Illumination Brush: Interactive Design of All-frequency Lighting

Makoto Okabe* Yasuyuki Matsushita† Li Shen† Takeo Igarashi*,‡

*The University of Tokyo †Microsoft Research Asia ‡SORST, JST

Abstract

We present an appearance-based user interface for artists to efficiently design customized image-based lighting environments. [1] *Our approach avoids typical iterations of parameter editing, rendering, and confirmation by providing a set of intuitive user interfaces for directly specifying the desired appearance of the model in the scene. Then the system automatically creates the lighting environment by solving the inverse shading problem. To obtain a realistic image, all-frequency lighting is used with a spherical radial basis function (SRBF) representation. Rendering is performed using precomputed radiance transfer (PRT) to achieve a responsive speed. User experiments demonstrated the effectiveness of the proposed system compared to a previous approach.*

1. Introduction

Image-based lighting, or the environment map, is a method of representing a large-scale lighting environment around a target scene as a texture map [5]. It compactly represents complicated incoming light from distant sources and enables real-time rendering of the scene with realistic lighting effects [28]. However, most systems rely on captured environments for image-based lighting [6, 10], and few attempts have been made to manually design such complex lighting environments. The artist might paint the environment map directly, but this is labor-intensive and makes it very difficult to obtain the desired rendering result due to the non-intuitive relation between lighting conditions and the appearance of objects. Because a captured environment is not always available, a practical method for manually designing complicated lighting environments is in great demand.

We propose an appearance-based user interface for designing image-based lighting environments. Instead of placing lights in the surrounding environment, users can directly specify the appearance of the resulting image by painting and dragging the color of outgoing radiance on the target model. Then the system constructs an appropriate image-based lighting model by solving the inverse lighting problem. Other appearance-based interfaces for lighting design have been proposed [19, 24, 20, 12, 14, 1, 26, 2, 18], but all of them are limited to simple lighting models, such as point or directional lights, and are not designed for image-based lighting, which can produce more appealing results than simple lighting. It is particularly useful for adding realistic lighting effects to synthetic objects when compositing them into a photographed or video-recorded background. Our system also supports the design of high-dynamic range (HDR) lighting on a standard low-dynamic range (LDR) monitor by introducing interactive tone-mapping, in which users specify a region of interest (ROI) and the system automatically adjusts the tone-mapping parameter.

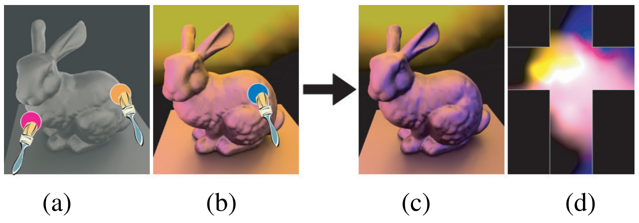


Figure 1. Designing image-based lighting with *Illumination Brushes*. The bunny model has a measured white glossy BRDF.(a) The user paints the desired diffuse appearance directly on the 3D model. Pink and orange diffuse brushes are shown. (b) The scene is rendered using the estimated environment map. The user paints a blue highlight on the bunny with specular brush. (c) All of the painted lighting effects are satisfied by rendering the bunny using the designed image-based lighting environment shown in (d).

Given the surface appearance specified by the user, the system must estimate the corresponding image-based lighting model. This process is formalized as an inverse lighting problem, that is, recovering unknown lighting from known geometry, outgoing radiance, and the surface bidirectional reflectance distribution function (BRDF) [21]. To solve this problem at a responsive speed, we represent an image-based lighting model using the spherical radial basis functions (SRBF) [29], and estimate coefficients in a precomputed

[1] This work was done while the first author was visiting Microsoft Research Asia.

radiance transfer (PRT) framework [27, 28]. The SRBF representation allows our system to represent all-frequency lighting effects and arbitrary BRDFs, which can be either analytic or measured. The system allows interactive view-changes and quick update of the lighting environment according to user input.

Our current implementation supports dynamic editing of HDR image-based lighting for a fixed geometry with arbitrary BRDF. It can handle both specular and diffuse lighting effects with appropriate treatment of self-occlusions (soft shadows). More advanced effects, such as texture mapping, interreflections, and deformation, will be addressed in future work.

2 Prior Work

Several methods have been proposed to manipulate lighting effects in a rendered 3D scene, as an alternative to manually setting raw lighting parameters. Pellacini *et al.* proposed using cast shadows for lighting design [19]. Some other methods use sketching or painting interfaces that allow users to directly draw lighting effects [24, 20, 12, 2, 26, 18]. While these methods are effective, they are limited to simple local lighting models, or simple geometry. Other related works have incorporated illumination control using psychophysical effects [14], and interactive control of highlighted shapes in cartoon animations [1]. Automated lighting design systems are also proposed [25, 9].

Our system utilizes inverse rendering, estimating the distribution of light sources given the scene geometry, surface properties, and the final image. Previous methods have estimated light source distributions from a variety of different input types, such as cast shadows [23] and shading [30]. Li *et al.* used integrated cues of shading, shadows, and specular reflections [15]. Nishino and Nayar proposed a method for capturing an environment map reflected by a human eye in an image, and using it for relighting [17]. Ramamoorthi and Hanrahan proposed methods for estimating either a material BRDF, an illumination condition, or both from a single image or multiple images of a scene [21]. Our work builds on these previous results and provides a practical method suitable for interactive editing without excessive precomputation.

3 User Interface

This section describes our prototype system for an appearance-based lighting design from the artist's point of view. The system allows users to directly specify the appearance of the final image by painting the outgoing radiance on the target model. The system also provides dragging tools to modify the lighting environment. The lighting environment and screen image are updated in real time. Users can freely change the viewpoint during the lighting design process. All light sources are assumed to be distant, and the painting process changes the positions, colors, intensities, and shapes of light sources. The system assumes a known geometry and a predefined BRDF for each point on the surface of the target model. Figure 2 shows a snapshot of the prototype system. Given a new 3D scene, the user must first run several minutes of precomputation before starting an interactive lighting design session.

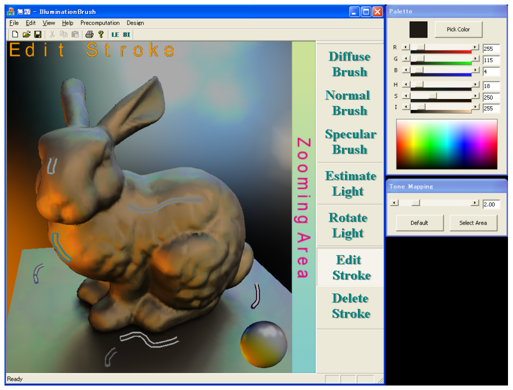


Figure 2. Our prototype system.

3.1 Illumination Brushes

Illumination brushes are painting tools for specifying the desired appearance of a surface. Users draw a stroke on the screen using a dragging operation. The stroke is projected onto the object surface and serves as a constraint that specifies the color of nearby mesh vertices to estimate the lighting environment. Color and position can also be changed later in the process. The system provides two types of illumination brushes: the *diffuse brush*, for designing diffuse lighting effects, and the *specular brush*, for editing specular lighting effects. An example is shown in Figure 3. Note that specular highlighting is view-dependent and moves along the surface as a user changes the viewpoint, while the diffuse component is independent of viewing direction.

For both brushes, users simply select a color and begin painting on the 3D model. The system immediately and continuously estimates the corresponding illumination and updates the image on the screen during the painting process.

However, user input can be inconsistent. Figure 4 shows two such examples. In (a), the user paints white and black lighting effects at different positions with the same surface normal. Because no lighting environment can exactly accommodate this input, the system tries to satisfy the contradictory constraints as much as possible in a least squares sense. In this example, Figure 4 (b) shows that the painted positions will have the same gray color, which is the average of the painted lighting effects. In (c), the user paints white and black lighting effects at different positions with the same surface normal. Such input seems to be inconsistent, however, because the area painted in black can be

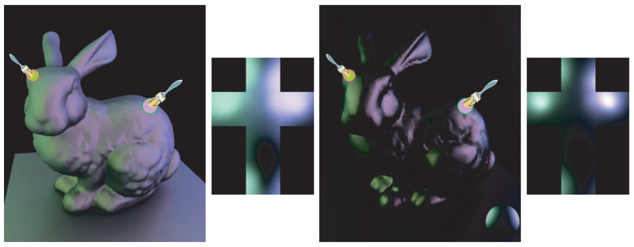

(a) (b)

Figure 3. Painting with illumination brushes. The bunny model has a glossy aluminum-bronze BRDF. (a) A user paints green and purple lighting effects on the 3D model using *diffuse brushes*. The resulting lighting environment (shown as a cross) is diffused and blurry, which is reflected in the rendered image. (b) The same colors are painted at the same positions using *specular brushes*, which results in a lighting environment with sharper features and specular lighting effects.

in the statue's shadow, the input is consistent and thus the system can compute the lighting environment accordingly, which results in a soft shadow under the statue (d).

3.2 Manipulation of Painted Strokes

The result of a painting operation is stored as a stroke on the object surface, whose color and position can then be modified. To change the color of a stroke, the user simply clicks it and adjusts its color using the color panel. To change the location of the stroke, the user clicks and drags the stroke on the object surface. As the dragged stroke moves, it adapts its shape to match the surface (Figure 5). These operations are useful for locally adjusting the appearance of the scene. Similar operations are used in [11, 3].

If the viewpoint is changed, the position of a stroke no longer corresponds to the created highlight. To help users modify strokes in a consistent manner, the system automatically recovers the original viewpoint each time a stroke is clicked.

3.3 Interactive Tone-Mapping

A standard monitor can display only LDR images, where color is represented as 8 bits per channel. When designing a lighting environment, however, editing in such a narrow range is a severe limitation. To enable users design in a HDR domain, we provide an interactive tone-mapping interface. Users specify the ROI using a rubber band on the screen, as shown in Figure 6, and the system automatically adjusts the exposure so that the contrast in the ROI is maximized. The manipulated color is always converted into the HDR domain using the exposure settings. Users can also change the exposure directly by controlling a slider.

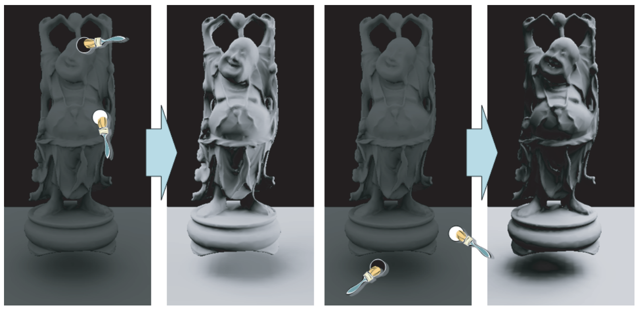

(a) (b) (c) (d)

Figure 4. Painting with inconsistent multiple colors. The Buddha model has a diffuse BRDF. (a) The user paints white and black lighting effects at different positions with the same normal, resulting in a uniform gray color (b). In (c), the user repeats this inconsistent input, but at a different location on the 3D model. Because the region painted black is identified as the statue's shadow, the input becomes consistent, resulting in a soft shadow under the statue (d).

3.4 Rotation of Lighting Environment

During the light-editing process, users may want to rotate the entire lighting environment. We provide an appearance-based tool to achieve this with an intuitive dragging operation on the scene surface. Users can grab any feature, such as highlights and shadows on the surface, and move it to another location (Figure 7). Internally, the system rotates the entire lighting environment so that the color under the mouse cursor remains constant while dragging.

Users can directly rotate the light sphere representing the entire lighting environment, but dragging inside the scene (e.g., dragging cast shadows around the scene) can be more convenient in some cases. Figure 8 illustrates an example, where identical dragging operations result in opposite rotations depending on the scene geometry and lighting conditions.

4 Algorithm

In this section, we describe the algorithms for the interactive rendering and user interfaces. To develop the interactive rendering algorithm, we used the SRBF-based PRT framework proposed by Tsai *et al.* [29], for several reasons. First, the rendering speed is fast, allowing users to preview rendering results in real time. Second, unlike spherical harmonics or wavelet-based systems, the basis function is in the spatial domain rather than the frequency domain. This makes the estimation algorithm simple and stable. Third, SRBFs can be added progressively to locally control the illumination resolution.

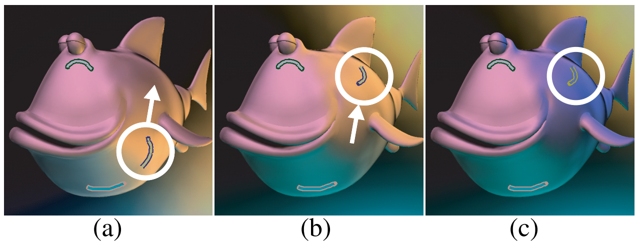


Figure 5. Manipulation of painted strokes. (a) The user selects the orange painted stroke at the top of the fish model. (b) The user drags the stroke to a different location, and the environment lighting is updated accordingly. (c) The color of the stroke is modified from orange to blue.

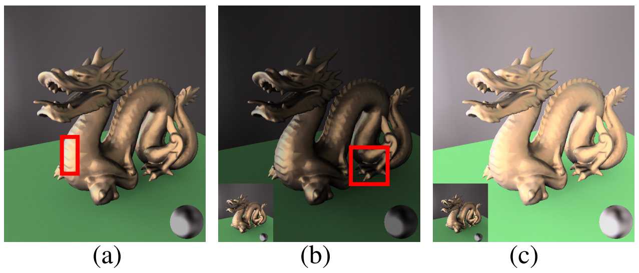


Figure 6. The system allows scene editing with tone mapping. (a) The ROI is selected using a red rubber band. Because the intensity distribution in the ROI is relatively bright, the system adjusts the exposure to make the entire scene darker (b). The bottom-left image of (b) shows the original exposure. The dark area is selected with the red rubber band, and the system makes the entire scene brighter (c).

The rendering algorithm is based on the previous PRT framework [29]: the system renders the scene in real time by fixing the geometry and BRDF while the user interactively changes the view and lighting. The major problem with the previous method is that the precomputation takes a prohibitively long time before start painting a new 3D scene (e.g., up to 24 h for a 3D scene with 50,000 vertices). Therefore, our system approximates the result using a simplified method to reduce precomputation time. First, instead of accurately modeling the shading of a triangle in the spirit of Phong's model, we simply interpolate the colors of the mesh vertices to render a triangle. Second, we use a simplified method to approximate a function with SRBF. Instead of solving a global least-squares system to compute the SRBF coefficients, we determine the coefficients of each SRBF independently assuming that each SRBF component is relatively independent. These changes shorten the precomputation time from 1 day to several minutes. It also slightly reduces rendering accuracy and speed (it takes 1 second to render a scene with 10,000 vertices when the view is changed), but remains sufficiently fast for a preview during interactive design.

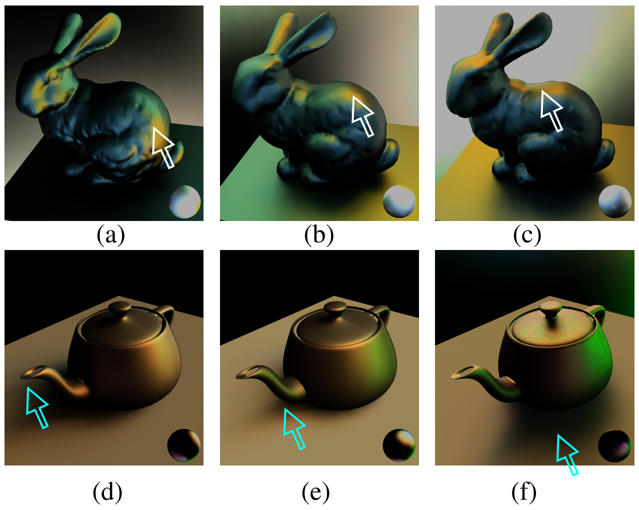


Figure 7. The user clicks and drags the white-lit part to rotate the lighting environment (a-c). The lighting environment rotates smoothly so that the surface colors under the mouse cursor remain constant. The user can also click and drag the shadow under the teapot (d-f).

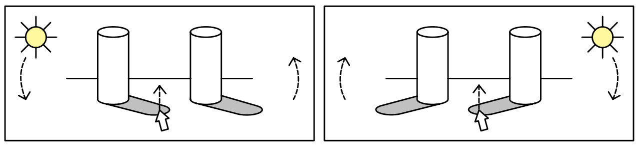


Figure 8. The same dragging operations can result in opposite rotations, depending on the geometry and lighting conditions.

4.1 Interactive Rendering

Representation For distant illumination, the outgoing radiance I from a surface point x is described by the rendering Equation [13] as follows:

$$I(x,\omega_o)=\int_{\Omega_i} L(\omega_i)R(x,\omega_i,\omega_o)V(x,\omega_i)\max(0,\omega_i\cdot\mathbf{n})d\omega_i, \quad (1)$$

where ω_i and ω_o are the light and view directions, respectively; L is the incoming ray from the environment lighting; R is the BRDF; V is the visibility map (which takes $[0|1]$); and $\mathbf{n}$ is the surface normal at x. We rewrite the Equation (1) as follows:

$$I(x,\omega_o) \quad = \quad \int_{\Omega_i} L(\omega_i)T(x,\omega_i,\omega_o)d\omega_i, \quad (2)$$

$$T(x,\omega_i,\omega_o) \overset{\text{def}}{=} R(x,\omega_i,\omega_o)V(x,\omega_i)\max(0,\omega_i\cdot\mathbf{n}), \quad (3)$$

If the viewpoint, ω_o, is fixed, then $T(x,\omega_i,\omega_o) = T_x(\omega_i)$ and $I(x,\omega_o) = I_x$. The subscript x indicates that the quantity is associated with the vertex x.

We represent the light transfer function, $T_x(\omega_i)$, and the distant illumination, $L(\omega_i)$, with SRBF expansions. The SRBF centers of light transfer function, $\Xi_T = \{\xi_{T,1},...,\xi_{T,n_t}\}$, and the SRBF centers of light sources, $\Xi_L = \{\xi_{L,1},...,\xi_{L,n_l}\}$, are the sets of distinct points on the unit sphere, S^2. The associated bandwidth parameters $\Lambda_T = \{\lambda_{T,1},...,\lambda_{T,n_t}\}$ and $\Lambda_L = \{\lambda_{L,1},...,\lambda_{L,n_l}\}$ are defined by sets of real numbers. $T_x(\omega_i)$ and $L(\omega_i)$ can be represented by the SRBF function G and coefficients as

$$T_x(\omega_i) \approx \sum_{i=1}^{n_t} t_{x,i}G(\omega_i,\xi_{T,i};\lambda_{T,i}), \quad (4)$$

$$L(\omega_i) \approx \sum_{j=1}^{n_l} l_j G(\omega_i,\xi_{L,j};\lambda_{L,j}), \quad (5)$$

where $G(\cdot,\xi;\lambda)$ is an SRBF centered at ξ with the bandwidth parameter λ. In the following sentences, *T-SRBF* is described by $G(\omega_i,\xi_{T,i};\lambda_{T,i})$, and *L-SRBF* is described by $G(\omega_i,\xi_{L,j};\lambda_{L,j})$. By combining Equations (2–5), we obtain the following matrix notation:

$$I_x = \mathbf{T}_x^T\mathbf{A}\mathbf{L}, \quad (6)$$

where $\mathbf{T}_x = [t_{x,1}\ t_{x,2}\ \ldots\ t_{x,n_t}]$ is a light transfer coefficient vector, $\mathbf{L} = [l_1\ l_2\ \ldots\ l_{n_l}]$ is a light source coefficient vector, and the (m,n) component of the matrix $\mathbf{A}$ is given by

$$\mathbf{A}_{mn} = \int_{\Omega_i} G(\omega_i,\xi_{T,m};\lambda_{T,m})G(\omega_i,\xi_{L,n};\lambda_{L,n})d\omega_i. \quad (7)$$

Equation (7) is called *spherical singular integral*, and it can be efficiently evaluated for Gaussian SRBF as described in [29].

Precomputation To make the initial set of T-SRBF centers, Ξ_T, we use an icosahedron subdivided three times to provide 642 vertices, and project them on the unit sphere S^2. Λ_T is assigned the same value based on the relationship between the variance and the bandwidth parameters for the Gaussian SRBFs: $\sigma^2 = 1-(coth(2\lambda)-(2\lambda)^{-1})$ [16]. We fixed the variance at $\pi/40$ radians to derive Λ_T. It is possible to increase the number of T-SRBF centers to increase the accuracy.

Several minutes of precomputation of the visibility map $V(x,\omega_i)$ are required for each new 3D scene. The visibility map of each vertex is obtained by setting the camera at the vertex and rendering the scene into a cube map with flat shading. To avoid aliasing, we first render the cube map at a resolution of $6\times128\times128$, and then downsample it to $6\times32\times32$.

Rendering Our system renders the scene by computing the right-hand side of Equation (6). We have implemented our system on a platform that includes a GeForce 8800 GPU and an Intel Core2 2.4-GHz CPU. Computation of $\mathbf{T}_x$ and multiplication of $\mathbf{AL}$ is performed using the CPU, and other operations take place on the GPU. When the user changes viewpoints, the system computes $\mathbf{T}_x$ for all visible vertices. Computation of $\mathbf{T}_x$ requires access to the visibility map of x and BRDF data. On average, this process takes about 1 second for 10,000 vertices.

Our system allows the user to change the lighting in real-time (20-60 FPS), but view changing runs at an interactive rate (several seconds) that is proportional to the number of vertices. For this reason, our system uses a simplified model during camera motion.

4.2 Estimation of Lighting Environment

We initialize the lighting environment with uniformly distributed L-SRBFs obtained by subdividing the icosahedron several times, usually twice to provide 162 L-SRBFs. We describe a method to estimate the light source coefficients from diffuse and specular brushes. We also describe a method of adaptively controlling the number of L-SRBFs.

Basic Formulas Let $\mathbf{T}'_x = (\mathbf{T}_x^T\mathbf{A})^T$. Then Equation (6) can be written as

$$I_x = \mathbf{T}'^T_x\mathbf{L}. \quad (8)$$

Given a set of the user-painted colors, $I' = \{I'_1,...,I'_{n_p}\}$, on the painted vertices, $P = \{p_1,...,p_{n_p}\}$, the system estimates a new lighting environment such that rendered vertex colors are as close as possible to the user-specified colors. The simple form of the estimation problem can be written in a least squares form as

$$\hat{\mathbf{L}} = \operatorname*{argmin}_{\mathbf{L}} \sum_{i=1}^{n_p} (I'_i - \mathbf{T}'^T_{p_i}\mathbf{L})^2, \quad (9)$$

where $\hat{\mathbf{L}} = [\hat{l}_1\ \hat{l}_2\ \ldots\ \hat{l}_{n_l}]^T$ is the estimated light source coefficient vector. To derive a unique solution in the under-constrained case, we add a regularization term. Equation (9) can then be written as

$$\hat{\mathbf{L}} = \operatorname*{argmin}_{\mathbf{L}} \sum_{i=1}^{n_p} (I'_i - \mathbf{T}'^T_{p_i}\mathbf{L})^2 + \kappa\mathbf{L}^2, \quad (10)$$

where κ, the regularization factor that penalizes models to prevent overfitting, is empirically set to $1.0\text{e}-4$.

Diffuse and Specular Brushes As we have described in the user interface section, using the diffuse brush changes the lighting of the overall environment while the specular brush changes the lighting only in the direction of specular reflection. Figure 9 shows our design philosophy for each brush. We obtain these effects by assigning a different weight to each light source. That is, when the specular brush is used, the system assigns smaller weights to the light sources located in the direction of specular reflection so that these light sources are mainly modified. On the other hand,

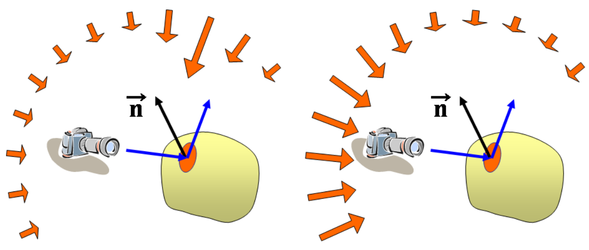


Figure 9. The user is painting an orange color on a glossy surface point. (a) The specular brush produces effects for light sources located in the direction of specular reflection. This results in a specular lighting effect. (b) The diffuse brush produces effects for light sources located off the axis of specular reflection. This results in a diffuse lighting effect.

when the diffuse brush is used, the system assigns smaller weights to the light sources off the direction of specular reflection. To incoporate these features in the process, we change Equation (10) to

$$\hat{\mathbf{L}} = \underset{\mathbf{L}}{\operatorname{argmin}} \sum_{i=1}^{n_p} (I'_i - \mathbf{T'}^T_{p_i}\mathbf{L})^2 + \kappa(diag(\mathbf{W})\mathbf{L})^2. \quad (11)$$

where $\mathbf{W} = [w_1\ w_2\ \dots\ w_{n_l}]^T$ are the weighting factors controlling the variability of the light sources. Smaller values of w_i means larger changes in l_i, and vice versa.

To determine components of $\mathbf{W}$, we use the light transfer coefficient vector, $\mathbf{T}'_x = [t'_{x,1}\ t'_{x,2}\ \dots\ t'_{x,n_l}]$, that represents which light source coefficient is important in determining $\mathbf{I}_x$ (see Equation (8)). The weighting factors $\mathbf{W}_x = [w_{x,1}\ w_{x,2}\ \dots\ w_{x,n_l}]^T$ at painted vertex x are determined as follows: if the vertex x is painted with the diffuse brush, the weighting factors are given by $w_{x,i} \propto t'_{x,i}/\max_j\{t'_{x,j}\}$, and if the vertex x is painted with the specular brush, the weighting factors are given by $w_{x,i} \propto 1 - t'_{x,i}/\max_j\{t'_{x,j}\}$. We assign weighting factors of zero to light sources that are invisible at the surface point x. Finally, $\mathbf{W}$ is computed as $\mathbf{W} = \sum \mathbf{W}_{p_i}$ for all painted vertices, p_i. To avoid a zero weight factor, we add $\varepsilon = 2.0\mathrm{e}-4$ to each element of $\mathbf{W}$.

Solving the Linear System To avoid negative lighting intensity, the estimated lighting $\hat{\mathbf{L}}$ must satisfy the following condition:

$$\hat{l}_i \geq 0 \qquad \forall i \in [1, n_l]. \quad (12)$$

This gives us a least square problem with inequality constraints. We use the Active Set method [7] on *UMFPACK Version 4.1* [4] to solve the problem iteratively.

Adaptive Control of Number of L-SRBFs The original environment map is represented by uniformly distributed L-SRBFs. As mentioned earlier, there are 162 L-SRBFs, as determined by the number of vertices in an icosahedron subdivided twice. While the original number of L-SRBFs is limited, the resolution can be locally enhanced by adding more. When the user provides new input to the system using the specular brush, an L-SRBF is added in the direction of the specular reflection. This mitigates the low-sampling problem of the illumination and ensures that the light source in the highlight direction always exists when edited.

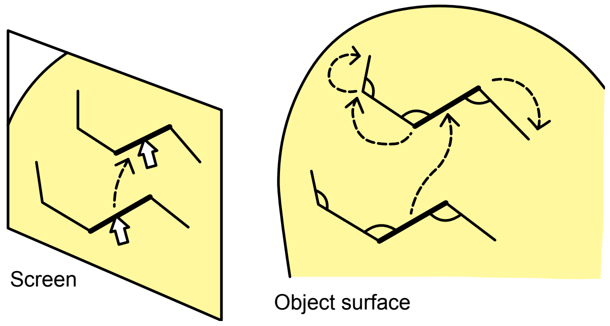


Figure 10. Dragging a stroke on the object surface. The system moves the grabbed edge and then determines the locations of the surrounding edges.

4.3 Dragging Painted Strokes

When the user grabs and drags a stroke, the system moves the grabbed edge on the surface and then determines the location of the neighboring vertices one by one so that the local positional relationships among neighboring vertices are maintained (Figure 10). This process is viewed as a walking operation on the surface: given the original location and the target location of an edge (v_i, v_{i+1}), the target location of a neighboring vertex v_{i+2} is determined so that both the angle between edges (v_i, v_{i+1}) and (v_{i+1}, v_{i+2}) projected onto the tangent plane at v_{i+1} and the geodesic distance between v_{i+1} and v_{i+2} remain constant. The location of the dragged edge is determined so that it follows the mouse cursor and its orientation on the screen remains constant.

4.4 Rotation of Environment Lighting

The lighting environment can be revolved by rotating the centers of the L-SRBFs (ξ). When users perform a dragging operation using the rotation tool, the system automatically updates the rotation angles so that the radiance under the mouse cursor remains the same. To be more precise, the computation proceeds as follows (Figure 11). Suppose that the mouse cursor moves from the screen coordinate p_0 to p_1. The corresponding surface vertices are v_0 and v_1. The system estimates the new orientation minimizing the difference between the original radiance I_{v_0} at v_0 and the new

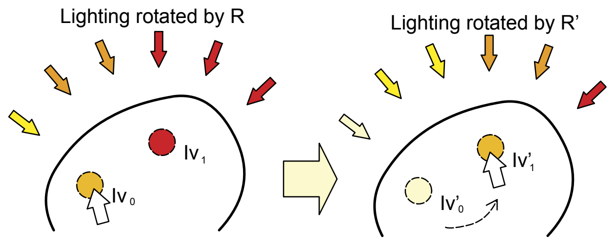


Figure 11. Rotation of light source using a dragging operation. The system computes R so that the resulting radiance at v_1 is identical to the previous radiance at v_0.

surface radiance I'_{v_1} at v_1 after rotation.

We solve this problem using a standard minimization method. The rotation matrix R is represented by three rotation angles: α, β, and γ. The system uses the smallest changes to the rotation angles ($d\alpha$, $d\beta$, and $d\gamma$) that satisfy $I'_{v_1} = I_{v_0}$. To achieve this, the system solves the following minimization problem using the Lagrange multiplier method:

$$\min\{d\alpha^2 + d\beta^2 + d\gamma^2\} \quad \text{s.t.} \quad dI_{v_1} = \frac{\partial I_{v_1}}{\partial \mathbf{J}} dJ, \tag{13}$$

where $\mathbf{J} = [\alpha\ \beta\ \gamma]^T$, and dI_{v_1} is the target differential scalar of the radiance at $v_1(I_{v_0} - I_{v_1})$. The system continuously solves this equation in the background process, and updates $\mathbf{J}$ using a small gradient step $\varepsilon_{\mathbf{J}} = 0.05$.

4.5 High-dynamic Range Support

To achieve the operation described in Section 3.3, we use Reinhard *et al.* 's global tone-mapping operator [22], which can be efficiently implemented using programmable graphics hardware [8]. The operator calculates the log average luminance $\bar{L}_w$ as

$$\bar{L}_w = \exp\left(\frac{1}{N}\sum_{x,y}\log(\delta + L_w(x,y))\right), \tag{14}$$

where N is the number of pixels in the image, (x,y) represents the pixel coordinate, and δ is a small constant used to avoid numerical underflow when $L_w(x,y) = 0$. The source luminance values $(L_w(x,y))$ are mapped to the relative luminance $(L_r(x,y))$ by

$$L_r(x,y) = \frac{a}{\bar{L}_w} L_w(x,y), \tag{15}$$

where a is a key value that controls whether the tone-mapping image appears relatively bright or relatively dark. Then $L_r(x,y)$ is mapped to the display luminance $L_d(x,y)$ by

$$L_d(x,y) = \frac{L_r(x,y)}{1 + L_r(x,y)}. \tag{16}$$

Finally, the RGB display values R_d, G_d, and B_d are determined by

$$R_d = L_d\left(\frac{R_w}{L_w}\right)^b, \; G_d = L_d\left(\frac{G_w}{L_w}\right)^b, \; B_d = L_d\left(\frac{B_w}{L_w}\right)^b. \tag{17}$$

	Buddha	Bunny	Dragon	Fish
Vertices	23,884	17,483	23,960	7,634
Precomputation (sec)	398	292	400	127
Viewpoint (msec)	2,136	1,575	2,166	696
162 L-SRBFs (FPS)	56	61	56	62
642 L-SRBFs (FPS)	26	27	26	30

Table 1. Time and rendering performance of the proposed approach. The run-time FPS in the last two rows lists the rendering performance for lighting environment changes with either 162 or 642 L-SRBFs.

In our implementation, a and $\bar{L}_w$ are initially set to 2.0 and 1.0, respectively. When the user selects the ROI, the system recalculates the log average luminance $\bar{L}_w$ to update the RGB values. The user can also change a by directly manipulating the slider bar to control the brightness distribution in the image. We use a fixed value of 0.6 for b. The LDR color values used can easily be converted to HDR values R_w, G_w, and B_w by solving Equation (17), and the system stores the HDR representation internally.

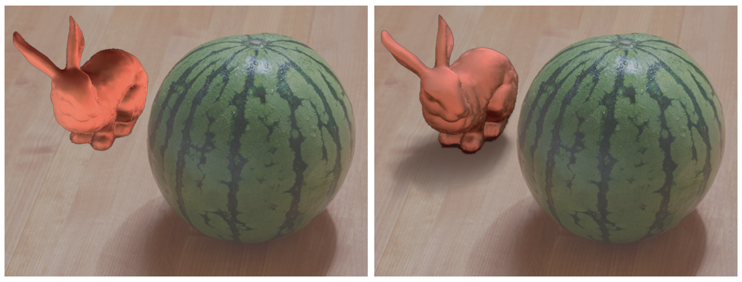
5 Result

For all of the glossy objects illustrated, we used both measured and synthetic BRDFs. The reflectance properties are unrelated to the numerical performance or stability in the computation. However, more user input is needed when an object's reflectance contains high frequency component.

Table 1 shows the time and rendering performance of the proposed approach. It takes a few minutes for precomputation, a few seconds to change the viewpoint, and less than 40 msec for the lighting change. The system estimates the illumination within a few seconds (e.g., 1-5 seconds for 162 L-SRBFs). Optimization speed depends not on the size of the 3D model but on the numbers of L-SRBFs and iterations of the Active Set method.

3D Object Insertion The system is also useful for adjusting photometric consistency when inserting a synthetic 3D object into a photograph. For example, users can load and display a background image behind the 3D object, and then paint and adjust the lighting effects of the 3D object so that it seamlessly matches the lighting conditions of the background photograph. Figure 12 and the accompanying video show the results of inserting a synthetic 3D object into a photograph using the system. Each 3D object has a measured glossy BRDF, and shadows under the 3D object are inserted using Adobe Photoshop.

Editing imported environment map The more reflective an object surface is, the greater is the number strokes that must be drawn on the object to design the lighting environment map. This is a fundamental limitation of any inverse-lighting algorithm. One reasonable solution is to edit an im-


(a) Diffuse:14 / Specular:0

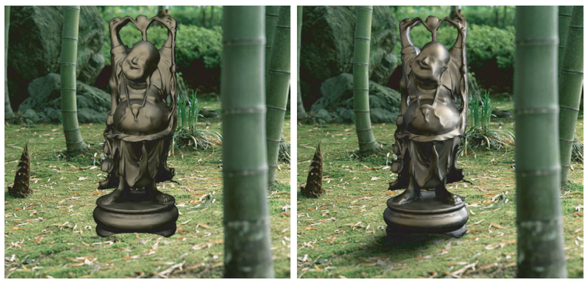
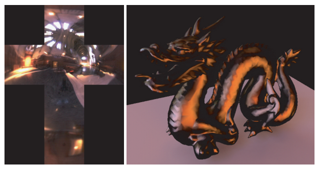
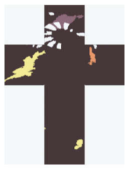
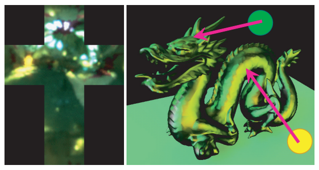

(b) Diffuse:12 / Specular:2

Figure 12. Adjusting the photometric effects of a superimposed 3D model with respect to the background image. The two images shown in (a) are the result of inserting a 3D bunny with pink plastic BRDF. The left image has an arbitrary illumination and the image on the right is rendered using an image-based lighting environment designed by our system. (b) The result of inserting a 3D budda with white glossy material.


(a) Original


(b) Mask Image


(c) Modified #1

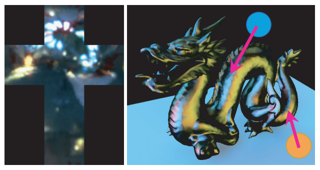

(d) Modified #2

Figure 13. Editing an existing environment map. (a) Original lighting environment and rendered scene. (b) Mask Image. (c-d) Modified versions. Painted colors are also shown.

ported environment map. This allows users to input a sparse collection of strokes and still retain the spatial structure of the original environment.

The proposed system can also be extended for this purpose as shown in Figure 13. This extension takes two images as input: an original environment map and a mask image that labels pixels as groups. During the design process, the system updates the lighting environment while retaining the visual structure of the original environment maps. The design process is as follows. First, the original environment map is segmented using image manipulation tools (e.g., Photoshop) to create a mask image as shown in Figure 13(b). Second, the environment map and the mask image are loaded into the system. The system fits L-SRBFs to the environment map and creates labeled groups of L-SRBFs based on the mask image. Finally, the user designs L-SRBF lights using the system tools. During lighting estimation, the system allows only one scaling factor to be updated for each group. In this way, the pixel intensities in a segment change together preserving the original image structure as shown in Figure 13(c) and (d). We used 2562 L-SRBFs. The mask image had four colors that created four labeled groups of L-SRBFs.

User Test To validate the effectiveness of the proposed system, we performed a user test to compare its performance to a combination of Adobe Photoshop and a HDR-Shop plug-in (the most popular combination for editing an environment map today). Eight computer science students, all novice users of our system and 3D graphics interfaces in general, participated in the study. Each subject was given a target image (Figure 15 a) and a 3D bunny model under a default lighting condition (Figure 15 b). Each scene has the same measured BRDF, aluminum-bronze. After a brief tutorial, the subjects were asked to adjust the lighting environment around the bunny model to that of the target image, using either our system or the alternative software (Photoshop+HDRShop). For both design tools, subjects are allowed to pick a color from the target image and use it for illumination design. The testing order of the software was alternated between subjects; four used Photoshop+HDRShop first, and the other four used our prototype system first. The subjects were allowed to work on the task until satisfied, for up to 20 min. Most of the subjects spent approximately 70 minutes for the study including tutorial. Figure 15 shows some of the resulting images.

Figure 14 (a) illustrates the time of design process across eight subjects and two design tools. Subjects using Illumination Brush overall took less than 75% of the time that they did when using Photoshop+HDRShop. Figure 14 (b) shows the subjective evaluation of the produced images. Sixteen people voted on the rendered images, regarding their similarity to the image rendered with the ground truth environment map (Figure 15 c). The numbers represent the votes for each system. A binomial test of the scores shows that our system performed significantly better than the Photo-

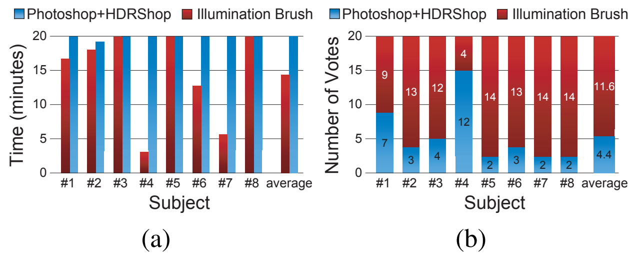


Figure 14. Subjective evaluation of the designed scenes. Each of eight users designed two environment lightings, one using Illumination Brush and the other using Photoshop+HDRShop, so that the rendered images were similar to the target image (shown in Figure 15 a). (a) illustrates the time of design process across eight subjects and two design tools. (b) Then, 16 people voted on the results, regarding how similar each rendered image matched the bunny image rendered using the ground truth lighting condition (shown in Figure 15 c). The numbers represent the votes for each system.

shop+HDRShop system in editing a lighting environment ($p < .001$). These results show that our system achieves better quality in less time than the combination of Adobe Photoshop and a HDRShop plug-in.

6 Limitations and Future work

The system approximates each transfer function using T-SRBFs. The ability to render strong specularity depends on the number of T-SRBFs. We tested the system using 642 T-SRBFs that can represent not specular but glossy BRDFs. If the number of T-SRBFs is increased, more specular material can be rendered, but it will result in a slower rendering speed and more memory usage.

Our future work also includes the evaluation/analysis of the relationship between reflectance property and illumination estimation algorithm. For the weighting factors described in 4.2, since it is difficult to obtain the best parameter analytically, we have used parameters and formulations that are empirically obtained. In the future, we want to find the best weighting factors with respect to reflectance property.

It would be interesting to extend our system to separate diffuse lighting and specular lighting. We could achieve this by incorporating two environment maps, one for each type of lighting. (A diffuse environment map can be represented more compactly by SH than L-SRBFs.) This could be useful for computer graphic designers, who often perform similar separations by rendering a scene into several components, which can be modified one by one and subsequently composed. The extension of our system may help them with the lighting design.

We are interested in appearance-based design tool not only for illumination but also for material, i.e., BRDF, BSSRDF, and spatially variant BRDF, etc. Our future research includes supporting BRDF editing in the unified framework.

Because recent works have designed methods for converting image-based lighting environments into a manageable number of light points [10], our system is useful for designing illumination for scenes that must be rendered with limited computation power, such as 3D games. Our future research includes such an application.

7 Acknowledgements

This work is supported in part by JSPS Research Fellowship and Microsoft Institute for Japanese Academic Research Collaboration.

References

[1] K. Anjyo and K. Hiramitsu. Stylized highlights for cartoon rendering and animation. *IEEE Comput. Graph. Appl.*, 23(4):54–61, 2003.

[2] F. Anrys and P. Dutré. Image based lighting design. *In The 4th IASTED International Conference on Visualization, Imaging, and Image Processing*, 2004.

[3] H. Biermann, I. Martin, F. Bernardini, and D. Zorin. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02*, pages 312–321. ACM Press., 2002.

[4] T. A. Davis. A column pre-ordering strategy for the unsymmetric-pattern multifrontal method. *ACM Trans. Math. Softw.*, 30(2):165–195, 2004.

[5] P. Debevec. Rendering synthetic objects into real scenes: bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH*, pages 189–198, 1998.

[6] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, pages 369–378, 1997.

[7] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, 2nd edition, 1987.

[8] N. Goodnight, R. Wang, C. Woolley, and G. Humphreys. Interactive time-dependent tone mapping using programmable graphics hardware. In *Proc. of the 14th Eurographics workshop on Rendering*, pages 26–37, 2003.

[9] S. Gumhold. Maximum entropy light source placement. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 275–282, Washington, DC, USA, 2002. IEEE Computer Society.

[10] V. Havran, M. Smyk, G. Krawczyk, K. Myszkowski, and H.-P. Seidel. Interactive system for dynamic scene lighting using captured video environment maps. In *Eurographics Symposium on Rendering 2005*, pages 31–42,311, 2005.

[11] T. Igarashi and J. F. Hughes. Clothing manipulation. In *UIST '02*, pages 91–100. ACM Press., 2002.

[12] T. Jung, M. D. Gross, and E. Y.-L. Do. Light pen – sketching light in 3d. In *Proc. of CAAD Futures*, pages 327–338, 2003.

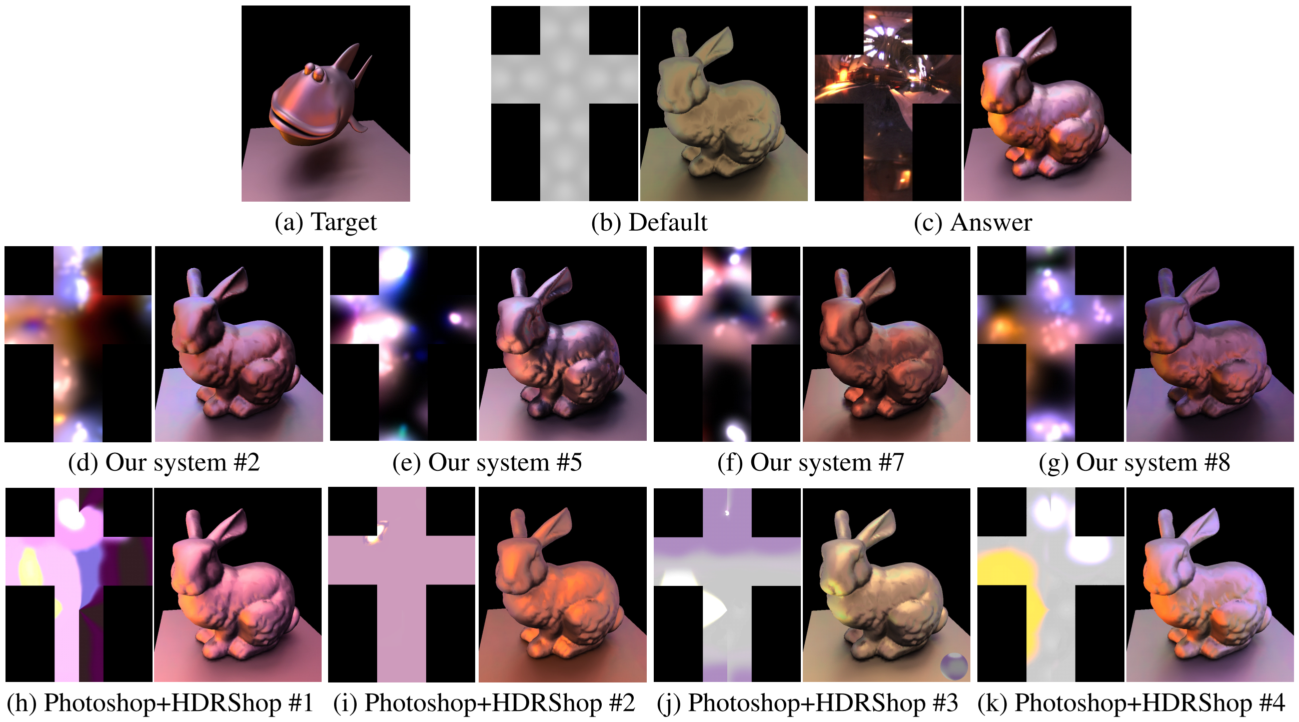


Figure 15. Results of the user test comparing the proposed system to Photoshop+HDRShop. The users were presented a target image (a) and asked to modify the bunny scene (b) to exhibit a similar illumination environment. The middle row (d-g) presents the resulting environment maps obtained by different users of the proposed system. The bottom row (h-k) shows the results using Photoshop+HDRShop. Note that users were not shown the ground-truth environment map. The ground-truth environment map and the rendered scene are given in (c).

[13] J. T. Kajiya. The rendering equation. In *SIGGRAPH*, pages 143–150, New York, NY, USA, 1986. ACM Press.

[14] J. K. Kawai, J. S. Painter, and M. F. Cohen. Radioptimization: goal based rendering. In *SIGGRAPH*, pages 147–154, 1993.

[15] Y. Li, S. Lin, H. Lu, and H.-Y. Shum. Multiple-cue illumination estimation in textured scenes. In *Int'l Conf. on Computer Vision*, pages 1366–1373, Nice, France, 2003.

[16] F. J. Narcowich and J. D. Ward. Nonstationary wavelets on the m-sphere for scattered data. *Applied and Computational Harmonic Analysis*, 3(4):324–336, 1996.

[17] K. Nishino and S. K. Nayar. Eyes for relighting. *ACM Trans. Graph. (SIGGRAPH)*, 23(3):704–711, 2004.

[18] F. Pellacini, F. Battaglia, R. K. Morley, and A. Finkelstein. Lighting with paint. *ACM Trans. Graph.*, 26(2):9, 2007.

[19] F. Pellacini, P. Tole, and D. P. Greenberg. A user interface for interactive cinematic shadow design. In *SIGGRAPH*, pages 563–566, 2002.

[20] P. Poulin, K. Ratib, and M. Jacques. Sketching shadows and highlights to position lights. In *Proc. of Conference on Computer Graphics International*, pages 56–63, 1997.

[21] R. Ramamoorthi and P. Hanrahan. A signal-processing framework for inverse rendering. In *SIGGRAPH*, pages 117–128, 2001.

[22] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. *ACM Trans. Graph. (SIGGRAPH)*, 21(3):267–276, 2002.

[23] I. Sato, Y. Sato, and K. Ikeuchi. Illumination from shadows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(3):290–300, 2003.

[24] C. Schoeneman, J. Dorsey, B. Smits, J. Arvo, and D. Greenburg. Painting with light. In *SIGGRAPH*, pages 143–146, 1993.

[25] R. Shacked and D. Lischinski. Automatic lighting design using a perceptual quality metric. *Computer Graphics Forum*, 20(3):1067–7055, 2001.

[26] A. Shesh and B. Chen. Crayon lighting: Sketch-based illumination of models. *Proceedings of Pacific Graphics 2006*, pages 113–116, 2006.

[27] P.-P. Sloan, J. Hall, J. Hart, and J. Snyder. Clustered principal components for precomputed radiance transfer. *SIGGRAPH*, 22(3):382–391, 2003.

[28] P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH*, pages 527–536, 2002.

[29] Y.-T. Tsai and Z.-C. Shih. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Trans. Graph.(SIGGRAPH)*, 25(3):967–976, 2006.

[30] Y. Wang and D. Samaras. Estimation of multiple directional light sources for synthesis of augmented reality images. *Graph. Models*, 65(4):185–205, 2003.