Chapter 8

Photometric Modeling for Mixed Reality

Katsushi Ikeuchi
Yoichi Sato
Ko Nishino
Imari Sato

The University of Tokyo, Japan

8.1 Introduction

The mixed reality technology is considered as one of the key technologies for enhancing a wide variety of applications ranging from manufacturing to entertainment. The mixed reality technology differs from the virtual reality technology in that users can feel immersed in a space which consists of not only computer-generated objects but also real objects. Thus seamless integration of the virtual and real worlds is essential for mixed reality in addition to reality of the virtual world.

Our efforts in the mixed reality research span two aspects: how to create models of virtual objects and how to integrate such virtual objects with real scenes. For model creation, we have developed two methods, the model-based rendering method and the eigen-texture method, both of which automatically create such rendering models by observing real objects. The model-based rendering method first analyzes input images of real objects, obtains reflectance parameters from this analysis, and then, using the determined reflectance parameters, generates the virtual image [1]. This method stores very compact information, namely surface shapes and reflectance parameters, and works well when an object surface follows a known reflectance model.

For other objects that do not follow simple reflectance models, we have developed the eigen-texture method [2]. This method samples appearances of a real object, pastes the images onto the surface of the 3D model of the object, and then compresses them on the 3D surface (not on the image plane). Later, the compressed data can be used for generating a virtual image of the object. The method does not assume any known reflectance model, nor does it require any detailed reflectance analysis, as was the case in model-based rendering. Using the 3D model of the object, the method can also generate shadows cast by the virtual object, something that the image-based rendering fails to do. Moreover, the method achieves a very compact storage of the object's appearances because the compression of the appearances is performed along pixels corresponding to the same physical points.

For integration of virtual object with real scenes, we have developed a method that renders virtual objects based on real illumination distribution [3]. First, a radiance distribution in the real scene is determined from two omni-directional images of the scene. Then the measured radiance distribution is used for rendering virtual objects superimposed onto the scene image. The proposed method has the ability to synthesize a convincing image even for a complex radiance distribution, a task at which other methods often fail.

8.2 Creating Models from Observation

Currently, most virtual reality systems utilize image-based rendering [4]–[6]. The image-based rendering samples a set of color images of a real object and stores them on the disk of a computer. A new image is then synthesized either by selecting an appropriate image from the stored set or by interpolating multiple images. Apple's QuickTime VR is one of the earlier successful image-based rendering methods. Image-based rendering does not assume any reflectance characteristics of objects nor does it require any detailed analysis of the reflectance characteristics of the objects; rather, the method needs only to take images of an object. The method can be applied to a wide variety of real objects. And because it is also quite simple and handy, image-based rendering is ideal for displaying an object as a stand-alone without any background for the virtual reality.

On the other hand, image-based methods have critical disadvantages on application to mixed reality. Few image-based rendering methods employ accurate 3D models of real objects. Thus, it is difficult to make cast shadows under real illuminations corresponding to the real background-image.

Unlike image-based rendering, model-based rendering assumes a reflection model of an object and determines reflectance parameters through detailed reflectance analysis. Later, the method uses those reflectance parameters to generate virtual images by considering illumination conditions of the real scene. Since the reflectance parameters are obtained at every surface point of the object, integration of synthesized images with the real background can be accomplished quite realistically, *i.e.*, the method can generate a realistic appearance of an object as well as of the shadows cast by that object onto the background.

8.2.1 Model-Based Rendering

We have developed a model-based rendering method for modeling object reflectance properties, as well as object shapes, by observing real objects [1]. First, an object surface shape is reconstructed by merging multiple range images of the object. By using the reconstructed object shape and a sequence of color images of the object, parameters of a reflection model are estimated in a robust manner. The key point of the proposed method is that, first, the diffuse and specular reflection components are separated from the color image sequence, and then, reflectance parameters of each reflection component are estimated separately. This approach enables estimation of reflectance properties of real objects whose surfaces show specularity as well as diffusely reflected lights. The recovered object shape and reflectance parameters are then used for synthesizing object images with realistic shading under arbitrary illumination conditions.

Shape modeling

The image acquisition system used in our experiments is illustrated in Figure 8.1. The object whose shape and reflectance parameters are to be recovered is mounted on the end of a robotic arm. Using the system, a sequence of range and color images of the object is obtained as the object is rotated at a fixed angle. A range image is obtained using a light-stripe range finder with a liquid crystal shutter and a color CCD video camera [7]. Using optical triangulation, 3D locations of points in the scene are computed at each image pixel. Each range-image pixel represents a 3D location of a corresponding point on an object surface. The same color video camera is used for acquiring color images. Therefore, the range image pixels and those of the color images directly correspond to one another. A single incandescent lamp is used as a point light source. In our experiments, the light source direction and the light source color are measured by calibration.

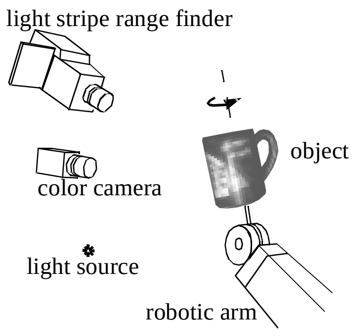

Figure 8.1 Image acquisition system.

A sequence of range images of the object is used to construct the object shape as a triangular mesh. For constructing object shapes as triangular mesh models from multiple range images, we used the volumetric method developed by Wheeler, Sato, and Ikeuchi [8]–[10]. In addition, surface orientation is necessary for estimating reflectance parameters. Polygonal orientation computed from a triangular surface mesh model can approximate real surface orientation only when the object surface is relatively smooth and does not have high curvature points. For instance, using the 3D points from the range images with a least square fitting, we compute surface orientation at regular grid points within each triangle.

Parameter estimation

Once the object shape has been obtained, we measure the reflectance parameters of the object surface using the obtained shape and the input color images. First, the two fundamental reflection components (*i.e.*, the diffuse and specular reflection components) are separated from the input color images. Then, the parameters for the two reflection components are estimated separately. Separation of the two reflection components enables us to obtain a reliable estimation of the specular reflection parameters. Also, the specular reflection component (*i.e.*, highlight) in the color images does not affect the estimated diffuse reflection parameters of the object surface.

The Torrance-Sparrow model [11] [12] is used for representing the diffuse and specular reflection components

$$L = K_d f_d(\theta) + K_s f_s(\psi, \alpha), \tag{8.1}$$

where$f_d(\theta) = \cos(\theta), f_s(\psi, \alpha) = e^{-\alpha^2/(2\sigma^2)}/\cos\psi$. θ is the angle between the surface orientation and the light source direction, ψ is the angle between the surface orientation and the viewing direction, α is the angle between the surface normal and the bisector of the light source direction and the viewing direction, K_d, K_s are constants for the diffuse and specular reflection components, and σ is the standard deviation of a facet slope.

We employ the separation algorithm, originally introduced for the case of a moving light source by Sato and Ikeuchi [13]–[15]. In order to handle the case of a moving object as opposed to a moving light source, we align color images from different poses onto a 3D-mesh model using known camera parameters. From this alignment, we can determine a sequence of observed color values at each point on the object surface during the object rotation.

Measurement at each point on the object surface provides an observation matrix, consisting of three sequences of intensity values (R, G, and B), where M is the number of measurements.

$$A = \begin{bmatrix} L_1^R & L_1^G & L_1^B \\ L_2^R & L_2^G & L_2^B \\ \vdots & \vdots & \vdots \\ L_M^R & L_M^G & L_M^B \end{bmatrix} \tag{8.2}$$

This $M \times 3$ observation matrix can be represented as a product of an $M \times 2$ intensity matrix, consisting of the intensity values of the diffuse and specular reflection component with respect to the illumination/viewing directions, and of a 2×3 color matrix, consisting of the diffuse and the specular color vectors.

$$\begin{bmatrix} f_1(\theta_1) & f_2(\psi_1, \alpha_1) \\ f_1(\theta_2) & f_2(\psi_2, \alpha_2) \\ \vdots & \vdots \\ f_1(\theta_M) & f_2(\psi_M, \alpha_M) \end{bmatrix} \begin{bmatrix} K_d^R & K_d^G & K_d^B \\ L_s^R & L_s^G & L_s^B \end{bmatrix} \tag{8.3}$$

From this observation, we can conclude that the rank of the observation matrix A is 2, and the observation matrix can be decomposed into an intensity matrix and a color matrix by using the singular value decomposition. By extracting the intensity matrix, we can obtain how intensity values of the diffuse and specular components vary depending on the illumination/viewing directions.

Using the angles of the illumination/viewing directions computed from the rotation angle, the diffuse reflection parameters are estimated by fitting the first term of the Torrance-Sparrow reflection model, a cosine function, to the separated diffuse reflection component, the first column of the intensity matrix. The diffuse reflection parameters are estimated at regular grid points within each triangle.

The specular reflection parameters are computed in the same way as the diffuse reflection parameters. However, since the specular reflection component is usually observed only from a limited range of viewing directions, we have to interpolate, over the entire surface, the specular reflection parameters obtained at a fixed number of points selected using several evaluation measures [1].

Image synthesis

Using the reconstructed object shape, the surface normals, the diffuse reflection parameters, the specular reflection parameters and the reflection model, we can synthesize color object images under arbitrary illumination/viewing conditions. Figure 8.2 shows three synthesized images as well as one frame of the input color image sequence. One of those synthesized images was generated using the same illuminating/viewing condition as the input color image. It can be seen that the synthesized image closely resembles the corresponding real image. In particular, highlights, which generally are a very important cue of surface material, appear on the side and the handle of the mug naturally in the synthesized images.

Summary

This section describes an algorithm for model-based rendering. The object surface shape is reconstructed by merging multiple range images of the object. By using the reconstructed object shape and multiple color images of the object, parameters of the Torrance-Sparrow reflection model are estimated. For estimating reflectance parameters of the object robustly, our method is based on separation of the diffuse and specular reflection components from a color image sequence. Using separated reflection components, reflection model parameters for each of the two components are estimated separately. Our experiments have demonstrated that our model-based rendering method can be effectively used for synthesizing realistic object images under arbitrary illumination and viewing conditions.

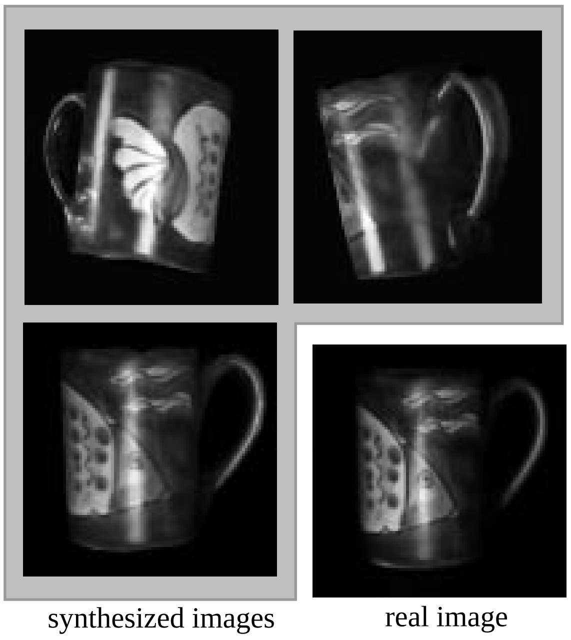

Figure 8.2 Input and synthesized images.

8.2.2 Eigen-Texture Method

The previous section describes a model-based rendering method. This method can generate realistic appearances of an object from reflectance parameters estimated from real objects. Unfortunately, however, some classes of object surfaces reveal complicated reflectance models, and the model-based rendering method cannot be applied to those classes of objects.

For those object surface classes, we propose a new rendering method, which we refer to as the eigen-texture method [2] [16] [17]. Figure 8.3 displays an overview of the proposed method. The eigen-texture method creates a 3D model of an object from a sequence of range images. The method aligns and projects color images of the object onto the 3D surface of the object model. Then, the method compresses those images in the coordinate system defined on the 3D-model surface. This compression is accomplished using the eigenspace method [18]. The synthesis process is achieved using the inverse transformation of the eigenspace method. Shadows cast by the object are generated by using the 3D shape of the object. A virtual image under a complicated illumination condition is generated by summation of component virtual images sampled under single illuminations thanks to the linearity of image brightness.

Appearance compression based on 3D shape model

The eigen-texture method samples a sequence of color and range images. Once these two sequences are input to the system, a 3D triangular mesh model of the object is

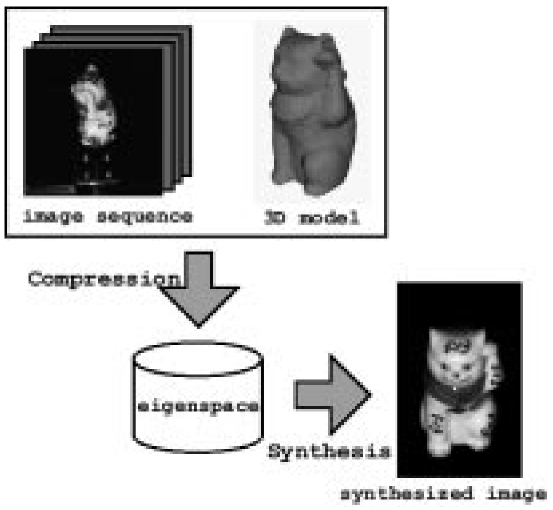

Figure 8.3 Eigen-texture method.

created from the range images. Each color image is then aligned with the 3D model, as was the case in the model-based rendering method. Each color image is divided into small areas that correspond to triangle patches on the 3D model. Each triangle patch is normalized to have the same shape and size as that of the others. Color images on the small areas are warped on a normalized triangular patch. We refers to this normalized triangular patch as a cell and to its color image as a cell image. A sequence of cell images from the same cell is collected as shown in Figure 8.4. Here this sequence depicts appearance variations on the same physical patch of the object surface under various viewing/illumination conditions.

These cell images are compressed using the eigenspace method [18]. Note that the compression is done in a sequence of cell images, whose appearance change is caused only by the change of shading, e.g., brightness and specularity of the triangular patch. Thus, high compression ratio can be expected with the eigenspace method. On the other hand, current image-based rendering methods usually compress images in the image coordinate systems or in other coordinate systems which are defined based on the image coordinate system. Each pixel in the sequence corresponds to different physical points in the image sequence. Thus, the variance carries both those from the imaging geometries and those from the physical locations.

Eigenspace compression on cell images can be achieved by the following steps. First, each cell image is converted into a $1 \times 3N$ vector X_m by arranging color values for each color band RGB in a raster scan manner. Here, m is the pose number of the real object, M is the total number of poses, and N is the number of pixels in each cell image.

$$X_m = \left[x^R_{m,1} \cdots x^R_{m,N} x^G_{m,1} \cdots x^G_{m,N} x^B_{m,1} \cdots x^B_{m,N} \right] \tag{8.4}$$

The sequence of cell images can be represented as a $M \times 3N$ matrix.

The average of all color values in the cell image set is subtracted from each

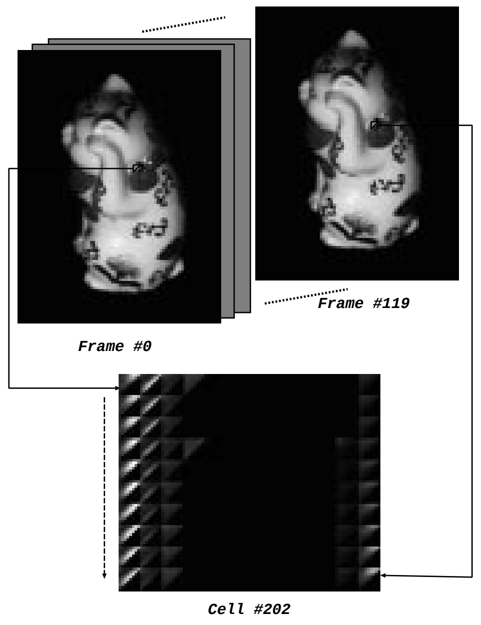

Figure 8.4 Cell images.

element of the matrix. This ensures that the eigenvector with the largest eigenvalue represents the dimension in eigenspace in which the variance of images is maximum in the correlation sense. The resulting matrix is represented as $X = [X_1 X_2 \cdots X_M]^T$.

With this resulting $M \times 3N$ matrix, we define a $3N \times 3N$ matrix Q, and determine eigenvectors e_i and the corresponding eigenvalues λ_i of Q by solving the eigenstructure decomposition problem: $Q = X^T X$ and $\lambda_i e_i = Q e_i$.

At this point, the eigenspace of Q is a high dimensional space, *i.e.*, $3N$ dimensions. Although $3N$ dimensions are necessary to represent each cell image in an exact manner, a small subset of them is sufficient to describe the principal characteristics as well as to reconstruct each cell image with adequate accuracy. Accordingly, we extract $k (k \ll 3N)$ eigenvectors which represent the original eigenspace adequately; by this process, we can substantially compress the image set. The k eigenvectors can be chosen by sorting the eigenvectors by the size of the corresponding eigenvalues, and then computing the eigenratio:

$$\sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{3N} \lambda_i \geq T, \tag{8.5}$$

where $T \leq 1$.

Using the k eigenvectors $\{e_i | i = 1, 2, \ldots, k\}$, each cell image can be projected onto the eigenspace composed by matrix Q by projecting each matrix X. And the projection of each cell image can be described as a $M \times k$ matrix $G : G = XV$ where $V = [e_1, e_2, \ldots, e_k]$.

To put it concisely, the input color image sequence is converted to a set of cell image sequences, and each sequence of cell images is stored as the matrix V, which is the subset of eigenvectors of Q, and the matrix G, which is the projection onto the eigenspace. Each sequence of cell images corresponding to $M \times 3N$ matrix X is compressed to $3N \times k$ matrix V and $M \times k$ matrix G, so that the compression ratio becomes: $k(M + 3N)/3MN$.

Each synthesized cell image can be computed by the following equation. Here, R is a $M \times 3N$ matrix which corresponds to the synthesized sequence of cell images, and each column of matrix R corresponds to the appearance of each triangular mesh (cell) of the object. Accordingly, a virtual object image of one particular pose (pose number) can be synthesized by aligning each corresponding cell appearance of every R to the 3D model:

$$R_m = \sum_{i=1}^{k} g_{m,i} e_i^T. \tag{8.6}$$

Implementation

We have implemented the eigen-texture method, and have applied it to a real object. The same experimental setup in the model-based rendering experiment is used to sample sequence of color and range images of a real object. Figure 8.5 shows the images synthesized by using 8 dimensional eigenspace. Details of the images are reconstructed accurately, and the synthesized images are indistinguishable from the input images. The compression ratio of the input image sequence was 7.9%. This ratio is quite high when compared with that of other image compression methods, and this experimental result demonstrates that the eigen-texture method is effective in terms of compression. This is because the eigen-texture method compresses the sequence of cell images that come from the same physical area on the object surface. The variances in the cell images are only caused by the relative directional change of the light source and the position of the viewer. Cell images are thus highly correlative.

The eigen-texture method cannot create images of an object under unfamiliar illumination conditions. Thus, we have to sample all possible illumination conditions. This is impossible. Fortunately, however, the irradiance at one point on the surface of the object from the whole illumination environment is a linear combination of the irradiance due to each light source composing the illumination environment. For this reason, we first decompose the real illumination environment into base point light sources and then sample the color images of the object under each base point light source separately. Those color images are compressed for each point light source by using the eigen-texture method. Then, arbitrary images of the object can be obtained as a linear combination of images synthesized for those point light sources.

As an example, we took three color images under 3 different point light sources, lighting them separately. Under each point light source, we took 40 color images and synthesized 120 virtual object images for each light source with the threshold 0.999 in eigenratio with interpolation in eigenspace. Then we synthesized the virtual object image sequence with all lights on by taking a linear combination of each point light source virtual object image. Figure 8.6 shows the result of this linear combination compared with a real image with all three lights turned on.

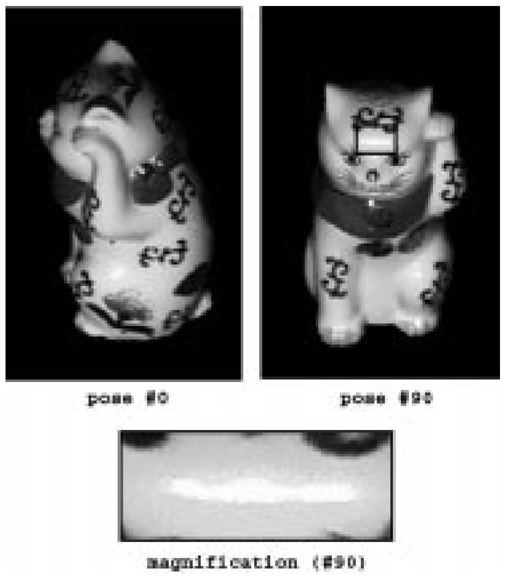

Figure 8.5 Synthesized images (8 dimensions).

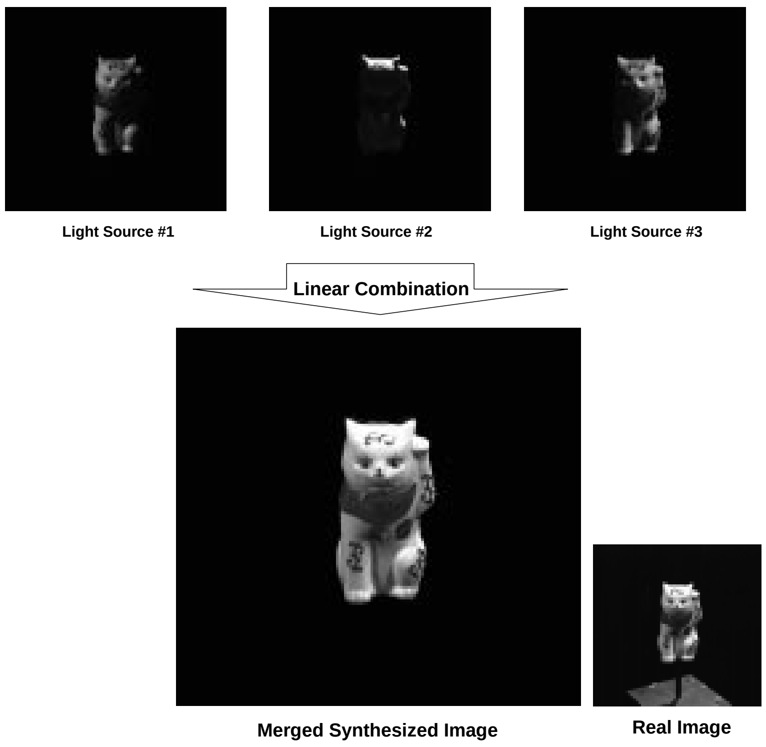

Figure 8.6 Linear combination of light sources.

Summary

This section describes the eigen-texture method for synthesizing virtual images of an object from a sequence of range and color images. Experimental results of the method proves its effectiveness, in particular, its high compression ratio. Using the model-based rendering method and the eigen-texture method, we can create appearances of a virtual object. The next section considers how to integrate these appearances with real scenes for mixed reality.

8.3 Integrating Virtual Objects with a Real Scene

For superimposing virtual objects onto a real scene appropriately [19] [20], the following three aspects have to be taken into account: geometry, illumination, and time. More specifically, the virtual object has to be located at a desired location in the real scene, and the object must appear at the correct location in the image (**consistency of geometry**). Also, shading of the virtual object has to match that of other objects in the scene, and the virtual object must cast a correct shadow, *i.e.*, a shadow whose characteristics are consistent with those of the shadows in the real scene (**consistency of illumination**). Lastly, motions of the virtual object and the real objects have to be coordinated (**consistency of time**).

In this section, we are mainly concerned with illumination consistency. Illumination of the real scene is directly measured and used for rendering virtual objects [3]. Our method measures a radiance distribution from all directions by a pair of omni-directional images taken by a CCD camera with a fisheye lens. If we use only one omni-directional image, we can determine only the radiance distribution seen from the particular point where the omni-directional image was captured. To overcome this limitation, our method uses an omni-directional stereo algorithm for measuring a radiance distribution of the real scene as a 3D spatial distribution (3D triangular mesh). Once the radiance distribution is obtained as a triangular mesh, the measured radiance distribution is used for rendering a virtual object and for generating shadows cast by the virtual object onto the real scene wherever the virtual object is placed in the scene.

8.3.1 Constructing a Radiance Map Using Omni-Directional Stereo

A CCD camera with a fisheye lens is placed at two known locations in the scene to capture two omni-directional images from different locations. An acquisition system for the omni-directional images is illustrated in Figure 8.7. From a pair of omni-directional images, an illumination distribution of the scene is estimated.

We first extract feature points with high contrast in the two omni-directional images by using the feature extraction algorithm proposed by Tomasi and Kanade [21]. In the algorithm, an image pixel with high gradient values in two orthogonal directions, e.g., a corner point, is extracted as a feature point.

Most of incoming light energy in a real scene comes from direct light sources such as a fluorescent lamp and a window to the outside, while the rest of the incoming light energy comes from indirect illumination such as reflection from a wall. Thus it is important to know the accurate locations of direct light sources to represent an illumination distribution of a real scene. Fortunately, direct light sources usually appear as significantly bright points in an omni-directional image. Therefore, it should be relatively easy to identify direct light sources in the image.

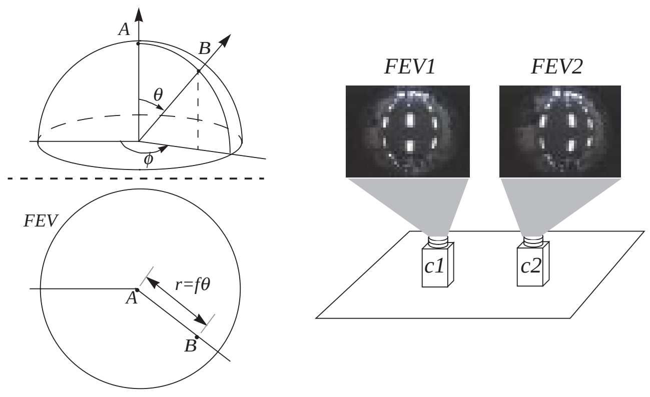

Figure 8.7 Acquisition of omni-directional images.

After feature points are extracted, 3D coordinates of points in the real scene corresponding to the extracted feature points are determined by using a stereo algorithm. This is done by obtaining an intersection point of a pair of lines-of-sight given by a pair of feature points. The direction of line-of-sight is determined from the image coordinate of the point corresponding to the light-of-sight using a known geometry of the fisheye lens.

In order to include indirect light sources such as walls and ceilings, we set up a 3D triangular mesh of the entire scene. First, we construct a 2D triangular mesh by applying 2D Delaunay triangulation to the feature points in the first omni-directional image. That determines the connectivity of a 3D triangular mesh whose vertices are the 3D points corresponding to the feature points in the image. Then, using the connectivity, a 3D triangular mesh is created from the 3D feature points.

The obtained triangular mesh approximates an entire shape of the real scene, e.g., the ceiling and walls of a room, which act as direct or indirect light sources. After the shape of the real scene is obtained as a triangular mesh, the radiance of the scene is estimated by using the image brightness of the omni-directional images. This radiance value is warped on the corresponding triangular mesh. The 3D textured mesh is referred to as a radiance map. An example of an obtained radiance map is shown in Figure 8.8(d).

8.3.2 Generating Soft Shadow

For superimposing virtual objects onto an image of a real scene, the ray casting algorithm is used. In this section, we assume that a virtual object always exists between the camera projection center and real objects in the scene and that the virtual object is located on the table, of which the position is known.

For each pixel in the input image of the real scene, a ray extending from the camera projection center through the pixel is generated by using the transformation between the world coordinate system and the image coordinate system. If the ray intersects a virtual object, we consider that the pixel corresponds to a point on the virtual object surface. Then we compute a color to be observed at the surface point under the measured radiance distribution of the real scene with the Torrance-Sparrow reflectance model. The computed color is stored in the pixel as the surface color of the virtual object at the pixel.

If a ray through an image pixel does not intersect with a virtual object, the color of a real object surface corresponding to the image pixel needs to be modified so that a shadow cast by the virtual object is created on the real object surface.

First, we obtain a 3D coordinate of a point on a real object surface where a ray through an image pixel intersects the real object. Then, a total irradiance E_1 at the surface point is computed from the measured illumination distribution by assuming that a virtual object does not occlude incoming light at the surface point.

$$E_1, c = \int_{-\pi}^{\pi} \int_{0}^{\frac{\pi}{2}} L_c(\theta_i, \phi_i) cos\theta_i \sin\theta_i d\theta_i d\phi_i \qquad c = R, G, B \tag{8.7}$$

where $L(\theta_i, \phi_i)$ $(i = 1, 2, .., n)$ are the estimated illumination radiance.

Then, a partial irradiance E_2 at the surface point given by non-occluded light sources is computed as

$$E_{2,c} = \int_{-\pi}^{\pi} \int_{0}^{\frac{\pi}{2}} L_c(\theta_i, \phi_i) S(\theta_i, \phi_i) cos\theta_i \sin\theta_i d\theta_i d\phi_i \qquad c = R, G, B \tag{8.8}$$

where $S(\theta_i, \phi_i)$ are occlusion coefficients; $S_i = 0$ if $L(\theta_i, \phi_i)$ is occluded by the object; Otherwise $S_i = 1$.

The ratio of the total radiance E_2 to the total radiance E_1 is multiplied to the observed color at the intersection between the ray and the plane on which the cast shadow is generated.

$$I'_c = I_c \frac{E_{2,c}}{E_{1,c}} \qquad c = R, G, B \tag{8.9}$$

8.3.3 Experimental Results

An image was taken of a tabletop and miscellaneous objects on the tabletop in our laboratory. From the same camera position, another image of the tabletop with a calibration board was taken. The input image and the calibration image are shown in Figure 8.8(a) and (b).

First, regularly spaced dots on the calibration board were extracted in the calibration image to determine their 2D image coordinates. From pairs of the 2D image coordinates and the 3D world coordinates that were given a priori, the transformation between the world coordinate system and the image coordinate system was estimated by using the camera calibration algorithm [22]. Two omni-directional images of the scene, e.g., the ceiling of the laboratory in this experiment, were taken. Figure 8.8(c) shows the two omni-directional images. From this pair of images, the radiance map shown in Figure 8.8 is obtained using the omni-stereo algorithm. Then

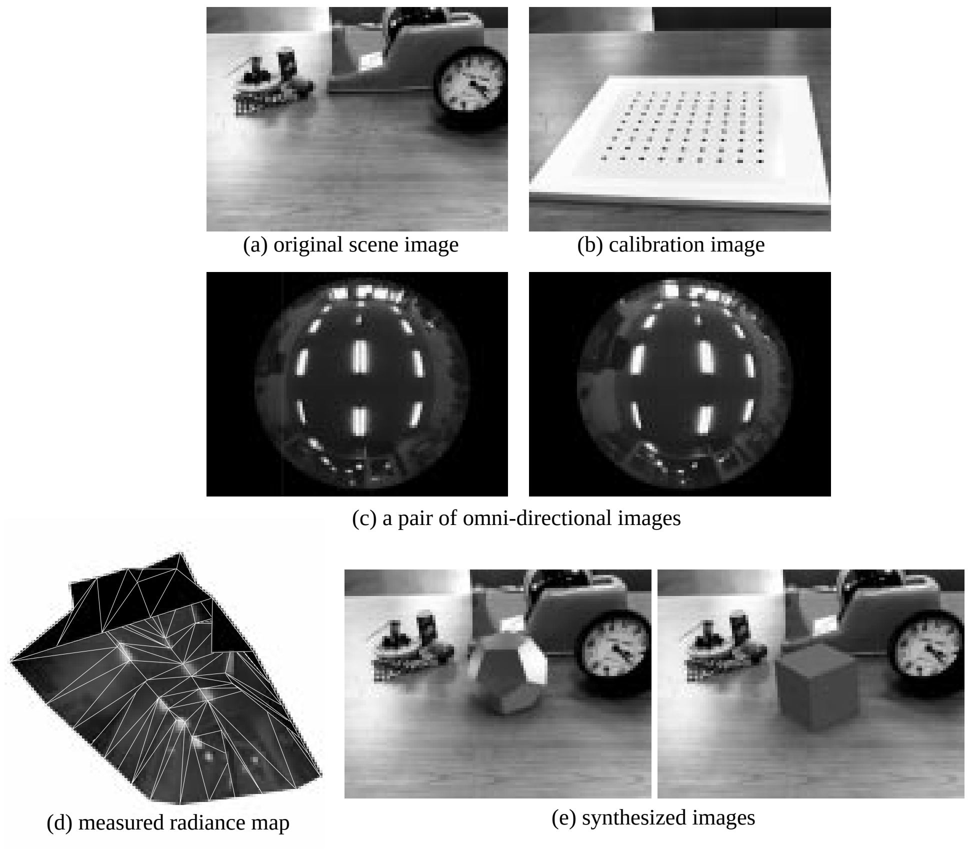

Figure 8.8 Results.

the radiance map was used for rendering virtual objects in the real scene. Figure 8.8 (e) shows the resulting image that contains a virtual object with a soft shadow on the tabletop.

Figure 8.9 shows another example of integration of a virtual image with a real background.

8.4 Conclusions

We have explored the automatic generation of photorealistic object models from observation. For those objects with known reflectance models, we proposed a model-based rendering method that obtains reflectance parameters from sequences of range and color images. For other classes of objects, we have developed the eigen-textured method that stores and compresses appearances of objects on their 3D mesh models. Our experiments have shown that both of our rendering methods can be effectively used for synthesizing realistic object images. We have also developed a new method of superimposing virtual objects onto images of real scenes by taking into account

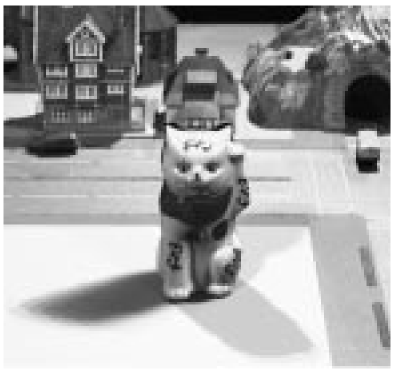

Figure 8.9 Another example of a mixed reality image.

the radiance distribution of the scenes. To demonstrate the effectiveness of the proposed method, we have successfully tested our method by using real images.

Acknowledgments

This research has been supported in part by Ministry of Education, Japanese Government, under grant number 09NP1401(shin-pro) and 09450164(kakenhi).

References

[1] Y. Sato, M. Wheeler, and K. Ikeuchi: "Object shape and reflectance modeling from observation," *Proc. SIGGRAPH'97*, pp.379–387, 1997.

[2] K. Nishino, Y. Sato, I. Sato, and K. Ikeuchi: "Eigen-texture method - Appearance compression based on 3D model -," *Information Processing Society of Japan Symposium Series*, vol.98, no.10, pp.I19–I26, July 1998 (in Japanese).

[3] I. Sato, Y. Sato, and K. Ikeuchi: "Acquiring a radiance distribution to superimpose virtual objects onto a real scene," *IEEE Trans. on Visualization and Computer Graphics,* 1999 (to appear).

[4] M. Bajura, H. Fuchs, and R. Ohbuchi: "Merging virtual objects with the real world," *Proc. SIGGRAPH'92,* pp.203–210, 1992.

[5] S. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen: "The Lumigraph," *Proc. SIGGRAPH'96,* pp.43–54, 1996.

[6] M. Levoy and P. Hanrahan: "Light field rendering," *Proc. SIGGRAPH'96,* pp.31–42, 1996.

[7] K. Sato, H. Yamamoto, and S. Inokuchi: "Range imaging system utilizing nematic liquid crystal mask," *Proc. 1st Int'l Conf. on Computer Vision,* pp.657–661, 1987.

[8] M. Wheeler, Y. Sato, and K. Ikeuchi: "Consensus surfaces for modeling 3D objects from multiple range images," *Proc. 6th Int'l Conf. on Computer Vision,* pp.917–924, 1998.

[9] B. Curless and M. Levoy: "A volumetric method for building complex models from range images," *Proc. SIGGRAPH'96,* pp.303–312, 1996.

[10] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle: "Surface reconstruction from unorganized points," *Proc. SIGGRAPH'92,* pp.71–78, 1992.

[11] K. Torrance and E. Sparrow: "Theory for off-specular reflection from roughened surface," *J. Opt. Soc. Am.,* vol.57, pp.1105–1114, 1967.

[12] S. Nayar, K. Ikeuchi, and T. Kanade: "Surface reflection," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol.13, no.7, pp.611–634, 1991.

[13] Y. Sato and K. Ikeuchi: "Temporal-color space analysis of reflection," *J. Opt. Soc. Am. A,* vol.11, no.11, pp.2990–3002, November 1994.

[14] K. Ikeuchi and K. Sato: "Determining reflectance properties of an object using range and brightness images," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol.13, no.11, pp.1139–1153, 1991.

[15] G. K. Klinker, S. Shafer, and T. Kanade: "The measurement of highlights in color images," *Int'l J. Computer Vision,* vol.2, pp.7–32, 1988.

[16] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle: "View-based rendering," *Proc. 8th Eurographics Workshop on Rendering,* June 1997.

[17] Z. Zhang: "Modeling geometric structure and illumination variation of a scene from real images," *Proc. 6th Int'l Conf. on Computer Vision,* pp.1041–1046, 1998.

[18] H. Murase and S. K. Nayar: "Visual learning and recognition of 3-D objects from appearance," *Int'l J. Computer Vision*, vol.14, pp.5–24, 1995.

[19] R. T. Azuma: "A survey of augmented reality," *Presence,* vol.6, no.4, pp.355–385, August 1997.

[20] A. State, G. Hirota, D. Chen, W. Garrett, and M. Livingston: "Superior augmented-reality registration by integrating landmark tracking and magnetic tracking," *Proc. SIGGRAPH'96,* pp.429–438, August 1996.

[21] C. Tomasi and T. Kanade: "Shape and motion from image streams under orthography: a factorization method," *Int'l J. Computer Vision*, vol.9, no.2, pp.137–154, 1992.

[22] R. Tsai: "A versatile camera calibration technique for high accuracy machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Trans. on Robotics and Automation,* vol.3, no.4, pp.323–344, 1987.

[23] S. Chen: "Quicktime VR," *Proc. SIGGRAPH'95,* pp.29–38, 1995.

[24] A. Fournier, A. Gunawan and C. Romanzin: "Common illumination between real and computer generated scenes," *Proc. Graphics Interface '93,* pp.254–262, 1993.

[25] G. Kay and T. Caelli: "Inverting an illumination model from range and intensity maps," *CVGIP: IU,* vol.59, pp.183–201, 1994.