A Method of Shading and Shadowing in Image-Based Rendering

Akihiro Katayama, Yukio Sakagawa, and Hideyuki Tamura
Mixed Reality Systems Laboratory Inc.
{katayama, sakagawa, tamura}@mr-system.co.jp
http://www.mr-system.co.jp/

Abstract

This paper presents a method to generate dynamic shading of image-based objects without geometric models. Since conventional rendering techniques cannot be used to render the shading of this type of objects, image processing operation, such as interpolation or combination of image data, is performed. However, the variation of the shading of objects is not linear. Therefore, complicated processing is required to generate the desired shading, and they may not be compatible with real-time systems. Then, to perform the variation of the shading of image-based objects in a real-time system, we adopted a very simple method. From a set of image-based data with several different light conditions we select the image-based data with the light condition that is the closest to that of the rendering time, and use it to reconstruct the image. However, as it is necessary to save a large number of image data in this method, the volume of data is huge. Then we developed a method to reduce the data volume while maintaining real-time rendering. In this paper, we present not only our method, but also its application in the real-time system CyberMirage.

1. Introduction

Image-based rendering (IBR) method such as Light field rendering [1] and Ray space method [2] made possible the generation of photo-realistic images of complex shaped objects without the use of geometric models. These techniques can be used in applications that require photo-realism in its images, like virtual museums or cyber-shopping. However, as image-based objects do not hold geometric shape data or information about the surface reflectance properties, they have functional limitations compared to conventional geometric data. Consequently, the key to IBR future prosperity is how close it can get to the functionalities given by the conventional rendering that uses geometric models.

We have already designed the CyberMirage system [3], that can generate images of virtual environments that are more photo-realistic than the images generated by previous systems. In this system, we have implemented interactions like walk-through of the virtual space and manipulation of the image-based objects, like rotating or translating them [4].

The next topic is the addition of shading to image-based objects. In the previous work [5] of shading of image-based objects that do not have geometric models, Wong et al. proposes the use of apparent Bidirectional Reflectance Distribution Function (BRDF) to be used with the Lumigraph [6]. However, generally it is difficult to obtain the apparent BRDF. In [7], Shashua shows that when the relative position of the observer and an object is immutable and if the surface of the object is Lambertian, it is possible to generate an image of the object illuminated by a light source in any position, performing a linear combination of three images of the object illuminated by a light source in different positions. However, depending on the light source position, self cast shadow can occur on the surface of the object. Besides, ambient light and specular reflection can exist. In this case, images can not be generated by linear combination of other images.

This paper describes the method used in the CyberMirage system to generate dynamic shading of image-based objects due to moving light sources. In order to keep CyberMirage system working in real time, the method that we adopted is very simple: 1) shoot several images of an object under several different light conditions; 2) during rendering time, select the image set with the nearest illumination condition to that of the virtual environment; 3) use the selected image set to render an image of the object.

The CyberMirage system is going to be described in Section 2. In Section 3, we describe our method of shading and shadow-casting of image-based objects. In Section 4, we conclude and give the future works.

2. Fundamental System

The CyberMirage system is a real-time virtual reality system that merges image-based data and geometric-based data to make possible the construction of photo-realistic virtual environments. Ray-based representation [2] is utilized as IBR technique. It can be considered as a simplified version of Adelson's 'plenoptic function [8]', where the dimensions were reduced to make its use easier. The ray description method used in ray-based representation is different from that of light field, but they are equivalent in their principles.

2.1. Ray-Based Representation

In this representation, a light ray passing through a reference plane is defined by the position (x, y) where the ray passes, and the direction (θ, ϕ) of the ray (see Fig 1). If we ignore the vertical parallax information, the degrees of freedom of the ray can be restricted to (x,θ). Let, $u = \tan\theta$ then the ray passing through the point in the real space is mapped to the straight line

$$X = x + u \cdot Z \qquad (1)$$

in the *x*-*u* space as Fig.2 shows.

An image captured by a camera can be considered as a register of the rays that passed through the lens focal point. In this way, the *x*-*u* space can be filled if several images are collected. Conversely, the image from a new point of view can be generated by reading the data from the *x*-*u* space on the straight line, in the *x*-*u* space, that correspond to the new point of view. Nevertheless, when only one reference plane is used the direction of the rays that can be described is limited. To solve this problem, multiple reference planes arranged radially about the Y axis can be used to describe rays in any direction.

2.2. CyberMirage System

In order to integrate the ray-based data and the geometric shape model, it is necessary to solve the problem of mutual occlusion of objects in the virtual

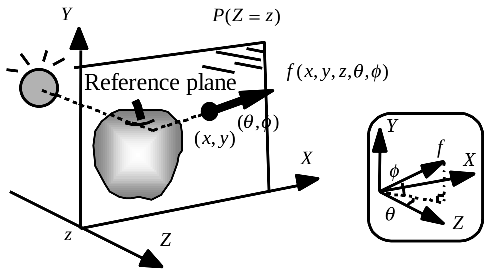
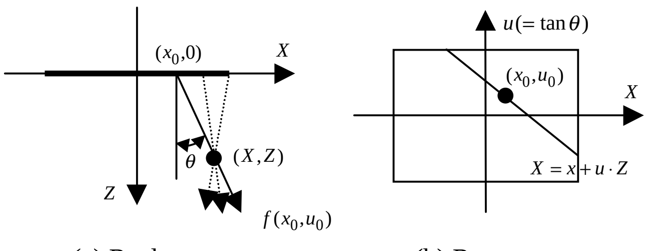
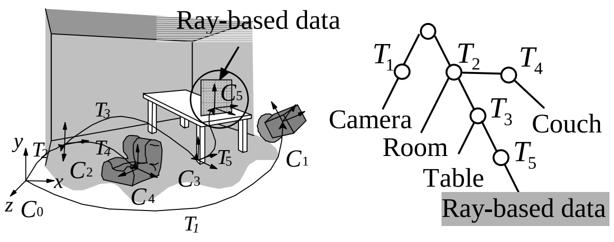


Fig. 1 Definition of ray space



(a) Real space (b) Ray space

Fig.2 Relationship between real space and ray space



(a) Virtual space (b) Tree structure

Fig.3 Virtual space representation

environment. The occlusion of geometric shape models can be correctly processed using Z-buffer algorithms. However, ray-based data do not hold the 3D shape data explicitly, and hidden surfaces elimination can not be straightly performed. Therefore, the depth of the object represented by ray-based data is approximated to the depth of the reference plane, and this depth value is used to perform elimination of hidden parts.

In CyberMirage, to put together geometric model data and ray-based data, ray-based data is represented as nodes in the tree structure of the VRML.

Fig. 3 shows a simple example of such representation. Consider the room, table and couch of (a). From the world coordinate C_0, the coordinate transformation T_2 is used to obtain C_2, the coordinate system of the room. The desk and the couch in that room are described in the coordinate systems C_3 and C_4, which are derived by the coordinate transformations T_3 and T_4, respectively, from the coordinate system C_2. Fig 3b shows the tree structure that represents these relations. To render the virtual space, a tree like the one of Fig 3 is traced starting from the left node, through all nodes by a depth-first search and the scene is drawn based on the information written in the nodes. In this process, the image of the geometric shape model data is generated by ordinary rendering process. In order to render from the ray-based data, on the other hand, it is necessary to have the position of the viewpoint and the direction of view line as seen from the reference plane, the field of view and the size of the display area. The view position and direction can be determined by successively applying the coordinate transformations that appear while tracing the tree. In the case of Fig. 3, let the position of the camera in coordinate system C_1 be P_1, then the position of the camera P_5 on the ray-based data coordinate C_5 is given by

$$P_5 = T_5^{-1} \cdot T_3^{-1} \cdot T_2^{-1} \cdot T_1 \cdot P_1 \qquad (2)$$

The view direction of the camera is similarly obtained. The field of view and the size of the display area are calculated based on the field of view of the virtual camera and the size of the drawing area that were configured for the rendering process.

3. Method of Shading and Shadow-Casting

Conventional rendering methods use information about the shape and reflectance properties of an object to generate its shading and cast shadow. However, for objects that do not have explicit shape information, as is the case of ray-based objects, there is no other alternative than applying image processing to generate the shaded images.

3.1. Shading of Ray-Based Objects

Our method is to have a collection of ray-based data sets with illuminations corresponding to the possible illumination conditions in the virtual space. In other words, only allow virtual illumination conditions that are close to those used to generate the ray-based data. Then, during reconstruction time, the data set which illumination condition is the most similar to the required one is used to generate the image.

[Generation of ray-based data set]

The generation of the ray-based data set with shading is done as follows:

1. the object is placed on a turn table, and the positions of the turn table and the illumination device (light source) are fixed, thus the relative position of the object and light source does not change.
2. the object is rotated, and images of the object are taken for some given angles. These images are used to generate the ray-based data set for light position p_i (p_i represents the position vector of the light source, i=1,...,N, for N light positions)
3. the relative position of the object and light source is altered, and the ray-based data set for light point m is generated (m is a serial numbering of the light positions). See Fig. 4.

[Data format]

As images of the object under several light positions are necessary, the problem of this method is the huge volume of data. Compression algorithms could be used to reduce the data volume, but it would consume processing power. In order to reduce the volume of data whilst maintaining rendering speed, we adopted a simple compression method. To simplify our explanation, lets consider a set of images of an object, shot with the same pose and camera parameters, but illuminated by a light source in different positions. The light source shall be far enough from the object so it can be considered to be a point light source.

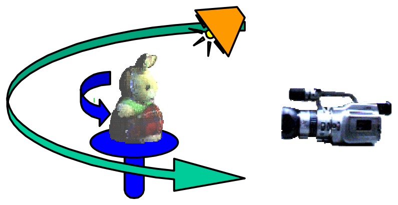


Fig. 4 For each different light position, the object is rotated to capture images to generate the ray-based data.

Let's consider that the surface of the subject has no specular reflection, and also that

$$I_c = I_{p,c} \cdot k_c \cdot L(p_i), \quad c = Y, U, V \qquad (3)$$

where I_c is the resulting intensity of the color component c, $I_{p,c}$ is the point light source's intensity; k_c is the material reflection coefficient, and $L(p_i)$ is a coefficient that says how much of the light source intensity is applied to the surface, according to their relative position p_i. This characteristic can be seen when considering Lambertian surfaces, but in our case it is not necessary to have the normal of the surface of the subject. Then, for a set of images of an object where only the position of the point light source changes, for each pixel of the image it is enough to save one base value (k_Y, k_U, k_V), and the value $L(p_i)$ for each light position p_i. What we basically do is to work in the YUV space, and use the relative luminance intensity of the pixel as the relative length of the YUV vector.

The generation of the base color image (Y_b, U_b, V_b), that corresponds to (k_Y, k_U, k_V), and the relative luminance intensity $L(p_i)$ is as follows:

1. pictures of the object under several different light positions are taken, and for each light position the luminance $Y(p_i)(x, y)$ is calculated, where (x,y) is the position of the pixel in the image.
2. picture of the object under an illumination condition as close as possible to ambient light is taken, and (Y_a, U_a, V_a) are calculated.
3. the values for the base color image are set as

$$Y_{\max}(x, y) = \max\{Y(p_i)(x, y)\} \qquad (4)$$

$$c(x, y) = Y_{\max}(x, y) / Y_a(x, y) \qquad (5)$$

$$H_b(x, y) = c(x, y) \cdot H_a(x, y), \quad H = Y, U, V \qquad (6)$$

4. for each light position p_i,

$$L(p_i)(x, y) = Y(p_i)(x, y) / Y_b(x, y) \qquad (7)$$

is saved.

However, when using real images, the values of (Y,U,V) may not behave as in equation (3). We try to minimize this problem including a test in step 3 above.

3. if $U_b(x, y)$ or $V_b(x, y)$ overflows, then adjust $c(x, y)$ as to avoid it, and recalculate (Y_b, U_b, V_b).

The obtained set of base color images and relative luminance are then used to generate the ray-based data of the object.

[Reconstruction of the image]

When rendering the scene, the position of the virtual light v_l is calculated from the scene graph and the relative luminance data set with nearest light position p_n to that of the virtual light is selected. The base color image (Y_b, U_b, V_b) of the object is generated from the ray-based data, and the relative luminance data $L(p_n)$ is applied as

$$H_r(v_l) = H_b \cdot L(p_n),\ H = Y,U,V \tag{8}$$

for each pixel and transformed to the RGB color space to be displayed. The intensity of the virtual light source can be used during the reconstruction of the image of the object as

$$H_r(v_l) = H_b \cdot I_H \cdot L(p_n),\ H = Y,U,V \tag{9}$$

where I_H is the relative intensity of the components (Y,U,V) of the light source. When S light sources illuminate the object, the resulting image can be considered as the linear combination of the participation of each light source individually:

$$H_r(v) = H_b \cdot \sum_{s=1}^{S} I_{s,H} \cdot L_s(p_n),\ H = Y,U,V \tag{10}$$

3.2. Shadow-casting

Conventional shadow algorithms consider that surfaces that can not be "seen" from the light source are in shadow. To generate the shadow of a ray-based object, the image of the object seen from the light source is generated, and its silhouette is extracted. The silhouette is then projected according the geometry of the surface where the shadow is supposed to be cast (Fig.5). As ray-based objects do not hold shape information, it is not possible to calculate shadows that are projected on these objects, but notice that the self-shadowing component is captured with the shading information.

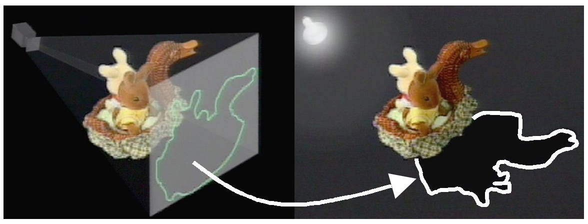


Fig. 5 Generation of cast shadow

3.3. Implementation

Our shading and shadow-casting method was implemented in the CyberMirage 99 system, a virtual mall for cyber shopping. In one of the rooms of the CyberMirage 99 virtual mall, the exhibition room, virtual objects are illuminated by virtual point lights that can be controlled by the user. The user can turn on/off the virtual light, change its position or its intensity.

The subject was a set of small dolls: a rabbit, a squirrel, and a baby squirrel playing in a seesaw. The main parts of the subject that we selected had velvet, cloth or fur as surface texture, and only a few parts presented surfaces with specular reflection. This set was considered as one single object. To generate the ray-based data set with shading, the object was put in a turn table, and the light source could move in a semi-circle of 180 degrees around the turn table. For a given light position, an image of the object was taken for every 4 degrees of rotation of the turn table. Then new images sets were taken changing the light position to every 3 degrees of arc around the object.

In this implementation, the shadow cast by the ray-based data object can not be generated as the silhouette of the object image seen from the light source, because the ray-based object does not have vertical disparity. To perform adequate shadow-casting, then, silhouettes of the object when seen from the light was prepared beforehand. Photographs of the object on a turn table were taken for several angles of rotation from the light position and the silhouettes were extracted and saved. When generating the synthetic image, the relative

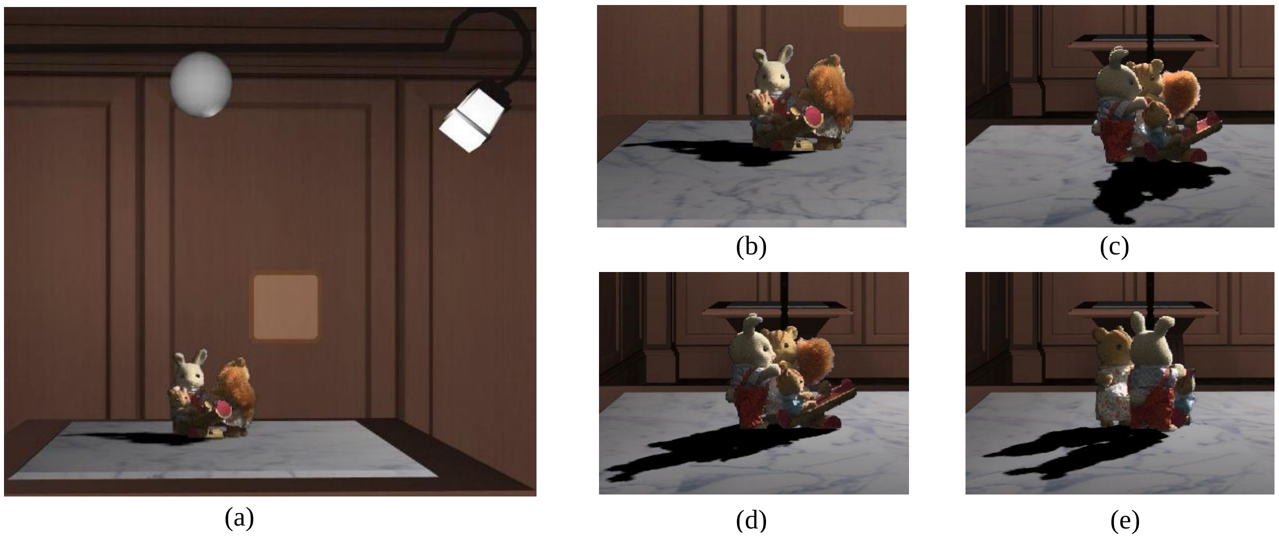
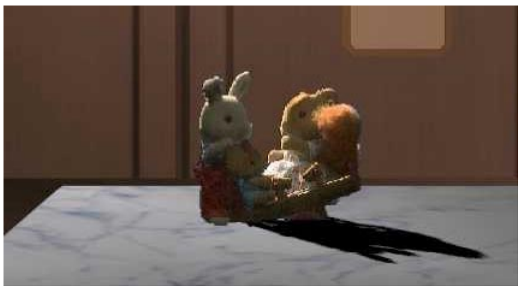
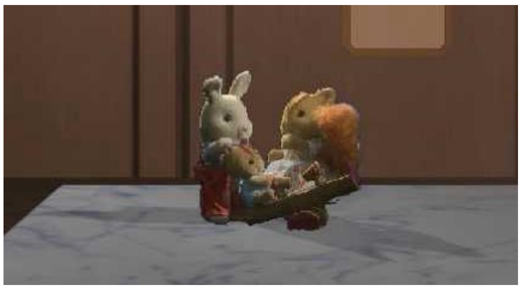

(a) (b) (c) (d) (e)

Fig. 6 Views of the objects under different lighting conditions



(a) Light coming from the back of the objects



(b) Light coming from the back and the side of the objects

Fig.7 Views of the objects illuminated by one and two lights

position of the object and light is used to select an approximate silhouette of the object, and it is used as the image to be cast as shadow. In this implementation we consider that the shadow is always cast on a plane surface, so a texture mapping board is provided for each ray-based object to project the shadow image in the virtual space. The size and position of the shadow board is calculated from the geometry of a bounding box that is associated with the ray-based data object and its relative position to the light source. This bounding box is also used to speed rendering of the ray-based data, and for its manipulation.

Fig. 6(a) shows an image of objects illuminated by a certain light source. It is noticed that the objects are appropriately shaded and cast an appropriate shadow. Fig. 6(b) is the image when an observer comes near to the objects and Fig. 6(c) is the image when an observer comes to another side of the objects. Fig. 6(d) and 6(e) are images when the light source is moved and when the objects are rotated, respectively. These figures show that the shading and cast shadow are appropriately adjusted. Fig. 7 shows images of the ray-based object illuminated by one and two light sources. It is noticed that two cast shadows and shading appear appropriately in Fig. 7(b).

The viewer program of the system is based in IRIS Performer, with a VRML 2.0 loader, and a module program to render the ray-based data. This system runs in an Onyx2 with 8 CPUs and 3 graphics pipelines. Each pipeline sends its image to a different screen of a 3-screen display. The screens, of 120 inches each, are arranged to give 180 degrees of view to the user. This system was also designed to generate pairs of stereoscopic images, that can be seen through shutter glasses.

4. Conclusion

In this paper, we described a method of adding shading to the image-based data, and showed the implementation. Here, we placed image-based objects in an environment based on geometric models, and we worked with the case where shading changes.

However, a situation where varying shading gives more effective results is in Mixed Reality Systems, which deals with both real and virtual environments, where this method can be applied to place image-based data in real environments. However, when working with the real world, it is necessary not only to infer the shading of the image-based data, but it is also necessary to estimate the illumination condition of the real world.

Furthermore, the present method is simple, but as it is necessary to shoot and save all the images with shading, the volume of data becomes huge. The next research topic is the development of a more sophisticated compression method that can be used in real-time processing, to reduce the volume of data.

References

[1] M. Levoy and P. Hanrahan: “Light field rendering,” *Proc. SIGGRAPH 96*, pp.31-42, 1996.

[2] T. Naemura, M. Kaneko, and H. Harashima: “Orthographic approach to representing 3-D images and interpolating light rays for 3-D image communication and virtual environment,” *Signal Process. Image Commun.*, vol.14, pp.21-37, 1998.

[3] S. Uchiyama, A. Katayama, A. Kumagai, H. Tamura, T. Naemura, M. Kaneko, and H. Harashima: “Collaborative CyberMirage: A shared cyberspace with mixed reality,” *Proc. VSMM’97*, pp.9-18, 1997.

[4] S. Uchiyama, H. Yamamoto, A. Katayama, and H. Tamura: “Presentation and interaction of virtual 3D objects without geometric model,” *Proc. HCI International’97*, vol.2, pp.869-872, 1997.

[5] T. Wong, P. Heng, S. Or, and W. Ng: “Image-based rendering with controllable illumination,” *Proc. 8th Eurographics Workshop on Rendering*, pp.13-22, 1997.

[6] S. J. Gortler, R. Grzeszczuk R. Szeliski, and M F. Cohen: “The Lumigraph,” *Proc. SIGGRAPH 96*, pp.43-54, 1996.

[7] A. Shashua: “Geometry and photometry in 3D visual recognition,” Ph. D thesis, Dept. Brain and Cognitive Science, MIT, 1992.

[8] E. H. Adelson and J. R. Bergen: “The plenoptic function and the elements of early vision,” in (Landy and Movshon, Eds.) *Computational Models of Visual Processing*, MIT Press, pp.3-20, 1991.