envyLight: An Interface for Editing Natural Illumination

Fabio Pellacini
Dartmouth College

Abstract

Scenes lit with high dynamic range environment maps of real-world environments exhibit all the complex nuances of natural illumination. For applications that need lighting adjustments to the rendered images, editing environment maps directly is still cumbersome. First, designers have to determine which region in the environment map is responsible for the specific lighting feature (e.g. diffuse gradients, highlights and shadows) they desire to edit. Second, determining the parameters of image-editing operations needed to achieve specific changes to the selected lighting feature requires extensive trial-and-error.

This paper presents *envyLight*, an interactive interface for editing natural illumination that combines an algorithm to select environment map regions, by sketching strokes on lighting features in the rendered image, with a small set of editing operations to quickly adjust the selected feature. The *envyLight* selection algorithm works well for indoor and outdoor lighting corresponding to rendered images where lighting features vary widely in number, size, contrast and edge blur. Furthermore, *envyLight* selection is general with respect to material type, from matte to sharp glossy, and the complexity of scenes' shapes. *envyLight* editing operations allow designers to quickly alter the position, contrast and edge blur of the selected lighting feature and can be keyframed to support animation.

Keywords: lighting design interfaces, natural illumination

1 Introduction

Image-Based Illumination. Over the past decade, image-based illumination was shown to be an accurate representation to model direct lighting of real scenes. By using a high dynamic range environment map of a real-world environment, all the nuances of natural illumination are faithfully reproduced [Debevec 1998]. The investigation of efficient interactive and offline rendering methods has made lighting with environment maps a practical alternative for lighting design. For applications where precise control of scenes' look is desirable, adjustments to these environment maps are still needed. To the best of our knowledge, lighting designers commonly use image-based editing tools, like Photoshop [Adobe Systems Inc 2009], to edit these maps.

Motivation. Making image-based adjustments to environment maps to achieve specific changes in the rendered image is very complex. In a typical *select-and-modify* workflow with a tool like Photoshop, a designer would select a region of the environment map, separate it into a layer, and then apply operations to it. In the context of lighting design, the most challenging aspect of this workflow is determining which regions of the environment maps are responsible for an effect in the rendered image. For example, in Fig. 1.b, which parts of the *grove* map cause which highlight? Or, in Fig. 1.a, which regions of the *grace* map cause the reddish tone on the statue?

Even after the selection is performed, it is often unclear how to set the parameters of an image processing operation to achieve a specific edit. For example, in Fig. 1.b, how to transform the environment map to move the selected highlight or, in Fig. 1.c, how to scale image intensities to change the contrast of the shadow without altering overall image tones.

envyLight. This paper presents *envyLight*, an interface to simplify the editing of natural illumination. *envyLight* works within the same *select-and-modify* workflow that artists are familiar with. We found this workflow to work well in our testing, with the additional advantage of allowing designers to integrate *envyLight* with existing tools, thus obtaining the benefits of all of them. In our prototype implementation, *envyLight* is interactive, further enhancing the benefit of this workflow.

The foremost focus of *envyLight* is to simplify selection by letting designers indicate which lighting feature they desire to edit in the rendered image (e.g. diffuse gradients, highlights and shadows). Our selection algorithm splits the environment map into two layers, a foreground and a background, such that edits to the foreground environment map modify the lighting feature marked by the user in the rendered image and such that the sum of the two layers is equal to the original map. In *envyLight*, designers mark lighting features with two rough strokes, a stroke to indicate parts of the image that belong to that feature, and another stroke to indicate parts of the image that do not. The same stroke-based metaphor works for all lighting features; for example, we can select diffuse illumination, specular highlights and shadows as shown in Fig. 1.

In addition, *envyLight* provides a set of editing operations that allows designers to quickly alter the selected lighting features by appropriately changing the split environment map layers. In particular, we support enhancing or reducing the contrast of the selected lighting features, moving them by specifying rotations or by sketching strokes on the rendered image, and blurring or sharpening them. Examples of such edits are shown in Fig. 1.

All-Frequency Effects. As we will show throughout the paper, we found *envyLight* selection to work well with a large variety of natural illumination, from indoor to outdoor, exhibiting a variety of frequency distributions and contrast ranges within the environment map. This variety in natural illumination causes lighting features to vary greatly in terms of their number, spatial size, contrast and edge blur. For example, shadows vary from the sharp, high-contrast ones of sunny days to the subdued, blurred ones of overcast days. Highlights exhibit similar variety, while diffuse effects show complex blending of gradients. We found our selection to work well in all cases. Selections can be made regardless of the material type, from matte to low gloss to sharp specular finishes. Finally, intricate geometry does not pose a problem for our method; for example in Fig. 1 we show the selection of the shadow of a bush and the selection of a sharp highlight on a bumpy statue.

Contributions. In summary, as an interface for editing natural illumination, *envyLight* has the following advantages:

ACM Reference Format
Pellacini, F. 2010. *envyLight*: An Interface for Editing Natural Illumination. *ACM Trans. Graph. 29*, 4, Article 34 (July 2010), 8 pages. DOI = 10.1145/1778765.1778771 http://doi.acm.org/10.1145/1778765.1778771.

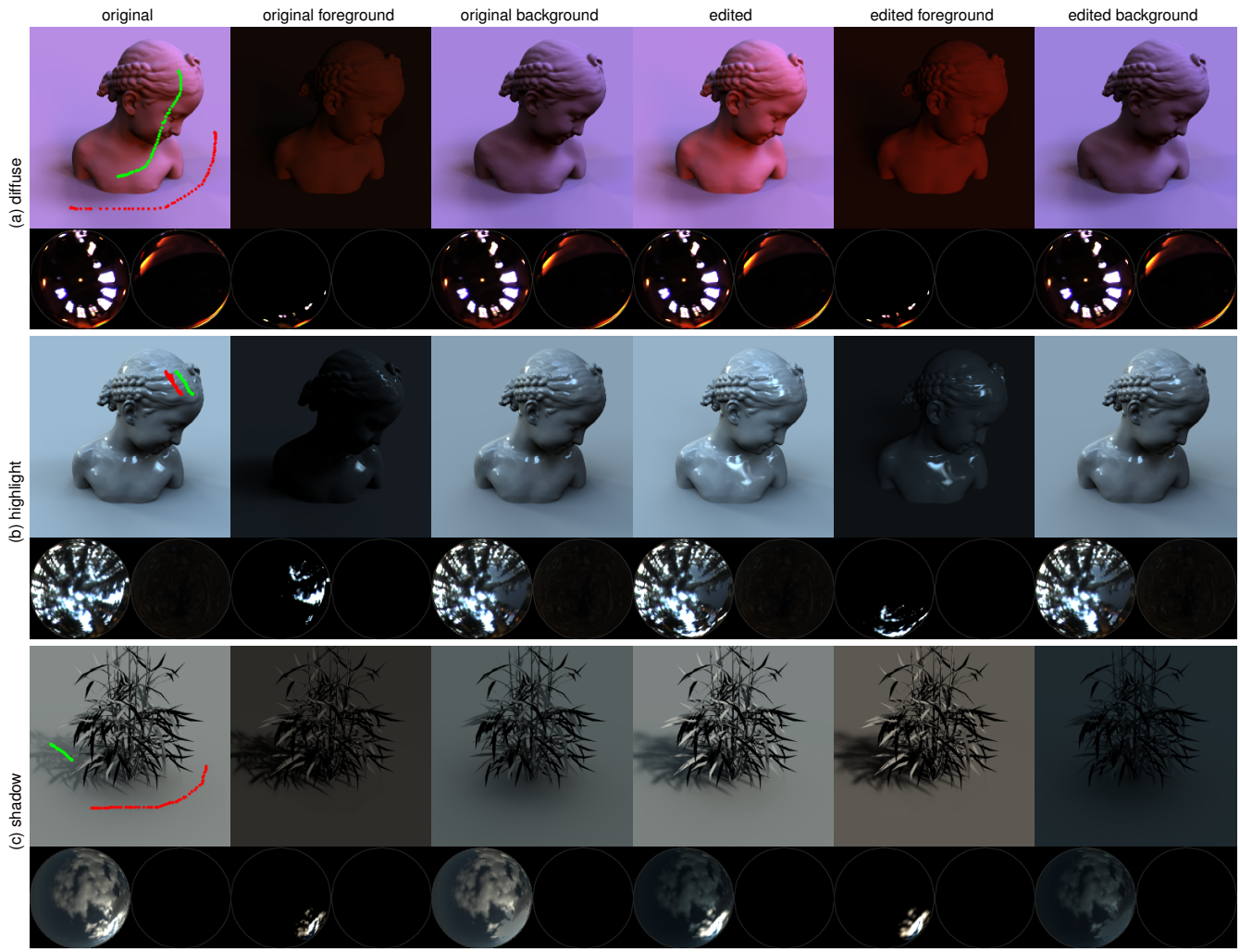

Figure 1: *Example edits performed with* envyLight*. For each row, we show the environment map at the bottom, as top and bottom hemispheres, and the corresponding rendered image at the top. Designers mark lighting features (e.g. diffuse gradients, highlights, shadows) with two strokes: a stroke to indicate parts of the image that belong to the feature (shown in green), and another stroke to indicate parts of the image that do not (shown in red).* envyLight *splits the environment map into a foreground and background layer, such that edits to the foreground directly affect the marked feature and such that the sum of the two is the original map. Editing operations can be applied to the layers to alter the marked feature. (a) Increased contrast and saturation of a diffuse gradient. (b) Translation of a highlight. (c) Increased contrast and blur of a shadow.*

- it allows users to select environment map regions by quickly sketching strokes on the lighting features a designer intends to edit; one selection metaphor works for all lighting features, e.g. diffuse gradients, highlights and shadows
- it supports selections for a wide range of indoor and outdoor lighting, corresponding to rendered images in which lighting features vary widely in number, size, contrast and edge blur
- it supports selections that are robust with respect to material type, from matte to highly glossy finishes, as well as the complexity of the lit shapes
- it provides a set of editing operations to quickly alter the position, contrast, and edge blur of the selected lighting feature
- it supports editing in scenes with animated objects, where environment map edits can be keyframed as well
- it allows editing at interactive rates.

2 Related Work

Interfaces. Our work is motivated in part by the user study on lighting design presented in [Kerr and Pellacini 2009]. The authors group lighting interfaces for point-like illumination into three categories. *Direct interfaces* (e.g. [Autodesk Inc 2009]) require users to act on light parameters, e.g. selecting and moving a light. *Indirect interfaces* [Gleicher and Witkin 1992; Poulin and Fournier 1992; Pellacini et al. 2002] allow users to edit lighting features in the rendered image, e.g. clicking and dragging a shadow. In *painting interfaces* [Schoeneman et al. 1993; Poulin et al. 1997; Anrys and Dutré 2004; Mohan et al. 2005; Pellacini et al. 2007], users paint a goal image, while light parameters are optimized to attempt to match it. [Kerr and Pellacini 2009] found that painting interfaces were not effective, due to the inability of users to paint a precise depiction of lighting, while users did well with direct and indirect interfaces, preferring the latter slightly. Within this categorization, editing environment maps with Photoshop-like tools is considered

a direct interface, since designers edit the light directly, while *envyLight* is considered an indirect interface, since designers mostly work in the rendered view. The results of this study motivated the design of *envyLight* in two ways. First, we excluded using paint-like interfaces. Second, we specifically designed *envyLight* to fit in the same workflow as image-based editing, giving designers the ability to work with the two together.

The work most related to our own is [Okabe et al. 2007], an interface that supports the creation of synthetic, mostly low-frequency, environment illumination using an inverse rendering method. *envyLight* focuses instead on editing existing real-world illumination, working in a fundamentally different design space, where the methods of [Okabe et al. 2007] would not be applicable.

Representations. While environment maps can be represented by simple images, e.g. spherical or cube environment maps, a variety of representations have been proposed for more efficient rendering. [Ramamoorthi and Hanrahan 2002; Sloan et al. 2002] adopt spherical harmonics as a compact representation for low frequency effects. [Ng et al. 2003] proposes the use of wavelets to capture all frequency effects. While these are all linear basis, [Tsai and Shih 2006] proposes the use of non-linear spherical radial basis functions to achieve better compression. For editing operations, spherical harmonics and wavelets do not have particular benefits over using pixels as a basis, with the latter remaining simpler and easier to integrate with existing image-based editing toolsets. While the non-linear basis of [Tsai and Shih 2006] would allow designers to rotate and scale the spherical functions directly, the high number required would still be impractical for direct manipulation, and the need for non-linear fitting while editing would slow down the workflow considerably. Our implementation uses cubic environment maps as its internal representation, since it matches our renderer, while displaying hemispherical projections, since they are easier to interpret when viewed.

3 Editing Natural Illumination

Workflow. We propose to edit environment maps with a *select-and-modify* workflow, where designers first select which part of an environment map to adjust, move it to a separate layer, then apply editing operations to it. This is the same workflow supported by widely-adopted image editing tools, such as Photoshop. This is advantageous since designers are already familiar with it and can integrate *envyLight* with other editing tools, if so desired. Fig. 1 shows example edits created interactively with this workflow in *envyLight*, where, for each environment map, we show the corresponding rendered image.

Notation. For each point x in the rendered image, direct illumination from an environment is computed by

$$B_x(\omega_o, \lambda) = \int_\Omega L(\omega, \lambda)\rho_x(\omega \rightarrow \omega_o, \lambda)V_x(\omega)(\omega \cdot n_x)d\omega$$

where ω_o is the viewing direction, ω is the incoming lighting direction, L is the environment map, V the visibility function, ρ the reflectance function and $(\omega \cdot n_x)$ the cosine of the incident angle. To simplify notation in the rest of the paper, we will drop the dependence from ω_o, since for a given camera location this is defined by x, as well as the wavelength λ, since our algorithms are performed independently on each channel. We thus obtain

$$B_x = \int_\Omega L(\omega)\rho_x(\omega)V_x(\omega)(\omega \cdot n_x)d\omega = \int_\Omega L(\omega)t_x(\omega)d\omega$$

where, at each point x, the transport function t is defined as the product of reflectance, visibility and cosine.

3.1 Selection

Interface. *envyLight* simplifies the selection of environment map regions by allowing designers to mark which lighting feature they desire to edit. Our selection algorithm automatically splits the environment map into a foreground and background layer, such that edits to the foreground layer directly affect the feature the designer marked and such that the sum of the two layers is equal to the original map. We adopt a sketching metaphor, where designers quickly mark the rendered image with rough strokes, since interfaces based on this metaphor have been shown to work well in image and material editing (e.g. [Pellacini and Lawrence 2007]). In *envyLight*, designers mark lighting features with two strokes, a stroke to indicate parts of the image that belong to that feature, and another stroke to indicate parts of the image that do not. We will refer to these strokes as *in* and *out* respectively.

Lighting Features. To gain intuition on how our selection algorithm works, let us consider how some lighting features form. A highlight is formed in the rendered image if the product $t_x(\omega)L(\omega)$ is high for some ω at a set of locations X_{in} and low for a set of nearby locations X_{out}. Diffuse gradients form for the same reason, but typically have a larger spatial extent in the rendered image. Shadows form when the opposite is true, i.e. the product $t_x(\omega)L(\omega)$ is low for some ω for locations in X_{in} and high for nearby locations X_{out}. Note that this is true regardless of the contrast and edge sharpness of the lighting feature. See Fig. 1 for example images. In *envyLight*, designers specify the sets X_{in} and X_{out} with the two strokes discussed above.

Selection Algorithm. *envyLight* splits the original environment map L into two layers, a foreground L^f and a background L^b, such that $L = L^f + L^b$ and that edits to L^f modify the marked lighting feature directly. To compute L^f and L^b, *envyLight* takes as input the designer's strokes X_{in} and X_{out}, the transport t and the environment map L. We first compute the average transport for each of the strokes as

$$\bar{t}_{in}(\omega) = \frac{1}{|X_{in}|}\sum_{x \in X_{in}} t_x(w)\,; \quad \bar{t}_{out}(\omega) = \frac{1}{|X_{out}|}\sum_{x \in X_{out}} t_x(w)$$

Let us indicate with $\Delta = L(\omega)\bar{t}_{in}(\omega) - L(\omega)\bar{t}_{out}(\omega)$ the difference between the products of these average transports and the environment map. For highlights and diffuse gradients, we simply define

$$L^f(\omega) = \begin{cases} L(\omega) & if \Delta(\omega) > \epsilon \\ 0 & otherwise \end{cases} \; ; \quad L^b(\omega) = L(w) - L^f(\omega)$$

For shadows, we invert the order of the transports, corresponding to a comparison of $-\Delta(\omega) > \epsilon$. The ϵ value in the comparison ensures that locations with vanishingly small transport are excluded, together with compensating for designer imprecision in stroke placement. For all results in this paper, we fixed ϵ as $0.2\bar{L}\bar{\rho}$, where $\bar{L}$ is the average intensity of the input environment map and $\bar{\rho}$ the average of the scene albedo.

Example Selections. Example selections are presented in Fig. 2, where the first column shows the stroke pairs, and the second shows the difference of products. The *in* and *out* strokes are indicated in green and red while the difference of the product is green if $\Delta > 0$ and red otherwise. The third and fourth column of the figure show the split light layers together with images of the scene lit by them. Inspection of these images shows how well *envyLight* allows designers to isolate specific lighting features, from high contrast diffuse gradients (Fig. 2.a) to more subtle ones (Fig. 1.a), from splitting soft highlights from large lights (Fig. 2.b) to selecting sharp highlights on bumpy surfaces (Fig. 1.b), from sharp shadows of

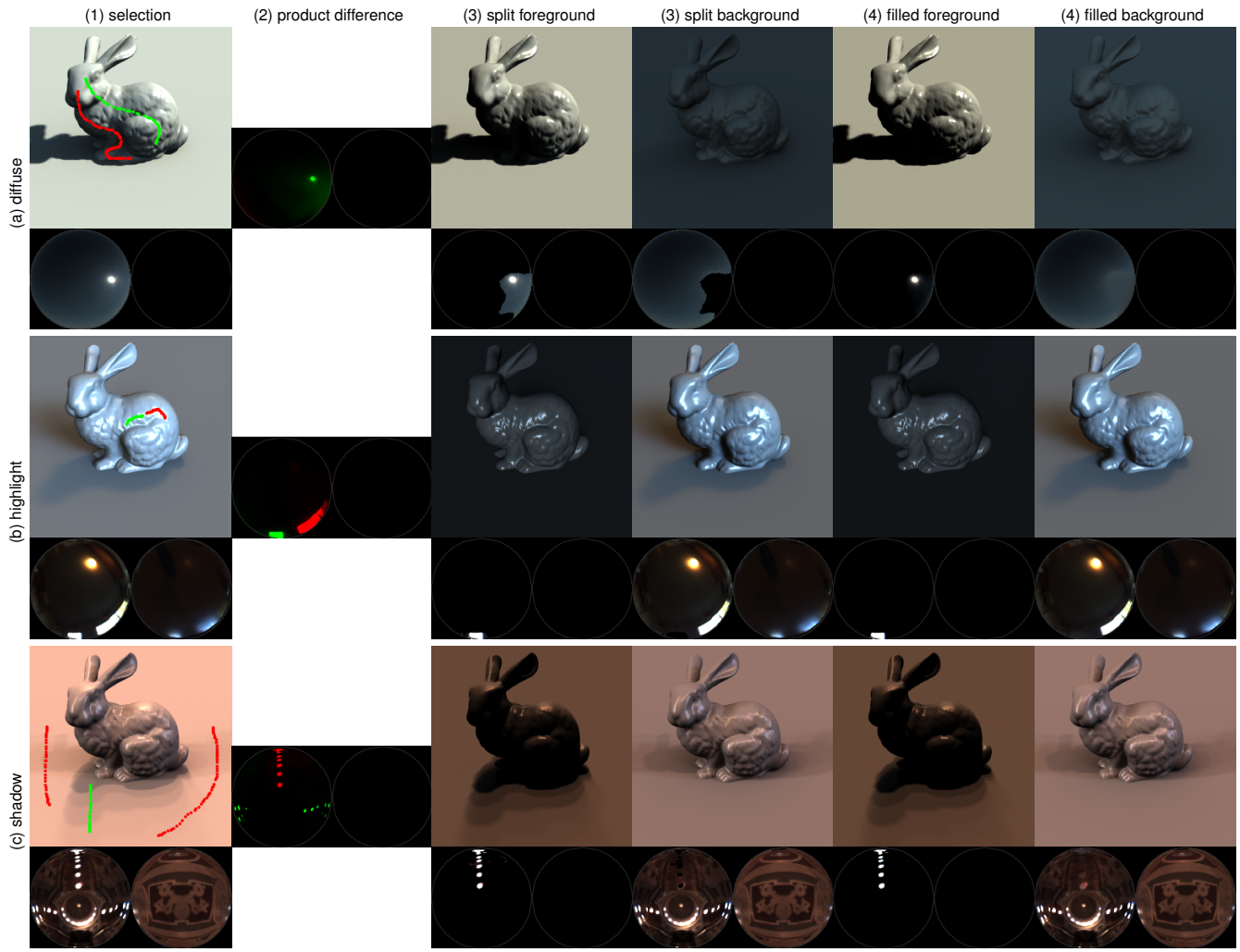

Figure 2: *Example selections performed with* envyLight. *(a) Diffuse gradient. (b) Highlight. (c) Shadow. (1) Designers mark lighting features with* in *(green) and* out *(red) strokes. (2) Our selection algorithm computes the product of the light and the difference between the average transports of the* in *and* out *strokes. Here we display the product in green if positive or red otherwise. (3) For highlights and diffuse gradients, environment map pixels are assigned to the foreground if the product is above an epsilon; to the background otherwise. For shadows, we invert the test. (4) Optionally, we can fill the background layer to avoid "holes" and correct the foreground appropriately.*

complex lights (Fig. 2.c) and shapes (Fig. 1.c) to soft shadows from large area sources (Fig. 3) and complex geometry (Fig. 6).

Fill-in. One potential drawback of this selection scheme is that the background layer has zero values where the foreground is selected. This is clearly visible in Fig. 2. While this provides well-separated features in the rendered images, a designer may be interested in preserving smoother background layers for layering operations in Photoshop-like editing tools. This desire needs to be balanced with the fact that adding energy to the background layer has the possible effect of decreasing the separation between features. Given these contradicting needs, *envyLight* provides a simple fill-in algorithm for the background, while leaving the choice to enable it or not to designers. Fig. 2 shows an example of filled background layers. All results in this paper where generated with fill-in enabled.

Since boundaries of the selection have low Δ, filling the background in the selected region with a smooth membrane is not likely to add significant difference to the separation. We considered creating a smooth membrane by using mean-value coordinates defined over the space of direction ω. We found though that using radial basis functions to compute a weighted average of edge values for each direction gave us similar results with much higher performance; we thus choose it for *envyLight*. Specifically, for each ω in the selected region, we compute an interpolated value $\hat{L}(\omega) = \sum_{\omega'} L(\omega')/\sin(\omega,\omega')/\sum_{\omega'} 1/\sin(\omega,\omega')$ where ω' are directions on the selection boundary. We use one over the sine of the angle between the two directions since this is a rough approximation of the mean value coordinate weighting over the sphere. The values of foreground and background layers in the selected region are then modified as $L^b(\omega) = min(L(\omega), \hat{L}(\omega))$ and $L^f(\omega) = L(\omega) - L^b(\omega)$ respectively. If the selection has more than one connected component, each of them is filled separately. While more complex inpainting methods would certainly provide more realistic filled regions, the need to preserve separation of the selected lighting feature makes them not directly applicable to our work; we leave to future work the exploration of additional methods.

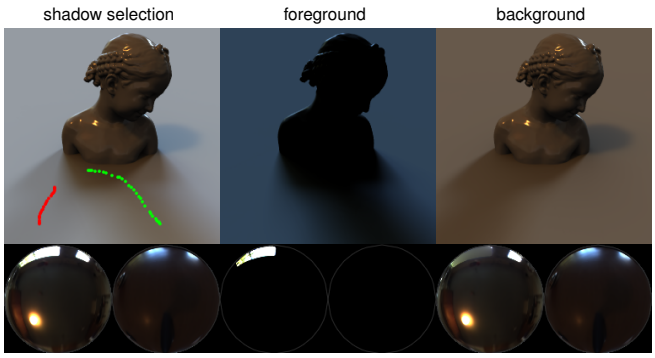

Figure 3: *Example of selecting soft overlapping shadows.*

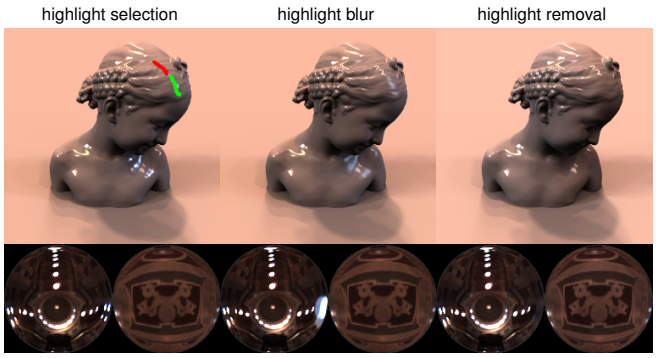

Figure 4: *Example of highlight edits.*

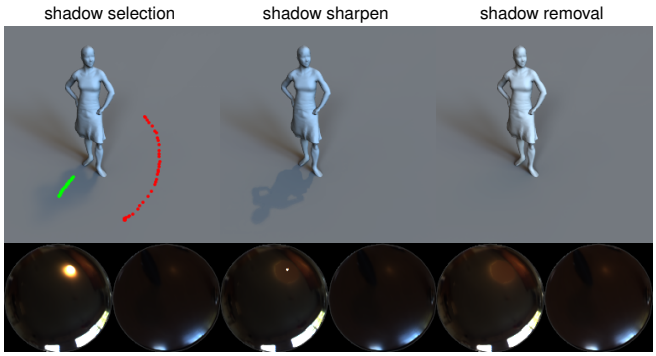

Figure 5: *Example of shadow edits.*

3.2 Editing Operations

Interface. Once the environment map is split into layers, editing operations can be applied to the layers themselves. We found, though, that applying standard image-processing operations to the environment map to achieve specific edits to the lighting features requires a lot of trial-and-error to determine the operation parameters. For example, changing the contrast of a lighting feature requires the concurrent scaling of the foreground and background intensities which is time consuming to perform. Translation and blurring of a lighting feature by applying image transforms and blurs are equally cumbersome since they work on projections with unavoidable distortions. In *envyLight*, we implemented a small set of simple editing operations we found useful in generating the results in this paper. We allow designers to quickly change the location, contrast and edge sharpness of a lighting feature using simple parameters. While we do not expect this list to be comprehensive, we hope these operations will demonstrate the effectiveness of the *envyLight* workflow and inspire designers to add other operations to their toolsets.

Brightness and Tint. Simply by scaling the values of the foreground map by a constant color, we can change the brightness and tint of the lighting feature. Specifically, we can write the edited foreground as $\widetilde{L}^f(\omega) = \alpha L^f(\omega)$. An example of this type of edit is presented in Fig. 1.a. Doing so though might change the overall tone of the image significantly.

Contrast. We would like an operation that changes the brightness of the lighting feature, while maintaing the brightness of its surrounding as constant as possible. This has the effect of altering the contrast of the lighting feature. We achieve this by scaling the foreground layer by a user-desired value, while scaling the background by an appropriate amount such that the average rendered color of pixels in the *out* stroke is unaltered. The edited foreground is defined as $\widetilde{L}^f(\omega) = \alpha L^f(\omega)$, while the edited background becomes $\widetilde{L}^b(\omega) = \beta L^b(\omega)$, where β is such that $\int \widetilde{L}\bar{t}_{out} = \int L\bar{t}_{out}$, i.e. $\beta = 1 + (1-\alpha)\left(\int L^f \bar{t}_{out}\right) / \left(\int L^b \bar{t}_{out}\right)$.

Examples of increasing contrasts are Fig. 1.a for diffuse gradients and Fig. 1.c. The contrast operation also provides an effective way to remove features by setting α to 0. Fig. 4 and Fig. 5 show highlight and shadow removal; diffuse gradients can be similarly removed. Note that for very large contrast increases, i.e. very large α, we cannot guarantee that the *out* stroke values remain unaltered, since β may become negative; said another way, as in tonal adjustment, there is a bound to how much contrast can be increased.

Translation. Lighting features can be translated by rotating the foreground layer. Designers can choose any interface to specify this rotation. In addition, *envyLight* provides an operation that allows users to specify the translation of lighting features directly in the rendered image by using two strokes: a reference stroke to indicate the original position of the feature and a target stroke to indicate its new desired location. We implemented two variants of this operation, one for highlights and one for shadows.

For highlights, we determine the reference ω^r and the target ω^t directions as those corresponding to the maximum value of the average transport for the reference and target strokes respectively. We then compute the rotation as the minimal rotation that aligns ω^r with ω^t. In [Pellacini et al. 2002], shadows are translated by rotating point lights around a pivot point determined as the intersection from a reference position to the light. We implement a similar scheme, where the pivot point is computed as the average location of the scene intersections corresponding to all points in the reference stroke. We compute these intersections by shooting rays from the direction of maximum intensity of the foreground layer toward each point in the reference. We then compute ω^r and ω^t as the average directions between the pivot and the reference and target points respectively. The use of strokes, rather than single points, gives these translation operations more robustness when used on complex geometry (Fig. 6) or bumpy surfaces (Fig. 1.b as demonstrated in the video). Note though, that these operations may still fail in cases where no rotation of the environment map can translate the feature to the desired location, i.e. if designers inadvertently specify an impossible target.

Blur. To blur the edges of highlights and shadows, we convolve the foreground layer with normalized spherical gaussians whose size $1/k$ is controlled by the designer. The edited foreground layer then

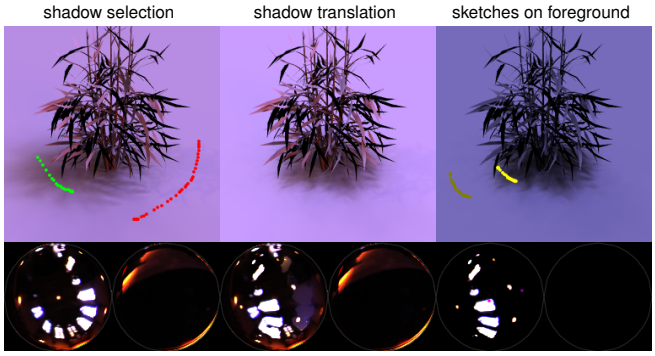

Figure 6: *Example of shadow translation from a reference (dark yellow) to a target (yellow) location marked by strokes on the foreground rendering.*

becomes $\widetilde{L}^f(\omega) = \int L^f(\omega')e^{k(\omega\cdot\omega')}d\omega' / \int e^{k(\omega\cdot\omega')}d\omega'$. Fig. 1.a shows blurring of shadow edges, while Fig. 4 shows blurring of a highlight. Note that blurring the environment map with an isotropic image-space kernel is not robust due to the unavoidable distortions in the projections.

Sharpen. To sharpen highlights and shadows, we shrink the foreground layer around a central direction ω^c. The edited layer can be defined as $\widetilde{L}^f(\omega) = L^f(R_\omega(\omega^c))$ where R_ω is the rotation around the vector $\omega^c \times \omega$ of an angle $s\theta$, where θ is the angle between ω and ω^c and s is a scaling ratio. In our implementation, we ensure that total foreground energy is not changed after shrinking to avoid tonal shift in the rendered image. By default, we choose ω^c to be the direction of brightest intensity in the foreground layer, but allow the designer to specify a different preference if so desired. If the selection has multiple connected components, we apply the operation independently on each. Fig. 5 shows an example of sharpening a shadow.

3.3 Animation

Interface. The *envyLight* workflow fits well with animation. Selection is performed on any one frame of the animation. Designers can then apply and preview edits to the layers at any other frame. The output of the editing operation is a new static environment map that can be used to render the whole animation. More interestingly, the simple parameters of our editing operations can be keyframed and interpolated over time. By applying the corresponding edit to map layers at each frame, *envyLight* computes a new environment map for each frame that can be used to light the corresponding frame. Specifically, we interpolate the scaling value α for brightness and contrast, the kernel size $1/k$ for blurring and the shrink ratio s for sharpening. Rotations defined by our various tools are also interpolated. This is demonstrated in the video. Note that this would not be possible when using image processing operations since their parameters are not directly linked to lighting features' appearances.

4 Implementation

Environment map representation. In our prototype implementation, environment maps are represented as cube maps. We edit maps at 128×128 resolution for each cube face.

Rendering. We implemented *envyLight* in a cinematic lighting setting, where camera and animation are fixed at the time of lighting. In our implementation, we use the all-frequency relighting algorithm of [Ng et al. 2003] to provide real-time previews. As a brief recap, this algorithm precomputes the transport at each image pixel and compresses it lossily using wavelets. At run-time, the environment map itself is compressed in the wavelet domain and a sparse vector multiply is performed for each pixel. We refer the reader to the original paper for a complete description. While other precomputed radiance transfer algorithms would have worked well (see [Wang et al. 2009] for a recent review), we chose this algorithm since it provides accurate previews while maintaining interactivity. We render images at 256×256 resolution with downsampled environment maps at 64×64. We provide anti-aliased previews by using 9 samples per pixel when computing transport. We keep 500 wavelet coefficients for the light and pixel transport for all scenes, except the *bush* scene where we use 1000, corresponding to cache sizes of roughly 0.5 and 1 GB respectively.

Since *envyLight* outputs standard environment maps after editing, any rendering algorithm can be used for the final rendering. We use our real-time preview to generate the images in this paper, while a raytracer was used to render the animation in the supplemental video.

Selection. To perform real-time selections, *envyLight* needs quick access to the transport values of the stroked pixels. We experimented with using the wavelet compressed values and found that compression artifacts were shown in the selection. We settled on storing the uncompressed transport on disk during precomputation. During interaction, the transport of the stroked pixels is loaded from disk and upsampled to the light resolution. While this allows *envyLight* to remain interactive, a better alternative might be to use GPUs to sample transport on-the-fly; we leave this to future work.

5 Results

Edits. We have already shown several selection and editing results throughout this paper. In this section, we will quickly summarize timings, and scene and light characteristics used in our testing. All edits shown in this paper were performed in a few seconds, including trial-and-error. The supplemental material includes high dynamic range images for the figures in this paper as well as several more editing examples.

Timings. We tested *envyLight* on a quad-core Intel processor running at 3 GHz with 4 GB of RAM with an NVidia 9600 GT. Our implementation uses 4 cores when rendering and for slower operations like environment map convolution. While editing, we found it useful to provide rendered previews for foreground and background lighting, together with their sum. *envyLight*'s execution speed is split into rendering, selection and applying editing operations, and varies with selection and edit type. In a typical session, selection takes roughly 0.3 s and is done once. Applying edits and rendering them takes 0.05 s and 0.2 s respectively and is done multiple times while adjusting editing parameters. As seen by these numbers and the supplemental video, *envyLight* allows designers to work interactively. Furthermore, since we use an image-based rendering algorithm, execution speed does not depend on scene complexity.

Scenes. We tested *envyLight* on four scenes, starting from the *bunny* scene (Fig. 2) containing a simple object. The *bimba* statue (Fig. 1, Fig. 3, Fig. 4), courtesy of CNR-IMATI/INRIA by Aim@Shape, combines fine details on the head with low curvature regions on the body, generating highlights that vary in size and shape significantly. The complex geometry of the *bush* [Deussen et al. 1998] (Fig. 1, Fig. 6) casts intricate overlapping shadows. Finally, the *samba* sequence [Vlasic et al. 2008] (Fig. 5) shows a realistic human and cloth animation. Materials in these scenes vary widely in their glossy property, from matte surfaces to the low-gloss *bunny* to the high-gloss *bimba*. We used the modified Phong model [Lafortune and Willems 1994] with specular exponents up to 500

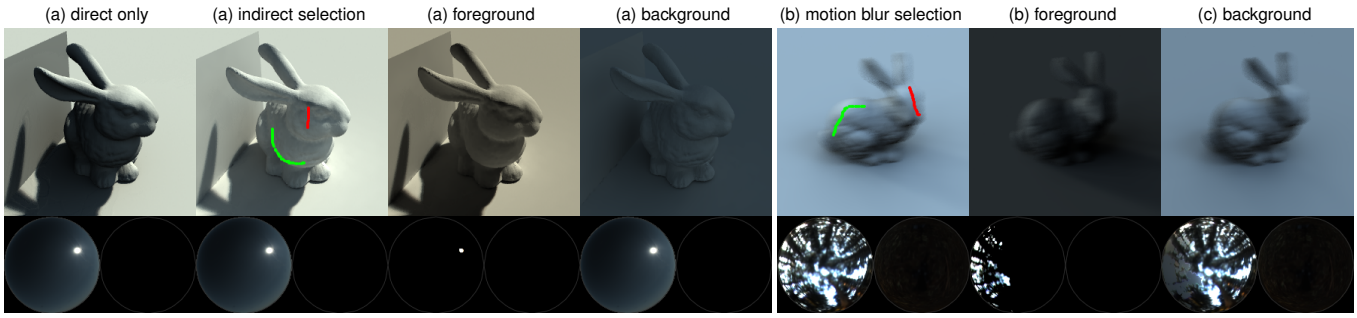

Figure 7: *By including (a) indirect illumination and (b) motion blur in the transport computation,* envyLight *selection algorithm can support editing under these additional features.*

to obtain very sharp highlights. We found *envyLight* selection and editing operations to work well in all these cases.

Environment Maps. We tested *envyLight* on a set of indoor and outdoor environment maps, from [Debevec 1998; Stumpfel et al. 2004], with very different illumination characteristics. We tested three indoor environment maps: *st. peters* (Fig. 2.c), *grace* (Fig. 1.a, Fig. 6) and *kitchen* (Fig. 2.b, Fig. 3, Fig. 5, Fig. 4). They render lighting features with middle contrast but whose size, number and edge sharpness vary greatly, e.g. from the sharp overlapping shadows of *st. peters*, to softer ones in *grace* to very blurry *kitchen*'s. We tested five outdoor environment maps: *clear* (Fig. 2.a), *sunny* (supplemental), *cloudy* (Fig. 1.c), *overcast* (supplemental) and *grove* (Fig. 1.b). They render lighting features with a wide range of contrast, e.g. from the high contrast shadows of *clear*, to lower contrast ones in *cloudy*, to the very subdued ones in *overcast*. *grove* is possibly the hardest case since the intricate tree shapes crossing the sky create a large number of subtle lighting features. We found *envyLight* selection and editing operations to work well in all these cases. We have included several additional edits as supplemental material to further strengthen this claim.

Extensions. While this work focuses primarily on editing under direct illumination, we found that the *envyLight* selection algorithm works well with indirect illumination, motion blur and depth-of-field, by simply including these effects in the transport computation. Fig. 7.a shows an example of selecting a bright spot in the environment map by marking its indirect contribution. Fig. 7.b shows a selection of diffuse gradients on a motion blurred object. While the *envyLight* contrast operation works in these cases, further investigation is needed to provide additional operations specifically tailored to indirect lighting.

6 Discussion and Limitations

Editing Speed. We found working with *envyLight* to be very efficient. Within a few seconds, we were able to isolate specific lighting features with *envyLight*'s selection, as well as edit them with *envyLight*'s editing operations. We found this speed to be typical, regardless of the complexity of the lighting environment. We believe this would have required much longer with image-based tools.

Robustness and Generality. The use of two strokes allows designers to mark differences of illumination they perceive as lighting features, regardless of how they were generated, and the selection algorithm will find which part of the environment map is responsible for this difference. The same interface works whether designers are selecting diffuse gradients, highlights or shadows, and regardless of the size, contrast or edge blur of the lighting feature, as can be seen in the various results presented in this paper and supplemental materials. This is in stark contrast with similar selection tools presented for point-like illumination (see [Kerr and Pellacini 2009] for a review), that require a different selection tool for each type of lighting feature, and cannot handle low contrast or overlapping features and blurry ones. Furthermore, *envyLight* is general with respect to surface materials, from matte to near-mirror finishes, as well as the complexity of objects' geometry.

Integrated Workflow. We believe that *envyLight* could be easily integrated into existing lighting design workflows. First, since *envyLight* adopts a layer-based *select-and-modify* workflow, it can work together with existing Photoshop-like image editing tools, that, to be best of our knowledge, are today the choice when editing measured environment maps. Furthermore, artists' familiarity with such workflows should be beneficial. Second, *envyLight* works with animated scenes, just like traditional toolsets, while also allowing edits to the environment maps to be keyframed, a requirement in animation production. Third, *envyLight* is quite simple to implement, thus lowering the cost of integration.

Limitations. *envyLight* selection has two main limitations. First, the interface returns empty selections if a designer places the *in* and *out* strokes on the same feature or on two different features that are generated by the same region of the environment map. While this is the stated semantic of our interface, a designer might inadvertently mark these features. For example, bumpy surfaces with mirror-like reflections often exhibit highlights that appear to form at random on the surface. In attempting to separate some highlights from the others, designers might inadvertently place strokes on different image points whose highlights correspond to the same environment map locations. Similar issues might arise with several overlapping shadows.

Second, when editing with *envyLight*, designers need to be aware that editing the selected lighting feature might affect others. For example, editing shadows will also change highlights. This coupling of lighting features is inherent in all physical lighting models. While experts are familiar with this, novices might not be. While we believe *envyLight* selection will be beneficial to novices, *envyLight* editing operations might still require more understanding of lighting than what novices have.

Comparison with Image-Based Editing. It is our belief that *envyLight* is a more efficient way to edit environment maps than using Photoshop-like tools. Throughout the development of this paper, we experimented with making selections and edits directly in Photoshop and found it to be remarkably cumbersome in all but the simplest cases. We also implemented a very simple interface to interactively paint environment map selections in our prototype, but still found it to require more trial-and-error than *envyLight*. It is our opinion, though, that a quantitative user study should be eventually

performed to compare these interfaces meaningfully. Doing so requires the implementation of a large subset of complex Photoshop tools within *envyLight*, a task we leave to future work.

7 Conclusion and Future Work

We presented *envyLight*, an interface for editing natural illumination, that combines sketch-based selection with efficient editing operations. *envyLight* works for a variety of lighting, material and geometry types, as well as animation. For the future, we are interested in exploring editing operations specifically tailored to editing indirect illumination effects as well as investigating the possibility of concurrent editing of materials and lighting to merge these two interrelated tasks that are typically performed separately.

Acknowledgements

We would like to thank William B. Kerr, Jonathan Denning and Lori Lorigo for their help in preparing this paper. This work was supported by NSF (CNS-070820, CCF-0746117), Intel and the Sloan Foundation.

References

ADOBE SYSTEMS INC, 2009. Photoshop CS 4.

ANRYS, F., AND DUTRÉ, P. 2004. Image based lighting design. In *4th IASTED International Conference on Visualization, Imaging, and Image Processing*.

AUTODESK INC, 2009. Maya 2009.

DEBEVEC, P. 1998. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *Proceedings of SIGGRAPH 98*, 189–198.

DEUSSEN, O., HANRAHAN, P. M., LINTERMANN, B., MECH, R., PHARR, M., AND PRUSINKIEWICZ, P. 1998. Realistic modeling and rendering of plant ecosystems. In *Proceedings of SIGGRAPH 98*, 275–286.

GLEICHER, M., AND WITKIN, A. 1992. Through-the-lens camera control. In *Proceedings of SIGGRAPH '92*, 331–340.

KERR, W. B., AND PELLACINI, F. 2009. Toward evaluating lighting design interface paradigms for novice users. *ACM Transactions on Graphics 28*, 3 (July), 26:1–26:9.

LAFORTUNE, E. P., AND WILLEMS, Y. D. 1994. Using the modified phong reflectance model for physically based rendering. Tech. rep., Katholieke Universiteit Leuven.

MOHAN, A., TUMBLIN, J., BODENHEIMER, B., GRIMM, C., AND BAILEY, R. 2005. Table-top computed lighting for practical digital photography. In *Rendering Techniques 2005: 16th Eurographics Workshop on Rendering*, 165–172.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-frequency shadows using non-linear wavelet lighting approximation. *ACM Transactions on Graphics 22*, 3 (July), 376–381.

OKABE, M., MATSUSHITA, Y., SHEN, L., AND IGARASHI, T. 2007. Illumination brush: Interactive design of all-frequency lighting. In *Proc. of Pacific Graphics*, 171–180.

PELLACINI, F., AND LAWRENCE, J. 2007. Appwand: Editing measured materials using appearance-driven optimization. *ACM Transactions on Graphics 26*, 3, 54:1–54:9.

PELLACINI, F., TOLE, P., AND GREENBERG, D. P. 2002. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics 21*, 3, 563–566.

PELLACINI, F., BATTAGLIA, F., MORLEY, R. K., AND FINKELSTEIN, A. 2007. Lighting with paint. *ACM Transactions on Graphics 26*, 2, 9:1–9:14.

POULIN, P., AND FOURNIER, A. 1992. Lights from highlights and shadows. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, 31–38.

POULIN, P., RATIB, K., AND JACQUES, M. 1997. Sketching shadows and highlights to position lights. In *Proceedings of Computer Graphics International 97*, 56–63.

RAMAMOORTHI, R., AND HANRAHAN, P. 2002. Frequency space environment map rendering. *ACM Transactions on Graphics 21*, 3 (July), 517–526.

SCHOENEMAN, C., DORSEY, J., SMITS, B., ARVO, J., AND GREENBERG, D. 1993. Painting with light. In *Proceedings of SIGGRAPH '93*, 143–146.

SLOAN, P.-P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Transactions on Graphics 21*, 3 (July), 527–536.

STUMPFEL, J., TCHOU, C., JONES, A., HAWKINS, T., WENGER, A., AND DEBEVEC, P. 2004. Direct hdr capture of the sun and sky. In *Proc. of AFRIGRAPH '04*, 145–149.

TSAI, Y.-T., AND SHIH, Z.-C. 2006. All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation. *ACM Transactions on Graphics 25*, 3 (July), 967–976.

VLASIC, D., BARAN, I., MATUSIK, W., AND POPOVIĆ, J. 2008. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics 27*, 3 (Aug.), 97:1–97:9.

WANG, J., REN, P., GONG, M., SNYDER, J., AND GUO, B. 2009. All-frequency rendering of dynamic, spatially-varying reflectance. *ACM Transactions on Graphics 28*, 5 (Dec.).