The Journal of the Society for Art and Science, Vol. 20, No. 1, pp. 55-71 (2021)

View-dependent Projection Mapping Enhanced by Real Background

Khuslen Battulga[1] (Nonmember) **Tadahiro Fujimoto**[1] (Member)

1) Graduate School of Engineering, Iwate University

khuslenb@gmail.com fujimoto@cis.iwate-u.ac.jp

Abstract

"Spatial augmented reality (SAR)" or "projection mapping" projects an image of a virtual object on the surface of a real object. A "view-dependent" display shows the virtual object with its correct appearance for an arbitrary viewer's position. If the virtual object and the real object have different shapes, the virtual object's image to project needs to be correctly distorted according to the viewer's position. Besides, the difference causes the real object's surface to have some empty areas onto which the virtual object is not projected. Such empty areas seriously degrade the viewer's feeling that the virtual object merges into the real world. We propose a method to eliminate undesired empty areas by projecting the real background behind the real object in a view-dependent way. Our method converts a real background's image captured by a fixed camera to an appropriate image for a viewer's position based on homography. This image conversion approximates the background's shape by a plane adjusted for an arbitrary viewer's position. The plane is determined by practical parameters interpolated on a 3D grid space. Consequently, the viewer does not feel the presence of the real object and feels the virtual object merging into the real world.

1. Introduction

1.1 Background

Augmented reality (*AR*) [1][2] is a computer-based technology to generate a *virtual world* by adding a variety of virtual things, such as *virtual objects*, to the *real world* in order to augment the real environment. The virtual world is usually displayed on various kinds of devices, such as a monitor, a tablet, a smart phone, a head-mounted display, and special glasses. A viewer usually has his/her own display device to see an image of the virtual world generated for only him/her according to his/her viewing position. This means that multiple viewers see different images respectively. The viewer feels as if a virtual object actually existed in the real world if the appearance of the object is observed correctly from an arbitrary position around it. The display to show a virtual object's *correct appearance* according to a viewer's position is usually called *view-dependent* display, which is so important in AR. One typical way for view-dependent display is to use a *marker* to indicate the exact 3D position of a virtual object to obtain a correct image of the object seen from a viewer's position. Some systems use a screen made from special material to reflect a different correct image according to a viewer's position. "Parallax" is a visual effect caused by the difference of the positions of left and right eyes. In a typical stereoscope system, a viewer wears special glasses that give the respective eyes different images generated by the parallax effect. In AR, usually, a displayed image depends on a viewer's position.

Spatial augmented reality (*SAR*) [1][2][3][4] augments the real world without display devices described above. SAR usually uses a projector to display an image by projecting it onto the surface of a *real object*, such as a screen, a wall, a building, a house, a car, and even a human in the real world. *Projection mapping* is a well-known technology to achieve SAR. In this paper, we use "SAR" and "projection mapping" as the same meaning if the usage causes no problem. In SAR, a viewer sees an image projected on a real object's surface by the naked eyes; that is, multiple viewers see the same image simultaneously. In this sense, the displayed image does not depend on the viewer's position [2]. Actually, an SAR system sometimes, or often, does not need the view-dependent display of a virtual object. If it is needed, one easy way is to project an image of a virtual object onto the surface of a real object whose shape is the same as the virtual object's shape. In this case, the virtual object's appearance projected on the real object's surface is observed correctly from an arbitrary viewer's position. However, it is practically difficult to prepare such a real object for an arbitrary virtual object. Then, the difference between their shapes causes a problem of "incorrect appearance" of the virtual object. The difference between the shapes causes another problem. The real object's surface has "empty areas" on which the virtual object is not projected. Such empty areas seriously degrade a viewer's feeling that the virtual object merges into the real world.

1.2 Objective

The above "incorrect appearance" problem has been one of main problems in SAR. Many view-dependent methods to solve this problem have been proposed. On the other hand, as far as we know, there is no existing method to treat the "empty area" problem. In this paper, we propose a method to eliminate undesired empty areas by projecting the real background behind the real object in a view-dependent way. In order to treat a far background in a large space, even outdoors, the real background is not measured by an RGB-D camera with a depth range limit but captured by a usual RGB camera and approximated by a plane according to a viewer's position. Then, the captured image is converted to an appropriate image for the viewer's position based on homography. Although the plane is determined manually, a practical way is proposed to adjust it easily, intuitively, and efficiently by devising effective background parameters. A simple grid-based interpolation mechanism for the parameters provides an appropriate plane to an arbitrary viewer's position. Consequently, the projected real background shows its appearance as correctly as possible so as to match the real background directly seen by the viewer seamlessly along the real object's contour. Our method makes the viewer not feel the presence of the real object but feel the virtual object merging into the real world.

Many view-dependent SAR methods have been developed to allow multiple viewers to see a virtual object's correct appearance simultaneously by relying on a specialized device with multiple projectors and a real object with specialized material. On the other hand, an existing method using a non-specialized configuration with an off-the-shelf projector and an ordinary real object sacrifices the important merit of SAR; only a single viewer is allowed to see the correct appearance. Our method has the same situation. However, the simplicity of our method enables the integration with an existing method developed for multiple viewers.

In Section 2, we describe related works and arguments for our method. In Section 3, we explain our method. In Section 4, we show experimental results. Finally, in Section 5, we conclude this paper.

2. Related Work and Argument

2.1 Related Work

One of pioneering technologies with respect to SAR is "the office of the future" [5] proposed in 1998. The representative works proposed in recent 20 years are surveyed intelligibly in [6][7]. The view-dependent projection needs an appropriate *projection image* to project on a real object surface by a projector. Such a projection image depends on a projector's position, pose, and optical properties as well as a real object surface's geometry and material properties such as texture and reflectance. The projector and the real object surface need to be calibrated in advance. A calibration in SAR consists of *geometric calibration* and *photometric calibration* [6][7]. A geometric calibration makes a *projected image* on a real object surface geometrically correct without distortion. When multiple projectors are used, projected images by the respective projectors are partially overlapped so as to reconstruct a virtual object by their correct arrangement. A photometric calibration makes a projected image

photometrically correct such that the projected image's colors become as similar as possible to its original image's colors according to a real object surface's material properties. When multiple projectors are used, the color intensities of the respective projected images are adjusted on the overlap areas. A typical calibration uses a camera to capture a projected image for automatic geometric and photometric calibration. Such a calibration is usually called *projector-camera calibration* [6][7].

Our method involves geometric calibration. Therefore, we mention it below. A geometric calibration estimates the geometry of a real object surface and the intrinsic and extrinsic parameters of a projector. A projector-camera calibration also estimates the intrinsic and extrinsic parameters of a camera. Some methods estimate unknown projector parameters by using known camera parameters while others estimate all unknown projector and camera parameters. Besides, some methods estimate an unknown real object surface's geometry while others estimate unknown projector and camera parameters by using a known surface's geometry. "Shader Lamps" [8] is an early well-known SAR work. First, the 3D shape of a real object is measured by a 3D touch probe scanner to obtain its 3D model as a virtual object. Then, multiple projectors are calibrated by projecting markers from each projector onto the real object's surface and matching the projector's marker pixels with 3D points on the surface. Geometric projector-camera calibrations are categorized into *semi-automatic calibration* and *self-calibration* [7]. A semi-automatic calibration uses a specialized apparatus [9][10][11][12] [13]. Typically, a projector-camera pixel correspondence is estimated by projecting structured light patterns, such as a chessboard pattern, onto a planar surface, such as a screen, by a projector and capturing it by a camera. A homography matrix is often used to relate the planar surface with the image planes of the projector and the camera. A self-calibration does not use a specialized apparatus [14][15][16][17][18]. Typically, structured light patterns are projected onto a non-planar surface whose geometry is unknown. Recently, various *dynamic projection mapping* methods have been proposed to calibrate a moving real object, such as a human face and clothes, and give an appropriate projection in real time. For example, an object silhouette [19], a rigid surface [20], and a non-rigid surface [21][22][23] have been treated.

An existing SAR method using a non-specialized configuration basically enables view-dependent projection for only a single viewer. For example, "HeatSpace" [24] and "OptiSpace" [25] systems use projectors and Kinect sensors. Each system measures and analyzes the environment in a room and the movement of a single viewer during a certain time, and determines the optimal surface on which a virtual object image should be projected in the environment. The viewer can see the correct appearance of the virtual object within an analyzed small area. In "dyadic SAR system" [26], two viewers stand face-to-face with each other near the opposite walls in a room and interact with a common virtual object. This system uses three projector-Kinect pairs mounted on the ceiling. The two pairs face the viewers respectively. Each pair measures the surfaces of one viewer and his/her back environment and projects a virtual object image on the surfaces. The projected image is seen by another viewer. The viewers perceive the virtual object existing in the same 3D position. The remaining pair measures and projects the environment between the viewers. Even in this system, each projected image gives the virtual object's correct appearance to only a single viewer. The above systems can work in only a non-large room-scale environment because the real-time measuring by a Kinect sensor, generally an RGB-D camera with a depth range limit, is necessary. On the other hand, many SAR systems to allow multiple viewers to see the correct appearance of a common virtual object simultaneously have been proposed [27][28][29] [30][31][32][33]. Such SAR is often called *light field projection*. These systems use specialized configurations with special devices and real object surfaces; typically, they use multiple projectors and an anisotropic or lenticular surface made from special material to reflect different images to different directions. The development of view-dependent projection technologies for multiple viewers using non-specialized configurations is an important future work in SAR.

2.2 Argument

The main purpose of AR is to augment the real world by adding a virtual object and making a viewer feel the object merging into the real world. Multiple viewers can experience the augmentation simultaneously by using their respective display devices. On the other hand, SAR has mainly two purposes. One purpose is to decorate a real object's surface by projecting various images. Multiple viewers can enjoy the decoration simultaneously by the naked eyes [8][19][20][21][22][23][34]. This purpose needs a 3D model of the real object as a virtual object. Another purpose is to display a virtual object on the surface of a real object from which the virtual object does not originate, such as a screen and a wall. In this case, the real object's shape has no relation with the virtual object's shape. A typical method aims to provide the view-dependent projection of the virtual object without any consideration for the relation between the shapes. When the virtual object is desired to merge into the real world, there is no problem if the real object should be shown as an element of the real world [26]. On the other hand, there are many cases in which the presence of the real object should not be shown. Consequently, the "empty area" problem happens. When a viewer moves freely within a wide range around the real object, this problem becomes more serious.

As far as we know, there is no method to solve the empty area problem intendedly. The system of [29] uses a specialized mirror rotating at high speed to reflect a virtual object image to a viewer. The real background behind the mirror is seen through it intermittently, which results in causing a visual effect by which the virtual object is floating in front of the background without feeling empty areas. However, the viewer unavoidably perceives the presence of the mirror. Our method solves the empty area problem without any specialized device.

Our method uses not only a projector but also a camera, which is used to capture a real background. We assume that the intrinsic and extrinsic parameters of the projector and the camera and the geometry of a real

object are already known by geometric calibration. An arbitrary photometric calibration method can be used for our method. However, in the experiments, photometric calibration was not applied and a white object with a diffuse reflection surface was used as a real object. Besides, a tracking sensor is used to track a moving viewer in real time. Our method obtains a color image of the background by the camera, approximates the shape of the background by a plane, and converts the image to an appropriate image for a viewer's position based on homography. This enables our method to treat a far background in a large space, even outdoors. The plane is determined by effective background parameters, which are interpolated on a 3D grid space to adjust the plane for an arbitrary viewer's position. Our method uses the "two-pass algorithm" [5] [35][36] to obtain a projection image containing a virtual object and a real background for view-dependent projection. The projection of this image results in displaying the virtual object's correct appearance on a real object's surface whose empty areas are eliminated by the projected real background according to a viewer's position.

3. View-dependent projection mapping enhanced by real background

3.1 Outline

If the surface of a real object is given the projection of a virtual object and their shapes are different from each other, the virtual object's correct appearance cannot be seen from an arbitrary viewer's position. This is explained in Figure 1, in which projection mapping is simulated virtually by a graphic library OpenGL. In the following, the notation "(a1,2)" means "(a1) and (a2)", and "(a,b1)" means "(a1) and (b1)". In this simulation, a virtual object is a *teapot* shown in (b1,2,3), and a real object is a *cube* shown in (c1,2,3). The images (a1,2) are top views to show the positions of the teapot, the cube, a projector, and a viewer on the horizontal plane. The images (b,c2) are seen from the projector's position while the images (b,c1) and (b,c3) are seen from the left and right viewer's positions. The images (b1,2,3) are obtained by rendering the teapot from the respective positions; they have the teapot's correct appearances. The gray colors in (c1,2,3) mean depth values from the respective positions. If the image (b2) is used as a projection image and projected onto the cube's surface by the projector, the surface is seen from the respective positions as shown in (d1,2,3); the red part on the cube's surface in (a2) is given the projection of the teapot. The teapot's appearance is correct in (d2), while it is incorrect in (d1,3). Our method solves the "incorrect appearance" problem by the "two-pass algorithm" [5][35][36], which "correctly distorts" an image with correct appearance to obtain an appropriate projection image to project by a projector according to the positional relation between the projector and a viewer.

In (d1,2,3), the gray-colored empty areas on the cube's surface are not given the projection of the teapot. The colored boxes in (e1,2,3) simulate the real background behind the cube seen from the respective positions in the real world. Then, the actual views seen from the positions become (f1,2,3). In addition to the incorrect appearance, the empty areas greatly

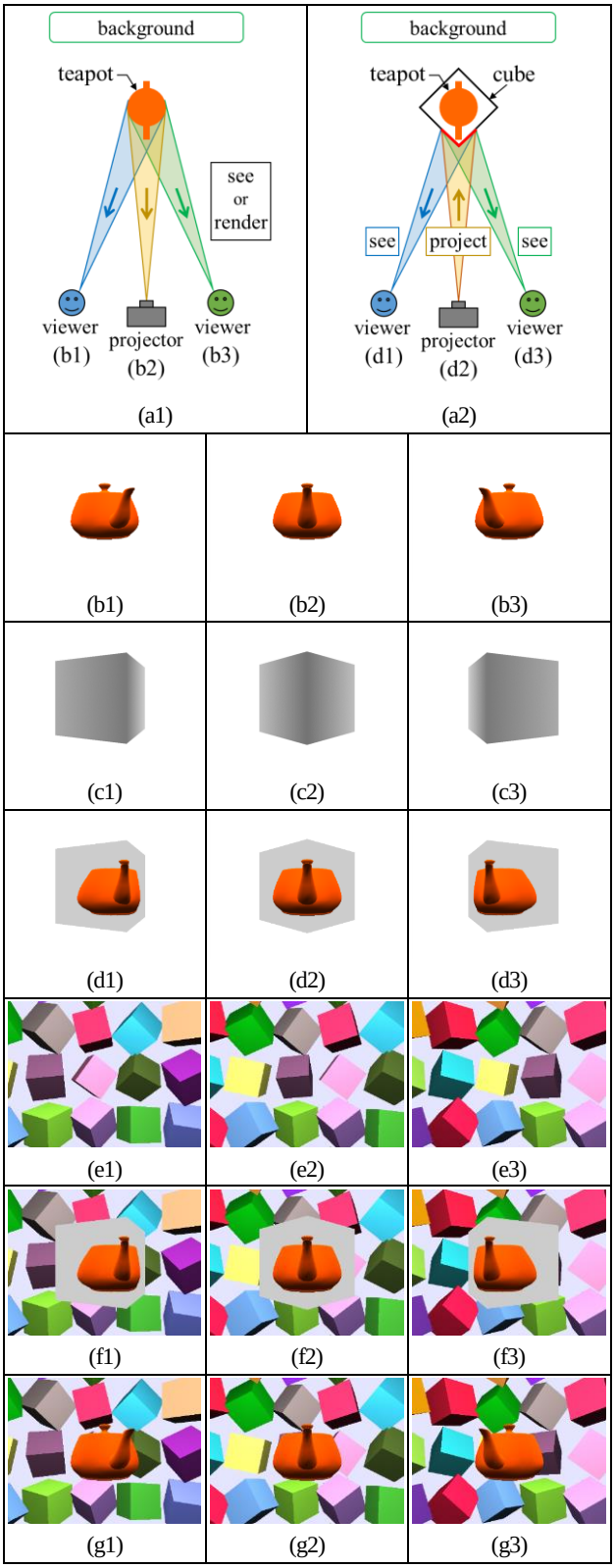

Figure 1. Correct/incorrect appearances of projected images and enhancement by real background.

degrade the viewer's feeling that the teapot merges into the real world. In order to avoid this problem, our method captures an image of the real background behind a real object by a camera and projects the image onto such empty areas. This results in making a viewer not feel the presence of the real object. The view-dependent projection of the teapot and the

real background with their correct appearances provides the ideal views in (g1,2,3), in which the teapot looks as if it were floating in the air in front of the real background.

We need to consider how to obtain the background image seen from the viewer's position. One solution is that the viewer carries a camera to directly use a captured background image. However, this imposes a burden on the viewer. Practically, the camera should be fixed at a certain position. Then, a background image captured from the camera's position needs to be converted into an image seen from the viewer's position. An easy solution is to use an RGB-D camera to obtain not only a color image but also a depth image of objects in the background. The depth image is used to obtain the 3D shapes of the background objects, which are rendered from the viewer's position to obtain the objective image. However, the quality of such an image obtained by using a reasonable RGB-D camera, such as a Kinect sensor, is not so high because of the measurement errors of depth values and the pixel correspondence errors between a color image and a depth image. Besides, as a crucial problem, the depth value that the RGB-D camera can measure is limited to a certain range; for example, the practical depth range of a Kinect sensor is limited from 0.5 to 4.0 meters. This means that a background far from the camera cannot be measured. A traditional solution in computer vision to estimate depth values from color images captured by multiple RGB cameras can treat a far background. However, it is troublesome and difficult to set up and calibrate the cameras accurately. Besides, the estimation is not always exact and stable. Therefore, our method uses only a color image captured by a single RGB camera to treat a far background in a large space, even outdoors, easily and stably. Our method approximates the 3D shape of the background, that is, the 3D shapes of the background objects by a *background plane* according to a viewer's position. Then, the captured image is converted into an image seen from a viewer's position based on homography. The converted background image by the approximation needs to be similar to the real background directly seen by the viewer as exactly as possible. Particularly, in order to avoid the viewer's strange feeling, the converted background projected on the real object's surface should match the directly-seen real background seamlessly along the real object's contour.

The algorithm of our method executes the following steps, as shown in Figure 2. Steps 1 and 2 are done in the *virtual space* of a computer while Step 3 is done in the *real space*, that is, the real world. Step 2 is known as the two-pass algorithm [5][35][36]. These are executed in real time for the sake of the movement of a viewer and background objects.

Step 1: A *viewer composite image* seen from a viewer's position is generated by the next sub-steps, as shown in Figure 2 (a1).

Step 1-1: A virtual object is rendered from the viewer's position to generate a *viewer virtual object image* (b1).

Step 1-2: A background image captured by the camera, called *camera background image* (c1*), is converted into a background image seen from the viewer's position, called *viewer background image* (c1).

Step 1-3: The viewer virtual object image (b1) and the viewer background image (c1) are combined into a viewer composite image (d1).

Step 2: The viewer composite image (d1) is used as a texture and projectively mapped on the real object's surface. The textured surface is rendered from the projector's position to generate a projection image (d2), as shown in (a2). The image (d2) consists of (b2) and (c2), which are generated from (b1) and (c1) respectively.

Step 3: The projection image (d2) is projected on the real object's surface, which results in showing the view of the virtual object's correct appearance (d3), as shown in (a3). The view (d3) consists of (b3) and (c3). The view (e3) shows the background directly seen by the viewer in the real world. Then, the actual view seen by the viewer is (d3'). The view (c3') contains only the background.

The details of these steps are explained below. The main novelty of our method is in Step 1-2. The viewer background image is generated from the camera background image by a background plane defined by appropriate background parameters interpolated on a 3D grid space. In the following, we use "background" instead of "real background" if it does not mislead a reader.

3.2 Practice environment

We need to prepare the same practice environment in both of the real space and the virtual space. In the real space, a real object, a projector, and a camera are prepared. In the virtual space, an accurate 3D geometric model of the real object is made. Then, the registration between the spaces is done by placing the real object, the projector, and the camera in the same positions and orientations in both spaces. Their positions and orientations are fixed during a viewer's SAR experience.

The world coordinate system $[x, y, z]$ by the rectangular coordinate system is given in the practice environment. The coordinates x and y define the horizontal plane, and the coordinate z defines the vertical direction. A viewer usually moves around the real object. Therefore, the real object is put on the origin O of the world coordinate system, and the polar coordinate system $[r, \theta, \varphi]$, which is defined by

$$x = r \cos\theta \sin\varphi, \quad (1)$$

$$y = r \sin\theta \sin\varphi, \quad (2)$$

$$z = r \cos\varphi, \quad (3)$$

is used for the viewer's position.

3.3 Generation of viewer composite image

A viewer composite image is generated for an arbitrary viewer's position by Step 1, which consists of the following three sub-steps.

3.3.1 Viewer virtual object image

A viewer virtual object image is generated in Step 1-1. In the virtual space, a 3D geometric model of a virtual object is made. It is scaled and positioned so as to fit inside the real object. Then, it is rendered from the viewer's position to generate a viewer virtual object image.

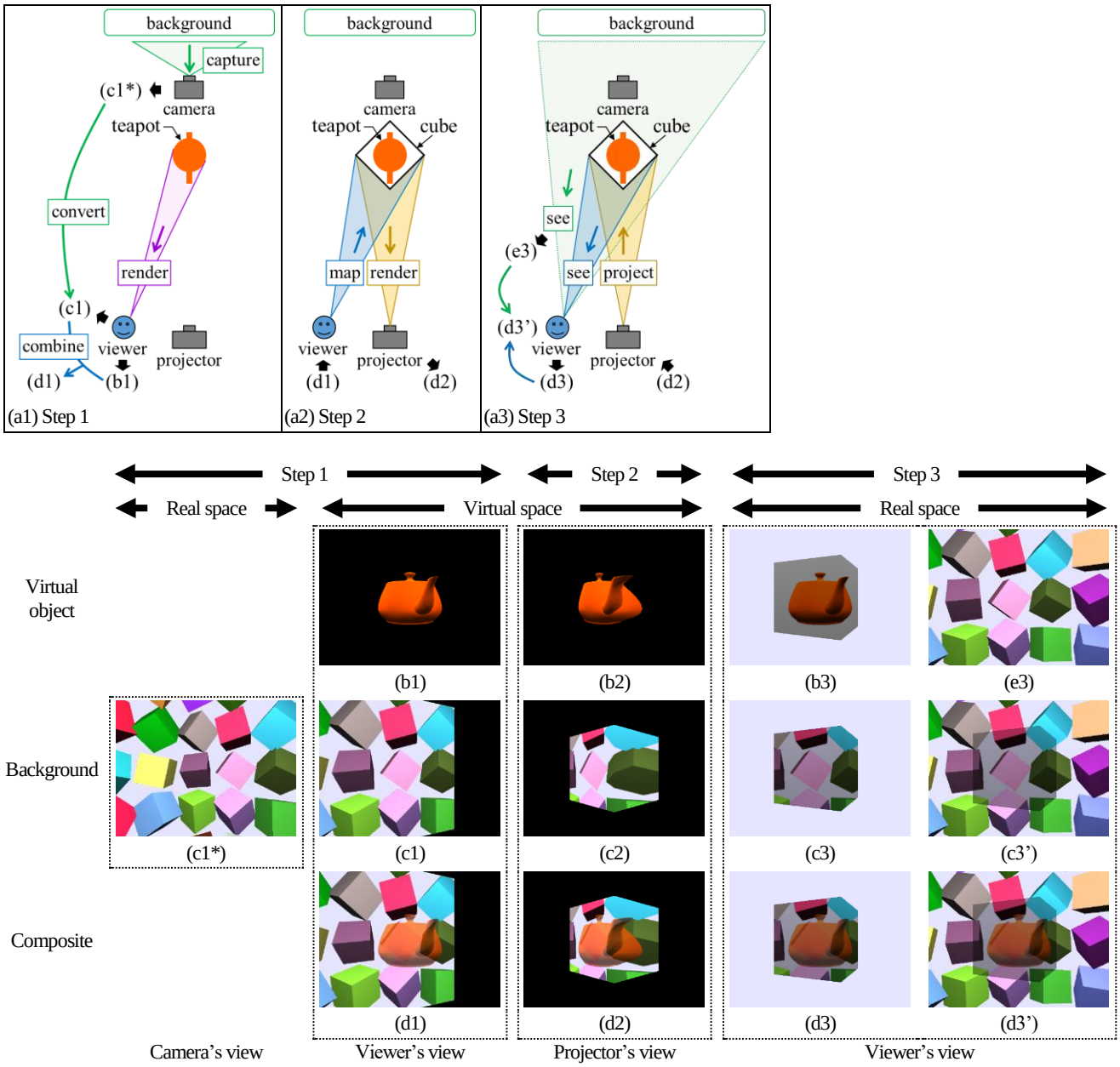

Figure 2. Our algorithm.

3.3.2 Viewer background image

A viewer background image is generated in Step 1-2. It is obtained from a camera background image by using a background plane to approximate the actual shape of a background. Our method uses a homography matrix.

When an image on a plane P is seen from two cameras, the images of these cameras are directly related to each other by a 3×3 homography matrix. The 2D pixel coordinates $\boldsymbol{U}_1 = [u_1, v_1]^T$ of a camera C_1 and $\boldsymbol{U}_2 = [u_2, v_2]^T$ of a camera C_2 have the relationship

$$\lambda\tilde{\boldsymbol{U}}_1 = \lambda\begin{bmatrix}\boldsymbol{U}_1\\ 1\end{bmatrix} = H\begin{bmatrix}\boldsymbol{U}_2\\ 1\end{bmatrix} = H\tilde{\boldsymbol{U}}_2 \tag{4}$$

by a homography matrix

$$H = \begin{bmatrix}h_{11} & h_{12} & h_{13}\\ h_{21} & h_{22} & h_{23}\\ h_{31} & h_{32} & h_{33}\end{bmatrix}, \tag{5}$$

where $\lambda \neq 0$ is a constant. The symbol T means a transposed matrix.

With respect to the camera C_m, $m = 1, 2$, the intrinsic parameter matrix A_m and the extrinsic parameter matrix $[R_m \ \boldsymbol{t}_m]$ defined by the rotation matrix R_m and the translation vector $\boldsymbol{t}_m$ are given by

$$A_m = \begin{bmatrix}f_{xm} & s_m & c_{xm}\\ 0 & f_{ym} & c_{ym}\\ 0 & 0 & 1\end{bmatrix}, \tag{6}$$

$$R_m = \begin{bmatrix}r_{11}^m & r_{12}^m & r_{13}^m\\ r_{21}^m & r_{22}^m & r_{23}^m\\ r_{31}^m & r_{32}^m & r_{33}^m\end{bmatrix}, \tag{7}$$

$$\boldsymbol{t}_m = \begin{bmatrix} t_{xm} \\ t_{ym} \\ t_{zm} \end{bmatrix}. \quad (8)$$

In the matrix A_m, the parameters f_{xm} and f_{ym} are the focal lengths in terms of pixels, c_{xm} and c_{ym} are the coordinates of the principal point, and s_m is the skew coefficient. The 2D pixel coordinates $\boldsymbol{U}_m$ and the 3D world coordinates $\boldsymbol{X} = [x, y, z]^T$ have the relationship

$$\lambda_m \tilde{\boldsymbol{U}}_m = \lambda_m \begin{bmatrix} \boldsymbol{U}_m \\ 1 \end{bmatrix} = A_m [R_m \ \ \boldsymbol{t}_m] \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = A_m [R_m \ \ \boldsymbol{t}_m] \ \tilde{\boldsymbol{X}}, \quad (9)$$

where $\lambda_m \neq 0$ is a constant. Besides, the plane P is defined by

$$\boldsymbol{N}^T \boldsymbol{X} + d = n_x x + n_y y + n_z z + d = 0, \quad (10)$$

where $\boldsymbol{N} = [n_x, n_y, n_z]^T$, $|\boldsymbol{N}| = 1$, is a unit normal vector, and d is a constant. Then, the homography matrix H is represented by

$$H = A_1 \{ (\boldsymbol{N}^T R_2^{-1} \boldsymbol{t}_2 - d) R_1 + (\boldsymbol{t}_1 - R_1 R_2^{-1} \boldsymbol{t}_2) \boldsymbol{N}^T \} R_2^{-1} A_2^{-1}. \quad (11)$$

The derivation of Equation (11) is described in Appendix A.

If the angles of view α_{xm} and α_{ym}, and the numbers of pixels M_{xm} and M_{ym} of the camera C_m are known, then the parameters f_{xm}, f_{ym}, c_{xm}, and c_{ym} are obtained by

$$f_{xm} = M_{xm} / \{2 \tan(\alpha_{xm}/2)\}, \quad (12)$$

$$f_{ym} = M_{ym} / \{2 \tan(\alpha_{ym}/2)\}, \quad (13)$$

$$c_{xm} = M_{xm}/2, \quad (14)$$

$$c_{ym} = M_{ym}/2. \quad (15)$$

Besides, it is often assumed that

$$s_m = 0. \quad (16)$$

We consider the camera C_1 as a camera to capture a background, the camera C_2 as a viewer, and the plane P as a background plane to approximate the shape of the background. Then, a viewer background image is obtained from a camera background image by determining the matrix H in Equation (11) and using the correspondence between the coordinates $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$ in Equation (4). This 2D computation using the 3×3 matrix H is so efficient compared to the usual 3D computation using a 4×4 matrix by considering the two image planes and the plane P in the 3D space.

The matrix H in Equation (11) consists of the intrinsic and extrinsic matrices of the cameras C_m, $m = 1, 2$, and the parameters of the plane P. These are obtained as follows. The camera C_1 is a camera to capture a background. The matrix A_1 can be obtained by camera calibration; it can be also obtained by Equations (12) to (16) although it is influenced by the errors of α_{xm} and α_{ym} in the specification and the assumption by Equation (16). The matrix R_1 and the vector $\boldsymbol{t}_1$ are obtained from the position and viewing direction of the camera C_1. The camera C_2 is a viewer. The matrix A_2 can be given arbitrarily and Step 2 needs to use the same matrix. The matrix R_2 and the vector $\boldsymbol{t}_2$ are obtained from the viewer's position and viewing direction, which are obtained by tracking the viewer in real time. Finally, with respect to the plane P, the constant d is obtained from the vector $\boldsymbol{N}$, the world coordinates $\boldsymbol{X}_0$ of an arbitrary reference point Q_0, and the distance D_P from the point Q_0 to the plane P as follows:

$$d = -\boldsymbol{N}^T \boldsymbol{X}_0 - D_P, \quad (17)$$

where the vector $\boldsymbol{N}$ has the same direction as the direction from the point Q_0 to the plane P. By using the origin O with world coordinates $\boldsymbol{O} = [0, 0, 0]^T$ as the point Q_0, we obtain

$$d = -D_P. \quad (18)$$

The world coordinates of the vector $\boldsymbol{N}$ can be defined by using two coordinates θ_P and φ_P and setting $r = |\boldsymbol{N}| = 1$ in Equations (1) to (3) as follows:

$$n_x = \cos\theta_P \sin\varphi_P, \quad (19)$$

$$n_y = \sin\theta_P \sin\varphi_P, \quad (20)$$

$$n_z = \cos\varphi_P. \quad (21)$$

Among the above-described elements to define the matrix H, only the three parameters D_P, θ_P, and φ_P to define the plane P cannot be determined automatically. We call them *background parameters*. We adjust them manually to make the plane P approximate the shape of a background optimally such that a resulting viewer background image becomes as similar as possible to the actual background view seen from the viewer's position. However, it is not practically easy to obtain the optimal plane by using the above three parameters. We improve them into other parameters in Section 3.6.1.

3.3.3 Viewer composite image

In Step 1-3, a viewer composite image is obtained by overlaying the viewer virtual object image on the viewer background image. The virtual object can be made semitransparent by alpha blending so that the background can be seen through the virtual object.

3.4 Generation of projection image

In Step 2, a projection image is generated from the viewer composite image by using the two-pass algorithm [5][35][36]. First, the viewer composite image is used as a texture and projectively mapped on the surface of the geometric model of the real object from the viewer's position. Then, the textured surface is rendered from the projector's position to generate a projection image. The resulting projection image is "correctly distorted" so that the virtual object and the background in the original viewer composite image can be seen with their "correct appearances" by the viewer in Step 3. If a graphic library or tool for *projective texture mapping* is available, such as OpenGL, the above can be easily realized.

3.5 Projection of projection image

In Step 3, the projection image obtained in Step 2 is projected on the surface of the real object by the projector in the real space. The viewer sees the virtual object and the background correctly.

3.6 Adjustment of background parameters

The background parameters to define a background plane need to be adjusted as easily as possible to approximate a real background's shape optimally. We present a practical adjustment method as follows.

3.6.1 Practical background parameters

The three background parameters D_P, θ_P, and φ_P to define a background plane P are presented in Section 3.3.2. The parameter D_P is the distance from the origin O to the plane P. The parameters θ_P and φ_P define the unit normal vector $\boldsymbol{N}$ of the plane P. The real object is put on the origin O. Practically, the plane P needs to be adjusted to obtain an optimal viewer background image by changing the parameters and observing the projected image on the real object's surface. This adjustment is not done efficiently if the parameters D_P, θ_P, and φ_P are used directly. In the following, the "viewer" means a person to prepare a projection mapping event and adjust the plane P in advance. In Figure 3 (a), the viewer adjusts the plane P by seeing the real object and the real background toward the origin O from the viewer's eye position Q_v; the viewer's viewing direction is represented by the red long arrow. In the following, we mean "viewer's eye position" by "viewer's position". The planes P_1 and P_2 have the unit normal vectors $\boldsymbol{N}_1$ and $\boldsymbol{N}_2$ defined by the parameter sets $[\theta_{P1}, \varphi_{P1}]$ and $[\theta_{P2}, \varphi_{P2}]$. The two planes have the same distance D_P from the origin O in the directions of their normal vectors. However, the distances D_1 and D_2 in the viewer's viewing direction are different. That is, when the viewer changes θ_P and φ_P to adjust the normal vector, the distance in the viewing direction also changes against the viewer's intention. This causes a serious difficulty for the viewer to obtain an optimal plane P.

Therefore, we present another practical way, as shown in Figure 3 (b). First, the base normal vector $\boldsymbol{N}_0 = [n_{x0}, n_{y0}, n_{z0}]^T$, $|\boldsymbol{N}_0| = 1$, is defined; its direction is the same as the viewer's viewing direction. It is obtained from the coordinates $\boldsymbol{R}_v = [r_v, \theta_v, \varphi_v]^T$ of the viewer's position Q_v:

$$\theta_{P0} = \theta_v + 180°, \tag{22}$$

$$\varphi_{P0} = 180° - \varphi_v, \tag{23}$$

$$n_{x0} = \cos\theta_{P0} \sin\varphi_{P0}, \tag{24}$$

$$n_{y0} = \sin\theta_{P0} \sin\varphi_{P0}, \tag{25}$$

$$n_{z0} = \cos\varphi_{P0}. \tag{26}$$

Next, the base plane P_0 is defined such that it has the normal vector $\boldsymbol{N}_0$ and a distance L_P from the origin O. An arbitrary unit normal vector $\boldsymbol{N}$ defined by parameters θ_P, φ_P, and Equations (19) to (21) is obtained based on the vector $\boldsymbol{N}_0$ using difference parameters $\Delta\theta_P$ and $\Delta\varphi_P$ by

$$\theta_P = \theta_{P0} + \Delta\theta_P = \theta_v + 180° + \Delta\theta_P, \tag{27}$$

$$\varphi_P = \varphi_{P0} + \Delta\varphi_P = 180° - \varphi_v + \Delta\varphi_P. \tag{28}$$

Then, the viewer can obtain an objective plane P based on the plane P_0 by changing the three parameters L_P, $\Delta\theta_P$, and $\Delta\varphi_P$. The base point Q_P is the intersection of the plane P_0 and the viewing direction; it has the world coordinates $\boldsymbol{X}_P = L_P\boldsymbol{N}_0$. The point Q_P is on an arbitrary plane P defined by L_P, $\Delta\theta_P$, and $\Delta\varphi_P$. This means that the coordinates $\boldsymbol{X}_P$ satisfy Equation (10), which gives

$$d = -\boldsymbol{N}^T\boldsymbol{X}_p = -L_P\boldsymbol{N}^T\boldsymbol{N}_0. \tag{29}$$

Equations (18) and (29) give the relation between D_P and L_P:

$$D_P = L_P\boldsymbol{N}^T\boldsymbol{N}_0. \tag{30}$$

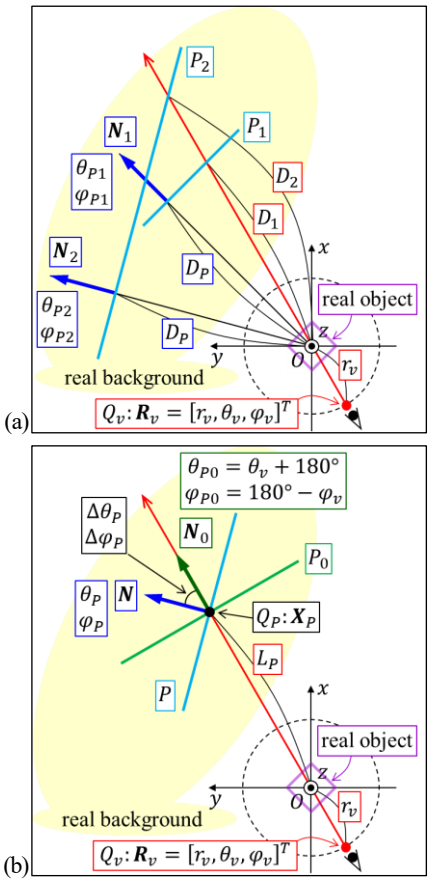

Figure 3. Background parameters.

The two parameter sets $[D_P, \theta_P, \varphi_P]$ and $[L_P, \Delta\theta_P, \Delta\varphi_P]$ are related by Equations (27), (28), and (30). The set $[L_P, \Delta\theta_P, \Delta\varphi_P]$ enables the viewer to adjust the plane P more easily than $[D_P, \theta_P, \varphi_P]$. Therefore, we use the parameters L_P, $\Delta\theta_P$, and $\Delta\varphi_P$ as background parameters practically. These parameters determine an arbitrary plane P for a viewer's position Q_v with coordinates θ_v and φ_v. The unit normal vector $\boldsymbol{N}$ is determined by Equations (19) to (21), (27), and (28). The constant d is determined by Equation (29), in which the vector $\boldsymbol{N}_0$ is determined by Equations (22) to (26). The plane P always has the base point Q_P, which is seen by the viewer at the center of his/her visual field. The point Q_P is away from the viewer by the distance $L_P + r_v$. For fixed values of $\Delta\theta_P$ and $\Delta\varphi_P$, the change of the distance L_P moves the plane P forward and backward in the viewing direction with the normal vector $\boldsymbol{N}$ unchanged. For a fixed value of L_P, the change of the angles $\Delta\theta_P$ and $\Delta\varphi_P$ causes the 3D rotation of the plane P around the point Q_P, which is fixed as a rotational center, by moving the normal vector $\boldsymbol{N}$ apart from the base normal vector $\boldsymbol{N}_0$. The above way enables the viewer to obtain an optimal plane P easily, intuitively, and efficiently.

3.6.2 Interpolation of background parameters

An optimal background view projected on the real object's surface by using a background plane P should be as similar as possible to the actual background view directly seen by a viewer. Such a plane P de-

pends on the viewer's position. Our method provides optimal background parameters L_P, $\Delta\theta_P$, and $\Delta\varphi_P$ according to the viewer's position. They are separated as follows:

$$L_P = L_P^G + L_P^L, \tag{31}$$
$$\Delta\theta_P = \Delta\theta_P^G + \Delta\theta_P^L, \tag{32}$$
$$\Delta\varphi_P = \Delta\varphi_P^G + \Delta\varphi_P^L. \tag{33}$$

The *global parameters* L_P^G, $\Delta\theta_P^G$, and $\Delta\varphi_P^G$ are common for every viewer's position to adjust the plane P roughly. The *local parameters* L_P^L, $\Delta\theta_P^L$, and $\Delta\varphi_P^L$ change according to the viewer's position to adjust the plane P in detail. For an arbitrary viewer's position with coordinates $\boldsymbol{R}_v = [r_v, \theta_v, \varphi_v]^T$, the following functions of $\boldsymbol{R}_v$ provide the local parameters:

$$L_P^L = F_{L_P^L}(\boldsymbol{R}_v) = F_{L_P^L}(r_v, \theta_v, \varphi_v), \tag{34}$$
$$\Delta\theta_P^L = F_{\Delta\theta_P^L}(\boldsymbol{R}_v) = F_{\Delta\theta_P^L}(r_v, \theta_v, \varphi_v), \tag{35}$$
$$\Delta\varphi_P^L = F_{\Delta\varphi_P^L}(\boldsymbol{R}_v) = F_{\Delta\varphi_P^L}(r_v, \theta_v, \varphi_v). \tag{36}$$

These functions are defined in the ranges

$$r_{min} \le r_v \le r_{max}, \tag{37}$$
$$\theta_{min} \le \theta_v < \theta_{max},\ \theta_{min} = 0,\ \theta_{max} = 360, \tag{38}$$
$$\varphi_{min} \le \varphi_v \le \varphi_{max}, \tag{39}$$

by using grid points $G_{[i,j,k]}$ with coordinates $[r_i, \theta_j, \varphi_k]$, $0 \le i < N_r$, $0 \le j < N_\theta$, $0 \le k < N_\varphi$, given by

$$r_i = \begin{cases} r_{min}(= r_{max}) & (N_r = 1), \\ r_{min} + \Delta r \cdot i, \quad \Delta r = \dfrac{r_{max} - r_{min}}{N_r - 1} & (N_r \ge 2), \end{cases} \tag{40}$$

$$\theta_j = \Delta\theta \cdot j, \qquad \Delta\theta = 360/N_\theta, \tag{41}$$

$$\varphi_k = \begin{cases} \varphi_{min}(= \varphi_{max}) & (N_\varphi = 1), \\ \varphi_{min} + \Delta\varphi \cdot k, \quad \Delta\varphi = \dfrac{\varphi_{max} - \varphi_{min}}{N_\varphi - 1} & (N_\varphi \ge 2). \end{cases} \tag{42}$$

Each grid point is given local parameters $L_P^L[i,j,k]$, $\Delta\theta_P^L[i,j,k]$, and $\Delta\varphi_P^L[i,j,k]$, which are interpolated linearly to define the functions. In order to obtain an optimal plane P for an arbitrary viewer's position, optimal global and local parameters L_P^G, $\Delta\theta_P^G$, $\Delta\varphi_P^G$, $L_P^L[i,j,k]$, $\Delta\theta_P^L[i,j,k]$, and $\Delta\varphi_P^L[i,j,k]$ need to be given in advance.

Our current system provides an interface to give the global and local parameters by trial and error manually. In the real space, the interface enables us to stand at a position near each grid point, check a background view projected on the real object's surface, and adjust the background plane by changing the parameters interactively to obtain an optimal background view in real time. It is an important future work to develop an efficient way to determine the optimal parameters automatically or semi-automatically.

3.7 Practice

The following are necessary for the practice of our method.

3.7.1 Tracking of viewer

We use a sensor to track a moving viewer in the real space in real time to obtain his/her position, which is used in Steps 1 and 2 in the virtual space. The tracking sensor is fixed near the real object and directed toward the area in which the viewer can move. Consequently, the tracking sensor and the camera to capture a real background are fixed near the real object and directed toward the sides opposite to each other.

3.7.2 Generation of viewer virtual object image

Our system has two options to obtain a viewer virtual object image in Step 1-1. The first option renders the 3D geometric model of a virtual object in real time during a viewer's SAR experience. This option can generate an image for an arbitrary position of a moving viewer with less memory than the second one. Besides, a desired rendering and shading algorithm can be used. However, the algorithm needs to be implemented so as to work efficiently in real time in our system, and the quality of the image is restricted by the real-time processing time. The second option renders the model in advance for predetermined $M_r \times M_\theta \times M_\varphi$ discrete viewer's positions and saves the rendered images to a storage device such as a hard disk. In a viewer's experience, all the saved images are stored on a memory, and the image for the discrete position nearest to the actual position of a moving viewer is used. This option can use high-quality images generated by time-consuming advanced rendering and shading algorithms and graphic tools. However, it needs large storage/memory; in particular, an amination of a virtual object needs huge storage/memory to treat all frame images for all discrete positions.

4. Experiment

We made some experiments in virtual and real environments. The demo videos of the experiments can be accessed on the journal website. Besides, the additional experiments and demo videos can be accessed on the following website: http://www-cg.cis.iwate-u.ac.jp/~fujimoto/demo/projmap_3d_back-ASv20n1/projmap_3d_back.html

4.1 Experiment in virtual environment

We firstly made an experiment in a virtual environment constructed by OpenGL. It is shown in Figure 4. The top view (a) shows the horizontal xy plane, and the side view (b) shows the xz plane. A cube is a real object, which is fixed at the origin O; it is denoted by the violet square. A viewer sees it at a position Q_v with coordinates $\boldsymbol{R}_v = [r_v, \theta_v, \varphi_v]^T$. The three small round red dots denote typical viewer's positions and the long red arrows denote their viewing directions; in each of (a) and (b), the middle dot denotes the viewer's initial position Q_v^{ini} with $\boldsymbol{R}_v^{ini} = [r_v^{ini}, \theta_v^{ini}, \varphi_v^{ini}]^T$. A projector is fixed at a position Q_{pr} with $\boldsymbol{R}_{pr} = [r_{pr}, \theta_{pr}, \varphi_{pr}]^T$; it is directed toward the cube. A camera is fixed at a position Q_c with $\boldsymbol{R}_c = [r_c, \theta_c, \varphi_c]^T$; it is directed toward the side opposite to the cube. The green and blue dots denote the positions Q_{pr} and Q_c, and the green and blue long arrows denote their viewing directions. The viewer, the projector, and the camera are given the values in Table 1. The height of the floor is $z = -1.2$. In this experiment, the viewer's coordinate φ_v moves from 80 to 100 degrees; the coordinate z relative to the floor's height moves

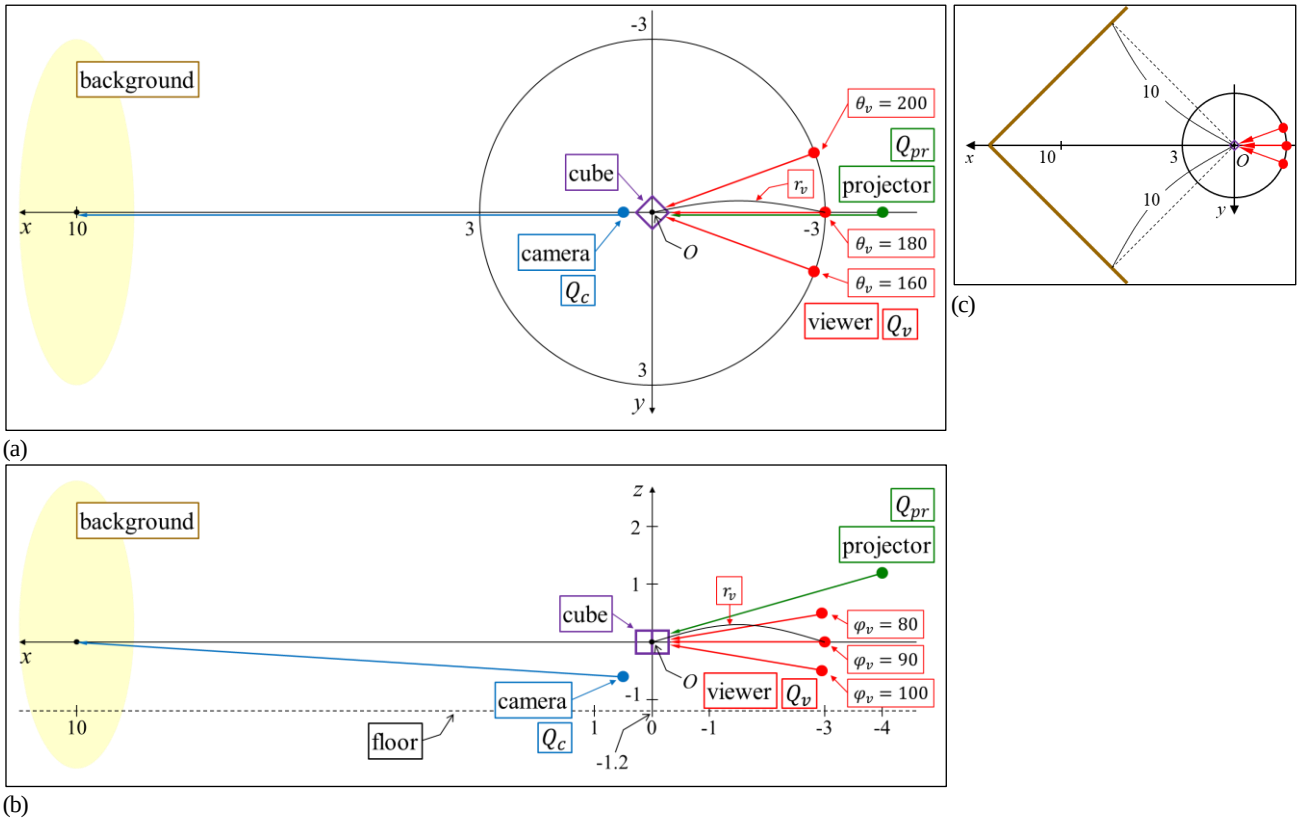

Figure 4. Virtual environment.

Table 1. Values for experiment in virtual environment.

	Position $[r, \theta, \varphi]$ (r: meters) (θ, φ: degrees)	Position $[x, y, z]$ (meters)	Gaze point $[x, y, z]$ (meters)	Angles of view $[\alpha_x, \alpha_y]$ (degrees)	Numbers of pixels $[M_x, M_y]$
Viewer: Q_v^{ini}	$3.0, 180.0, 90.0$	$-3.0, 0.0, 0.0$	$0.0, 0.0, 0.0$	$20.0, 15.0$	$640, 480$
Projector: Q_{pr}	$4.2, 180.0, 74.0$	$-4.0, 0.0, 1.2$	$0.0, 0.0, 0.0$	$38.0, 23.0$	$640, 400$
Camera: Q_c	$0.8, 0.0, 140.0$	$0.5, 0.0, -0.6$	$10.0, 0.0, 0.0$	$57.0, 43.0$	$640, 480$

roughly from 1.7 to 0.7 meter for $r_v = 3.0$ meters. This range of z simulates the height of an actual viewer's eyes. The size of the cube is 0.4^3 meters. The cube's top and bottom faces are parallel to the floor, while its four vertical edges are put toward $\pm x$ and $\pm y$ directions. The angles of view $[\alpha_x, \alpha_y]$ and the numbers of pixels $[M_x, M_y]$ of the projector and the camera are the same as those of our actual projector and camera used in the experiments in Section 4.2, while those of the viewer were selected to appropriately evaluate the performance of our method.

The result of the experiment is shown in Figure 5. The images are "augmented" views seen from twenty-five viewer's positions given by $\theta_v = 160, 170, 180, 190, 200$ and $\varphi_v = 80, 85, 90, 95, 100$ at a constant distance $r_v = r_v^{ini} = 3.0$. An orange teapot is a virtual object. The colored boxes simulate objects in a real background. The boxes are arranged close to two planes vertical to the xy plane; the brown lines in Figure 4 (c) represent the planes. The two planes intersect perpendicularly, and they are 10 meters away from the origin O. The boxes are rotated randomly. Their centers are arranged regularly with random displacements within ± 10 % of the interval between neighboring boxes horizontally and vertically on the planes and within ± 0.1 meter perpendicularly to the planes. The size of each box is 0.55^3 meters. We call the planes "box planes". The distribution of the boxes close to the two box planes needs to be approximated by a single background plane P. Each image in Figure 5 has two areas. The dark central area is the cube's surface onto which the teapot and the background are projected by the projector; the cube's top face is bright and its bottom face is black due to the position of the projector. The remaining area is the background directly observed. Our method aims to remove the presence of the cube by projecting the background on its surface. Thus, the background plane P should be adjusted such that the projected background matches the directly-observed background along the boundary of the cube's contour as seamlessly as possible. The background parameter interpolation provides the "boundary match" for an arbitrary viewer's position.

Table 2. Global background parameters for Figure 5.

Figure 5	L_P^G	$\Delta\theta_P^G$	$\Delta\varphi_P^G$
(a)	12.8	0.0	0.0
(b)	10.0	0.0	0.0

Table 3. Definition of grid points for Figure 5 (b).

	Minimum coordinate	Maximum coordinate	Number of points	Interval
r_i	$r_{min} = 2.5$	$r_{max} = 3.5$	$N_r = 3$	$\Delta r = 0.5$
θ_j	$\theta_{min} = 0$	$\theta_{max} = 360$	$N_\theta = 18$	$\Delta\theta = 20$
φ_k	$\varphi_{min} = 80$	$\varphi_{max} = 100$	$N_\varphi = 3$	$\Delta\varphi = 10$

Table 4. Local background parameters given to grid points for Figure 5 (b). Other grid points are given $[L_P^L, \Delta\theta_P^L, \Delta\varphi_P^L] = [0.0, 0.0, 0.0]$.

$r_i = 3.0$									
θ_j	160			180			200		
φ_k	L_P^L	$\Delta\theta_P^L$	$\Delta\varphi_P^L$	L_P^L	$\Delta\theta_P^L$	$\Delta\varphi_P^L$	L_P^L	$\Delta\theta_P^L$	$\Delta\varphi_P^L$
80	1.203993	−25.0	−10.0	2.997460	0.0	−10.0	1.203993	25.0	−10.0
90	1.033779	−25.0	0.0	2.8	0.0	0.0	1.033779	25.0	0.0
100	1.203993	−25.0	10.0	2.997460	0.0	10.0	1.203993	25.0	10.0

The images in Figure 5 (a) were obtained when the interpolation was not used. Local parameters were not used, and the global parameters in Table 2 were carefully selected so as to provide the best boundary match along the cube's contour for the viewer's initial position Q_v^{ini}. The resulting plane P was parallel to the yz plane and 12.8 meters away from the origin O. The central image for $\boldsymbol{R}_v^{ini} = [3.0, 180, 90]$ has an appropriate boundary match although there are some noticeable "gaps" between the projected background and the directly-observed background; the gaps were caused by the approximation of the two box planes by the single plane P as well as the approximation of the 3D shapes of the boxes by the plane P. However, the other images have worse boundary matches with serious gaps due to the same parameters selected only for Q_v^{ini}.

The images in Figure 5 (b) were obtained when the interpolation was used. The grid points $G_{[i,j,k]}$ were given by Table 3, and the global and local parameters in Tables 2 and 4 were selected so as to provide the best boundary matches for the viewer at the respective grid points. The grid point with $\boldsymbol{R}_v^{ini}$ was given the same parameters in total as those given in the "without-interpolation" case. Consequently, the central images in (a) and (b) are the same by the same plane P. The two grid points with $[r_i, \theta_j, \varphi_k] = [3.0, 180, 80]$ and $[3.0, 180, 100]$ were given the parameters to make the same plane P remain. The three grid points with $\theta_j = 160$ in Table 4 were given the parameters to make the plane P coincide with the right box plane in $y \leq 0$ seen from the origin O; a viewer at $\theta_v = 160$ sees mainly the boxes close to the right plane. In the same way, the parameters given to the three grid points with $\theta_j = 200$ make the plane P coincide with the left box plane in $y \geq 0$. The nine images with black frames were obtained directly from the parameters given to the grid points. These images have appropriate boundary matches although there are some gaps. The three images for $\theta_v = 180$ have more noticeable gaps than the others have. The projected background in each of the three images has the boxes, some of which are close to the right box plane and others of which are close to the left one. The noticeable gaps were caused by the imperfect approximation for the two box planes by the single plane P. The projected background in each of the remaining six images for $\theta_v = 160$ and 200 has the boxes, all of which are close to only one box plane. In this case, the single plane P worked quite well. On the other hand, the images other than the above nine images were obtained from interpolated parameters. These images also have appropriate boundary matches although there are also some gaps. The two images for $\theta_v = 180$ have some noticeable gaps and the four images for $\theta_v = 160$ and 200 have less gaps by the same reason as described above. The ten images for $\theta_v = 170$ and 190 have noticeable gaps although the boxes in their projected backgrounds have the same situations as those for $\theta_v = 160$ and 200. The parameters for $\theta_v = 170$ are given by interpolating the parameters for $\theta_v = \theta_j = 160$ and 180. This means that the plane P_{170} is the intermediate between the two planes P_{160} and P_{180}, where P_{θ_v} means a plane P for θ_v. Thus, the noticeable gaps for $\theta_v = 170$ were caused by the plane P_{170} that does not coincide with the box plane. The gaps for $\theta_v = 190$ have the same cause. Compared to Figure 5 (a), the boundary matches are greatly improved in Figure 5 (b) by the background parameter interpolation although there are some gaps, part of which are noticeable.

The above result shows that our method has the fundamental ability to appropriately eliminate undesired empty areas on a real object's surface by projecting a real background. A background plane can be adjusted easily, intuitively, and efficiently by practical background parameters, which are interpolated to make the boundary match between a projected background and a directly-observed background as seamless as possible for an arbitrary viewer's position. This ability achieves the effective view-dependent projection to make a viewer not feel the presence of a real object but feel a virtual object merging into the real world.

4.2 Experiment in real environment

We secondly made some experiments in a real environment. We used a white cube made from styrofoam as a real object; its size is 0.4^3 meters. We used an off-the-shelf projector, *RICOH IPSiO PJ WX5150*. We

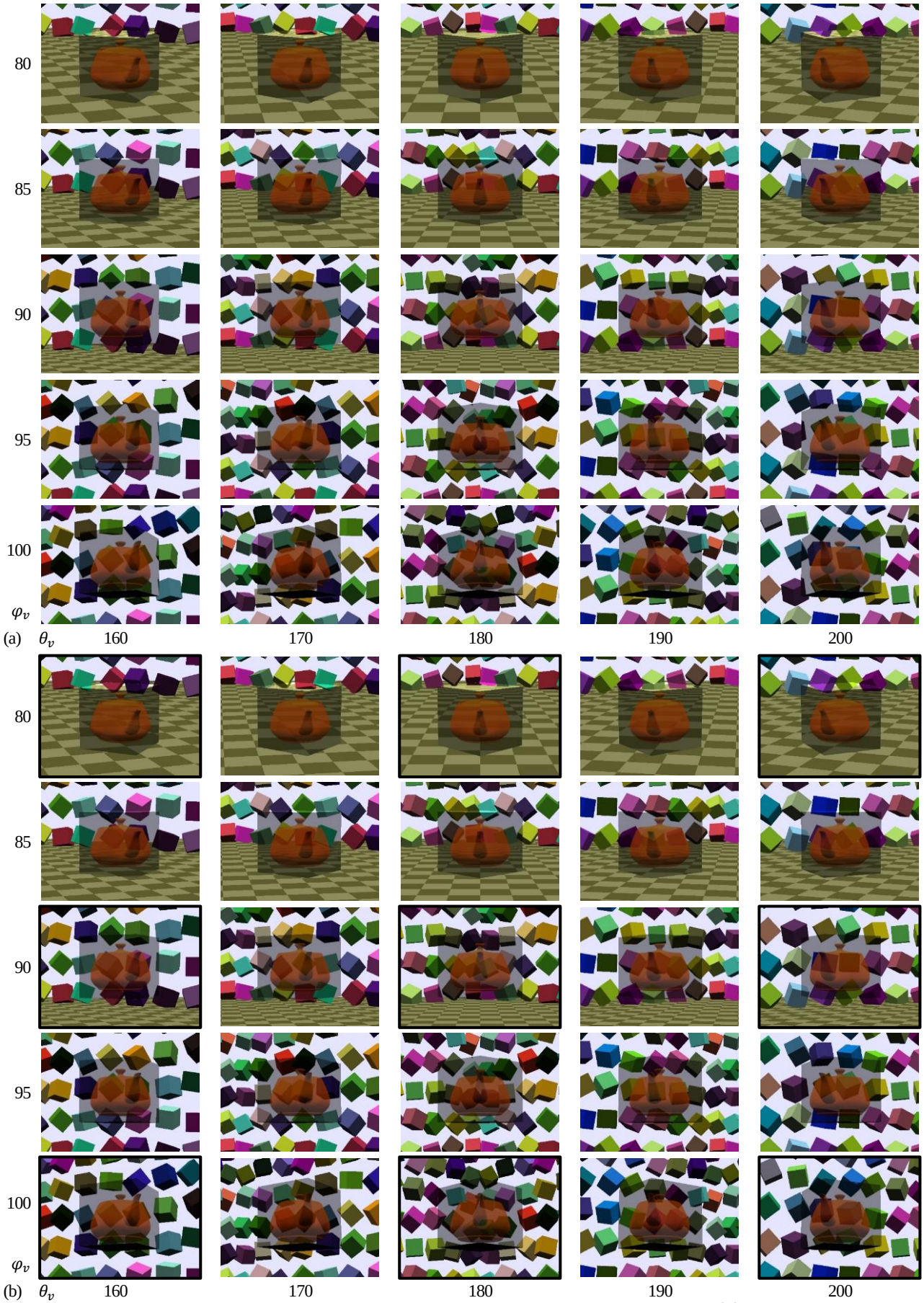

Figure 5. Augmented views for different viewer's positions in virtual environment. $r_v = r_v^{ini} = 3.0$.
(a) Background parameters are not interpolated. (b) Background parameters are interpolated.

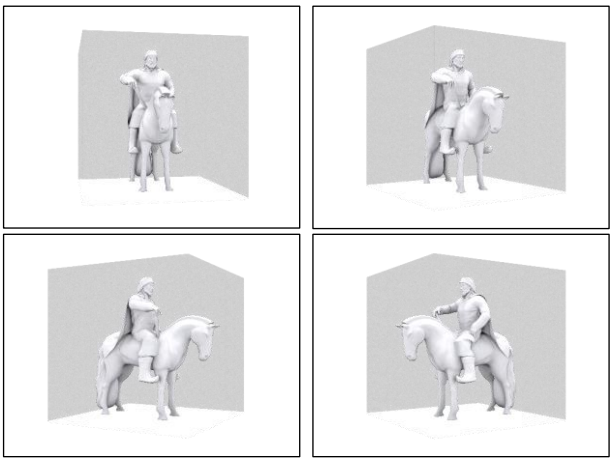

Figure 6. Virtual object.

used two Kinect sensors. One was used as a tracking sensor to track a viewer. Another was used as a camera to capture a real background. This Kinect sensor captured only color images of the background and did not capture depth images. We used it instead of a usual RGB camera because it has a wide angle of view to capture the background widely. We used old-type Kinect sensors, *Kinect for Windows*, because we needed to connect them to one personal computer; more than one new-type Kinect sensor cannot be connected to one personal computer. The two Kinect sensors were fixed near the cube and directed toward the sides opposite to each other. The angles of view and the numbers of pixels of the projector and the Kinect sensors are shown in Table 1. We used a personal computer with OS: Windows 10, Chipset: Intel(R) Z370 Express, CPU: Intel(R) Core i7-8700K (3.7-4.7GHz), GPU: NVIDIA(R) GeForce GTX 1060 6GB GDDR5, Mem: 32GB, SSD: 480GB, HDD: 2TB. We used the 3D geometric model of a virtual object shown in Figure 6. Viewer virtual object images were obtained by the second option in Section 3.7.2; the model was rendered in advance for $M_r \times M_\theta \times M_\varphi = 1 \times 360 \times 1$ discrete viewer's positions, that is, for every one degree around the model. We gave global background parameters and local parameters for grid points manually in advance. All the projection processes of our method were executed in real time during a viewer's SAR experience.

Figure 7 shows augmented views seen from some viewer's positions in the first experiment, in which the background was a planar wall with three figures near the cube. The views show that our method worked well. The correct appearance of the virtual object was always seen from different viewer's positions. The appropriate boundary match between the projected background and the directly-observed background was obtained by adjusting the background plane so as to fit the wall. The virtual object looked as if it actually existed in the real world. It was difficult to eliminate the presence of the cube perfectly due to the difference of the brightness of the projected background and the directly-observed background. As a result, the cube looked like a transparent box containing the virtual object inside. This visual effect was also yielded in the experiments below.

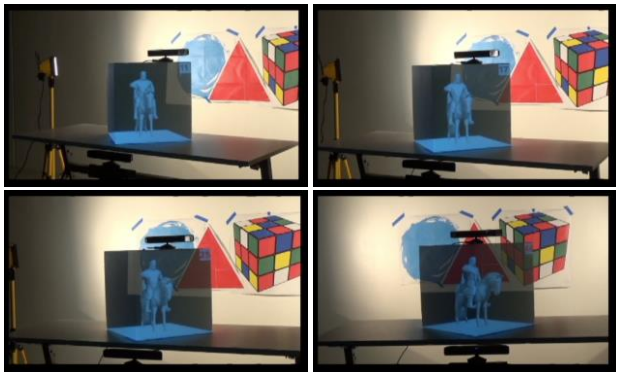

Figure 7. Augmented views in real environment with near planar background.

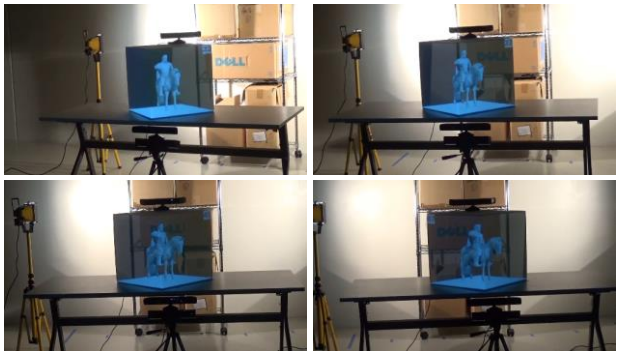

Figure 8. Augmented views in real environment with near non-planar background.

Figure 8 shows some augmented views obtained in the second experiment, in which a rack containing some boxes was put as a background object in front of the wall used in the first experiment. The correct appearance and the appropriate boundary match were also obtained. Some gaps between the projected background and the directly-observed background were caused by the difficulty of the approximation for the non-planar background by the background plane.

Figure 9 shows some augmented views obtained in the third experiment, in which the background was far from the cube and the background objects had complicated shapes. The black jaggy-shaped wall was about 10 meters and the white wall behind it was about 15 meters away from the cube. There were many background objects in front of the walls, such as desks and chairs. Figure 10 shows some projection images of the virtual object. Our method worked well to treat such a far complicated-shaped background in a large space. The correct appearance and the appropriate boundary match for the complicated background were obtained even if the viewer moved within a wide angle range around the cube. The boundary match over the wide range was achieved by the interpolation of background parameters to obtain an optimal background plane for each viewer's position.

5. Conclusion

In SAR, when a virtual object's shape is different from a real object's shape, the difference of the shapes causes undesired empty areas onto which the virtual object is not projected on the real object's surface. We proposed a view-dependent method to eliminate such empty areas by

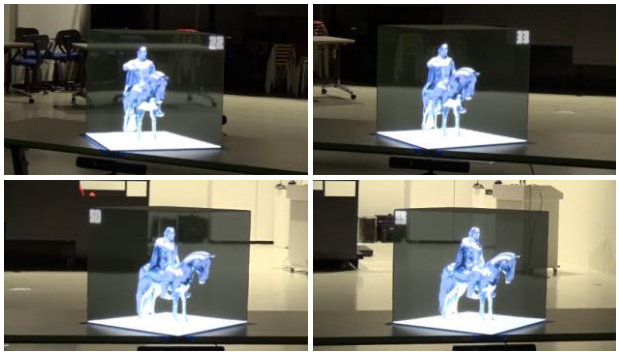

Figure 9. Augmented views in real environment with far complicated-shaped background.

projecting the real background behind the real object. In order to treat a far background in a large space, the real background's image is captured by an RGB camera and converted to an image for the viewer's position based on homography. The image conversion uses the approximation of the background's shape by a background plane, which is defined by practical background parameters interpolated on a 3D grid space. This adjusts the projected background so as to match the directly-observed background according to the viewer's position. Consequently, our method makes the viewer not feel the presence of the real object but feel the virtual object merging into the real world.

Currently, the presence of a real object is not eliminated perfectly due to the difference of the brightness of a projected background and a directly-observed background. This should be appropriately improved by photometric calibration. Our method has a serious problem; the view-dependent display is available to only a single viewer and not available to multiple viewers simultaneously. However, the simplicity of our method enables the integration with an existing method developed for multiple viewers. This is an important future work. Currently, a background plane is adjusted by giving background parameters manually. The development of an efficient way to determine optimal parameters automatically or semi-automatically is also a future work. In the current practice, a camera to capture a background and a tracking sensor to track a viewer are fixed near a real object and directed toward the sides opposite to each other. This configuration allows a viewer to see only one side of a virtual object. Actually, two Kinect sensors are used as the camera and the tracking sensor respectively. If each Kinect sensor works as both a camera and a tracking sensor, the viewer can see two opposite sides of the virtual object. By expanding this configuration, we are planning to develop a system to allow a viewer to enjoy a 360-degree view of a virtual object merging into its real background by using multiple Kinect sensors.

References

[1] Oliver Bimber and Ramesh Raskar, Spatial Augmented Reality: Merging Real and Virtual Worlds, A K Peters/CRC Press, 2005.

[2] Alessandra Meschini, Daniele Rossi, Enrica Petrucci, and Filippo Sicuranza, Expanded Cultural Heritage Representation: Digital Applications for Mixed-Reality Experiences, Handbook of Research on Emerging Technologies for Digital Preservation and Information Modeling, pp.256-287, IGI Global, 2017.

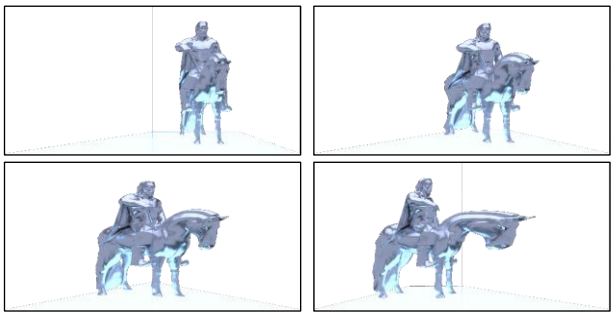

Figure 10. Projection images used in experiment of Figure 9.

[3] Microsoft Research, User Interfaces for Spatial Augmented Reality, 2016. https://www.youtube.com/watch?v=qZWpQ8Tuweo

[4] Ramesh Raskar, Spatial Augmented Reality: Inserting in the Real World, 2018. https://www.youtube.com/watch?v=4tRmDDM8lTs

[5] Ramesh Raskar, Greg Welch, Matt Cutts, Adam Lake, Lev Stesin, and Henry Fuchs, The Office of the Future: A Unified Approach to Image-Based Modeling and Spatially Immersive Displays, Proc. of ACM SIGGRAPH 1998, pp.179-188, 1998.

[6] Oliver Bimber, Daisuke Iwai, Gordon Wetzstein, and Anselm Grundhöfer, The Visual Computing of Projector-Camera Systems, Computer Graphics Forum, vol.27, 8, pp.2219-2245, 2008.

[7] Anselm Grundhöfer and Daisuke Iwai, Recent Advances in Projection Mapping Algorithms, Hardware and Applications, Computer Graphics Forum, vol.37, 2, pp.653-675, 2018.

[8] Ramesh Raskar, Greg Welch, Kok-lim Low, Deepak Bandyopadhyay, Shader Lamps: Animating Real Objects With Image-Based Illumination, Proc. of Eurographics Workshop on Rendering Techniques 2001, pp.89-102, 2001.

[9] Ramesh Raskar and Paul Beardsley, A Self-Correcting Projector, Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2001, pp.626-631, 2001.

[10] Samuel Audet and Masatoshi Okutomi, A User-Friendly Method to Geometrically Calibrate Projector-Camera Systems, Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) 2009, pp. 47-54, 2009.

[11] Tuotuo Li, Feng Hu, and Zheng Geng, Geometric Calibration of a Camera-Projector 3D Imaging System, Proc. of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry (VRCAI 2011), pp.187-194, 2011.

[12] Daniel Moreno and Gabriel Taubin, Simple, Accurate, and Robust Projector-Camera Calibration, Proc. of the 2012 Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization & Transmission, pp.464-471, 2012.

[13] I. Din, H. Anwar, I. Syed, H. Zafar, and L. Hasan, Projector Cali-

bration for Pattern Projection Systems, Journal of Applied Research and Technology, vol.12, 1, pp.80-86, 2014.

[14] Tyler Johnson and Henry Fuchs, Real-Time Projector Tracking on Complex Geometry Using Ordinary Imagery, Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) 2007, pp.1-8, 2007.

[15] Shuntaro Yamazaki, Masaaki Mochimaru, and Takeo Kanade, Simultaneous Self-Calibration of a Projector and a Camera Using Structured Light, Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) 2011, pp.60-67, 2011.

[16] Behzad Sajadi, Mahdi Abbaspour Tehrani, Mehdi Rahimzadeh, and Aditi Majumder, High-Resolution Lighting of 3D Reliefs Using a Network of Projectors and Cameras, Proc. of 2015 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), pp.1-4, 2015.

[17] Oliver Fleischmann and Reinhard Koch, Fast Projector-Camera Calibration for Interactive Projection Mapping, Proc. of 2016 23rd International Conference on Pattern Recognition (ICPR), pp.3798-3803, 2016.

[18] Simon Willi and Anselm Grundhöfer, Robust Geometric Self-Calibration of Generic Multi-Projector Camera Systems, Proc. of 2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pp.42-51, 2017.

[19] Jaewoon Lee, Yeonjin Kim, Myeong-Hyeon Heo, Dongho Kim, and Byeong-Seok Shin, Real-Time Projection-Based Augmented Reality System for Dynamic Objects in the Performing Arts, Symmetry, vol.7, 1, pp.182-192, 2015.

[20] Christian Siegl, Matteo Colaianni, Lucas Thies, Justus Thies, Michael Zollhöfer, Shahram Izadi, Marc Stamminger, and Frank Bauer, Real-Time Pixel Luminance Optimization for Dynamic Multi-Projector Mapping, ACM Transactions on Graphics, vol.34, 6, article no.237, pp.1-11, 2015.

[21] Yuichiro Fujimoto, Ross T. Smith, Takafumi Taketomi, Goshiro Yamamoto, Jun Miyazaki, Hirokazu Kato, and Bruce H. Thomas, Geometrically-Correct Projection-Based Texture Mapping onto a Deformable Object, IEEE Transactions on Visualization and Computer Graphics, vol.20, 4, pp.540-549, 2014.

[22] Gaku Narita, Yoshihiro Watanabe, and Masatoshi Ishikawa, Dynamic Projection Mapping onto Deforming Non-Rigid Surface Using Deformable Dot Cluster Marker, IEEE Transactions on Visualization and Computer Graphics, vol.23, 3, pp.1235-1248, 2017.

[23] Amit H. Bermano, Markus Billeter, Daisuke Iwai, and Anselm Grundhöfer, Makeup Lamps: Live Augmentation of Human Faces via Projection, Computer Graphics Forum, vol.36, 2, pp.311-323, 2017.

[24] Andreas Fender, David Lindlbauer, Philipp Herholz, Marc Alexa, and Jörg Müller, HeatSpace: Automatic Placement of Displays by Empirical Analysis of User Behavior, Proc. of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST 2017), pp.611-621, 2017.

[25] Andreas Fender, Philipp Herholz, Marc Alexa, and Jörg Müller, OptiSpace: Automated Placement of Interactive 3D Projection Mapping Content, Proc. of the 2018 CHI Conference on Human Factors in Computing Systems (CHI 2018), paper no.269, pp.1-11, 2018.

[26] Hrvoje Benko, Andrew D. Wilson, and Federico Zannier, Dyadic Projected Spatial Augmented Reality, Proc. of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST 2014), pp.645-655, 2014.
https://www.youtube.com/watch?v=D6yWWMwjkcg
https://www.youtube.com/watch?v=Df7fZAYVAIE

[27] Wojciech Matusik and Hanspeter Pfister, 3D TV: A Scalable System for Real-Time Acquisition, Transmission, and Autostereoscopic Display of Dynamic Scenes, Proc. of ACM SIGGRAPH 2004, pp.814-824, 2004.

[28] Tibor Balogh, Péter Tamás Kovács, and Zoltán Megyesi, HoloVizio 3D Display System, Proc. of 2007 3DTV Conference, pp.1-4, 2007.

[29] Andrew Jones, Ian McDowall, Hideshi Yamada, Mark Bolas, and Paul Debevec, Rendering for an Interactive 360° Light Field Display, Proc. of ACM SIGGRAPH 2007, article no.40, pp.1-10, 2007.

[30] Joel Jurik, Andrew Jones, Mark Bolas, and Paul Debevec, Prototyping a Light Field Display Involving Direct Observation of a Video Projector Array, Proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) 2011, pp.15-20, 2011.

[31] Toshiyuki Amano and Kensuke Minami, Structural Color Display on Retro-Reflective Objects, Proc. of the 25th International Conference on Artificial Reality and Telexistence and 20th Eurographics Symposium on Virtual Environments (ICAT-EGVE), pp.37-44, 2015.

[32] Andrew Jones, Jonas Unger, Koki Nagano, Jay Busch, Xueming Yu, Hsuan-Yueh Peng, Oleg Alexander, Mark Bolas, and Paul Debevec, An Automultiscopic Projector Array for Interactive Digital Humans, Proc. of ACM SIGGRAPH 2015 Emerging Technologies, article no.6, 2015.

[33] Shunsuke Yoshida, fVisiOn: Interactive Glasses-Free Tabletop 3D Images Floated By Conical Screen and Modular Projector Arrays, Proc. of ACM SIGGRAPH Asia 2015 Emerging Technologies, article no.12, pp 1-3, 2015.

[34] Min Ki Park, Kyu Je Lim, Myoung Kook Seo, Soon Jong Jung, and Kwan H. Lee, Spatial Augmented Reality for Product Appearance Design Evaluation, Journal of Computational Design and Engineering, vol.2, 1, pp.38-46, 2015.

[35] Ramesh Raskar, Michael S. Brown, Ruigang Yang, Wei-Chao Chen, Greg Welch, Herman Towles, Brent Seales, and Henry Fuchs, Multi-Projector Displays Using Camera-Based Registration, Proc. of IEEE Visualization, pp.161-168, 1999.

[36] Ruigang Yang and Greg Welch, Automatic and Continuous Projector Display Surface Calibration Using Every-day Imagery, Proc. of the 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG 2001), pp.328-335, 2001.

Appendix

A. Derivation of homography matrix

The homography matrix H of Equation (11) is obtained as follows. In the following, for convenience, some equations used in Section 3.3.2 are written again.

Each of two cameras C_m, $m = 1, 2$, has an intrinsic parameter matrix A_m and an extrinsic parameter matrix $M_m = [R_m \ \boldsymbol{t}_m]$ defined by a rotation matrix R_m and a translation vector $\boldsymbol{t}_m$ as follows:

$$A_m = \begin{bmatrix} f_{xm} & s_m & c_{xm} \\ 0 & f_{ym} & c_{ym} \\ 0 & 0 & 1 \end{bmatrix}, \tag{a.1}$$

$$R_m = \begin{bmatrix} r_{11}^m & r_{12}^m & r_{13}^m \\ r_{21}^m & r_{22}^m & r_{23}^m \\ r_{31}^m & r_{32}^m & r_{33}^m \end{bmatrix}, \tag{a.2}$$

$$\boldsymbol{t}_m = \begin{bmatrix} t_{xm} \\ t_{ym} \\ t_{zm} \end{bmatrix}. \tag{a.3}$$

For a point Q in the 3D space, its 2D pixel coordinates $\boldsymbol{U}_m = [u_m, v_m]^T$ and 3D camera coordinates $\boldsymbol{X}_m = [x_m, y_m, z_m]^T$ of the camera C_m and its 3D world coordinates $\boldsymbol{X} = [x, y, z]^T$ have the relationships

$$\boldsymbol{X}_m = [R_m \ \boldsymbol{t}_m] \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = M_m \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = M_m \widetilde{\boldsymbol{X}}, \tag{a.4}$$

$$\lambda_m \widetilde{\boldsymbol{U}}_m = \lambda_m \begin{bmatrix} \boldsymbol{U}_m \\ 1 \end{bmatrix} = A_m \boldsymbol{X}_m, \tag{a.5}$$

where $\lambda_m \neq 0$ is a constant. The symbol T means a transposed matrix. These are unified as follows:

$$\lambda_m \widetilde{\boldsymbol{U}}_m = A_m M_m \widetilde{\boldsymbol{X}}. \tag{a.6}$$

Equations (a.4), (a.5), and (a.6) are given other representations using 4×4 matrices $\tilde{M}_m$ and $\tilde{A}_m$ as follows:

$$\widetilde{\boldsymbol{X}}_m = \begin{bmatrix} \boldsymbol{X}_m \\ 1 \end{bmatrix} = \begin{bmatrix} R_m & \boldsymbol{t}_m \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \begin{bmatrix} M_m \\ 0 \ 0 \ 0 \ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \tilde{M}_m \widetilde{\boldsymbol{X}}, \tag{a.7}$$

$$\widetilde{\boldsymbol{U}}_m^* = \begin{bmatrix} \lambda_m \widetilde{\boldsymbol{U}}_m \\ 1 \end{bmatrix} = \begin{bmatrix} \lambda_m \begin{bmatrix} \boldsymbol{U}_m \\ 1 \end{bmatrix} \\ 1 \end{bmatrix} = \begin{bmatrix} A_m & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_m \\ 1 \end{bmatrix} = \tilde{A}_m \widetilde{\boldsymbol{X}}_m, \tag{a.8}$$

$$\widetilde{\boldsymbol{U}}_m^* = \tilde{A}_m \tilde{M}_m \widetilde{\boldsymbol{X}}. \tag{a.9}$$

The following is obtained from Equations (a.7) and (a.8):

$$\tilde{A}_m \tilde{M}_m = \begin{bmatrix} A_m & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} R_m & \boldsymbol{t}_m \\ 0 \ 0 \ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_m R_m & A_m \boldsymbol{t}_m \\ 0 \ 0 \ 0 & 1 \end{bmatrix}. \tag{a.10}$$

The next equations are obtained from Equation (a.7):

$$\widetilde{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{X} \\ 1 \end{bmatrix} = \tilde{M}_m{}^{-1} \widetilde{\boldsymbol{X}}_m = \begin{bmatrix} R_m{}^{-1} & \boldsymbol{t'}_m \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{X}_m \\ 1 \end{bmatrix}, \tag{a.11}$$

$$\boldsymbol{t'}_m = -R_m{}^{-1} \boldsymbol{t}_m. \tag{a.12}$$

The next equation is obtained from Equation (a.8):

$$\widetilde{\boldsymbol{X}}_m = \begin{bmatrix} \boldsymbol{X}_m \\ 1 \end{bmatrix} = \tilde{A}_m{}^{-1} \widetilde{\boldsymbol{U}}_m^* = \begin{bmatrix} A_m{}^{-1} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} \lambda_m \begin{bmatrix} \boldsymbol{U}_m \\ 1 \end{bmatrix} \\ 1 \end{bmatrix}. \tag{a.13}$$

The next equation is obtained from Equation (a.9):

$$\widetilde{\boldsymbol{X}} = \tilde{M}_m{}^{-1} \tilde{A}_m{}^{-1} \widetilde{\boldsymbol{U}}_m^*. \tag{a.14}$$

The following is obtained from Equations (a.11) and (a.13):

$$\tilde{M}_m{}^{-1} \tilde{A}_m{}^{-1} = \begin{bmatrix} R_m{}^{-1} & \boldsymbol{t'}_m \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} A_m{}^{-1} & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 \ 0 \ 0 & 1 \end{bmatrix} = \begin{bmatrix} R_m{}^{-1} A_m{}^{-1} & \boldsymbol{t'}_m \\ 0 \ 0 \ 0 & 1 \end{bmatrix}. \tag{a.15}$$

Equation (a.9) gives the camera C_1

$$\widetilde{\boldsymbol{U}}_1^* = \tilde{A}_1 \tilde{M}_1 \widetilde{\boldsymbol{X}}. \tag{a.16}$$

Equation (a.14) gives the camera C_2

$$\widetilde{\boldsymbol{X}} = \tilde{M}_2{}^{-1} \tilde{A}_2{}^{-1} \widetilde{\boldsymbol{U}}_2^*. \tag{a.17}$$

Then, the next equation is obtained from Equations (a.16) and (a.17):

$$\widetilde{\boldsymbol{U}}_1^* = \tilde{A}_1 \tilde{M}_1 \tilde{M}_2{}^{-1} \tilde{A}_2{}^{-1} \widetilde{\boldsymbol{U}}_2^*. \tag{a.18}$$

The following is obtained from Equations (a.10) and (a.15):

$$\tilde{A}_1 \tilde{M}_1 \tilde{M}_2{}^{-1} \tilde{A}_2{}^{-1} = \begin{bmatrix} A_1 R_1 & A_1 \boldsymbol{t}_1 \\ 0 \ 0 \ 0 & 1 \end{bmatrix} \begin{bmatrix} R_2{}^{-1} A_2{}^{-1} & \boldsymbol{t'}_2 \\ 0 \ 0 \ 0 & 1 \end{bmatrix} = \begin{bmatrix} A_1 R_1 R_2{}^{-1} A_2{}^{-1} & A_1 R_1 \boldsymbol{t'}_2 + A_1 \boldsymbol{t}_1 \\ 0 \ 0 \ 0 & 1 \end{bmatrix}. \tag{a.19}$$

Then, the following is obtained from Equations (a.12), (a.18), and (a.19):

$$\begin{aligned} \lambda_1 \widetilde{\boldsymbol{U}}_1 &= [A_1 R_1 R_2{}^{-1} A_2{}^{-1} \quad A_1 R_1 \boldsymbol{t'}_2 + A_1 \boldsymbol{t}_1] \begin{bmatrix} \lambda_2 \widetilde{\boldsymbol{U}}_2 \\ 1 \end{bmatrix} \\ &= \lambda_2 A_1 R_1 R_2{}^{-1} A_2{}^{-1} \widetilde{\boldsymbol{U}}_2 + A_1 R_1 \boldsymbol{t'}_2 + A_1 \boldsymbol{t}_1 \\ &= \lambda_2 A_1 R_1 R_2{}^{-1} A_2{}^{-1} \widetilde{\boldsymbol{U}}_2 + A_1 (\boldsymbol{t}_1 - R_1 R_2{}^{-1} \boldsymbol{t}_2). \end{aligned} \tag{a.20}$$

The world coordinates $\boldsymbol{X}$ of a point Q on a plane P seen from the camera C_m satisfy

$$\boldsymbol{N}^T \boldsymbol{X} + d = n_x x + n_y y + n_z z + d = 0, \tag{a.21}$$

where $\boldsymbol{N} = [n_x, n_y, n_z]^T$, $|\boldsymbol{N}| = 1$, is a unit normal vector and d is a constant. From Equation (a.11), the world coordinates $\boldsymbol{X}$ and the camera coordinates $\boldsymbol{X}_m$ of the point Q have the relationship

$$\boldsymbol{X} = R_m{}^{-1} \boldsymbol{X}_m + \boldsymbol{t'}_m. \tag{a.22}$$

Then, from Equation (a.21), the camera coordinates $\boldsymbol{X}_m$ satisfy

$$\boldsymbol{N}^T (R_m{}^{-1} \boldsymbol{X}_m + \boldsymbol{t'}_m) + d = 0. \tag{a.23}$$

From Equation (a.12), Equation (a.23) is arranged into

$$\boldsymbol{N}^T R_m{}^{-1} \boldsymbol{X}_m = \boldsymbol{N}^T R_m{}^{-1} \boldsymbol{t}_m - d. \tag{a.24}$$

Then, the following is obtained from Equations (a.5) and (a.24):

$$\lambda_m \boldsymbol{N}^T R_m{}^{-1} A_m{}^{-1} \widetilde{\boldsymbol{U}}_m = \boldsymbol{N}^T R_m{}^{-1} \boldsymbol{t}_m - d. \tag{a.25}$$

From Equation (a.22), the origin O_m of the camera coordinates $\boldsymbol{X}_m$ has world coordinates

$$\boldsymbol{O}_m = R_m{}^{-1} [0 \ 0 \ 0]^T + \boldsymbol{t'}_m = \boldsymbol{t'}_m. \tag{a.26}$$

Then, from Equations (a.22) and (a.26), the viewing direction vector V_m from the origin O_m to the point Q has world coordinates

$$\boldsymbol{V}_m = \boldsymbol{X} - \boldsymbol{O}_m = {R_m}^{-1}\boldsymbol{X}_m. \quad \text{(a.27)}$$

If $\boldsymbol{N}^T\boldsymbol{V}_m = \boldsymbol{N}^T{R_m}^{-1}\boldsymbol{X}_m = 0$, the vector V_m is parallel to the plane P. In this case, the plane P cannot be seen from the camera C_m. Therefore, it is assumed that

$$\boldsymbol{N}^T{R_m}^{-1}\boldsymbol{X}_m \neq 0. \quad \text{(a.28)}$$

Thus, the following is obtained from Equations (a.24) and (a.28):

$$\boldsymbol{N}^T{R_m}^{-1}\boldsymbol{t}_m - d \neq 0. \quad \text{(a.29)}$$

Then, the following is obtained from Equation (a.25):

$$\frac{\lambda_m}{\boldsymbol{N}^T{R_m}^{-1}\boldsymbol{t}_m - d}\boldsymbol{N}^T{R_m}^{-1}{A_m}^{-1}\tilde{\boldsymbol{U}}_m = 1. \quad \text{(a.30)}$$

Equation (a.30) is given $m = 2$ and substituted in Equation (a.20) to obtain

$$\lambda_1\tilde{\boldsymbol{U}}_1 = \lambda_2 A_1 R_1 {R_2}^{-1}{A_2}^{-1}\tilde{\boldsymbol{U}}_2 + A_1\left(\boldsymbol{t}_1 - R_1{R_2}^{-1}\boldsymbol{t}_2\right)\frac{\lambda_2}{\boldsymbol{N}^T{R_2}^{-1}\boldsymbol{t}_2 - d}\boldsymbol{N}^T{R_2}^{-1}{A_2}^{-1}\tilde{\boldsymbol{U}}_2. \quad \text{(a.31)}$$

Equation (a.31) is arranged as follows:

$$\lambda_1\left(\boldsymbol{N}^T{R_2}^{-1}\boldsymbol{t}_2 - d\right)\tilde{\boldsymbol{U}}_1 = \lambda_2 A_1\left\{\left(\boldsymbol{N}^T{R_2}^{-1}\boldsymbol{t}_2 - d\right)R_1 + \left(\boldsymbol{t}_1 - R_1{R_2}^{-1}\boldsymbol{t}_2\right)\boldsymbol{N}^T\right\}{R_2}^{-1}{A_2}^{-1}\tilde{\boldsymbol{U}}_2. \quad \text{(a.32)}$$

Then, the following is obtained from Equation (a.32):

$$\lambda\tilde{\boldsymbol{U}}_1 = A_1\left\{\left(\boldsymbol{N}^T{R_2}^{-1}\boldsymbol{t}_2 - d\right)R_1 + \left(\boldsymbol{t}_1 - R_1{R_2}^{-1}\boldsymbol{t}_2\right)\boldsymbol{N}^T\right\}{R_2}^{-1}{A_2}^{-1}\tilde{\boldsymbol{U}}_2, \quad \text{(a.33)}$$

where

$$\lambda = \lambda_1\left(\boldsymbol{N}^T{R_2}^{-1}\boldsymbol{t}_2 - d\right)/\lambda_2 \neq 0 \quad \text{(a.34)}$$

is a new constant. Equation (a.33) is arranged as follows:

$$\lambda\tilde{\boldsymbol{U}}_1 = H\tilde{\boldsymbol{U}}_2, \quad \text{(a.35)}$$

$$H = A_1\{(\boldsymbol{N}^T R_2^{-1}\boldsymbol{t}_2 - d)R_1 + (\boldsymbol{t}_1 - R_1R_2^{-1}\boldsymbol{t}_2)\boldsymbol{N}^T\}R_2^{-1}A_2^{-1}. \quad \text{(a.36)}$$

Equations (a.35) and (a.36) are the same as Equations (4) and (11).

Khuslen Battulga received a Bachelor and a Master of Business Administration in Management Information Systems from Mongolian University of Science and Technology in 2011 and 2012, respectively. She worked at Millennium Challenge Corporation from 2012 to 2013. She was a research student from 2013 to 2014 and a Ph.D. student from 2014 to 2020 in Graduate School of Engineering, Iwate University. Her research interests include projection mapping, 3D modeling, animation, and graphic design.

Tadahiro Fujimoto is currently a professor in the Department of Computer and Information Sciences at Iwate University. His research interests include computer graphics and computer vision. He received a BE in electrical engineering, and an ME and a Ph.D. in computer science from Keio University in 1990, 1992, and 2000, respectively. He worked at Mitsubishi Research Institute from 1992 to 1995. He was a research associate in the Department of Computer and Information Sciences at Iwate University from 1999 to 2002, a lecturer from 2002 to 2005, and an associate professor from 2005 to 2016. He is a member of SAS Japan, IEICE Japan, IPS of Japan, IEEE and ACM.