Eigen-Texture Method: Appearance Compression and Synthesis Based on a 3D Model

Ko Nishino, *Student Member*, *IEEE*, Yoichi Sato, *Member*, *IEEE*, and Katsushi Ikeuchi, *Fellow*, *IEEE*

Abstract—Image-based and model-based methods are two representative rendering methods for generating virtual images of objects from their real images. However, both methods still have several drawbacks when we attempt to apply them to mixed reality where we integrate virtual images with real background images. To overcome these difficulties, we propose a new method, which we refer to as the Eigen-Texture method. The proposed method samples appearances of a real object under various illumination and viewing conditions, and compresses them in the 2D coordinate system defined on the 3D model surface generated from a sequence of range images. The Eigen-Texture method is an example of a view-dependent texturing approach which combines the advantages of image-based and model-based approaches: No reflectance analysis of the object surface is needed, while an accurate 3D geometric model facilitates integration with other scenes. This paper describes the method and reports on its implementation.

Index Terms—Image synthesis, texture, appearance, model-based rendering, image-based rendering, principle component analysis.

1 Introduction

Recently, there has been extensive development of mixed reality systems for the purpose of integrating virtual images of objects with real background images. Main research issues include how to obtain virtual images of objects and then how to seamlessly integrate those objects with real images. Our interest is directed toward the former topic: rendering virtual object images from real images. The computer vision and computer graphics communities have proposed two representative rendering methods to obtain such virtual images from real objects: image-based and model-based methods.

Image-based methods [20], [3], [8], [12] are based on the fact that light rays can be parameterized as a 7D function called the plenoptic function [1]. Considering each pixel in real images as samples of this plenoptic function, image-based methods synthesize virtual images by selecting the most appropriate sampled ray, or by interpolating between the rays, if necessary. This is equivalent to taking an image sequence as an input and interpolating those images to create a virtual image. Since image-based rendering does not assume any particular reflectance characteristics of objects nor does it require any detailed analysis of the reflectance characteristics of the objects, the method can be applied to a wide variety of real objects. Since it is also quite simple and easy to use, 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 disadvantages for application to mixed reality which integrates such virtual objects with real background. To avoid complications, few image-based rendering methods, e.g., Lumigraph [8], employ accurate 3D models of real objects. Without a geometric model, it is difficult to generate cast shadows of virtual objects in real background scene, while shadows are important for seamless integration of virtual and real scenes.

Unlike image-based rendering, model-based rendering assumes reflectance models of an object and determines reflectance parameters through detailed reflectance analysis [18], [19]. Later, the method uses those reflectance parameters to generate virtual images by considering illumination conditions of the real scene. By using these reflectance parameters, integration of synthesized images with the real background can be accomplished quite realistically [9]. However, model-based rendering has nontrivial intrinsic constraints; it cannot be applied to objects whose reflectance properties cannot be approximated by using simple reflection models. Furthermore, since computation of reflectance parameters needs the surface to be normal at each point of the object surface, it cannot be applied to objects whose surfaces are rough.

Several other methods, such as voxel coloring, take intermediate approaches between purely image-based and model-based approaches. Voxel Coloring/Space Carving methods [21], [4], [10] represent the scene with a colored voxel model. By using the fact that surface points in a scene and voxels that represent them project to consistent colors in the input images, the volume containing the scene is iteratively carved out generating a volumetric representation of the scene with its texture color attached on each

- *K. Nishino and K. Ikeuchi are with Department of Information Science, Graduate School of Science, The University of Tokyo, 461 Komaba, Meguro-ku, Tokyo, Japan 153-8505. E-mail: {kon, ki}@cvl.iis.u-tokyo.ac.jp.*
- *Y. Sato is with the Institute of Industrial Science, The University of Tokyo, 461 Komba, Meguro-ku, Tokyo, Japan 153-8505. E-mail: ysato@iis.u-toyko.ac.jp.*

Manuscript received 25 July 2000; revised 28 Apr. 2001; accepted 14 June 2001.
Recommended for acceptance by S. Dickinson.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 112590.

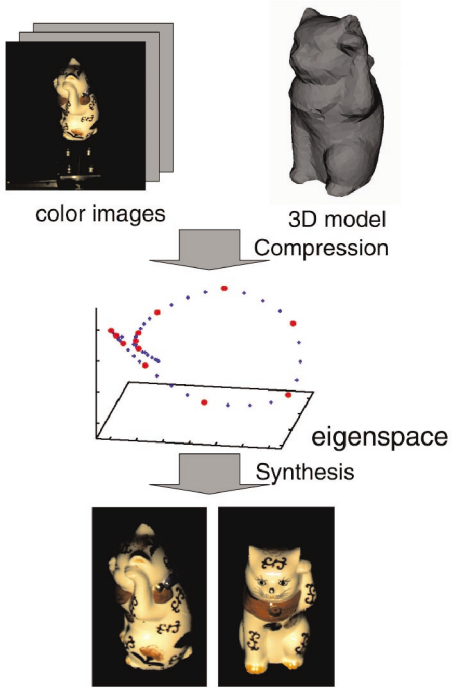


Fig. 1. Outline of the *Eigen-Texture Method*.

surface voxel. However, since these methods rely on the color consistency assumption that holds only on object surfaces that have Lambertian reflection property, it has difficulty in handling objects that have reflection properties that is not just diffusive. The approach we will propose is similar to voxel coloring with regard to the point we use geometric model; however, we obtain the geometric model directly from real objects by using a range scanner.

To avoid the need for surface reflectance analysis while making efficient use of geometric information which can also be used for cast shadow generation when integrating virtual objects into real scenes as mixed reality applications, we propose a new rendering method, which we refer to as the *Eigen-Texture method*. Fig. 1 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 pastes color images of the object onto the 3D surface of the object model and then compresses those appearances in the 2D coordinate system defined on the 3D model surface. This compression is accomplished in an eigenstructure decomposition framework. The synthesis process is achieved by using the inverse transformation of the eigenstructure decomposition. Thanks to the linearity of image brightness, virtual images under a complicated illumination condition can be generated by summation of component virtual images sampled under single illuminations. When superimposing virtual objects into real scene, the geometric consistency between the virtual object and the lighting distribution has to be maintained precisely. As we employ a geometric model to represent the virtual object, if we know the real illumination condition, we can precisely render the cast shadows and also the appearance by decomposing the real illumination distribution into a set of point light sources. A preliminary experiment will be shown in Section 4.

The proposed method can be considered to lie in the research field of view-dependent texture mapping. View-dependent texture mapping is a technique to combine multiple textures corresponding to each triangular patch of the geometric model, so that it can be rendered from view points that do not exist in the sampled images. On combining textures, weights are multiplied to the textures, e.g., weights depending on the angle between the triangular patch's surface normal and viewing angle [16], [7], [6], and blending is accomplished between the weighted textures. Our approach is different from these approaches in the respect that we derive the principle components of the textures and synthesize new textures by combining those principle components providing an intuitive interpolation method to render textures from new view points.

Analyzing real-world textures for deriving the essential characteristics of them is an ongoing research topic in the computer vision community. Especially for recognition tasks, several methods have been proposed to derive the basic components of real-world textures sampled under different illumination conditions and viewing directions. For instance, Leung and Malik [11] represent textures with *3D textons* extracted by applying filter banks, and Suen and Healey [23] derive the basis textures by applying orthonormal basis expansion and define a *dimensionality surface* which represents the number of basis textures necessary to represent the texture. We apply principle component analysis on sampled textures to represent the appearance variation of textures under varying illumination and viewing conditions. While our objective is view-dependent image synthesis, recognition of real-world texture can be a straightforward application of our approach.

Related work has been done by Zhang [28], who uses a stereo camera technique to obtain a partial 3D model of an object and compresses the pixel values of each point of the object surface with respect to the image coordinates. Our method differs from his method in that we treat the appearance of the object as dense texture, not sparse point set, and our method can generate a full 3D model of the object with all appearances from any degrees of viewer directions; this ability gives us great advantages in mixed reality systems.

The remainder of the paper is organized as follows: In Section 2, we describe the *Eigen-Texture method*. In Section 3, we describe the implementation of the proposed method and discuss the results of the experiments we conducted to demonstrate the efficiency of the proposed method. In Section 4, we describe the results of applying our method in integrating virtual images into real scenes. In Section 5, we discuss the merits and limitations of our method.

2 Eigen-Texture Method

This section describes the theory of the *Eigen-Texture method*. The method samples a sequence of color and range images. Once these two sequences are input to the system, a 3D geometric model of the object is created from the sequence of range images. Each color image is aligned with the 3D model of the object. In our system, this alignment is relatively simple because we use the same color CCD

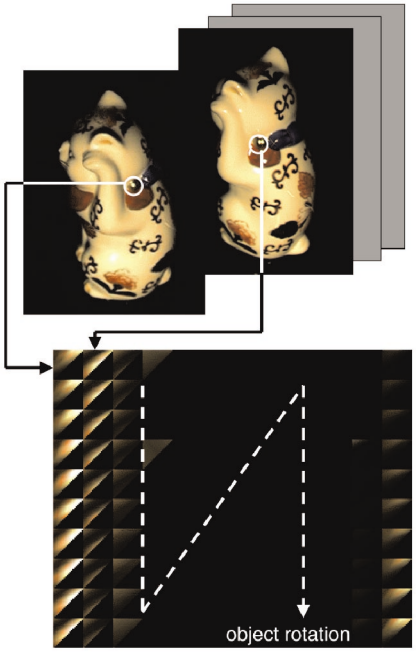


Fig. 2. A sequence of cell images.

camera for taking both range and color images. Each color image is divided into small areas that correspond to triangle patches on the 3D model. Then, the color images corresponding to the triangular patches are warped to have a predetermined normalized triangle shape by bilinearly interpolating the pixel values. This paper refers to this normalized triangular shape corresponding to each 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 Fig. 2. Here, this sequence depicts appearance variations on the same physical patch of the object under various viewing conditions. Principle component analysis (PCA) is applied for each cell image sequence to compress each cell image set separately. Note that the compression is done in a sequence of cell images. The appearance changes in the cell image sequence are mostly due to the change of brightness, when the object surface is well approximated with the triangular patch based 3D model, except some cell images may have highlights passing across. Thus, a high compression ratio can be expected with PCA. Furthermore, it is possible to interpolate appearances in the eigenspace.

Eigenspace compression on cell images can be achieved by the following steps: The color images are represented in RGB pixels, but the compression is accomplished in YC_rC_b using $4:1:1$ subsampling, based on the accepted theory that human perception is less sensitive to slight color changes than brightness changes. First, each cell image is converted into a $1 \times 3N$ vector $\mathbf{X}_m$ by arranging color values for each color band YC_rC_b in a raster scan manner (1). Here, M is the total number of poses of the real object, N is the number of pixels in each cell image, and m is the pose number.

$$\mathbf{X}_m = \left[x_{m,1}^{Y} \ \ldots \ x_{m,1}^{C_r} \ \ldots \ x_{m,N}^{C_b} \right]. \tag{1}$$

Then, the sequence of cell images can be represented as a $M \times 3N$ matrix as shown in (2).

$$\mathbf{X} = \left[\mathbf{X}_1^T \ \ \mathbf{X}_2^T \ \ldots \ \mathbf{X}_M^T \right]^T. \tag{2}$$

The average of all color values in the cell image set is subtracted from each element of matrix $\mathbf{X}$. 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.

$$\mathbf{X}_{ave} = \begin{bmatrix} \ldots & x_{1,N}^{R} - E & \ldots & x_{1,N}^{G} - E & \ldots & x_{1,N}^{B} - E \\ \ldots & . & \ldots & . & \ldots & . \end{bmatrix} \tag{3}$$

$$\mathbf{X}_a = \mathbf{X} - \begin{bmatrix} E & \ldots & E \\ . & \ldots & . \\ E & \ldots & E \end{bmatrix} \tag{4}$$

$$E = \frac{1}{3MN} \sum_{i=1,j=1,c\in\{Y,C_r,C_b\}}^{M\,N} x_{i,j}^{c}.$$

With this $M \times 3N$ matrix, we define a $3N \times 3N$ matrix $\mathbf{Q}$ and determine eigenvectors e_i and the corresponding eigenvalues λ_i of $\mathbf{Q}$ by solving the eigenstructure decomposition problem.

$$\mathbf{Q} = \mathbf{X}_a^T \mathbf{X}_a, \tag{5}$$

$$\lambda_i \mathbf{e}_i = \mathbf{Q}\mathbf{e}_i. \tag{6}$$

At this point, the eigenspace of $\mathbf{Q}$ 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 enough 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 (7).

$$\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{3N} \lambda_i} \geq T \ where \ T \leq 1. \tag{7}$$

Using the k eigenvectors $\{\mathbf{e}_i | i = 1, 2, ..., k\}$ (where $\mathbf{e}_i$ is a $3N \times 1$ vector) obtained by using the process above; each cell image can be projected on to the eigenspace composed by matrix $\mathbf{Q}$ by projecting each matrix $\mathbf{X}_a$. And, the projection of each cell image can be described as a $M \times k$ matrix $\mathbf{G}$.

$$\mathbf{G} = \mathbf{X}_a \times \mathbf{V} \ \ where \ \ \mathbf{V} = \left[\mathbf{e}_1 \ \ \mathbf{e}_2 \ \ldots \ \mathbf{e}_k \right]. \tag{8}$$

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 $\mathbf{V}$, which is the subset of eigenvectors of $\mathbf{Q}$, and the matrix $\mathbf{G}$, which is the projection onto the eigenspace. As we described in (1), each sequence of cell images corresponds to one $M \times 3N$ matrix $\mathbf{X}$ and is stored as $3N \times k$ matrix $\mathbf{V}$ and $M \times k$

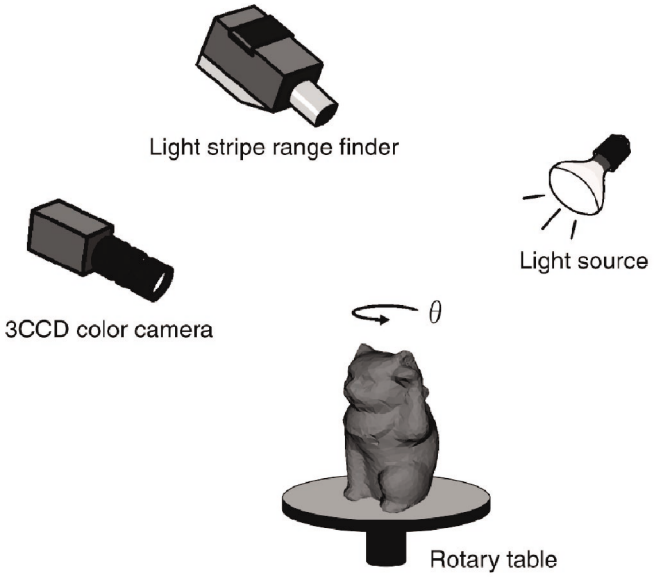


Fig. 3. The system setup we use.

matrix $\mathbf{G}$, so that the compression ratio becomes that described in (9).

$$compression\ ratio = k\frac{M+3N}{3MN}. \tag{9}$$

Each synthesized cell image can be computed by (10). A virtual object image of one particular pose (pose number m) can be synthesized by aligning each corresponding cell appearance (R_m) to the 3D model.

$$\mathbf{R}_m = \sum_{i=1}^{k} g_{m,i}\mathbf{e}_i^T + [E\ \ E\ \ \ldots\ \ E]. \tag{10}$$

3 Implementation

We have implemented the system described in the previous section and have applied the *Eigen-Texture method* to real objects.

3.1 System Setup

We built an experimental system to capture the input color images and range images. In our capturing system setup, the object is attached to a rotary table (Fig. 3). A single point light source fixed in the world coordinate is used to illuminate the object. A range image is taken through the 3 CCD color camera under a nematic liquid crystal mask [17]. A sequence of range images is taken by rotating the object 30° by each step in this experiment. Each range image is converted into a triangular mesh model and then registered to each other by iteratively minimizing the distances between corresponding points [2], [25]. After registration, all mesh models are integrated by obtaining the consensus surface of each mesh surface in a volumetric approach [26]. A sequence of color images is also taken by the same 3 CCD color camera, similarly rotating the object as with the range image sequence, but the rotation interval is smaller than that of the range image sequence. For instance, a step of 3° was used for the first experiment described in the next section.

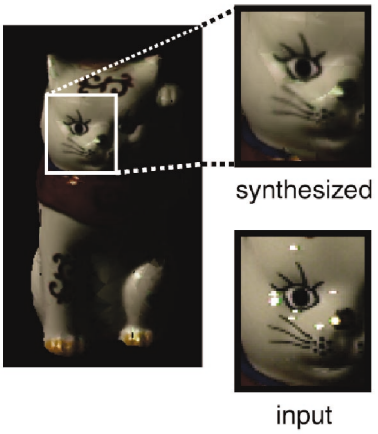


Fig. 4. Virtual object image synthesized by using three-dimensional eigenspaces (an example of a highly reflective object).

Since we use a light stripe range finder, the range images and the color images are taken with the same camera. Therefore, the color images can be easily aligned with the 3D model generated from the range images. Another alternative way to obtain accurate 3D models is to use laser range finders. When using other types of range finders, such as laser range finders, since the 3D model and the color images will be taken in different coordinate systems, they have to be aligned with each other. This can be accomplished by minimizing some error metric of the point correspondence between the color images and the 3D model, or by maximizing the correlation between them.

3.2 Cell-Adaptive Dimensional Eigenspace

Determining the number of dimensions of the eigenspace in which the sequence of cell images are stored is a nontrivial issue, as it has significant influence on the quality of the synthesized images. According to the theory of photometric stereo [27] and Shashua's linear theory [22], in an ideal case where the 3D model is perfectly aligned with the input images and when it approximates the real object surface well, i.e., when the object surface is planner or when the geometric model is tessellated enough, three dimensions are enough for compressing and synthesizing the appearance of an object with a Lambertian surface. However, as shown in Fig. 4, the images synthesized by using the data stored in only 3D eigenspace for every cell have an intolerable blurred effect around the highlights and the textures are matted. As the reflection of most general real objects cannot be approximated by simple Lambertian reflection model due to nonlinear factors, such as specular reflection and self shadowing, 3D eigenspace is not appropriate for storing all the appearance changes of each cell.

The quality of the geometric model has serious influence on the number of dimensions of eigenspace required to synthesize the image precisely. The simpler the construction of the geometric model, the higher the eigenspace dimensions that are needed since the triangle patches get far from approximating the object's real surface, and the correlation of each cell image becomes low. Fig. 5 shows a rendered image using only three eigenvectors of a duck made of wool. Although the material has a Lambertian reflectance property, as it has a fairly detailed structure that is difficult

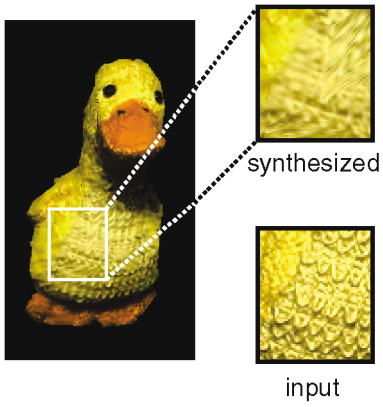


Fig. 5. Virtual object image synthesized by using three-dimensional eigenspace (an example of a object with rough surface).

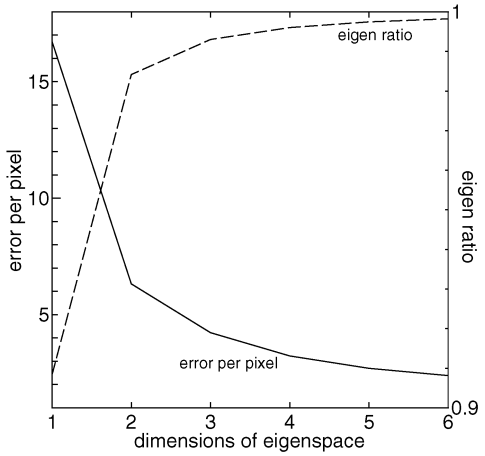


Fig. 6. Graph of the eigenratio and the error per pixel versus the number of dimensions of eigenspace.

to be approximated accurately by the 3D model surface, three dimensions do not suffice the final rendering quality; self shadows generated by the wool sticking out from the belly are smoothed out.

These facts indicate that the number of dimensions of eigenspace should differ for each of the cells, according to whether they have highlights or self shadows in their sequences and to the size of their triangle patch.

Taking these points into account, we determined the number of dimensions of the eigenspace independently for each cell so that each could be synthesized precisely. We used eigenratio to determine the number of dimensions for each cell. Fig. 6 describes the relationship between the eigenratio, the error per pixel versus the number of dimensions of the eigenspace. Here, the eigenratio is the ratio of the eigenvalues in the whole summation of them, and the error per pixel describes the average difference of the pixel values in 256 graduation in the synthesized and real image. The eigenratio is in inverse proportion to the error per pixel, so that it is reasonable to threshold the eigenratio to determine the number of dimensions of the eigenspace. For each sequence of cell images, we computed the eigenratio with (7) and used the first k eigenvectors whose corresponding eigenvalues satisfied a predetermined threshold of the eigenratio. The number of dimensions for each cell required to compress the sequence of input images and to reconstruct the synthesized images can be optimized by using these cell-adaptive dimensions. Yet, on the whole, the size of the database can be reduced. This cell-adaptive dimension method can be implemented because our method deals with the sequence of input images as with small segmented cell images.

Fig. 7 shows the images synthesized by using 0.999 as the threshold of the eigenratio for each cell. As can be seen in Fig. 7, the results described on the right side are indistinguishable from the input images shown on the left side. The number of dimensions of eigenspace used, the compression ratio, and the average error per pixel are summarized in Table 1. Due to the small protruding wool around the duck's stomach, tiny self shadows appear throughout the input sequence of color images. This causes the increase of the necessary number of dimensions for the duck compared with those necessary for the other objects. Because our method does not assume any reflectance model and does not require any analysis of the surface properties, it can be applied to objects with rough surfaces like the bear with fine fur and the duck with knitted wool, an application which is difficult to accomplish with model-based methods.

3.3 Compression Ratio

The compression ratio defined in (9) is the compression ratio of the cell image sequences; this ratio is computed from the number of dimensions, the size of each cell image, and the number of input color images. The essential compression ratio described in Table 1 is also the compression ratio between the cell image sequence and the data size to be stored in eigenspaces, but its value is computed from the real data size on the computer. The values of this essential compression ratio are relatively lower than the compression ratio computed by (9). This is caused by the difference of data types: The cell image sequences derived from the input color image sequence are represented by *unsigned char* type, while the data to be stored in eigenspaces—the projections and eigenvectors—are represented with *float* type. In future work, standard compression techniques such as vector quantization can be applied to avoid this loss in compression due to data types. As our main claim of the proposed method is to show the effectiveness of treating the appearance variation of real objects on the its surface, we applied PCA as a straight forward compression technique that allows intuitive interpolation as will be discussed in Section 3.4.

In this paper, we are not computing the compression ratio between the input image sequence and the data size stored in eigenspaces because an accurate comparison cannot be accomplished due to the resolution problem of cell images. When the *Eigen-Texture method* converts the input image sequence into a set of cell image sequences, the resolution of the cell images is determined by a fixed number with regard to the largest triangular patch on the 3D model surface. Although this enables us to synthesize virtual images in higher resolution as compared with the input images, it causes loss in compression ratio. We are now investigating a method to determine the resolution of cell images adaptively to their corresponding triangular patch size; in addition, we are considering a method to

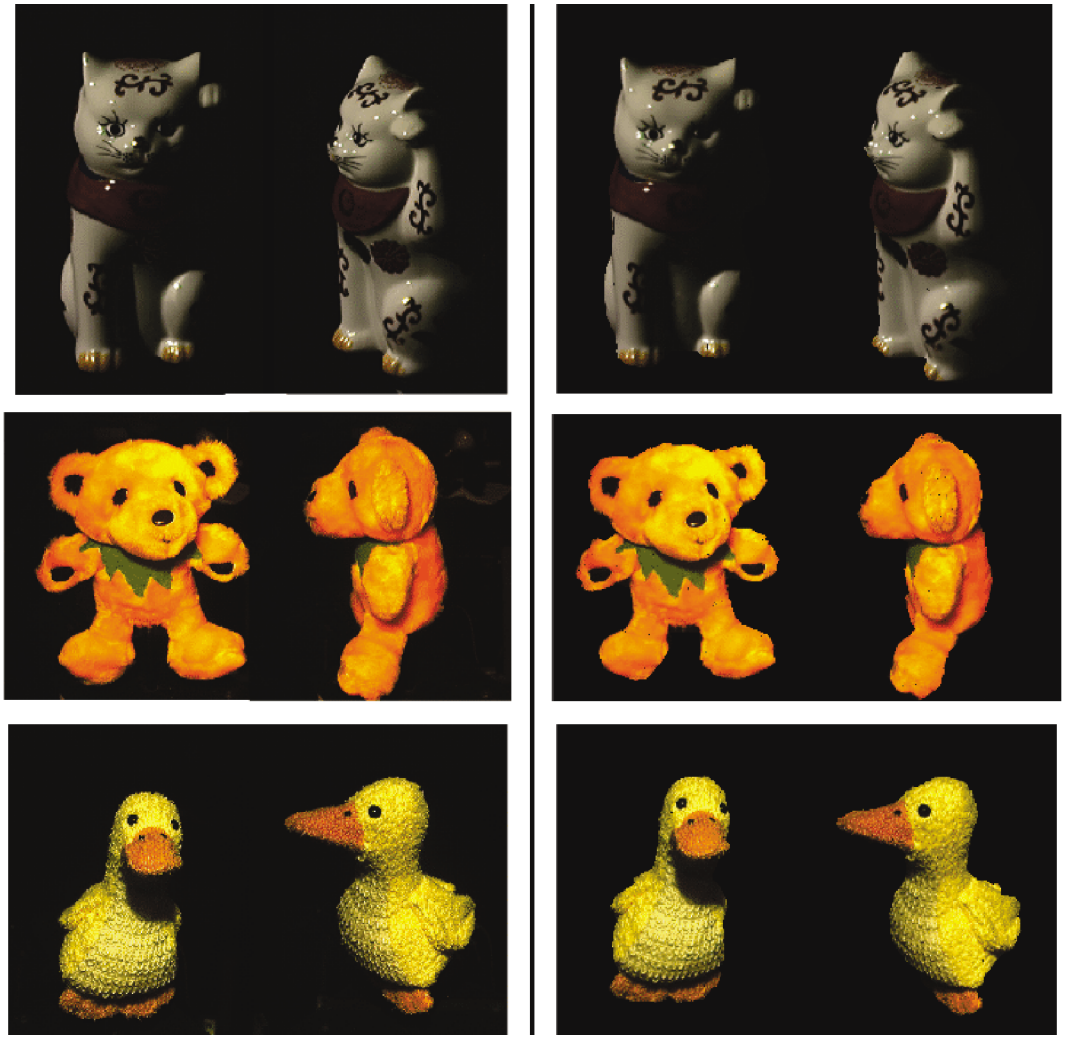


Fig. 7. Left: Input color images. Right: Synthesized images (by using cell-adaptive dimensional eigenspaces).

derive a higher compression ratio, mesh generation of the 3D model with regard to the color attributes on its surface.

3.4 Interpolation in Eigenspace

Once the input color images are decomposed into a set of sequences of cell images and projected onto their own eigenspaces, interpolation between each input image can be accomplished in these eigenspaces.

As an experiment, we took 30 images of a real object as the input image sequence by rotating the object at 12° by step. By interpolating the projections of these input images in the eigenspace, we obtained interpolated projections for 3° degrees rotation by each step. By synthesizing images using these interpolated projections, images of the object whose pose were not captured in the input color images could be synthesized.

Fig. 8 shows the synthesized images with interpolation in eigenspace. The results prove that we can reconstruct synthetic images by interpolation in eigenspace with adequate accuracy. The average dimensions of eigenspace for all cells used to fill 99.9 percent in eigenratio was $5:1$, and the compression ratio was about $20:1$. But, as we used only 30 images as the input image sequence and synthesized 120 images, the substantial compression ratio became almost $80:1$. The average error per pixel at this time was about 7.8.

TABLE 1
The Results

	Cat	Bear	Duck
Average number of dimensions	6.56	6.96	17.6
Compression ratio	15.3:1	14.5:1	5.72:1
Essential compression ratio	8.34:1	6.02:1	3.15:1
(MB)	357:42	702:117	290:92
Average error per pixel	1.46	1.79	1.79

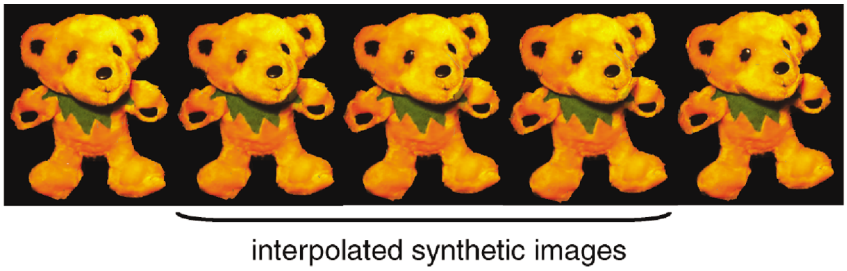


Fig. 8. Virtual images reconstructed by interpolating input images in eigenspace.

Fig. 9. Side by side comparison of input images and rendered images with interpolation in eigenspace. The original images are in the first column and the rendered images are in the second column. The first and last images in the original images were used as the input and intermediate images in the rendered sequence are the results of interpolation.

On the other hand, when interpolating the appearance of highly reflective objects, the highlights will not be interpolated smoothly. As can be seen in Fig. 9 and Fig. 10, the highlights will fade out and fade in, rather than moving across. This is an unavoidable effect as far as we interpolate the specular reflection linearly, in eigenspace in our case. However, from our experience, this highlight jumping effect can be tolerated when the sequence is shown as a movie when the input images are provided not too sparse, for instance, 12° by step for 360° reconstruction.

When using MPEGI to compress all the input color images into a single image sequence, the compression ratio becomes around $127:1$. Despite this high compression ratio achieved by MPEGI, the average error per pixel becomes around 7 to 8. As image sequence compression methods such as MPEGI compress the images in their 2D coordinates without any geometric information of the target object, the errors tend to appear around the nonlinear changes of the pixel values in the images (Fig. 11), i.e., edges of the occluding boundaries, edges of the texture, highlights, etc. These errors cannot be seen in the images synthesized with *Eigen-Texture method*. Even when compressing the input color images putting the priority on the quality by using MPEGII, it is hard to avoid these errors while keeping the compression ratio lower than *Eigen-Texture method*. This is because *Eigen-Texture method* accomplishes compression on the object surface. Other compression techniques than principle component analysis, e.g., discrete cosine transformation, may work as well as far as the appearance change is treated on the object surface.

4 Integrating into Real Scene

Since our method holds an accurate 3D object model constructed from a range image sequence, the synthesized virtual images can be seamlessly integrated into real scene images, taking the real illumination environment into account. As shown in (11), the irradiance at one point on

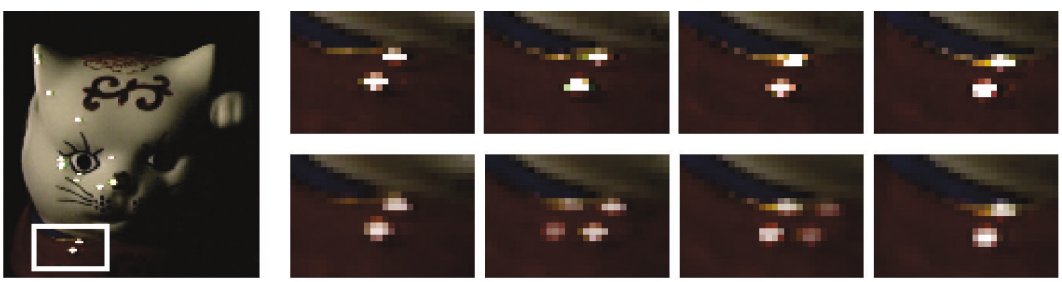


Fig. 10. Zoom ups of the images in Fig. 9.

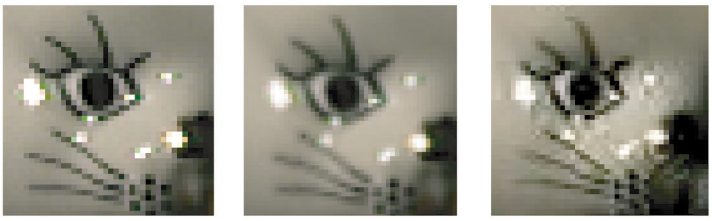


Fig. 11. A comparison with MPEGI compression accomplished on the image.

the surface of an object from the whole illumination environment is a linear combination of the irradiance due to each light source composing the illumination environment. For that reason, a virtual object image under the real illumination environment can be synthesized by decomposing the real illumination environment into several component light sources and then sampling the color images under each component light source separately.

$$I = \int\int \left\{ \sum_{j=1}^{n} L_j(\theta_i, \phi_i) \right\} R(\theta_i, \phi_i, \theta_e, \phi_e) cos\theta_i d\theta_i d\phi_i = \sum_{j=1}^{n} \int\int L_j(\theta_i, \phi_i) R(\theta_i, \phi_i, \theta_e, \phi_e) cos\theta_i d\theta_i d\phi_i, \quad (11)$$

where I: irradiance at one point on the object surface; L_j: radiance of one component light source; R: BRDF of the object surface; (θ_i, ϕ_i): illumination direction; (θ_e, ϕ_e): view direction.

As a preliminary experiment, we took input color images under three point light sources lighting them separately and synthesized images of the virtual object with all lights turned on. 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; we then synthesized an image sequence of the virtual object with all lights on by taking a linear combination of image sequences of the virtual object with each point light source on. Fig. 12 shows the result of the linear combination of a certain pose. In addition, we integrated the virtual object image sequence into a real scene image, which also had been taken under a condition in which all lights were on. The real illumination environment was captured with a stereo setup of cameras with fish-eye lenses [9], and the virtual coordinate system was transformed into the real scene coordinate by calibrating the camera to capture the real scene [24]. The result of the integration, shown in Fig. 13, proves that our method enables the creation of the accurate appearance of the object surface and the precise shadow of the virtual object according to the real illumination environment.

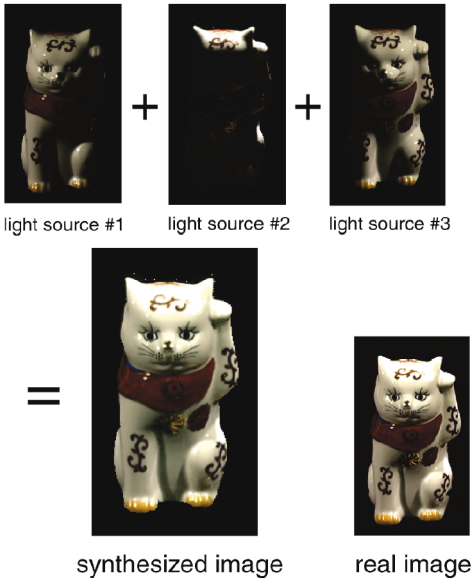


Fig. 12. Linear combination of light sources.

5 Conclusions

We have proposed the *Eigen-Texture method* as a new rendering method for synthesizing virtual images of an object from a sequence of range and color images. The implementation of the method proves its effectiveness, in particular, its high compression ratio. Also, we have demonstrated seamless integration of virtual appearance with real background images by using the method. The merits of the proposed method can be summarized as follows:

First, high compression ratio can be achieved because we compress a sequence of cell images corresponding to a physical patch on the object surface. Appearance variation in the sequence is approximately limited to brightness change due to illumination geometry. Second, interpolation in eigenspace can achieve synthesis of a virtual object image

Fig. 13. Integrating virtual object into real scene.

when object pose is not included in the input image sequence. Owing to this interpolation in eigenspace, we can reduce the necessary number of sampling color images, and reduce the amount of data to be stored. Third, a wide range in application is possible because we do not need any detailed reflectance analysis. The method can be applied to such objects as those with rough surfaces, i.e., the bear and duck shown in Fig. 7, or with strong highlights and whose color values saturate the dynamic range of a CCD camera, such as the cat in Fig. 7. Finally, since an accurate 3D model of the real object is constructed from the input range images, we can generate accurate cast shadows in integrating virtual images with real background images. Thanks to the linearity in image brightness, we can decompose the real illumination distribution into separate component light sources and sample the object under each component light source separately.

On the other hand, our method has a few drawbacks. In particular, the computational expense for compression using eigenspace could be large. As solving the eigenstructure decomposition problem requires a large number of iterations, the computational cost for storing the input image sequence in eigenspace becomes relatively expensive; although, once the input images are compressed and stored in eigenspace, the synthesis of virtual images can be computed in real time. With regard to this point, we believe our method can take advantage especially on applications for interactive mixed reality systems, such as virtual museums and electric shopping malls, where the targets can be compressed beforehand offline.

Currently, we treat each cell image sequence that stacks the appearance change of one triangular patch separately. However, if the cell image sequences can be classified into groups that have the same underlying texture and reflection property, accomplishing compression on the whole set of cell image sequences will achieve a better compression ratio. Constructing the geometric model with regards to the surface texture and reflection property for this purpose remains as future work.

REFERENCES

[1] E.H Adelson and J.R. Bergen, "The Plenoptic Function and the Elements of EarlyVision," *Computational Models of Visual Processing,* M. Landy and J.A. Movshon eds., pp. 3–20, 1991.

[2] P.J. Besl and N.D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 14, no. 2, pp. 239–256, Feb. 1992.

[3] S.E. Chen and L. Williams, "View Interpolation for Image Synthesis," *Proc. ACM SIGGRAPH 93,* pp. 279–288, Aug. 1993.

[4] W.B. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized Voxel Coloring," *Vision Algorithms: Theory and Practice,* B. Triggs, A. Zisserman, and R. Szeliski, ed., 1999.

[5] P. Debevec, "Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography," *Proc. ACM SIGGRAPH '98,* pp. 189–198, July 1998.

[6] P.E. Debevec, C.J. Taylor, and J. Malik, "Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach," *Proc. ACM SIGGRAPH '96,* pp. 11–20, Aug. 1996.

[7] P.E. Debevec, Y. Yu, and G. Borshukov, "Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping,"

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

[9] Y. Sato, I. Sato, and K. Ikeuchi, "Acquiring a Radiance Distribution to Superimpose Virtual Objects onto a Real Scene," *IEEE Trans. Visualization and Computer Graphics,* vol. 5, no. 1, pp. 1–12, Jan.-Mar. 1999.

[10] K.N. Kutulakos and S.M. Seitz, "What Do N Photographs Tell Us about 3D Shape?" Technical Report 680, Computer Science Dept., Univ. Rochester, NY, Jan. 1998.

[11] T. Leung and J. Malik, "Recognizing Surfaces Using Three-Dimensional Textons," *Proc. Int'l Conf. Computer Vision '99,* vol. 2, pp. 1010–1017, 1999.

[12] M. Levoy and P. Hanrahan, "Light Field Rendering," *Proc. ACM SIGGRAPH '96,* pp. 31–42, Aug. 1996.

[13] H. Murase, S. K. Nayar, "Visual Learning and Recognition of 3-D Objects fromAppearance," *Int'l J. Computer Vision,* vol. 14, pp. 5–24, 1995.

[14] K. Nishino, Y. Sato, K. Ikeuchi, "Appearance Compression and Synthesis based on 3D Model for Mixed Reality," *Proc. Seventh Int'l Conf. Computer Vision (ICCV '99),* vol. 1, pp. 38–45, Sept. 1999.

[15] K. Nishino, Y. Sato, and K. Ikeuchi, "Eigen-Texture Method: Appearance Compression Based on 3D Model," *Proc. Conf. Computer Vision and Pattern Recognition '99,* vol. 1, pp. 618–624, June 1999.

[16] K. Pulli, M. Cohen, T. Duchamp, H. Hoppe, L. Shapiro, and W. Stuetzle, "View-Based Rendering: Visualizing Real Objects from Scanned Range and Color Data," *Proc. Eighth Eurographics Workshop Rendering,* June 1997.

[17] K. Sato and S. Inokuchi, "Range-Imaging System Utilizing Nematic Liquid Crystal Mask," *Proc. First Int'l Conf. Computer Vision,* pp. 657–661, 1987.

[18] Y. Sato and K. Ikeuchi, "Temporal-Color Space Analysis of Reflection," *J. Optical Soc. of America,* vol. 11, no. 11, pp. 2990–3002, 1994.

[19] Y. Sato, M.D. Wheeler, and K. Ikeuchi, "Object Shape and Reflectance Modeling from Observation," *Proc. ACM SIGGRAPH '97,* pp. 379–387, Aug. 1997.

[20] S. Seitz and C.R. Dyer, "View Morphing," *Proc. ACM SIGGRAPH '96,* pp. 21–30, Aug. 1996.

[21] S.M. Seitz and C.R. Dyer, "Photorealistic Scene Reconstruction by Voxel Coloring," *Proc. Computer Vision and Pattern Recognition,* pp. 28–34, 1997.

[22] A. Shashua, "Geometry and Photometry in 3D Visual Recognition," PhD thesis, Dept. Brain and Cognitive Science, Massachusetts Inst. Technology, 1992.

[23] P. Suen and G. Healey, "The Analysis and Recognition of Real-World Textures in Three Dimensions," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 5, pp. 491–503, May 2000.

[24] R.Y. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses," *IEEE J. Robotics and Automation,* vol. 3, no. 4, pp. 323–344, Aug. 1987.

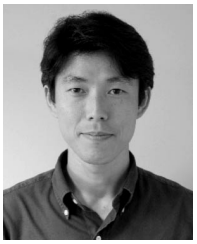
[25] G. Turk and M. Levoy, "Zippered Polygon Meshes from Range Images," *Proc. ACM SIGGRAPH '94,* pp. 311–318, July 1994.

[26] M.D. Wheeler, Y. Sato, and K. Ikeuchi, "Consensus Surfaces for Modeling 3D Objects from Multiple Range Images," *Proc. Sixth Int'l Conf. Computer Vision,* pp. 917–924, 1998.

[27] R.J. Woodham, "Photometric Stereo: A Reflectance Map Technique for Determining Surface Orientation from a Single View," *Image Understanding Systems and Industrial Applications,* vol. 155, pp. 136–143. *Proc. SPIE 22nd Ann. Technical Symp.,* Aug. 1978.

[28] Z. Zhang, "Modeling Geometric Structure and Illumination Variation of a Scene from Real Images," *Proc. Sixth Int'l Conf. Computer Vision,* pp. 1041–1046, July 1998.

Ko Nishino received the BS and MS degrees in information and communication engineering from the University of Tokyo in 1997 and 1999, respectively. Since 1999, he has been a PhD student in the Department of Information Science, Graduate School of Science at the University of Tokyo. His research interests are in the field of computer vision and computer graphics; specifically, in physics-based vision, image-based rendering, and modeling. He is a student member of the IEEE.



Yoichi Sato received the BS degree in mechanical engineering from the University of Tokyo in 1990. He received the MS and PhD degrees in robotics from the School of Computer Science, Carnegie Mellon University, Pittsburg, Pennsylvania, in 1993 and 1997, respectively. In 1997, he joined the Institute of Industrial Science at the University of Tokyo, where he is currently an associate professor. His primary research interests are in the fields of computer vision (physics-based vision, image-based modeling), human-computer interaction (perceptual user interface), and augmented reality. He is a member of the IEEE.

Katsushi Ikeuchi received the BEng degree in mechanical engineering from Kyoto University, Kyoto, Japan, in 1973, and the PhD degree in information engineering from the University of Tokyo, Tokyo, Japan, in 1978. He is a professor at the Institute of Industrial Science, the University of Tokyo, Tokyo, Japan. After working at the Artificial Intelligence Laboratory at Massachusetts Institute of Technology, the Electrotechnical Laboratory of Ministry of International Trade and Industries, and the School of Computer Science, Carnegie Mellon University, he joined the University of Tokyo, in 1996. He has received various research awards, including the David Marr Prize in computational vision in 1990, and the IEEE R&A K.-S. Fu Memorial Best Transaction Paper award in 1998. In addition, in 1992, his paper, "Numerical Shape from Shading and Occluding Boundaries," was selected as one of the most influential papers to have appeared in the *Artificial Intelligence Journal* within the past 10 years. He is a fellow of the IEEE.