mathrm{y}_w \ , \tag{7}$$

$$\mathrm{z}_k = \mathrm{x}_w \sin\phi_k + \mathrm{z}_w \cos\phi_k - t_{xk}\sin\phi_k - t_{zk}\cos\phi_k \ . \tag{8}$$

Then we have the view plane equation for a 3D plenoptic function $P(x, z, \phi)$:

$$\mathrm{x}_w \cos\phi_k - \mathrm{z}_w \sin\phi_k = t_{xk}\cos\phi_k - t_{zk}\sin\phi_k \ . \tag{9}$$

In the next section, we present an IBR algorithm with a general scaling factor using Eq. (6), (7), (8) and (9).

3 IBR with General Scaling Factor

We propose an IBR algorithm with a general scaling factor in this section. First, we present the algorithm and then derive the scaling factor.

3.1 IBR Algorithm

The overview of the proposed algorithm is shown in Fig. 4. There are 3 curves in the figure, i.e., a camera path, a focal surface and a geometric proxy. Cameras for a ray database are arranged on the camera path. The focal surface denotes the positions where a virtual camera adjusts its focus [12]. The geometric proxy is used as an approximated shape of a scene. All the curves are represented by polygons whose vertexes are given by the sets of vectors: $\{\boldsymbol{c}_l\}_{l=1}^{L}$, $\{\boldsymbol{f}_m\}_{m=1}^{M}$ and $\{\boldsymbol{g}_n\}_{n=1}^{N}$, respectively. Each vector has the position of a vertex as well as additional information such as the direction of the camera for a ray database.

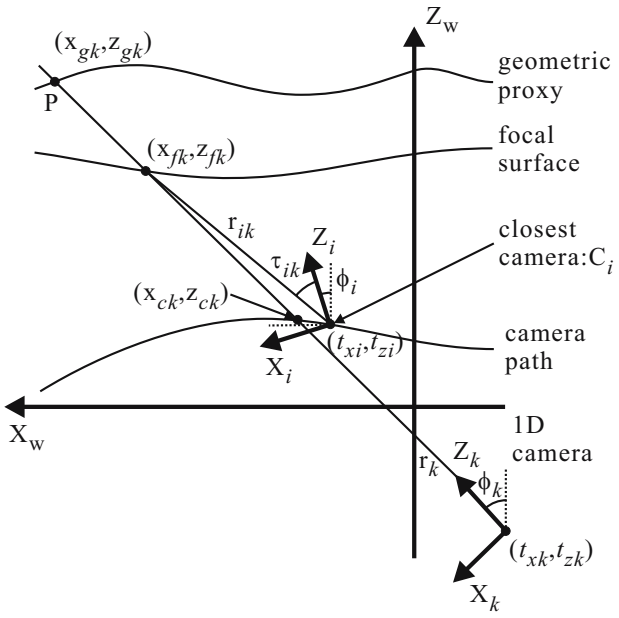


Fig. 4. The overview of the proposed IBR algorithm. The ray associated with the 1D camera with the position (t_{xk}, t_{zk}) and direction ϕ_k is r_k. The camera C_i, which is closest to the point of intersection of r_k and the camera path, $(\mathrm{x}_{ck}, \mathrm{z}_{ck})$, is found from a ray database to know its position (t_{xi}, t_{zi}) and direction ϕ_i. Then the point of intersection of r_k and the focal surface, $(\mathrm{x}_{fk}, \mathrm{z}_{fk})$, is computed. The angle of the ray r_{ik}, τ_{ik}, is obtained by the intersection point and the position of C_i. The position of a column of pixels corresponding to r_{ik} is calculated. After extracting the column at the position, the column is scaled based on the distances from the point of intersection of r_k and the geometric proxy, $(\mathrm{x}_{gk}, \mathrm{z}_{gk})$, to the 1D camera and the closest camera C_i.

The proposed IBR algorithm is summarized as follows:

[Step 1: Setting a view point] The position (t_{xk}, t_{zk}) and direction ϕ_k of a 1D camera are given based on a user request. The ray of the camera is expressed as r_k.

[Step 2: Finding the closest camera] The point of intersection of the ray r_k and the camera path is computed. This computation is performed by intersection check between the view plane of r_k obtained by Eq. (9) and the polygons $\{\boldsymbol{c}_l\}_{l=1}^{L}$. The vertex $\boldsymbol{c}_i$, which is closest to the point of intersection $(\mathrm{x}_{ck}, \mathrm{z}_{ck})$, is found and the camera at the vertex is regarded as the closest camera C_i. The vertex has the position and direction of C_i, and these are denoted as $\boldsymbol{c}_i = (t_{xi}, t_{zi}, \phi_i)^t$.

[Step 3: Finding the intersection point with the focal surface] The point of intersection between the ray r_k and the focal surface is computed. This computation is done by intersection check between the view plane of r_k and the polygons $\{\boldsymbol{f}_m\}_{m=1}^{M}$. The point of intersection is shown as $(\mathrm{x}_{fk}, \mathrm{z}_{fk})$.

[Step 4: Computing the angle of the ray] The ray of C_i passing through the point $(\mathrm{x}_{fk}, \mathrm{z}_{fk})$, i.e., r_{ik} shown in Fig. 4, corresponds to the column of pixels required for generating a new view. The view plane of r_{ik} is obtained by Eq. (9) as follows:

$$\begin{aligned} &\mathrm{x}_{fk}\cos(\phi_i + \tau_{ik}) - \mathrm{z}_{fk}\sin(\phi_i + \tau_{ik}) \\ &\quad = t_{xi}\cos(\phi_i + \tau_{ik}) - t_{zi}\sin(\phi_i + \tau_{ik}) \,. \end{aligned} \tag{10}$$

where, τ_{ik} is the angle of r_{ik}, which is needed to extract the required column of pixels. This angle is expressed as:

$$\tau_{ik} = \mathrm{T}an^{-1}\left(\frac{a - \tan\phi_i}{1 + a\tan\phi_i}\right),\ a = \frac{x_{fk} - t_{xi}}{z_{fk} - t_{zi}}. \tag{11}$$

[Step 5: Finding the position of the column of pixels] The position of the column of pixels for mosaicing, d_{ik}, is obtained as follows:

$$d_{ik} = f_i \tan\tau_{ik} \,. \tag{12}$$

where, f_i is the focal length of C_i.

[Step 6: Finding the intersection point with the proxy] The point of intersection P between the ray r_k and the geometric proxy is computed. This computation is performed by intersection check between the view plane of r_k and the polygons $\{\boldsymbol{g}_n\}_{n=1}^{N}$. The point of intersection is shown as $(\mathrm{x}_{gk}, \mathrm{z}_{gk})$.

[Step 7: Scaling the column of pixels] The column of pixels at the position d_{ik} is scaled using the scaling factor s_k.

$$s_k = \frac{f_k}{f_i}\frac{z_{ik}}{z_k} \,, \tag{13}$$

where,

$$\mathrm{z}_{ik} = \mathrm{x}_{gk}\sin\phi_i^{'} + \mathrm{z}_{gk}\cos\phi_i^{'} - t_{xi}\sin\phi_i^{'} - t_{zi}\cos\phi_i^{'} \,, \tag{14}$$

$$\phi_i^{'} = \phi_i + \tau_{ik}^{'} \,, \tag{15}$$

$$\tau_{ik}^{'} = \mathrm{T}an^{-1}\left(\frac{a^{'} - \tan\phi_i}{1 + a^{'}\tan\phi_i}\right),\ a^{'} = \frac{x_{gk} - t_{xi}}{z_{gk} - t_{zi}}\,, \tag{16}$$

$$\mathrm{z}_k = \mathrm{x}_{gk}\sin\phi_k + \mathrm{z}_{gk}\cos\phi_k - t_{xk}\sin\phi_k - t_{zk}\cos\phi_k\,, \tag{17}$$

f_k is the focal length of the 1D camera and $\tau_{ik}^{'}$ is the angle of the ray connecting the position of the closest camera and the intersection point P. The use of the angle implies that the scaling factor is determined by the position of the geometric proxy; the position of the focal surface is ignored in scaling.

[Step 8: Averaging the columns of pixels] Step1 to Step7 are applied not only to the closest camera but also to multiple cameras near the intersection point $(\mathrm{x}_{ck}, \mathrm{z}_{ck})$. Then the weighted average of the scaled columns of pixels is calculated. The number of cameras that plays the role of the aperture of a virtual camera [12] is 3 for this algorithm. The weights for averaging are given by the Gaussian distribution $N(0, 4)$.

[Step 9: Mosaicing] Step1 to Step 8 are applied to all 1D cameras in a virtual camera. By concatenating the averaged columns of pixels, we can generate a new view.

The scaling factor s_k of Eq. (13) is derived in the next section.

3.2 Deriving Scaling Factor

Assume that a 1D camera and the closest camera C_i observe the same point P on a geometric proxy with the coordinates (x_{gk}, z_{gk}). The points projected on the cameras are denoted by v and $v^{'}$, respectively. The ratio between v and $v^{'}$ is the required scaling factor. Using the perspective projection of the cameras, we can express the ratio as follows:

$$\frac{v}{v^{'}} = \frac{f_k \mathrm{y}_k}{\mathrm{z}_k}\frac{\mathrm{z}_{ik}}{f_i \mathrm{y}_{ik}}\,. \tag{18}$$

where, y_k, z_k and y_{ik}, z_{ik} are the camera coordinates of P for a 1D camera and C_i, respectively.

From Eq. (7), we know that $\mathrm{y}_k = \mathrm{y}_{ik} = \mathrm{y}_w$. Thus, Eq. (18) is equivalent to Eq. (13). We can represent z_{ik} and z_k as shown in Eq. (14) to (17) using Eq. (8), (11). Therefore, the scaling factor represented by Eq. (13) to (17) is obtained.

The scaling factor for polar coordinates is useful for the IBR algorithm using the circular camera path for a ray database [5,7,8,9,10]. The scaling factor is derived in Appendix A.

It is important to note that no assumption is imposed on the shape of the camera path in the above derivation. The derived scaling factor is, therefore, general in that it can be applied to the cases where camera paths are arbitrary curves.

4 Experimental Results

In this section, we show experimental results of the proposed algorithm. The main purpose of the experiments was to confirm the effect of the proposed scaling scheme.

Figure 5 shows a ray database created by using 3D rendering software, POV-Ray [13]. We rendered 900 frames including the 3 frames in the figure. Note that the camera path for the database was a sine curve, which leads to the changes in the depths between the cameras and the scene. The virtual camera was the central one, as shown in Fig. 2 (b). Since the distance between the virtual camera

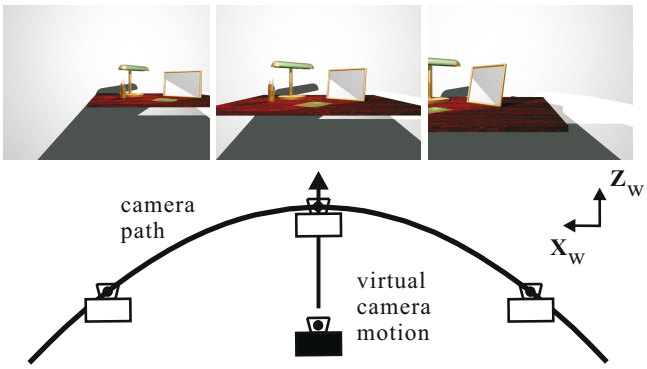
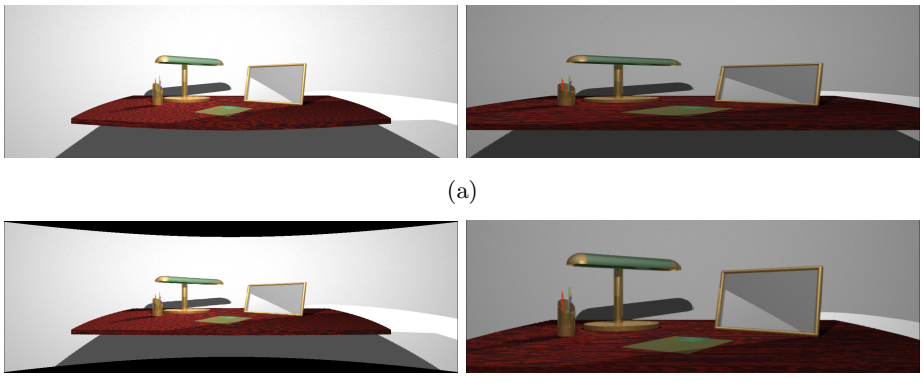


Fig. 5. The ray database obtained by POV-Ray [13]. The camera path was a sine curve. The virtual camera was the central one, as shown in Fig. 2 (a), and it was moved forward.



(a)

(b)

Fig. 6. The new views obtained from the virtual camera with the motion shown in Fig. 5. (a) without scaling. (b) with scaling. Left,Right: the new views corresponding to the different positions of the virtual camera. Note the changes in the aspect ratios of the objects in (a). The deformation of the left image in (b) demonstrates the compensation of the change in depth of the camera path in Fig. 5 by the scaling factor.

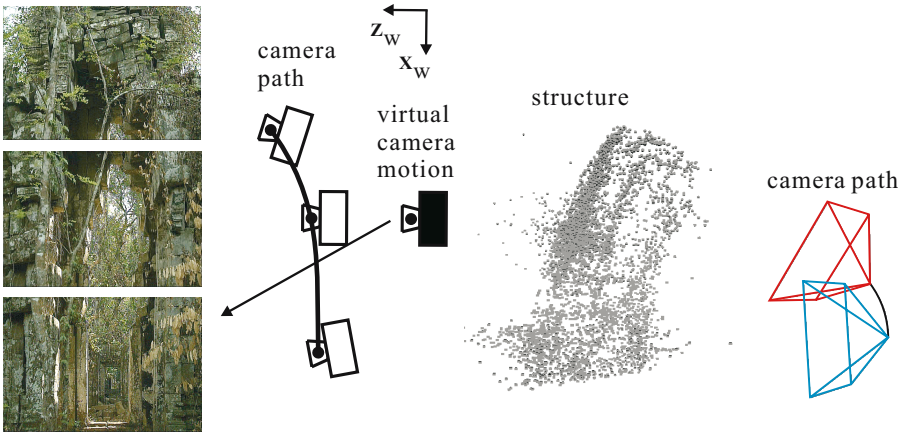


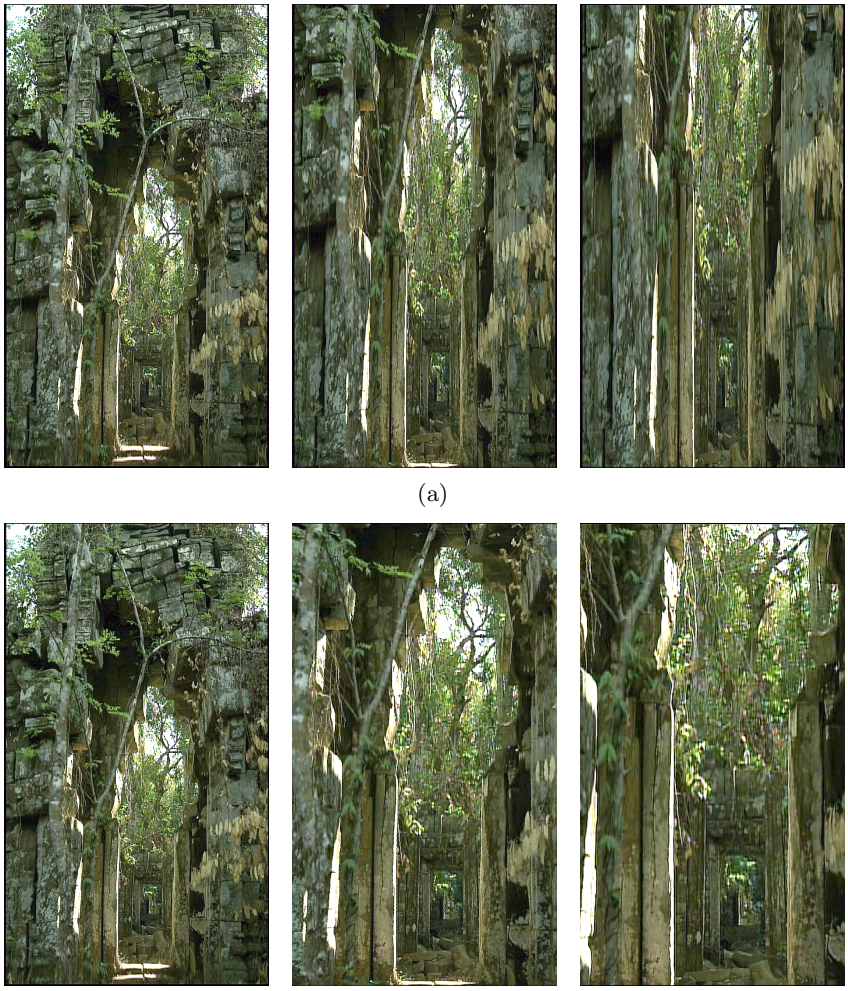
Fig. 7. The ray database for a remain in the Kingdom of Cambodia. (a) The virtual camera was the central one and it was moved downward from the center of the scene. (b) The camera path and the rough structure of the scene were estimated by using Voodoo Camera Tracker [14]. Note that the rough structure is enough for a geometric proxy.

and the scene has to be changed significantly in order to confirm the usefulness of the scaling factor, the virtual camera was moved forward.

The focal surface and geometric proxy were planes and these were placed at the same position as the average depth of the scene. The focal lengths of the cameras for the ray database and the virtual camera were $f_i = 480$ and $f_k = 960$[pixel], respectively. The change in the depth of the camera path, the position of a virtual camera and the difference between the focal lengths yielded the aspect ratio distortion.

The new views are shown in Fig. 6. Figure 6 (a) is the result obtained without scaling and Fig. 6 (b) is the result obtained with scaling. The left and right images of the figures correspond to the different positions of the virtual camera. Without scaling, the aspect ratios of the objects were changed as the virtual camera was moved. On the other hand, the aspect ratio distortion was corrected by scaling. The deformation of the left image in Fig. 6 (b) demonstrates that the scaling of the columns of pixels compensated for the change in the depth of the camera path.

We now present the results for IBR using real images. Figure 7 (a) shows the ray database using the images of a remain in the Kingdom of Cambodia. We used 330 frames, including the 3 frames in the figure. The positions and directions of the cameras for the frames were estimated by using Voodoo Camera Tracker [14], software for a structure from motion algorithm, as shown in Fig. 7 (b). Since the camera motion was controlled by a dolly, the camera moved on a plane. This fact enables us to use the 3D plenoptic function. The virtual camera was the central one and it was moved downward from the center of the scene as depicted in Fig. 7 (a).



(a)

(b)

Fig. 8. The new views obtained from the virtual camera with the motion shown in Fig. 7. (a) without distortion correction. (b) with distortion correction. Note the serious aspect ratio distortions in the new views of (a).

The fixed focal length of the camera for the ray database and the 3D structure of the scene were also estimated by the software. The focal surface and geometric proxy were planes and these were placed at the same position as the average depth of the scene. The focal lengths of the cameras for the ray database and the virtual camera were $f_i = 46$ and $f_k = 91$[pixel], respectively. The focal length f_k was determined to cover the entire scene at the initial position of the virtual camera.

Figure 8 shows the new views. Since no aspect ratio correction was done in Fig. 8 (a), the new views had the serious aspect ratio distortion which deteri-

orated the quality of the new views. For example, the width of the gate in the scene did not change although the virtual camera was moved. In Fig. 8 (b) using the proposed scaling scheme, the width of the gate gradually changed as the virtual camera was moved owing to the compensation of the aspect ratio. The quality of the new views was sufficient to emulate the virtually generated camera motion.

In summary, the experimental results with large changes in the distances between the virtual cameras and the scenes demonstrate that our algorithm can correct aspect ratio distortions.

5 Summary

We have proposed an IBR algorithm with a scaling scheme based on the modeling of a virtual camera using 1D cameras. The scaling scheme is general in that it can handle arbitrary camera paths. We demonstrated the usefulness of the proposed IBR algorithm by the experiments in which new views were generated using several ray databases. We believe that the scaling scheme presented here will facilitate the use of IBR based on a 3D plenoptic function.

References

1. Shum, H.Y., Chan, S.C., Kang, S.B.: Image-Based Rendering. Springer, Heidelberg (2007)
2. Adelson, E.H., Bergen, J.R.: The Plenoptic Function and the Elements of Early Vision, pp. 3–20. MIT Press, Cambridge (1991)
3. Levoy, M., Hanrahan, P.: Light field rendering. In: Proc. SIGGRAPH 1996, pp. 31–42 (1996)
4. Gortler, S.J., Grzeszczuk, R., Szeliski, R., Cohen, M.F.: The lumigraph. In: Proc. SIGGRAPH 1996, pp. 43–54 (1996)
5. Shum, H.Y., He, L.W.: Rendering with concentric mosaics. In: Proc. SIGGRAPH 1999, pp. 299–306 (1999)
6. Weinshall, D., Lee, M.S., Brodsky, T., Trajkovic, M.: New view generation with a bi-centric camera. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 614–628. Springer, Heidelberg (2002)
7. Bakstein, H., Pajdla, T., Vecerka, D.: Rendering almost perspective views from a sparse set of omnidirectional images. In: Proc. BMVC, pp. 241–250 (2003)
8. Zomet, A., Feldman, D., Peleg, S., Weinshall, D.: Mosaicing new views: The crossslits projection. IEEE Trans. PAMI 25(6), 741–753 (2003)
9. Bakstein, H., Pajdla, T.: Rendering novel views from a set of omnidirectoinal mosaic images. In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW 2003), pp. 74–79 (2003)
10. Bakstein, H., Pajdla, T.: Omnidirectional image-based rendering. In: Proc. Computer Vision Winter Workshop (CVWW 2006), pp. 99–104 (2006)
11. Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M.: Unstructured lumigraph rendering. In: Proc. SIGGRAPH 2001, pp. 425–432 (2001)
12. Isaksen, A., McMillan, L., Gortler, S.J.: Dynamically reparameterized light fields. In: Proc. SIGGRAPH 2000, pp. 297–306 (2000)
13. `http://www.povray.org/`
14. `http://www.digilab.uni-hannover.de/docs/manual.html`

A Scaling Factor for Polar Coordinate System

The camera coordinates z_{ik} and z_k of Eq. (14) and (17) are expressed as follows:

$$z_{ik} = d_k \sqrt{1 - \left\{ \frac{R_i}{d_k} \sin\left(\xi_i - \phi_i^{'}\right) \right\}^2} - R_i \cos\left(\xi_i - \phi_i^{'}\right), \tag{19}$$

$$z_k = d_k \sqrt{1 - \left\{ \frac{R_k}{d_k} \sin\left(\beta_k - \phi_k\right) \right\}^2} - R_k \cos\left(\beta_k - \phi_k\right), \tag{20}$$

where,

$$d_k = \sqrt{x_{gk}^2 + z_{gk}^2}\,, \tag{21}$$

$$R_i = \sqrt{t_{xi}^2 + t_{zi}^2},\ \ \tan\xi_i = t_{xi}/t_{zi}\,, \tag{22}$$

$$R_k = \sqrt{t_{xk}^2 + t_{zk}^2},\ \ \tan\beta_k = t_{xk}/t_{zk}\,. \tag{23}$$

Using the expression, we can derive the scaling factor of Eq. (13). This scaling factor is useful for the IBR algorithms using the polar coordinate system to represent the rotation of cameras for a ray database [5,7,8,9,10]. For example, if the camera path is circular, R_i in Eq. (22) is the radius of the circle and ξ_i the angle of the closest camera C_i.

We show the derivation of Eq. (19) as follows. Equation (14) is denoted as:

$$\begin{aligned} z_{ik} &= d_k \left(\sin\alpha_k \sin\phi_i^{'} + \cos\alpha_k \cos\phi_i^{'}\right) - R_i \left(\sin\xi_i \sin\phi_i^{'} + \cos\xi_i \cos\phi_i^{'}\right), \\ &= d_k \sqrt{1 - \sin\left(\alpha_k - \phi_i^{'}\right)^2} - R_i \cos\left(\xi_i - \phi_i^{'}\right). \end{aligned} \tag{24}$$

where, $\phi_i^{'}$, d_k and R_i are given by Eq. (15), (21) and (22), and $\tan\alpha_k = x_{gk}/z_{gk}$. The view plane of the ray of C_i passing through the coordinates (x_{gk}, z_{gk}) is given by Eq. (9):

$$x_{gk}\cos\phi_i^{'} - z_{gk}\sin\phi_i^{'} = t_{xi}\cos\phi_i^{'} - t_{zi}\sin\phi_i^{'}\,. \tag{25}$$

From the expression, we have:

$$\sin\left(\alpha_k - \phi_i^{'}\right) = \frac{R_i}{d_k}\sin\left(\xi_i - \phi_i^{'}\right). \tag{26}$$

Substituting Eq. (26) in Eq. (24), we obtain Eq. (19).

We can obtain Eq. (20) for the camera coordinate z_k by applying the same procedure as that used for Eq. (19).