Illumination Brush: Interactive Design of Image-based Lighting

Makoto Okabe[1] Yasuyuki Matsushita[2] Takeo Igarashi[1] Heung-Yeung Shum[2]

August, 2006
Technical Report
MSR-TR-2006-112

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
http://www.research.microsoft.com

[1]The University of Tokyo
[2]Microsoft Research Asia

Abstract

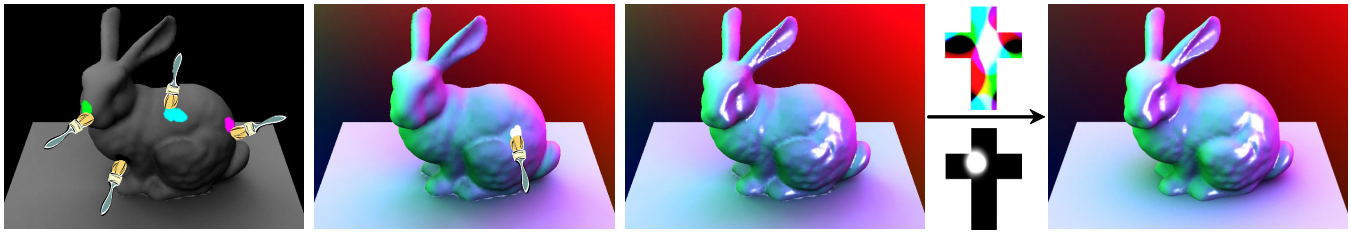
To help artists design customized lighting environments, we present a simple user interface for designing image-based lighting. Our system allows the artist to directly specify the appearance of the resulting image by painting and dragging operations. Our system constructs an image-based lighting model that produces the desired (and painted) appearance by solving the inverse shading problem. To obtain realistic lighting effects, we design diffuse and specular lighting effects separately. We represent diffuse lighting using low frequency spherical harmonics and pre-computed radiance transfer. For specular lighting, we simply project the painted highlights onto a cube map texture. We also provide an interface with which the artist can drag highlights and shadows on the surface to control the orientation of the entire lighting environment. We demonstrate that our technique is useful and practical for adding lighting effects to synthetic objects. We also show an application of seamlessly merging synthetic objects into photographs.

1 Introduction

Image-based lighting, or the environment map, is a method for representing a large-scale lighting environment surrounding the target scene as a texture map [Debevec 1998]. It compactly represents complicated incoming light from distant sources and enables real-time rendering of the scene with realistic lighting effects [Sloan et al.]. However, most systems rely on captured environments for image-based lighting [Debevec and Malik 1997; Havran et al. 2005] and few attempts have been made to manually design such complicated lighting environments. The artist might paint the environment map directly, but this is labor intensive and it would be very difficult to obtain the desired rendering result, because of the non-linear mapping of a lighting condition and the appearance of objects. Since a captured environment is not always available, a practical method for manually designing complicated lighting environments is in great demand.

In this paper, we propose an appearance-based user interface for designing image-based lighting environments. Instead of asking the artists to place lights in the surrounding environment, our system allows them to directly specify the appearance of the resulting image by painting and dragging surface colors on the target model. The system then constructs an appropriate image-based lighting model that produces the desired appearance by solving the inverse lighting problem. Appearance-based interfaces for lighting design have been proposed before [Pellacini et al. 2002; Schoeneman et al. 1993; Poulin et al. 1997; Jung et al. 2003; Kawai et al. 1993; Anjyo and Hiramitsu 2003], but they are limited to simple lighting models and are not designed for image-based lighting models. Image-based lighting can produce more appealing results than simple lights in general. It is particularly useful for adding realistic lighting effects to synthetic objects when compositing them into a photographed or video-recorded background.

Given the surface appearance specified by an artist, the system needs to estimate the corresponding image-based lighting model. This process is formalized as the inverse lighting problem, that is, recovering unknown lighting from known geometry, outgoing radiance, and surface bidirectional reflectance distribution function (BRDF) [Ramamoorthi and Hanrahan]. To solve this inverse lighting problem in real-time, we represent the image-based lighting model using spherical harmonics (SH), and obtain its coefficients using precomputed radiance transfer (PRT) [Sloan et al.]. Since this approach only works for low-frequency lighting on non-specular surfaces, the system requires the artist to specify specular components separately and constructs the high-frequency lighting model by simply projecting painted highlights on to a cube-map texture. This paper also discusses implementation details for dragging surface colors and for rotating global lighting environments according to a user's interactive operation.



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

Figure 1: The design process of image-based lighting with *Illumination Brushes*. (a) The user paints the desired color appearance directly on the 3D model. Four brushes with different colors are shown. (b) The recovered diffuse environment map is then used to relight the entire bunny. Also in (b), the user continues to paint highlights. (c) Now both diffuse and specular lighting effects are applied on the bunny. For comparison, the final image rendered by ray-tracing using the designed image-based lighting environment (the vertical cross maps of diffuse and specular lighting on the top and bottom respectively) is shown in (d).

2 Previous Work

Several methods have been proposed to manipulate lighting effects in a rendered 3D scene instead of manually setting raw lighting parameters. Pellacini *et al.* proposed an approach of using cast shadows for lighting design [Pellacini et al. 2002]. Some prior methods use sketching/painting interfaces which allow a user to directly draw lighting effects [Schoeneman et al. 1993; Poulin et al. 1997; Jung et al. 2003]. While these methods are effective, they are all limited to simple local lighting models, or to simple geometry. Other related work includes illumination control using psychophysical effects [Kawai et al. 1993] and interactive control of highlight shapes in cartoon animations [Anjyo and Hiramitsu 2003].

A part of our work is related to inverse rendering; more specifically, estimation of a light source distribution given the scene geometry, surface properties, and the resulting image. There have been methods to estimate light source distributions from a variety of different input types, such as cast shadows [Sato et al. 2003] and shading [Wang and Samaras 2003]. Li *et al.* used integrated cues of shading, shadows and specular reflections [Li et al. 2003]. Nishino *et al.* proposed a method to capture an environment map that is reflected on a human eye in an image and used it for relighting [Nishino and Nayar 2004]. Ramamoorthi and Hanrahan proposed methods of estimating either a material BRDF, an illumination condition, or both from a single image or from multiple images of a scene [Ramamoorthi and Hanrahan]. These methods provide us with a solid basis for the development of a user interface for interactive illumination design.

For the purpose of interactive rendering with image-based lighting, we adopt the PRT method [Sloan et al. ; Sloan et al. 2003] and a SH representation of environment maps [Green 2003].

3 User Interface

This section describes our prototype system for appearance-based lighting design from the artist's point of view. Our system allows the user to directly specify the appearance of the resulting image by painting diffuse and specular colors on the target model. The system also provides dragging tools to modify the lighting environment later. The system updates the lighting environment and screen image in real-time. The user can change the view point during the lighting design process. All the light sources are assumed to be distant and the user's painting process affects the positions, colors, intensities and shapes of light sources. Our system assumes known geometry and a pre-defined BRDF for each point on the surface of

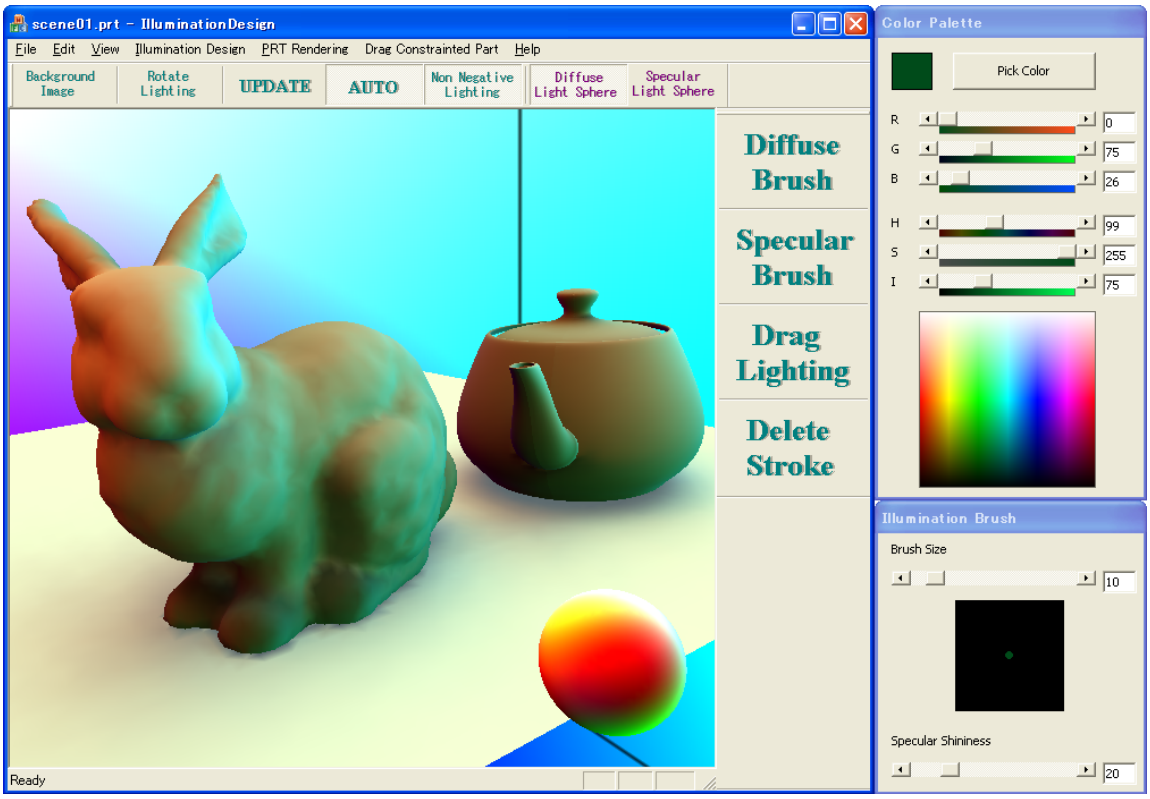


Figure 2: A snapshot of our prototype system.

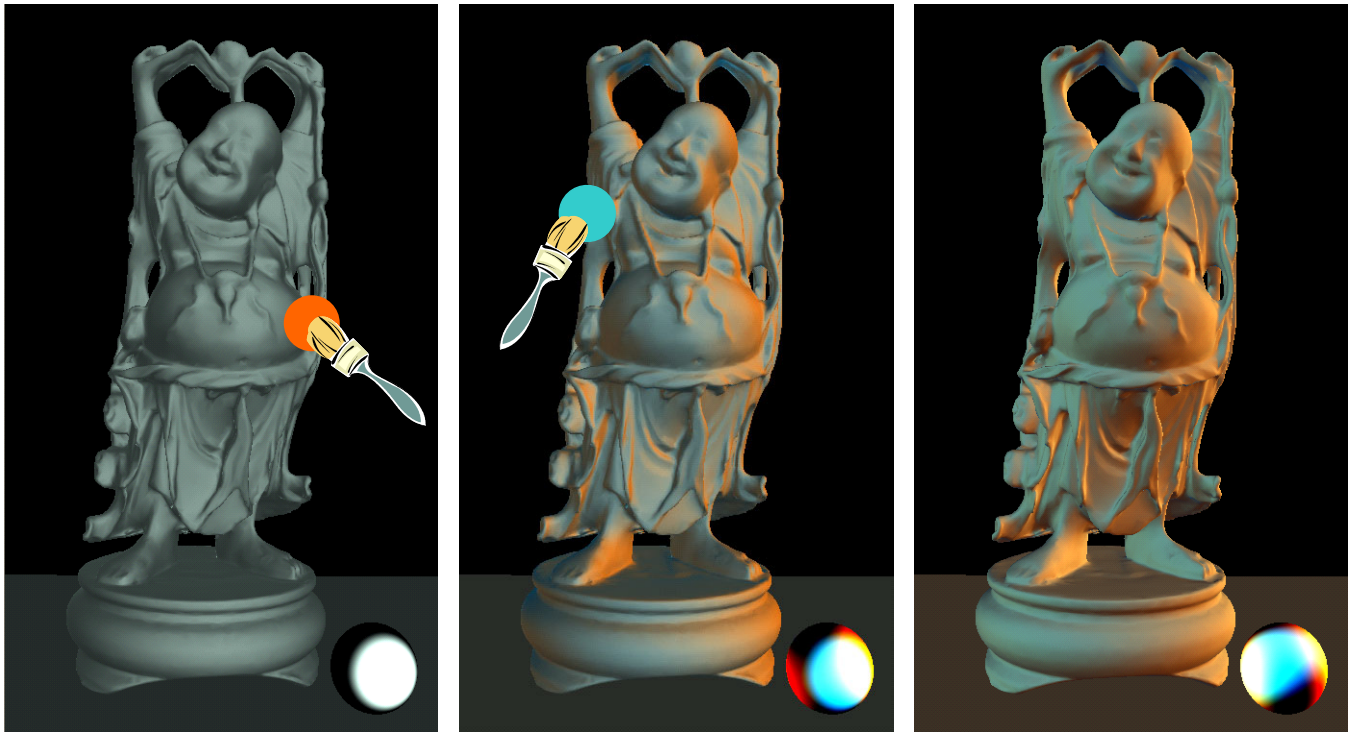
the target model.

Figure 2 shows a snapshot of our prototype system. After loading a 3D scene and corresponding PRT data, the user can immediately start painting on the surface by using illumination brushes. We currently support two types of brushes. One is for diffuse components and the other is for specular components. The current environmental lighting is always visualized on the bottom-right sphere.

3.1 Illumination Brushes

Illumination brushes are painting tools for specifying the desired appearance of the surface. The user draws a stroke on the screen by a dragging operation and the stroke is projected onto the object surface. The radiance of each mesh vertex is determined by the nearest stroke and the information is used for the lighting estimation. The painted stroke stays on the surface and its color and position can be modified later. Our system provides two types of Illumination Brushes. One is the Diffuse Brush for designing view-independent diffuse lighting effects, and the other is the Specular Brush for view-dependent specular lighting effects. It might be possible to provide a single user interface and estimate diffuse and specular components later. However, we choose to provide separate interfaces because this distinction is easy to understand for most artists, and the estimation result is more robust and predictable.

Diffuse Brush The Diffuse brush allows the user to specify the diffuse components of the outgoing radiance. The user simply selects the color, and starts painting on the 3D model. As the user paints, the system immediately estimates the corresponding illumination and updates the image on the screen. Figure 3 illustrates the process. In addition to this immediate mode, we also provide a deferred mode where



(a) (b) (c)

Figure 3: The initial image-based lighting environment has only a white light, which is shown in the bottom-right sphere. The user paints orange light directly on the 3D model (a). The lighting environment is updated and the user adds blue light effects to the image. (b). The rendered image with the final lighting environment is shown in (c).

the illumination estimation and screen update occurs only when the user requests them. The immediate mode is useful for a preliminary design phase to explore the possibilities, and the deferred mode is useful for the artist who has a specific goal in mind and does not want to be disturbed by instant updates.

Specular Brush The Specular brush allows the user to directly paint highlights on the 3D model to specify specular lighting effects. As with the diffuse brush, the lighting estimation occurs immediately and the result is shown on the screen (Figure 4). Note that the painted highlight is view-dependent and moves along the surface as the user rotates the model, while the diffuse component is independent of the viewing direction. The material's specular properties, such as shininess, are used to back-project the user input onto the environment map (Figure 5).

Manipulation of Painted Strokes The result of a paint operation is stored as a stroke on the object surface, and the user can modify its color and position later. To change the color of a stroke, the user clicks on the target stroke and adjusts its color using the color panel. To change the location of a stroke, the user clicks and drags the stroke on the object surface. As the dragged stroke slides over the object surface, it adapts its shape to match the surface (Figure 6). Similar operations are illustrated in [Igarashi and Hughes 2002; Biermann et al. 2002]. The lighting environment continues to be updated automatically during these operations. These operations are useful for locally adjusting the appearance of the scene.

When a user changes the view point, the position of a specular stroke no longer corresponds to the highlight it created. To help the user modify specular strokes in a consistent manner, whenever a user clicks on a specular stroke, our system automatically recovers the viewpoint from where it was painted.

(a) (b) (c)

Figure 4: The user paints specular highlights on the 3D 'ring' model. Orange highlights are painted in (a), and white highlights are added in (b). A rendered image from a different viewpoint is shown in (c). The light source distribution estimated from the user input is visualized in the vertical cross in (a) and (b).

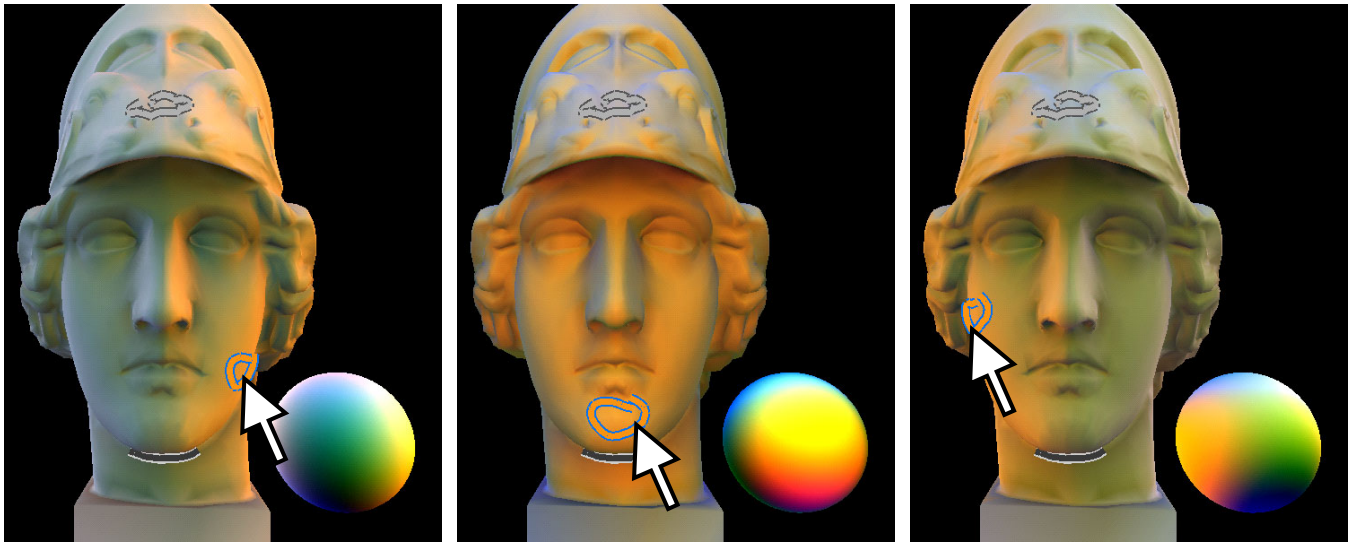
(a) (b) (c)

Figure 5: Specular painting onto objects with different shininess parameters, 10.0 (a), 20.0 (b), and 50.0 (c).

3.2 Rotating Image-based Lighting Environment

It is convenient to be able to rotate existing image-based lighting, especially when working with sky-lit environments. For example, after creating a sunset scene using illumination brushes, the user might want to change the orientation of the sun without changing the details of image-based lighting. To facilitate this process, we provide a tool for rotating the entire environmental lighting by a simple dragging operation on the object surface. The user can grab features such as highlights and shadows on the surface and move them to another location (Figure 7). Internally, the system rotates the entire environmental light so that the surface colors under the mouse cursor stay constant.

It is possible to have the user directly rotate the sphere that represents the environmental light, but we found that dragging within the scene is helpful in cases where the geometry and the lighting interact with each other. Figure 8 shows an example where identical dragging operations result in opposite rotations according to the different geometry and lighting conditions.



(a) (b) (c)

Figure 6: The user paints a gray light effect on the armor, a black light effect under the chin, and an orange light effect on the left cheek (a). The user clicks and drags the orange stroke to the chin (b) and to the right cheek (c).

4 Algorithm

Our system provides separate interfaces for diffuse components and specular components for the purpose of intuitive illumination design. Accordingly, the internal data structures in our system are also independent for the two lighting components, as supported by many existing libraries and software such as OpenGL and DirectX. We use a SH representation to compactly represent the low-frequency diffuse lighting, and use a cube map to represent high frequency specular lighting effects. We call the lighting environment for the diffuse component a 'diffuse lighting' and that for the specular component a 'specular lighting'.

4.1 Estimating Diffuse Lighting Environment

The input to the algorithm is a set of vertices that are painted by a user. Our goal is to estimate an image-based lighting environment, which reproduces the user-specified radiance at the painted vertices. In the estimation process, a SH representation is used to efficiently approximate the environment map. It compactly represents a rich lighting environment and facilitates the computation of forward and inverse rendering. In the following section, we describe the algorithm for one color channel to simplify the notation. In the actual implementation, the same steps are applied for each color channel independently.

We use PRT for accelerating the estimation and visualization process. Denoting the set of SH coefficients $\mathbf{C}_{lm} = \{c_{lm} | l \in \mathbf{Z}^+, -l \leq m \leq l\}$, where l is the band index, and PRT coefficients $\mathbf{P}^r_{lm} = \{p^r_{lm} | l \in \mathbf{Z}^+, -l \leq m \leq l\}$ for each vertex r, the radiance of a vertex r, I^r, is described by a dot product of $\mathbf{C}$ and $\mathbf{P}^r_{lm}$, i.e., $I(r) = \sum_{l,m} c_{lm} p^r_{lm}$. For convenience, let us rewrite $\mathbf{C}_{lm}$ and $\mathbf{P}_{lm}$ by single-indexed vectors $\mathbf{C} = \{c_0, \ldots, c_{n-1}\}$ and $\mathbf{P}^r = \{p^r_0, \ldots, p^r_{n-1}\}$, where $n = (l+1)^2$.

Given a set of vertices $r \in \Omega$ with associated PRT coefficients $\mathbf{P}^r$ and user-painted colors I^r_u, our algo-

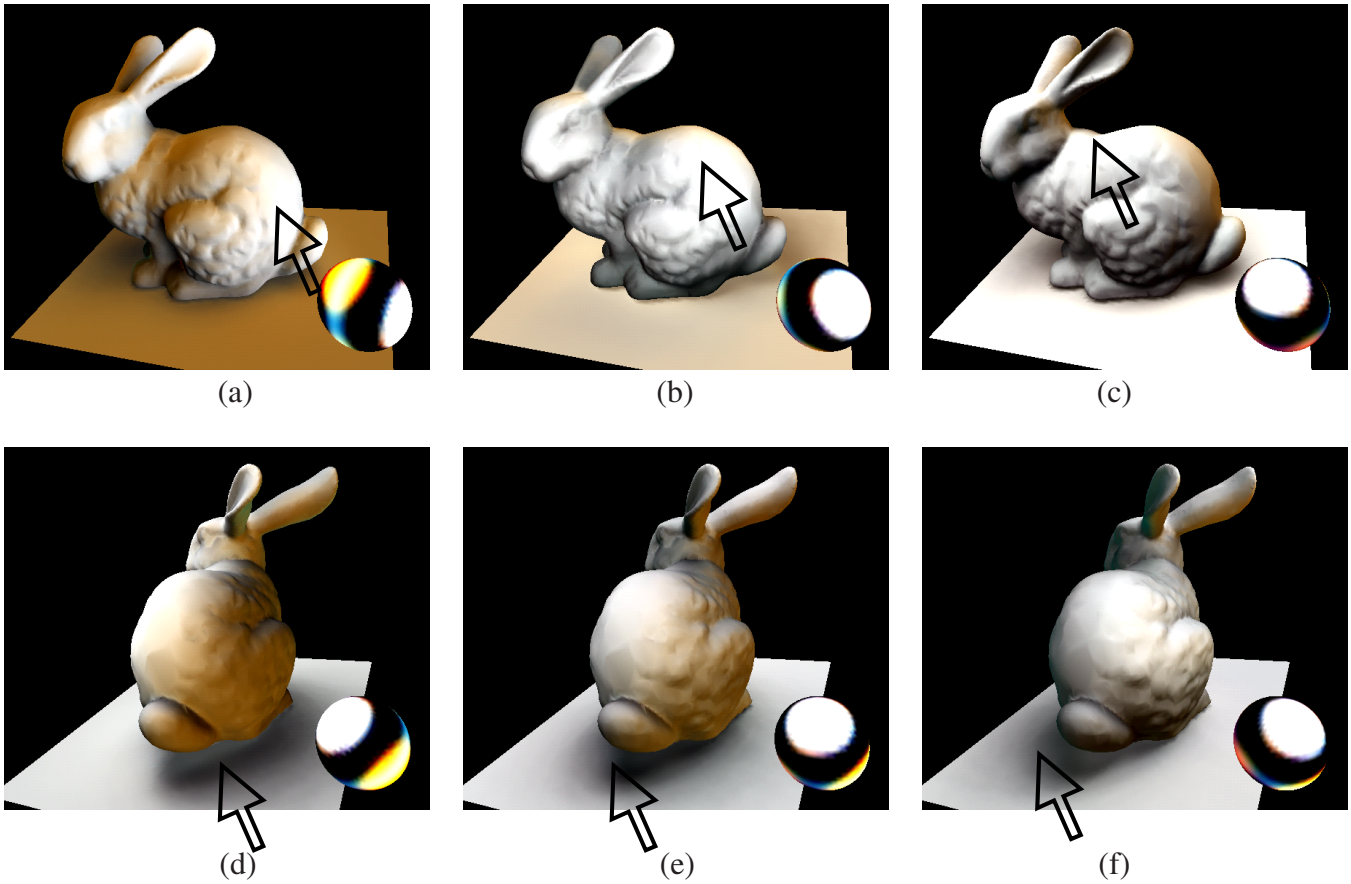


Figure 7: The user clicks and drags the white-lit part to rotate the lighting environment (a, b and c). The lighting environment rotates smoothly so that the surface colors under the mouse cursor stay constant.The user can also click and drag the shadow under the bunny's tail (d, e and f).

rithm solves the inverse problem of estimating $\mathbf{C}$ by minimizing the quadratic error function E:

$$E = \sum_{r \in \Omega} (I_u^r - \mathbf{P}^r \mathbf{C})^2, \tag{1}$$

where Ω is the set of indices of vertices that are painted by the user. Equation (1) can be rewritten using a matrix representation as

$$E = ||\mathbf{I}_u - \mathbf{P}\mathbf{C}||_2^2. \tag{2}$$

Denoting $|\Omega|$ as the number of colored vertices by the user, $\mathbf{I}_u$ is a $|\Omega| \times 1$ vector, $\mathbf{P}$ is a $|\Omega| \times n$ matrix, and $\mathbf{C}$ is a $n \times 1$ vector. The SH lighting coefficients $\mathbf{C}$ can be estimated by deriving the least squares solution from the linear system ([Ramamoorthi and Hanrahan]); however, additional considerations are required for the purpose of illumination design.

Constrained Quadratic Programming. From the view point of illumination design, it is preferred that the estimated lighting condition be physically realizable. Unfortunately, a naive solution of Equation (2) can produce negative lighting. To avoid this unnatural result, we impose a constraint that every point (θ, ϕ) in the estimated environment map has a non-negative value f:

$$f(\theta, \phi) = \sum_i c_i Y_i(\theta, \phi) \geq 0, \tag{3}$$

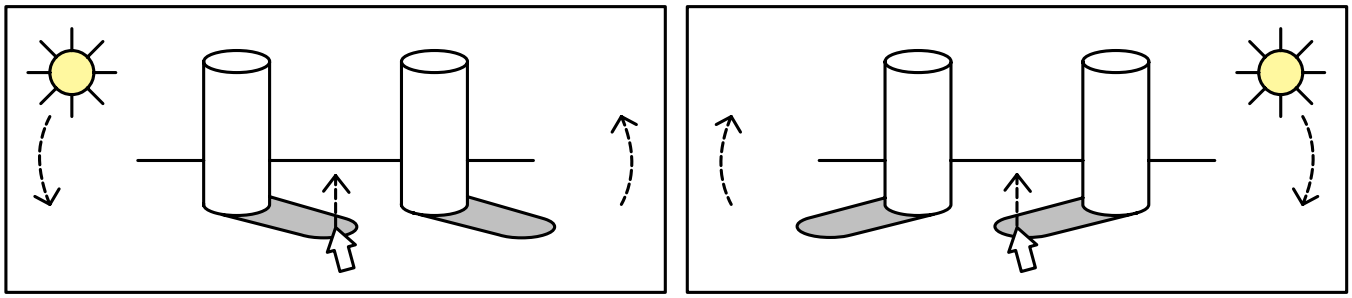


Figure 8: Exactly the same dragging operations can result in opposite rotations according to the different geometry and lighting conditions.

where Y is the singly-indexed SH function. Therefore, the problem of deriving the best SH coefficients $\mathbf{C}$ becomes a constrained quadratic programming problem, i.e., solving Equation (2) under the inequality constraint Equation (3). We use the active set method [Fletcher 1987] to efficiently derive the solution.

Adaptive Estimation Method. It is easy to imagine that user input is not always consistent and is often contradictory. Sometimes, the input information is not enough for estimating n SH coefficients (under-constrained case), and more often too much information is given (over-constrained case). Therefore, we adopt an adaptive estimation method [Sato et al. 2003], which changes the number of SH coefficients being estimated depending on the user input. Namely, we evaluate the rank of matrix $\mathbf{P}$, $R(\mathbf{P})$, and the best number of SH coefficients k is determined by taking the maximum integer value under the following conditions: $k \leq R(\mathbf{P}), k \leq n$. In this way, given a small number of input strokes, only low-frequency SH coefficients are estimated. On the other hand, given a high enough number of input strokes, high-frequency coefficients (up to n coefficients) can be estimated.

4.2 Estimating Specular Lighting Environment

The specular lighting environment is generated by simply back-projecting the input stroke onto the environment map, and assuming an isotropic refection model. The light source position is determined by mirror-reflecting the user's current eye-point about the surface normal direction, and the intensity of the light source is computed by dividing the user painted specular color by BRDF$(\theta, \phi, \theta, \phi + \pi)$. A shininess parameter given by the user is used to determine the size of the light source according to the Phong shading model.

4.3 Dragging Painted Strokes

When the user grabs and drags a stroke, the system first moves the grabbed edge on the surface and then determines the location of the surrounding vertices one by one so that the local positional relationship between neighboring vertices are maintained (Figure 9). 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.

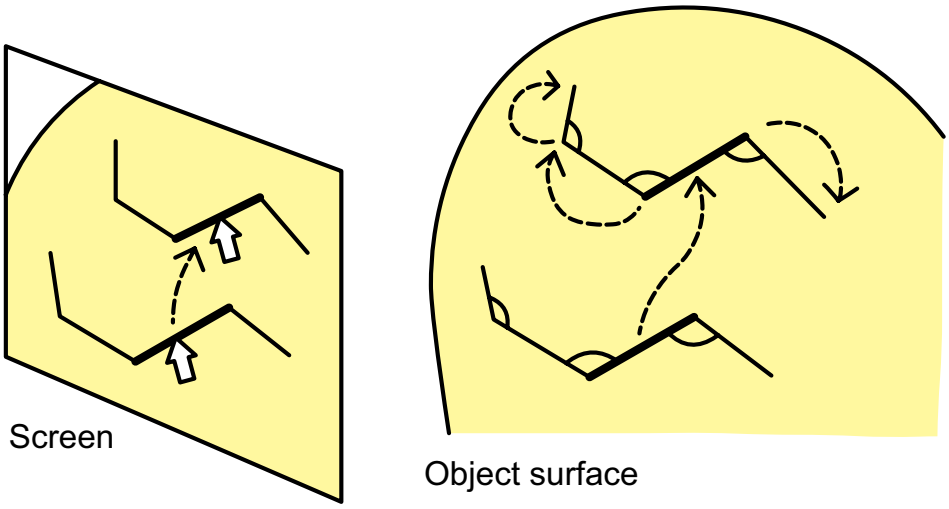


Figure 9: Dragging a stroke on the object surface. The system first moves the grabbed edge and determine the location of surrounding edges.

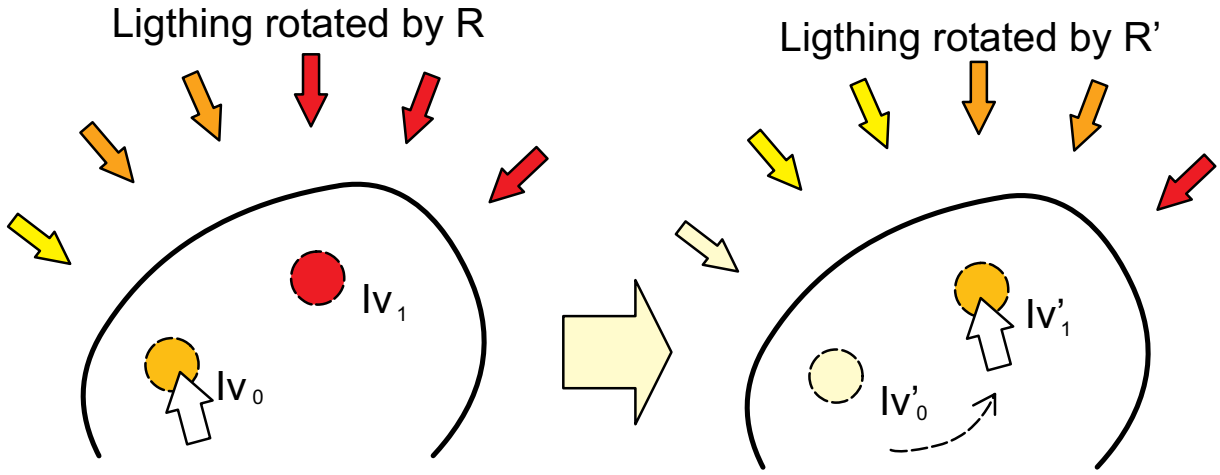


Figure 10: Rotation of SH lighting with dragging operation. The system computes $\mathbf{R}'$ so that the resulting radiance at v_1 is identical to the previous radiance at v_0.

4.4 Rotating Lighting Environment

The orientation of spherical harmonics-based lighting is represented as a $n \times n$ rotation matrix $\mathbf{R}$, which is calculated by extending a normal 3×3 rotation matrix [Choi et al. 1999]. The rotation matrix $\mathbf{R}$ is multiplied by the SH coefficients when computing the surface radiance, i.e., $I = \mathbf{PRC}$. When the user performs a dragging operation using the rotation tool, the system updates the rotation matrix so that the radiance under the mouse cursor remains constant. To be more precise, computation proceeds as follows (Figure 10). Suppose that the mouse cursor moves from p_0 to p_1 in screen coordinates. These points are projected to the object surface and corresponding surface positions v_0 and v_1 are obtained. The system calculates the surface radiance I_{v_0} at the position v_0 in the current orientation $\mathbf{R}$. Then the system estimates the new orientation R' minimizing the difference between I_{v_0} and I'_{v_1} (the surface radiance at position v_1 under the new orientation $\mathbf{R}'$).

We solve this problem using a standard minimization method. The rotation matrix is represented as three-rotation angles α, β, and γ. The system seeks the smallest changes, $\mathrm{d}\alpha$, $\mathrm{d}\beta$ and $\mathrm{d}\gamma$, to the rotation

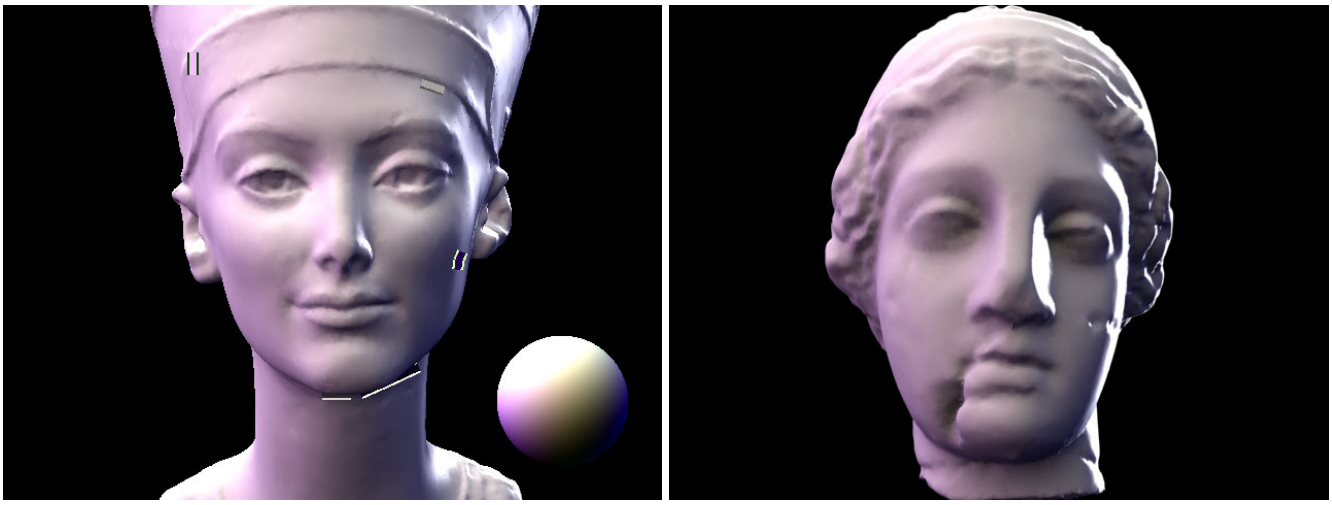


Figure 11: In the left image, an illumination condition is designed by a user. The designed illumination condition is exported to the other scene and rendered using a ray-tracer (right). This example shows the capability of our system of exporting designed lighting environment to other existing renderers.

angles that satisfies $I'_{v_1} = I_{v_0}$. To achieve this, the system solves the following minimization problem using the Lagrange multiplier method:

$$\min\{\mathrm{d}\alpha^2 + \mathrm{d}\beta^2 + \mathrm{d}\gamma^2\} \quad \text{s.t.} \quad \mathrm{d}I_{v_1} = \frac{\partial I_{v_1}}{\partial \mathbf{A}}\mathrm{d}\mathbf{A}, \tag{4}$$

where $\mathbf{A} = (\alpha, \beta, \gamma)$, and $\mathrm{d}I_{v_1}$ is the target differential scalar of the radiance at v_1 ($I_{v_0} - I_{v_1}$). The system continuously solves the problem and updates $\mathbf{A}$ using a small gradient step $\varepsilon = 0.05$.

5 Result

To validate the effectiveness of our system, we have performed user tests. Furthermore, we show a useful application of adjusting photometric consistency when inserting synthetic 3D objects into photographs. Throughout the experiment, we set the maximum number of SH coefficients n to be 9, because it was shown that more than 99% of the energy can be described by 9 coefficients [Ramamoorthi and Hanrahan].

Since our output is designed illumination conditions, not just the appearance of a particular scene, the designed environment map can be naturally exported in the form of high-dynamic range images [Debevec and Malik 1997] and applied to different scenes. Figure 11 shows this example. In the left image, the lighting condition is designed by a user using our system. The designed lighting condition is exported, and a different scene (right) is rendered by a ray tracer using the designed lighting condition.

Figure 12 shows qualitative results obtained by the authors using our illumination design system. Four lighting conditions in photographs are chosen as goal illumination environments. They are the scenes of early-morning, a hotel room, fine weather, and a night club. The user interaction time and the number of diffuse and specular strokes are shown below each result.

Our method is also useful for adjusting photometric consistency when inserting a synthetic 3D object into a photograph. In this scenario, our system allows the user to load and display a background image behind the 3D object, and the user can paint and adjust the light effects of the 3D object so that it seamlessly matches the lighting conditions of the background photograph.

Figure 13 shows the results of inserting a synthetic 3D object into a photograph by the authors. These results are rendered by a ray-tracer using exported image-based lighting environments designed using our system.

We also performed a user study to test the usability of our system in this scenario. Five computer science students, who are all novice users of the system and 3D graphics interfaces in general, participated the study. After presenting a brief tutorial, we gave two 3D models to each subject and asked him or her to insert each model to a photograph adjusting the illumination condition using the system. The subjects were allowed to work on the task until they are satisfied with the result and most of them spent approximately 30 minutes for the study including the tutorial. Figure 14 shows some of the resulting images. The subjects performed the task without major difficulty and reported that the function to pick a color from the background image was particularly useful for this task.

6 Limitations and Future Work

Due to the limitation of the SH representation, our current implementation cannot generate an environment map that generates sharp cast shadows. Our future research includes supporting high frequency shadows using wavelets or other representations of image-based lighting. Our current implementation only supports simple isotropic reflections for specular paintings. Extending it to support arbitrary BRDFs is one of our most important future goals. The output of our system is a cubic environment map. Since recent works can convert image-based lighting environments into a manageable number of point lights [Havran et al. 2005], our system is useful for designing illumination for scenes that need be rendered with limited computation power, such as 3D games.

References

ANJYO, K., AND HIRAMITSU, K. 2003. Stylized highlights for cartoon rendering and animation. *IEEE Comput. Graph. Appl. 23*, 4, 54–61.

BIERMANN, H., MARTIN, I., BERNARDINI, F., AND ZORIN, D. 2002. Cut-and-paste editing of multiresolution surfaces. In *SIGGRAPH '02*, ACM Press., 312–321.

CHOI, C. H., IVANIC, J., GORDON, M. S., AND RUEDENBERG, K. 1999. Rapid and stable determination of rotation matrices between spherical harmonics by direct recursion. *J. Chem. Phys. 111*, 19, 8825–8831.

DEBEVEC, P. E., AND MALIK, J. 1997. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, 369–378.

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

FLETCHER, R. 1987. *Practical Methods of Optimization*, 2nd ed. John Wiley & Sons.

GREEN, R. 2003. Spherical harmonic lighting: The gritty details. In *Proceedings of the Game Developer Conference, March 2003*.

HAVRAN, V., SMYK, M., KRAWCZYK, G., MYSZKOWSKI, K., AND SEIDEL, H.-P. 2005. Interactive system for dynamic scene lighting using captured video environment maps. In *Eurographics Symposium on Rendering 2005*, 31–42,311.

IGARASHI, T., AND HUGHES, J. F. 2002. Clothing manipulation. In *UIST '02*, ACM Press., 91–100.

JUNG, T., GROSS, M. D., AND DO, E. Y.-L. 2003. Light pen – sketching light in 3d. In *Proc. of CAAD Futures*, 327–338.

KAWAI, J. K., PAINTER, J. S., AND COHEN, M. F. 1993. Radioptimization: goal based rendering. In *SIGGRAPH*, 147–154.

LI, Y., LIN, S., LU, H., AND SHUM, H.-Y. 2003. Multiple-cue illumination estimation in textured scenes. In *Int'l Conf. on Computer Vision*, 1366–1373.

NISHINO, K., AND NAYAR, S. K. 2004. Eyes for relighting. *ACM Trans. Graph. 23*, 3, 704–711.

PELLACINI, F., TOLE, P., AND GREENBERG, D. P. 2002. A user interface for interactive cinematic shadow design. In *SIGGRAPH*, 563–566.

POULIN, P., RATIB, K., AND JACQUES, M. 1997. Sketching shadows and highlights to position lights. In *Proc. of Conference on Computer Graphics International*, 56.

RAMAMOORTHI, R., AND HANRAHAN, P. A signal-processing framework for inverse rendering. In *SIGGRAPH*.

SATO, I., SATO, Y., AND IKEUCHI, K. 2003. Illumination from shadows. *IEEE Trans. Pattern Anal. Mach. Intell. 25*, 3, 290–300.

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J., AND GREENBURG, D. 1993. Painting with light. In *SIGGRAPH*, 143–146.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *SIGGRAPH*.

SLOAN, P.-P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered principal components for precomputed radiance transfer. *SIGGRAPH 22*, 3, 382–391.

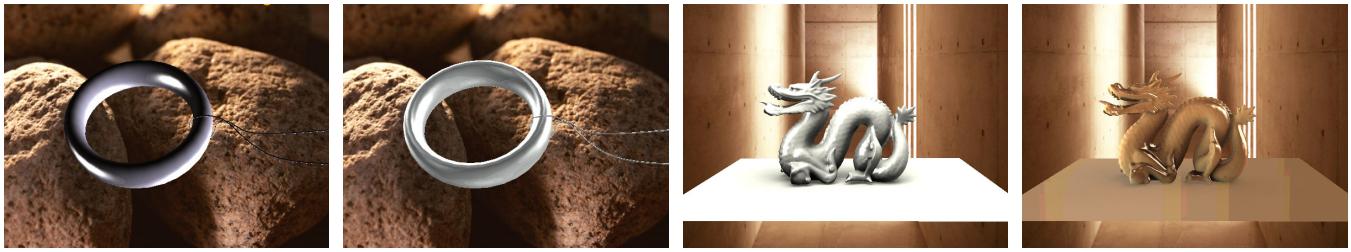
WANG, Y., AND SAMARAS, D. 2003. Estimation of multiple directional light sources for synthesis of augmented reality images. *Graph. Models 65*, 4, 185–205.

(a) 4.0 min / Diffuse:13 / Specular:1 (b) 5.5 min / Diffuse:11 / Specular:5 (c) 2.5 min / Diffuse:3 / Specular:7 (d) 3.0 min / Diffuse:4 / Specular:7

Figure 12: Illumination design results. Four lighting conditions(early-morning, a hotel room, fine weather, and a night club) are chosen as the goals of designed illumination. The interaction time and the number of diffuse and specular strokes are shown below each result.

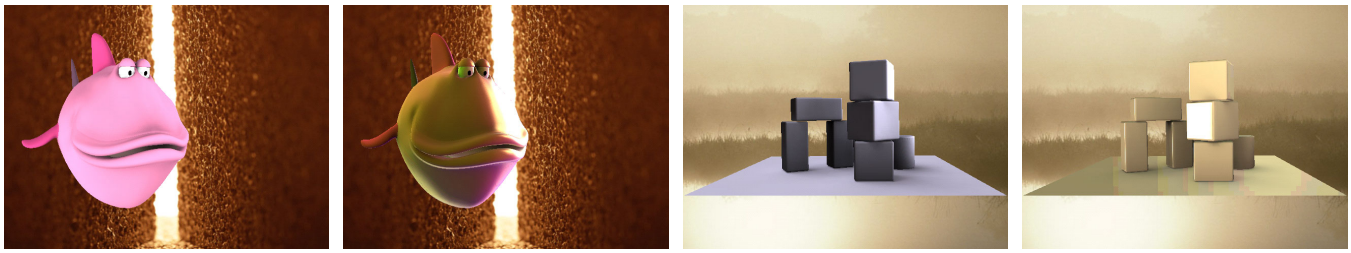
(a) Statue (b) Panther

Figure 13: Adjusting photometric consistency of a synthetic 3D object in a photograph. The two images on the left are the result of inserting a 3D statue model into a photograph. The left image has an arbitrary illumination and the image on the right is rendered by a ray tracer using an image-based lighting environment designed by our system. The two images on the right are the result of inserting a 3D panther model to the photograph. The top and bottom images are before and after illumination design using our system.



(a) 9.5 min / Diffuse:6 / Specular:23

(b) 4.0 min / Diffuse:2 / Specular:2



(c) 8.5 min / Diffuse:4 / Specular:3

(d) 10.5 min / Diffuse:7 / Specular:2

Figure 14: Illumination design results by the test users. The left images show the rendering results in default lighting conditions, and the right images show those in lighting conditions designed by the test users. The interaction time and the number of diffuse and specular strokes are shown below each result.