Acquiring Bidirectional Texture Function for Image Synthesis of Deformed Objects

Ryo Furukawa
Hiroshima City University
Faculty of Information Sciences
Asaminamiku, Hiroshima, 731-3194 ,Japan
ryo-f@cs.hiroshima-cu.ac.jp

Hiroshi Kawasaki , Katsushi Ikeuchi
Institute of Industrial Science, University of Tokyo
4–6–1 Komaba, Meguro-ku
Tokyo, 153-8505, Japan
h-kawa@sak.iis.u-tokyo.ac.jp, ki@iis.u-tokyo.ac.jp

Abstract

Image-based rendering (IBR) is one of the most promising techniques for photorealistic image synthesis. However, the naive IBR framework has some drawbacks, including restrictions on viewing direction, and difficulties in managing illumination change or object deformation. In this paper, we present a new IBR method capable of composing images of deformable objects from arbitrary view points, and under arbitrary illumination. To do this, we measured and utilized geometric object model and bidirectional texure funcion (BTF). BTF is genaralization of BRDF with spatial variation along object's surface, represented as a set of texture databases which are captured from every viewing angle and light direction. To evaluate the efficiency of this method, we performed several experiments on objects with non-rigid characteristics (e.g., cloth with varying shininess, a 3D object with complicated surface attributes) which are difficult to render correctly by general model-based CG techniques. Compared to previous BRDF based rendering approaches, our work more fully utilizes the 4-dimensional lighting/viewing parameters of BTF, since it is essential for image composition of deformed objects. Further, our implementation sorts the data based on BRDF parameters, resulting in compact data representation.

1. Introduction

Image-based rendering (IBR) techniques have recently become one of the major topics of the computer graphics (CG) and vision research; this is due to IBR's great potential for photorealistic image synthesis with complicated shapes and non-rigid effects (e.g., animal fur, velvet, specular, etc.) whose rendering has been historically difficult. In recent years, a multitude of papers have been published about IBR, describing principles, various kinds of implementation and theoretical analyses.

One of the key concepts used in IBR is the plenoptic function. Originally, a 7D plenoptic function was proposed to define the intensity of light rays passing through the camera center at every location, at every viewing angle, for every wavelength and at any time[1]. "Plenoptic Modeling"[9] uses a continuous 5D plenoptic function, ignoring time and wavelength. In reality, it is still difficult to apply this theory for rendering real world images. Practical simplification and implementation of this approach was presented as "Lumigraph"[7] and "Light Field Rendering"[8], both of which use a 4D plenoptic function with clever parameterization. H.Shum et al.[13] presented 3D plenoptic function which, as the name suggests, creates concentric mosaics.

Although these approaches are very powerful in reproducing real world scenes, they are not without some problems. Among them, (1) difficulty in acquiring, storing and managing huge image data, (2) limitations of rendering conditions (for example, restrictions on viewing direction, rotation of objects, or illumination change), and (3) difficulty in deformation of objects, are significant.

The first problem of huge data stems from the 7 dimensional expression of rays in real world, which amounts to too much information for efficient handling and storage. Roughly speaking, research shown above ([7] [8] [13]) attempt to reduce the huge quantity of data without degrading the quality.

The latter two problems are difficult to avoid when using a naive plenoptic function based approach. Generally, there is a trade-off relationship between the first problem and the others because the approach is essentially "replaying" recorded information of real rays. The more conditions you have to deal with, the more data you need. Obviously, it is impractical to record rays for every possible rendering condition and for every deformation.

A possible solution for realizing arbitrary viewing direc-

tion, object's rotation, or illumination is to use geometric information. View-dependent texture mapping[5, 6] is a kind of IBR, but is more closely related to model-based rendering because it uses geometric information. The surface light field, a term coined by Miller et al. [10], is a function that assigns an RGB value to every ray emanating from every point on a surface. Using this function and geometric data, composing scenes from an arbitrary viewpoint is possible. This method gives not only high data compression rate, but also a solution to the problem of synthesizing the glossy reflection on a surface. Nishino et al.[11, 12] and Wood et al.[16] extend surface light field rendering in an efficient manner. The basic idea of their research is almost the same, but their purpose and data compression algorithms are different. Nishino realizes the synthesis of the object with arbitrary direction of lighting using principal component analysis (PCA) for data compression, while Wood achieves image composition from arbitrary viewpoint and a high compression rate using PCA and a vector quantization. Wood et al. also proposed an editing system and "mimiced" illumination change and object deformation by moving the locations of specular peaks.

Essentially, the surface light field is a subset of the bidirectional reflectance distribution function (BRDF), a 4 dimensional function which represents dependency of reflection on illumination direction [3]. Theoretically, if you have all the information of BRDF for every surface points of a 3D object, you can synthesize the object's image from arbitrary viewpoints under arbitrary illumination with arbitrary deformation. BRDF with 2D spatial variation is called bidirectional texure function, or BTF [4]. BRDF information along the object's surface can be thought of as BTFs mapped onto the surface.

Obtaining BTFs of a 3D object is a challenging problem. Marschner et al.[15] constructed a BRDF map of a human face by modulating the BRDF of skin which is measured in advance. Dana et al.[4] acquired a BTF database of various materials with 4D lighting/viewing parameters, but the materials were flat shaped. For the purpose of image composition, Wood et al.[16] acquired a 2D subset of BRDF with fixed lighting direction, using photographs taken from every direction. Nishino et al.[11, 12] used a 3D subset with arbitrary lighting direction and a viewpoint of 1 degree of freedom (rotation with fixed axis). The photographs were acquired using a turntable. Chen et al.[18] used efficient representation of 2D subset of BRDF and proposed a method for exploiting hardware graphics accelerator for real-time rendering.

This paper represents a new BTF based rendering method capable of composing images of deformable objects from arbitrary viewpoints, and under arbitrary illumination. To realize this goal, we acquire and manage a BTF dataset with 4D lighting/viewing parameterization, because deformation causes changes of both illumination and viewing directions. In the following sections, we describe how we construct an original data structure based on BTF, and propose a synthesis method for our purpose. Then we present several experiments on objects with non-rigid characteristics (e.g., cloth with varying shininess, a 3D object with complicated surface attributes) which are difficult to render correctly by traditional model-based CG techniques.

2. Theory

To synthesize an object whose position and pose change arbitrarily, we have to consider the relative relationship between the light source intensity and direction as well as the surface orientation and view direction. Certainly, if the surface property is completely lambertian, we do not need to concern ourselves with the relationship. However, the actual material usually has non-rigid effects, e.g., animal fur and velvet. Therefore, to synthesize such objects, we use the function (1) for rendering.

$$F_i(\vec{\omega_l}, \vec{o}, \mathbf{R}) \tag{1}$$

Here, $\vec{\omega_l}$ denotes the lth light source direction, $\vec{o}$ the viewing direction in the world coordinates. $\mathbf{R}$ is the rotation matrix which translates world coordinates into local coordinates of the surface (see Fig.1). This function represents the RGB value and is defined at each mesh i. Therefore, when we render the mesh with this function, we select the appropriate texture from a stored texture database(we will explain this database in the next section) and simply texture map it onto the mesh. For actual calculation using this function, we also need two more parameters, d_i and u_l, which represent the weight function of distance between the light source and mesh and light source intensity, respectively. In terms of parameters, we assume that both d_i and u_l are linear functions ; d_i is inversely proportional to distance and u_l is proportion to light intensity. Thus, the actual RGB value can be calculated as follows:

$$\sum_l d_i u_l F_i(\vec{\omega_l}, \vec{o}, \mathbf{R}) \tag{2}$$

To realize function F_i, we have to configure a suitable data structure for actual implementation. F_i has parameter of 7 (= 2 + 2 + 3) degree of freedom, but we can assume that , if the view direction $\vec{o}$, the illumination direction $\vec{\omega_l}$, and the surface are transformed by the same rotation matrix $\mathbf{R}'$, the function F_i remains constant. Thus

$$\begin{aligned} F_i(\vec{\omega_l}, \vec{o}, \mathbf{R}) &= F_i(\mathbf{R}'\vec{\omega_l}, \mathbf{R}'\vec{o}, \mathbf{R}\mathbf{R}'^{-1}) \\ &= F_i(\mathbf{R}^{-1}\vec{\omega_l}, \mathbf{R}^{-1}\vec{o}, \mathbf{E}) \\ &= F_i(\vec{\omega_l'}, \vec{o'}). \end{aligned} \tag{3}$$

F_i in the last line is in reparametrized form omitting **E**, $\vec{\omega_l'}$ and $\vec{o'}$ are local coordinates expression transformed from $\vec{\omega_l}$ and $\vec{o}$. Therefore, the texture database has 4 arguments with the parameters of light source direction (θ_l, ϕ_l) and view direction (θ_w, ϕ_w). This 4D texture database represents the hi-resolution map of BRDF or, more precisely, the BTF (bidirectional texture function). Since we consider not only the 2D texture, but also the 3D object, we have to make a set 4D-parametrized BTF patches for each individual mesh of the surface.

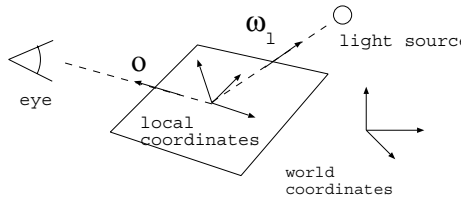


Figure 1. Definition of coordinates and directions for BTF

Once we have database of BTFs rendering objects under arbitrary conditions (i.e. object rotation or illumination direction) and deformation is quite simple. First, we calculate the 4D parameter (θ_l', ϕ_l', θ_w', ϕ_w'), of each geometrical mesh point based on the geometrical data of deformed (or undeformed) objects and rendering conditions. Then select the appropriate texture from the texture database and map it onto the the mesh.

3. Algorithm

To make this 4D-parametrized BTF, we configured the original data acquisition platform shown in Fig.2(a). In this system, the object we want to model is mounted onto the platform and turned by three servo mortors. The rotation angle of each of the mortors corresponds to parameters of Euler angle representation of 3D rotation of the object: α, β and γ (Fig.2(b)). The control of these rotation parametes and chapturing images with CCD cameras are done automatically. With this 3D rotation, we can cover all the viewing directions from objects' surface. In terms of the change of illumination position, the 3D rotation by the platform covers it up to rotation around viewing diretions. The one remaining parameter to represent BTF information is the angle between the viewing direction and the lighting direction, θ (Fig.2(b)).

Changing the parameters $\alpha, \beta, \gamma, \theta$, we capture the images of the object. From the images, we extract the object silhouette by background subtraction (this is quite robust and accurate, because we fix the camera and light source position for capturing process) and project this silhouette onto the voxel space to construct the 3D shape.

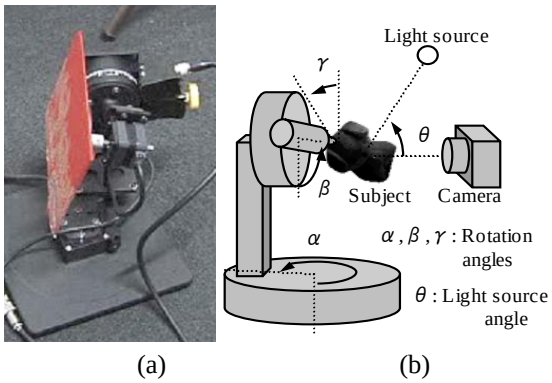


Figure 2. Image capturing platform

Since this 3D shape is made from the image itself, captured images and polygons are accurately aligned from the very beginning. However, our 3D shape acquisition method based on image silhouettes theoretically cannot reconstruct convexity. Also, image- based approaches usually have estimation errors because of noise and shadows in the background. Therefore, we use the 3D shape acquired by the range sensor to construct the texture database. Although the 3D polygons are not yet aligned to the image, we carry out the alignment between these two 3D objects and then project polygons of the 3D object to the images for acquiring the mesh texture. For alignment of the 3D objects, we adopt the Wheeler[14]'s iterative method.

At this stage, all the textures in one texture image originate from one photograph. Since each mesh of the 3D object has its own surface normal direction, the texture parameters (θ_l, ϕ_l, θ_w, ϕ_w) of an image vary. This condition is undesirable for effective texture handling using 4D parameters as a search key.

Now we have a set of texture images, each of which has constant parameters. If we can assume correct alignment, a set of pixels picked at the same positions for all the images originate from the same surface point. To compress texture data, we fit each set of pixels to reflection models, analyze residual data to remove errors or noises, and apply PCA to compress remaining data.

As the reflection function, we used Phong's model[17]. Ignoring ambient light and attenuation of rays, we obtain the following form:

$$I_c = k_{dc}(\vec{n'}, \vec{\omega_l'}) + k_{sc}(\vec{r'}, \vec{\omega_l'})^p \quad (4)$$

$$\vec{r'} = 2\vec{n'}(\vec{n'}, \vec{\omega_l'}) - \vec{\omega_l'} \quad (5)$$

where I_c is observed intensity for color channel c, $(,)$ denotes inner product, k_{dc} and k_{sc} are coefficients for diffuse

and specular reflections for color c respectively. $\vec{n'}$ is the surface normal vector expressed in the local coordinates.

Since we have *a priori* information of parameters for each pixels, we can easily fit pixel values to equation (4) using multivariable regression, obtaining k_{dc} and k_{sc} for each pixel set.

Although the object's surfaces do not always obey the form, (that's the reason why we obtain BTF), the bulk of BTF can be explained by the model in many cases. If the form (4) is not appropriate at all, we can provide several different models, made by shifting directions of $\vec{n}$ or $\vec{r}$.

After the fitting precess, we clamp residual values limiting maximum absolute values, assuming values which exceed certain range as errors. The remaining residuals are compressed with a PCA approrach, treating each texture as a vector, as done in [11, 12]. All the residual textures are expressed as linear combinations of a relatively small number of textures.

Because we already have 4D texture database, the rendering process is quite simple. We translate $\vec{\omega_l}$ and $\vec{o}$ into a local coordinate expression, determine the 4D parameter (θ_l, ϕ_l, θ_w, ϕ_w), and select the appropriate texture from the 4D texture database. When we make the novel texture data, because the 4D database is discrete, we apply the bilinear interpolation from the closest texture images for the actual rendering.

4. Experiments and results

To evaluate our method, we captured images of several types of objects (cloth, can, stuffed animal) using our capturing platform, and made 4D texture data for each of the objects. Then, we deformed the shape of the objects and rendered the deformed objects using our proposed method.

As the first experiment, we spread a Japanese cloth and applied our method. Because this cloth has texture of shininess, BRDF property along the surface varies sharply. Fig. 3 shows the results. We can observe the changes of shininess along the surface very clearly. Furthermore, when we deform the object continuously, the locations of specular lights move accordingly. This can be of much help for us to sense the texture and change of property along the surface.

The second type of objects we modeled were cans with non-uniform, shiny texture. These objects, cans wrapped in handmade Japanese paper, also had complicated reflection property. Unlike 2D objects, alignment between 3D objects and captured images is significantly important in order to render the object with consistent appearance. We synthesized sequence of images of the objects by bending and twisting them. Fig.4 shows the result we obtained. We can see that the appearance with lighting direction is consistent for all surface of the objects. Also, note that the locations of specular moves naturally as the renderd cans are bent or twisted.

Figure 3. Rendered images of cloth with textured shininess

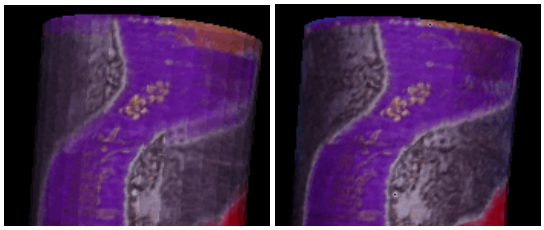
Figure 4. Renderd images of deformed cans

To demonstrate modeling of more complex sheped objects, we used the method to render a staffed dog. Fig.5 shows rendered images. Each of the redered images depicts the object deformed in different ways. Lighting on the doll's surface is correctly represented on the images.

We performed the compression scheme on the texture database of a can described before. The size of BTF patches was 14 $\times$14, ,the object was covered by 1764 patches of mesh, which composes an texture image of 14112 pixels. The number of images used were 72. After fitting to the model of form4, the data of each color channel reduces to 2 images ,14112 pixels each. We performed rendering experiment using these textures appropriated only by the model(Fig6(a)). In the result, seams appeared between textures, which implies that the residual pixels should be recorded to provide good quality of textures.

For error correction and data reduction, we clamped the residuals so that their absolute value is at most 40, and compressed the clamped data using PCA. PCA was performed

Figure 5. Rendering results of a stuffed dog

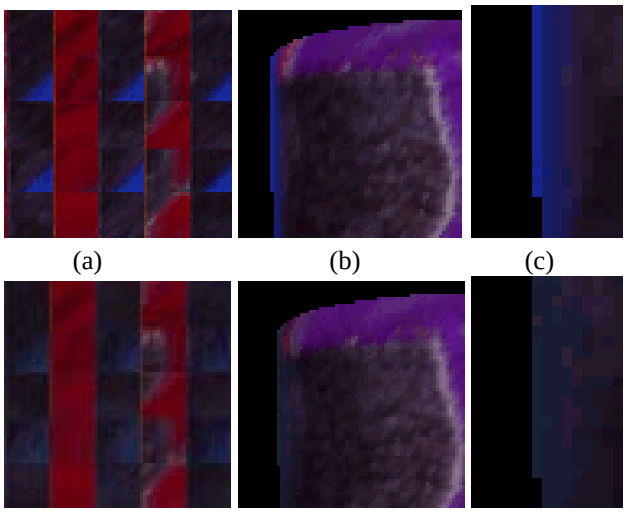


(a) (b)

Figure 6. Redering results using (a) texurtes described only by model fitting, and (b) texurtes described by model and PCA

for each set of patches corresponding to a mesh point so that approximation error for the set becomes less than 1 % of the original signals. The average number of eigenimages for each set of patches was 4.96. Thus the average number of images needed was 6.96. Without clamping values, the average number of images was 8.96. For comparison, compression with only PCA was performed, resulting in 6.95 eigenimages for a set of patches, on the average. The rendering result for this texture is shown in Fig6(b). The quality of the result was almost as good as the result using the original, uncompressed textures. Affect of clamping residuals on rendering quality were very small.

Fig7 show the result. Fig7 (a) and (d) are texture patches before and after the error correction. In Fig7 (a), there were blue pixels originating from the background of the original photo. By error correction, we could remove the blue pixels as in (d). Fig7 (b),(c), Fig7(e) and (f) are rendering results. In Fig7 (b) and (c), we could see that narrow band of blue pixels remained around the contour of the can. In In Fig7 (e) and (f), the band was substantially reduced.



(a) (b) (c)

(d) (e) (f)

Figure 7. Rendering results before error removal are (a) texture, (b) rendering result, and (c) magnified rendering results. Rendering results after error removal are (d) texture, (e) rendering result, and (f) magnified rendering results.

5. Further works

Currently, in our approach, certain patches of texture are impossible to obtain because the capturing platform itself occludes the objects. To alleviate this problem, we are searching for a way to interpolate textures from neighbor textures.

Shadow on the object which disappears after deformation is also a big problem, because we could not capture the actual ray for this situation. One Possible way to this problem is removing shadow from the texture data, since we can synthesize shadow if there's no shadow in the texture.

Also, we are now looking for more efficient ways for representing huge texture database. As a preliminary work, we are experimenting with 6discrete wavelet transformation(DWT). Because BTF is 6 dimensional and the pixels are strongly correlated along each of these axis, we think that 6-dimensional DWT may achieve good compression. Currently, we obtain compression rate of 3.1%, although it was far worse than our expectations.

6. Conclusion

We proposed an extended method for applying IBR techniques to deal with rotation or deformation of objects, and change of illumination direction. To achieve this goal, we proposed and implemented an efficient IBR algorithm and a 4D data structure. The algorithm was realized by function $F_i(\vec{\omega_l'}, \vec{o'})$ which expresses BRDF. Calculating the 4D parameter (θ_l, ϕ_l, θ_w, ϕ_w), we can select appropriate texture image from BTF database.

To implement our method, we made a special facility "light dome" for the data acquisition process. With this dome, we easily captured the sequential image data and 3D model of the object and subsequently generated the texture data sets with 4D lighting/viewing parameters.

To evaluate the effectiveness of our proposed method, we conducted several experiments using various objects. With our proposed algorithm, we successfully rendered the deformed objects with non-rigid surface effects. For future work, we shall pursue applications of this method to CG animation and mixed reality systems.

References

[1] E H Adelson and J Bergen. *The Plenoptic function and the elements of early vision*. MIT Press Cambridge, MA, 1991.

[2] Micheal Ashikhmin, Simon Premoze, and Peter Shirley. A microfacet-based BRDF generator. *ACM SIGGRAPH*, pages 65–74, July 2000.

[3] Shree K Nayar, Katsushi Ikeuchi and Takeo Kanade. Shape from interreflections. In *International Journal of Machine Vision*, 6(3),173-195,1991.

[4] Kristin J Dana, Bram van Ginneken, Shree K Nayar, and Jan J Koenderik. Reflectance and texture of real-world surfaces. In *Computer Vision and Pattern Recognition97*, 1997.

[5] Paul E Debevec, C J Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *ACM SIGGRAPH*, pages 11–20, August 1996.

[6] Paul E Debevec, Yizhou Yu, , and George D Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Eurographics Rendering Workshop*, pages 105–116, June 1998.

[7] S J Gortler, R Grzeszczuk, R Szeliski, and M F Cohen. The lumigraph. *ACM SIGGRAPH*, pages 43–54, 1996.

[8] Marc Levoy and P Hanrahan. Light field rendering. *ACM SIGGRAPH*, pages 31–42, 1996.

[9] L McMillan and G Bishop. Plenoptic modeling: An image-based rendering system. *ACM SIGGRAPH*, pages 39–46, 1995.

[10] Gavin Miller, Steven Rubin, and Dulce Ponceleon. Lazy decompression of surface light fields for precomputed global illumination. In *Rendering Techniques(Eurographics Proceedings)*, pages 281–292, June 1998.

[11] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. Appearance compression and synthesis based on 3D model for mixed reality. In *Proc. of SeventhInternational Conference on Computer Vision*, volume 1, pages 38–45, September 1999.

[12] Ko Nishino, Yoichi Sato, and Katsushi Ikeuchi. Eigentexture method: Appearance compression based on 3D model. In *Computer Vision and Pattern Recognition*, volume 1, pages 618–624, June 1999.

[13] Heung-Yeung Shum and Li-Wei-He. Rendering with concentric mosaics. *ACM SIGGRAPH*, pages 299–306, 1999.

[14] Mark D. Wheeler and Katsushi Ikeuchi. Sensor modeling, probabilistic hypothesis generation, and robust localization for object recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence97*, 17(3):252–265, 1995.

[15] Stephen R. Marschner, Brian Guenter, and Sashi Raghupathy Modeling and Rendering for Realistic Facial Animation *11th Eurographics Rendering Workshop*, 2000.

[16] Daniel Wood, Daniel Azuma, Wyvern Aldinger, Brian Curless, Tom Duchamp, David Salesin, and Werner Steutzle. Surface light fields for 3d photography. *ACM SIGGRAPH*, July 2000.

[17] Bui-Tuong Phong, Illumination for Computer Generated Pictures. *CACM*, 18(6),311-317, June 1975. Also in *BEAT82*,449-455

[18] Wei-Chao Chen, Radek Grzeszczuk, Jean-Yves Bouguet Light Field Mapping: Hardware-Accelerated Visualization of Surface Light Fields. *ACM SIGGRAPH, Course*, 2001.