Steerable Illumination Textures

MICHAEL ASHIKHMIN
State University of New York at Stony Brook
and
PETER SHIRLEY
University of Utah

We introduce a new set of illumination basis functions designed for lighting bumpy surfaces. This lighting includes shadowing and interreflection. To create an image with a new light direction, only a linear combination of precomputed textures is required. This is possible by using a carefully selected set of steerable basis functions. Steerable basis lights have the property that they allow lights to move continuously without jarring visual artifacts. The new basis lights are shown to produce images of high visual quality with as few as 49 basis textures.

Categories and Subject Descriptors: I.3.7 [**Computer Graphics**]: Three-Dimensional Graphics and Realism

General Terms: Algorithms

Additional Key Words and Phrases: Bump mapping, displacement mapping, relighting, steerable functions, textures

1. INTRODUCTION

The color of a point on a diffuse surface is a function of only light direction and intensity, and thus does not vary with viewer position. This property has been heavily leveraged by radiosity researchers. It allows them to encode lighting for all viewpoints into a single texture for each surface. However, if the lighting changes, the textures must be recomputed. It is possible to store lighting for a variety of light positions and reconstruct lighting for a new position from these precomputed basis lights. The most sophisticated method for such reconstruction used *steerable lights*[1] [Nimeroff et al. 1994], where in a certain sense reconstruction is ideal, that is, the shape of the light is preserved as it moves across some domain. Unfortunately, with steerable lights, many basis images are needed to reconstruct basic effects such as shadows. This, despite the elegance of steerable technology, has prevented steerable functions from finding a major niche in graphics applications. If too few basis functions are used, the fidelity of lighting features is not captured. In this article, we apply steerable functions to the illumination of

[1]Note that this use of "steerable" has nothing to do with "steering" used in computational science, and is instead terminology from the computer vision community.

This work was supported by NSF grants 96-23614, 97-96136, and 98-18344 as well as start-up funds from the State University of New York at Stony Brook.
Authors' addresses: M. Ashikhmin, State University of New York at Stony Brook; P. Shirley, Computer Science Department, 50 Central Campus Drive, University of Utah, Salt Lake City, UT 84112; e-mail: shirley@hayduke.cs.utah.edu.

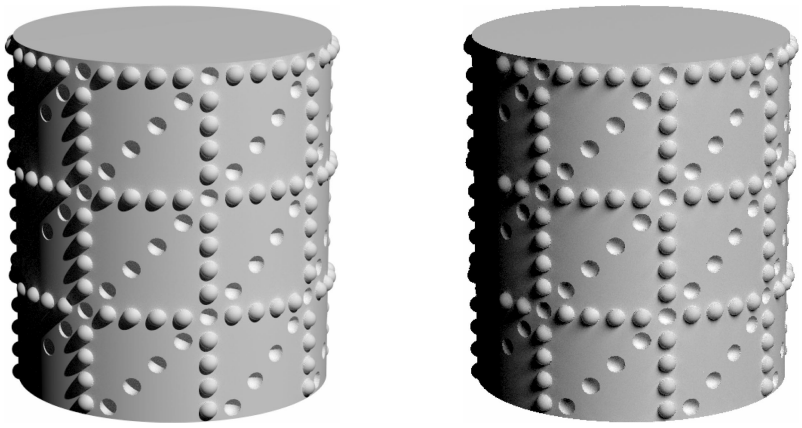

Fig. 1. Left: A bumpy cylinder with full shadows and interreflections from a small light source computed with batch global illumination. Note the interreflection in the dimples and on the bumps preventing a black color. Right: The same cylinder with a linear combination of 49 precomputed textures.

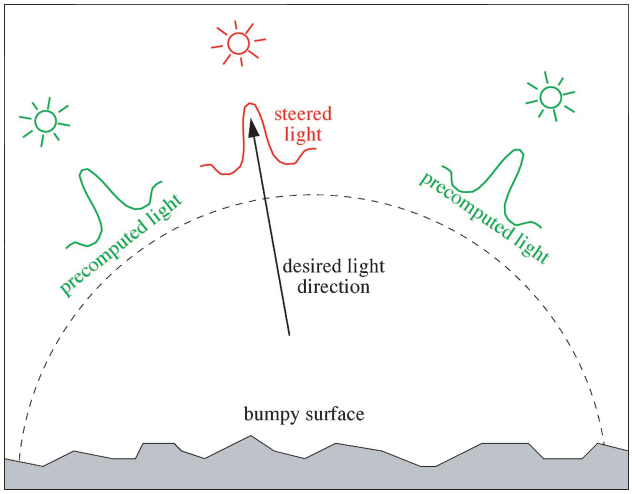

Fig. 2. The two green distributions are precomputed approximations that mimic a small light source. The red distribution is computed on the fly as a linear combination of several precomputed images, and can be moved continuously by changing the weights in the linear combination.

bumpy surfaces from distant distributed light sources. The lighting of such surfaces does not require the high accuracy needed in full scene illumination, and by carefully choosing basis functions we are able to relight such surfaces without an explosion in the size of the basis set. An example result is shown in Figure 1.

The basic idea of our technique is to take a bumpy surface and precompute a lighting texture for each of N lighting directions. Any of the these textures can be mapped directly onto a surface to give the results of applying an illumination computation. If we have a new lighting direction, we use the a linear combination of the precomputed illumination textures to create a new illumination texture. This idea is shown in Figure 2. The key idea in that figure is that the red lights are approximated by

the green radiance distributions. We linearly combine a number of the green distributions to create the red distribution. Because the green functions are part of a steerable basis, the red function can be reconstructed with the same shape as the green functions. In addition, the red distribution can be moved continuously over the hemisphere, so a continuous change in lighting can be simulated with no cross-fading artifacts. This is useful if the number of basis lights the number of basis lights N is not too large, and if we can find steerable lights that will reconstruct the types of lighting distributions we desire, such as the small area lights found in real scenes. This article argues that such lighting distributions do exist, although many careful choices must be made.

Our envisioned application is to the lighting of bumpy surfaces such as displacement maps. This could be performed in a Reyes-style renderer [Cook et al. 1987], ray tracing, or z-buffer environment. Our technique generates the intensity textures for height fields using just a linear combination of precomputed images, so in principle bumpy surfaces could be moved interactively with appropriate shadows and interreflection. Because we use very carefully chosen basis functions, we can get the desired lighting effects with a relatively small number of basis textures ($N = 49$) compared to alternative techniques.

There are two main contributions of this article:

—A new method of constructing approximately steerable lights is proposed. Although our technique is based on previously published methods, we apply it in a new domain: the hemisphere of directions. Most earlier approaches were designed to analyze images and were concerned about a single-parameter set of directions in a plane.

—A novel application of steerable functions to relighting bumpy surfaces is proposed.

In Section 2, we review previous work related to bumpy surface rendering and steerable functions. In Section 3, we review the basic mathematics of steerable functions. In Sections 4 and 5, we develop the new steerable basis set used for surface relighting in this paper. Section 6 provides practical implementation information. In Section 7, we present results, and conclude in Section 8.

2. PREVIOUS WORK

We review work related to the appearance of "bumpy" surfaces. We consider these surfaces as being defined by smooth underlying geometry and some kind of roughness or displacement function to give it "geometric texture". We also review some of the literature of steerable functions, as well as the few papers that have applied such functions to graphics problems.

Creating the geometry of bumpy surfaces via a procedure is usually called *displacement mapping*. This was introduced by Cook et al. [1987] and implemented via offsets to subdivision. Displacements are more difficult in a ray-tracing environment but several techniques have been developed [Heidrich and Seidel 1998; Pharr and Hanrahan 1996; Smits et al. 2000]. Recently, height-based displacements on polygons have been made practical for interactive settings using *relief textures* [Oliveira et al. 2000]. Our article does not address the issue of how geometric displacements are done. Rather it provides lighting for a displaced surface. It is well suited for any of the above methods when they have a standard (u, v) parameterization.

The color and lighting of bumpy surfaces was first addressed by texture mapping [Blinn and Newell 1976] and bump mapping [Blinn 1978] that modulated the reflectance and shading of a surface without changing the base geometry. Shadows for bump maps were addressed by Max [1988], and the scale and transition issues for bidirectional reflection distribution function (BRDF) to bump to displacement maps have been discussed by Becker et al. [1993]. The concept of a BRDF and (u, v) texture have been combined to encode more complex surface appearance by Dana et al. [1999] into a function

called the bidirection texture distribution function (BTDF). The BTDF encode occlusion properties as well as reflection properties of a surface. Information gathered from BTDFs have been used to efficiently render objects with changing viewpoints by preserving histogram properties [Ginneken et al. 1999]. A more brute-force technique is used by Dischler [1998] who stores a BTDF-like function at all points on a surface, along with a transparency index. This allows rendering of images of objects such as woven baskets. Heidrich et al. [2000] used a multipass algorithm to render globally illuminated displacement mapping by explicitly computing interreflections on the fly. Our technique is aimed at a different problem from any of these works except for the multipass technique of Heidrich et al. [2000]. The main difference between the two is that our technique uses precomputed images and theirs does not.

The general idea of using precomputed images to create new ones has been dealt with extensively in the *image-based rendering* (IBR) community. Typically, radiance point samples are taken and intermediate results are interpolated (e.g., Gortler et al. [1996] and Levoy and Hanrahan [1996]). This sampling can be adaptive to create arbitrary precision [Bala et al. 1999]. Precomputed images have been used to create novel relighting of scenes [Debevec et al. 2000; Dorsey et al. 1991, 1995], but that work did not explicitly use steerable lighting functions. The most related work to ours are the *polynomial texture maps* of Malzbender et al. [2001]. They take images of a surface under different illumination conditions and fit the color change at each pixel to a polynomial whose coefficients are then stored at each pixel. Their method works very well in practice with an extremely compact representation, but may not produce shadows in its current form.

The general tools of steerable functions have been used for a wide variety of applications, particularly in Computer Vision. A good literature review is available in Teo's [1998] recent dissertation. The use of approximate steering was introduced to 2D computer graphics by Gotsman [1994] for image filtering. Steering has received less attention in 3D computer graphics. It was first used by Nimeroff et al. [1994] for interpolating the indirect lighting on naturally lit scenes. Dobashi et al. [1995] implemented a system similar to Nimeroiff et al. [1994], but they used spherical harmonics, which are also steerable. The steerable properties of spherical harmonics had previously been harnessed by Sillion et al. [1991]. Ramamoorthi and Hanrahan [2001] have used spherical harmonics to create an error-bounded representation for irradiance from an environment map. Their method works extremely well, but makes the standard environment map assumption that all of the hemisphere of incident directions is visible, so local shadowing is not addressed.

Arvo [1995] used steerable phong lights in his dissertation, but did not explicitly use them in a steering framework. Teo et al. [1997] used steerable lights for theater lighting, where the light position had limited motion and only light direction was variable. They used SVD to reduce the basis size but their technique required generation of large number of images first our work differs from these in two ways. First, we restrict ourselves to the application of lighting bumpy surfaces, which allows us to use with relatively wide light distributions for reasons discussed in Section 4. This reduces the size of basis sets needed, and we believe large basis sets is the main drawback to steerable functions, especially for problems larger than one dimensions. Second, we design a special-purpose steerable light using several techniques including approximate steering. This further reduces the number of basis functions needed to a point we think makes the technique feasible for real applications.

3. BACKGROUND

In this section, we provide some general intuition of what steering is without attempting a mathematically rigorous presentation. There exist several somewhat different definitions of the steerability property, each of which reflects the needs of particular area it comes from. We call a function f *steerable*

if *all* transformed versions of the function can always be expressed as a linear combination of a *fixed*, *finite* set of basis functions $b(\vec{x})$:

$$f(\vec{x},\vec{x}_0)=\sum_{i=1}^{N}c_i(\vec{x}_0)b_i(\vec{x})=[c_1(\vec{x}_0)\cdots c_N(\vec{x}_0)][b_1(\vec{x})\cdots b_N(\vec{x})]^T \tag{1}$$

Functions $c_i(\vec{x}_0)$ are called *steering* functions (in contrast with a *steerable* function f) of steering coefficients. Depending on the application, "transformed" can mean one- or two-dimensional translation, rotation, scaling, etc., and $\vec{x}_0$ is the vector of parameters for the particular transformation, that is, a set of angles for rotations or displacements for translations. The usefulness of steerable functions relies on the fact that, if we are interested in responses of some linear operator L to a steerable function f with particular orientation, we can precompute a finite number of responses $L\{b_i\}$ for the basis functions b_i and then use this information to obtain a response to arbitrarily transformed f:

$$L\{f(\vec{x},\vec{x}_0)\}=L\left\{\sum_{i=1}^{N}c_i(\vec{x}_0)b_i(\vec{x})\right\}=\sum_{i=1}^{N}c_i(\vec{x}_0)L\{b_i(\vec{x})\}, \tag{2}$$

where L is any linear operator. Note that the c_is are constants for particular transformed f, which makes the second step in the last equation possible.

If we consider N particular positions $\vec{x}_0^1\cdots\vec{x}_0^N$ of functions f, we can write Eq. (1) in matrix form as $F=CB$ where $F=[f(\vec{x},\vec{x}_0^1)\cdots f(\vec{x},\vec{x}_0^N)]^T$, $B=[b_1(\vec{x})\cdots b_N(\vec{x})]^T$ and C is the corresponding coefficient matrix. Inverting this equation and substituting back into Eq. (1), we get

$$f(\vec{x},\vec{x}_0)=\sum_{i=1}^{N}([c_1(\vec{x}_0)\cdots c_N(\vec{x}_0)]C^{-1})_i\,f\left(\vec{x},\vec{x}_0^i\right), \tag{3}$$

which shows that N transformed versions $f(\vec{x},\vec{x}_0^i)$ of function f itself can be used as a basis. This particular basis is called a sampled basis since it consists of N samples of the function we want to steer. Notice that there are arbitrarily many different basis sets one can use to represent the same set of steerable functions, all related to each other through nondegenerate change of basis transformations. Applications using steerable functions can be considered a special case of a much wider class that uses linear combinations of some set of basis functions. As such, they inherit many nice properties of linear combination such as the applicability of rich mathematical apparatus of linear algebra and matrix analysis, computational efficiency, and simplicity of implementation.

What makes steerable functions unique is the amazing fact that a *continuum* of transformed versions of the *same* function can be represented by a *finite* basis set. This, obviously, is a severe restriction on this class of functions. Nevertheless, steerable functions are much more common than one might think. The simplest example is probably trigonometric functions in one dimension. A trigonometric identity states

$$\cos(\omega\phi-\phi_0)=\cos(\phi_0)\cos(\omega\phi)+\sin(\phi_0)\sin(\omega\phi). \tag{4}$$

This is exactly Eq. (1) with basis functions $\cos(\omega\phi)$ and $\sin(\omega\phi)$ and steering coefficients $\cos(\phi_0)$ and $\sin(\phi_0)$. This identity simply shows the familiar fact that an arbitrary shifted version of cosine is a linear combination of just two basis function with appropriately chosen coefficients. Because trigonometric functions are steerable and the Fourier decomposition provides a linear expansion in terms of sines and cosines, we can conclude that any function which has a discrete spectrum with only a finite number of terms in its Fourier series is steerable given enough basis functions. The Fourier decomposition for this class of functions has several useful optimality and uniqueness properties [Freeman and Adelson

1991; Perona 1995]. An example of a steerable function that does not belong to this class involves rotated versions of the first derivative of a 2D Gaussian that can be represented as linear combinations of the derivatives along the x and y axes. These functions are used in steerable filters and filter pyramids [Simoncelli and Freeman 1995], which are popular tools for feature detection on the early stages of computer vision and for texture synthesis [Heeger and Bergen 1995].

In this article, we are most interested in the case of 3D rotations; we call a function defined on a hemisphere *steerable* if an arbitrarily rotated version of this function can be expressed as a linear combination of a finite number of basis functions defined on the hemisphere. The analog of Fourier decomposition for the spherical domain is decomposition in spherical harmonics that are steerable with respect to 3D rotations. As a result, any function that can be written as a linear combination of a finite number of them is also steerable. Unlike with the 1D Fourier basis, dealing with spherical harmonics and related steering relations is both tedious and error-prone, but this is, of course, not the only possible basis set on a hemisphere. In the next two sections, we consider our approach to steering on a hemisphere and apply it to a particular problem of steered illumination for textures.

4. SCENE AND TEXTURE RELIGHTING

Of particular interest to us is the L in Eq. (2) that represents the rendering operator and f that represents light sources in the scene. It is well known that rendering operator is linear with respect to illumination [Kajiya 1986]. Therefore, if we use a light with an angular intensity distribution represented by a steerable function on a hemisphere, we can precompute a number of basis images and later combine them to produce an image corresponding to a light with arbitrary orientation. The remarkable fact is that as long as the viewpoint does not change, *all* light paths which took part in the computation of basis images will be correctly accounted for in the final image. In addition to direct lighting, this can include computationally expensive effects such as shadows, interreflections, and caustics. Moreover, a final rendering will not exhibit artifacts of a simple linear combination of ordinary (nonsteerable) light sources, such as cross-fading of shadows, caustics, specular reflection highlights and other features that depend on light direction. These artifacts are due to the different roles basis images and final results play in the usual linear combination. While basis images were computed with a "true" light source, the final result corresponds to a very different light. In the case of steerable lights, all the lights have the same shape. These properties were used by several researches to interactively relight scenes with arbitrary geometry and reflection properties for a fixed viewpoint [Dobashi et al. 1995; Nimeroff et al. 1994; Teo et al. 1997]. Changing the viewpoint introduces a nonlinearity to the rendering process and we cannot simply combine the basis images to obtain the correct result. This restricts previous applications of steerable functions in computer graphics to a rather specific, although important, domain of architectural and theatrical lighting design where one is often interested in one specific viewpoint.

Another problem with steerable functions is the large number of basis functions (and, therefore, images) needed to represent a typical light source. There is a version of "uncertainty principle" for steerable functions: the sharper the function is, the greater the number of basis functions needed to steer it over a given domain is. Basis size also grows with the size of this domain if the sharpness of the function is constant. Most light sources used in graphics are relatively sharp and we want to steer them over at least a hemisphere of directions. To make an approach based on precomputation and later linear combination of the results feasible, the number of basis functions should be as low as possible. It is reasonable to assume that this number should be below a hundred to be practical. These two facts together restrict current applications of steerable light sources in computer graphics to ones

with relatively broad intensity distributions, such as skylight [Nimeroff et al. 1994] or broad directional spotlights [Teo et al. 1997]. Note that the sharp shadows present in these papers were created using techniques other than steering very sharp light sources.

We present a novel application of steerable lights: bumpy texture relighting. On a high level, we design a suitable basis light set on a hemisphere, precompute and store the images of a texture lit by these lights, and then use them during runtime to create a texture lit from an arbitrary direction. These images are created from directly above the height field using an orthographic camera which makes them suitable for direct use with standard texture mapping techniques. There are several reasons to believe that this application is well suited for steerable functions.

First of all, illumination effects on bumpy textures are visually important, and arguably more important than visibility effects, especially in still images and far from the silhouette. This is what makes bump mapping, which also ignores visibility, useful. In fact, steerable textures can potentially replace bump maps for diffuse surfaces, since our method includes both shadows and interreflections and, in theory, only minor modifications are needed to the current texture mapping hardware to implement it. Another advantage of our method is that, unlike bump maps which resist antialiasing even for perfectly diffuse surfaces, any common antialiasing technique, such as mip-mapping [Willams 1983], can be used in our case. Since all runtime operations are linear, one simply prepares scaled-down versions of original basis images and performs steering on the appropriate level of the mip-map. Since we do not currently capture all light paths in the presence of viewpoint changes, directly applying our technique to a specular surface is more problematic. Another disadvantage is the extra space needed to store all the basis images.

Second, using a relatively broad light will not be as noticeable on textures as it would be in a more general environment since textures are usually not intended to be the center of attention. In addition, the human visual system is not highly sensitive to the accuracy of fine details of the lighting, shadows and interreflections as long as the overall character is conveyed correctly in an image. Moreover, many bumpy textures have random character and the viewer does not know how the surface should appear under a sharp light. This is usually not true in a general scene where one can easily tell that the light is broad. We do want the light to be as sharp as possible for a given number of basis functions and in the next section we describe a method we use to narrow down the light. To our knowledge, this work is the first attempt to use steerable functions to represent *direct* illumination due to common light sources and not just the very broad special purpose ones.

Finally, the whole purpose of texture is to inexpensively enhance the visual appearance of a surface. For this reason, and the ease of hardware implementation, the simple linear combination needed to obtain the final image from a precomputed set is very appealing. Until recently, such effects as illumination or visibility change due to surface bumps were considered too expensive to represent in textures. Relief textures [Oliveira et al. 2000] deal with the first problem by moving pixels in the original texture to new positions according to their depth values. Relief textures are well tailored for polygon rendering environment and our approach can be used together with this technique. If lighting direction is constant throughout a texture patch, which would be the case, for example, for a building facade under sunlight, steering coefficients will be constant for every pixel and we can first create a correctly illuminated texture by our technique and then warp pixels using the relief texture method. If lighting direction is also changing along a texture patch, as is the case for a polygon intended to represent a curved surface, we can apply steering to every pixel when its color is requested by relief texture algorithm to be pasted into the final image. We do not rely on any specific mechanism used by relief textures, all we need is a method to compute visibility. This means that steerable textures can be used, for example, in a raytracer. In fact, all images with correct bump visibility shown in this article use raytracing to compute it.

5. DESIGN OF BASIS LIGHTS

There are two main approaches to designing a basis for steerable functions: analytical and numerical. Early versions of the first approach [Freeman and Adelson 1991] assumed a certain steerable basis set and then attempted to find functions with needed properties in the linear subspace spanned by this basis. All scene relighting applications of steerable functions known to us follow this pattern. More sophisticated analytical approaches [Michaelis 1995; Teo and Hel-Or 1996] use relationship of steering property to the mathematical theory of Lie transformation groups. Basis sets constructed by these methods have some optimal properties but the process is rather involved and is usually restricted to one-parameter Lie groups or Abelian multi-parameter groups, which does not include the group of 3D rotations $SO(3)$ we are interested in. Numerical approaches [Gotsman 1994; Perona 1995] compute the set of basis and steering functions necessary to steer a given function under given family of transformations. They are usually more general but can be computationally expensive if number of transformation parameters is greater than two. In addition, the computed functions are available in the sampled form, and this is less convenient than an analytic expression given by the first family of methods. Teo and Hel-Or [1996] use a combined approach that first chooses a large set of analytic basis functions and then reduces this set using numerical technique. The method we present here is similar in spirit but works in the opposite direction: we first compute the basis numerically and then fit an analytic function to the result.

As previously noted, not every function can be steered exacly. It is always possible, however, to approximate a function with a steerable one. This is the approach we use. We choose a light source with desired properties and then attempt to find a steerable light which approximates it. For the special case of a hemisphere, Eq. (1) has the form:

$$f(\theta, \phi, \theta_0, \phi_0) = \sum_{i=1}^{N} c_i(\theta_0, \phi_0) b_i(\theta, \phi), \tag{5}$$

where (θ_0, ϕ_0) give axis orientation of the light source. We found that working directly on the hemisphere, however, is quite inconvenient. Fortunately, we do not have to. We note that if we have an arbitrary one-to-one mapping M from a hemisphere to a plane patch $(\theta, \phi) = M(x, y)$, we can rewrite Eq. (5) for the function f^p representing our light source in the plane as

$$\begin{aligned} f(\theta, \phi, \theta_0, \phi_0) &= f(M(x, y), M(x_0, y_0)) = f^p(x, y, x_0, y_0) \\ &= \sum_{i=1}^{N} c_i^p(x_0, y_0) b_i^p(x, y)) \\ &= \sum_{i=1}^{N} c_i^p(M^{-1}(\theta_0, \phi_0)) b_i^p(M^{-1}(\theta, \phi)) \\ &= \sum_{i=1}^{N} c_i(\theta_0, \phi_0) b_i(\theta, \phi), \end{aligned} \tag{6}$$

which shows that we can create a basis set b_i^p on a plane patch and then simply map it onto hemisphere using *arbitrary* (in particular, not necessarily linear) one-to-one mapping and the steering coefficients will be just transformed versions on the coefficients used in the plane. The price we have to pay for the more convenient domain is potential distortions of light source shape due to this mapping. On the other hand, we get an extra degree of flexibility since the mapping is arbitrary. For example, we can choose it in such a way that the shape of the light is the best when it is steered in some particular subregion of the hemisphere if this area is known to be more important. An example can be outdoor rendering

for given geographic area where the sun never visits some parts of the sky. Since we want to explore the most general case, in our method we simply use popular area preserving mapping from a point (x_h, y_h, z_h) in the upper hemisphere to a point (x, y) inside a circle of radius one:

$$(x, y) = \left(\frac{x_h}{\sqrt{1+z_h}}, \frac{y_h}{\sqrt{1+z_h}} \right). \tag{7}$$

We can now choose a light source f^p we would like to use on a plane and we choose a two-dimensional Gaussian in the form

$$f^p(x, y, x_0, y_0) = \exp(-((x-x_0)^2 + (y-y_0)^2)/0.01), \tag{8}$$

as our target light. Here (x_0, y_0) is the center of the Gaussian that we want to be able to put anywhere inside the unit circle. At this point, the choice of particular width (0.01) is rather arbitrary as long as the functional form conveys the fact that we want a narrow light source. A 2D Gaussian is a good candidate since it can be represented as a product of two one-dimensional gaussians. This means that we can further reduce the problem to finding a steerable approximation to a one-dimensional Gaussian defined on the interval $[-1, 1]$ and, once this is done, simply use a direct product of two such approximations as the steerable approximation in 2D:

$$\begin{aligned} f^p(x, y, x_0, y_0) &= f(x, x_0) f(y, y_0) \\ &\approx \left\{ \sum_{i=1}^{N_1} c_i(x_0) b_i(x) \right\} \left\{ \sum_{j=1}^{N_1} c_j(y_0) b_j(y) \right\} \\ &= \sum_{i=1}^{N_1} \sum_{j=1}^{N_1} \{c_i(x_0) c_j(y_0)\} \{b_i(x) b_j(y)\}. \end{aligned} \tag{9}$$

The 2D basis and steering coefficients are just the products of corresponding 1D quantities. The number of basis functions N in 2D will be square of the 1D basis size N_1. In this way, we obtain an approximation steerable with respect to 2D translation within the unit square, but only the area inside unit circle is actually used. The procedure does not guarantee that the approximation will be least-squares optimal in 2D even if optimal one-dimensional functions are used. We performed some experiments with a true 2D least-squares procedure using a straightforward extension of our approach described below for 1D. Although the resulting fully 2D approximations are slightly better, the difference is not sufficient, in our opinion, to sacrifice the simplicity of the direct product representation.

To design now our basis set numerically in one dimension, we follow the general approach of Gotsman [1994] who noted that the problem of finding basis functions and steering coefficients in Equation 1 to approximate some given, not necessarily steerable, function f can be cast as a singular value decomposition (SVD) problem. In particular, let us assume that f is a function of discrete variables x and x_0 each of which can take a value only from given vectors $\{x^1, \ldots, x^n\}$ and $\{x_0^1, \ldots, x_0^n\}$, both of which are uniformly sampled values on the interval $[-1; 1]$. We can form an n by n matrix with elements $F_{ij} = f(x^i, x_0^j)$. Using matrix SVD, we can write $F = U^T S V$ or, in the element form,

$$f(x, x_0) = \sum_{k=1}^{n} \sigma_k u_k(x) v_k(x_0), \tag{10}$$

where σ_k are singular values of matrix F (for which we will assume $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$), u_k and v_k are rows of orthonormal matrices U and V, respectively. But this is exactly the expression for steering if we set $c_k(x_0) = \sqrt{\sigma_k} v_k(x_0)$ and $b_k(x) = \sqrt{\sigma_k} u_k(x)$. Hence, SVD can be used to compute both basis functions and steerable coefficients for a function defined over discrete domain. The matrix F is symmetric in

our case because of the symmetry of f we chose. Thus, SVD becomes an eigenvalue decomposition that guarantees that $U = V$ and *both* basis functions and steering coefficients are given by the same functions.

For a continuous domain, we need to make n sufficiently large so that values of functions in between sampling points can be reliably interpolated. This convergence can be easily checked for by comparing the results for two different n values. If they are the same, we can assume convergence and use either of these values. For final computations, we used *Matlab* and set $n = 100$. The mathematical properties of SVD found in standard matrix analysis texts [Horn and Johnson 1990] guarantee that for a given N_1, the best possible approximation of $f(x, x_0)$ on the discrete domain by a sum of form of Eq. (10) where $u_k(x)$ and $v_k(x_0)$ are arbitrary vectors, will be obtained by performing SVD and simply truncating the sum after N_1 terms. We restrict ourselves to the case $N_1 = 7$, which produces a manageable number of 2D basis functions, $N = 49$. We consider this number to be a reasonable trade-off between quality of the lights (and images) that increases with extra basis functions and the work needed, both during preprocess and runtime, to create the textures. A more informed choice of the number of the basis lights can be based on recent research results on human perception of surface details depending of the width of the illuminating light source.

We now have the complete set of basis and steering functions, but they are represented numerically, as the output of SVD process. For steering with respect to a general transformation, it is hard to predict the analytic form of the basis functions produced by a purely numerical process we just described. This is demonstrated by results obtained by Gotsman [1994]. However, we have reduced the problem to steering with respect to a simple 1D translation and it would be reasonable to assume that a simple analytic functional form exists for our basis functions. Trigonometric functions are a good initial guess for this problem due to their steerability properties in 1D given by Eq. (4). Indeed, we were able to obtain an excellent fit of our numerical basis using analytic formula

$$b_i(x) = \alpha_i \cos\left(i\omega x + 3(i-1)\frac{\pi}{2}\right), \tag{11}$$

where $i = 1, \ldots, 7$, and ω and amplitudes α_i were the fitting parameters while the phase was fixed to ensure the orthogonality of the basis. We obtained the value of base frequency $\omega = 1.492$. Note that this base frequency is different from that of Fourier basis on $[-1; 1]$. Expressions for steering coefficients c_i are identical.

To avoid any confusion, we reiterate that the function representing the light source are only *approximately* steerable. One inevitable consequence of this is that their shape will change slightly as their position on the hemisphere changes. We have empirically found these distortions within acceptable perceptual limits for our main application of bumpy surface relighting.

Some results are presented on Figures 3 and 4, which show a steerable approximation to a narrow gaussian in one dimension obtained with our procedure and a direct product of two such approximations. Once mapped from the unit square to the upper hemisphere by the inverse of transformation (Eq. (7)), this function form the light source energy distribution.

Notice that the steerable approximation is much broader than the target Gaussian. This is a manifestation of the "uncertainty principle" we mentioned earlier, since we use only seven functions as the basis. This is the best (in the least squares sense) we can do with so few functions. An approach suggested by Simoncelli and Farid [1995] can also be used to get a sharp steerable light, but the first negative lobe of their function would be comparable in amplitude with the main lobe while in our case the ratio of these amplitudes is about one to five. Since we take a product of two such 1D functions to obtain the light, only a relatively small portion of the light energy concentrates outside main lobe, as demonstrated by Figure 4. For comparison, we also show a light of approximately the same width of the

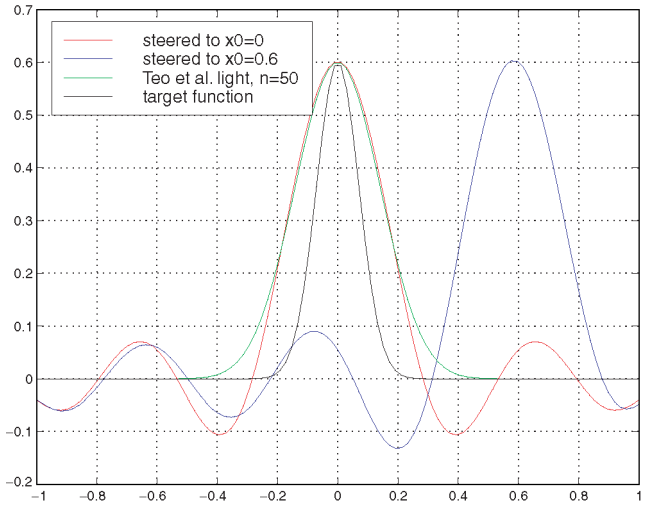

Fig. 3. Steerable approximation to a narrow Gaussian. To show relative width, the target function and Teo et al. [1997] light sources were normalized to have the same peak height as our results.

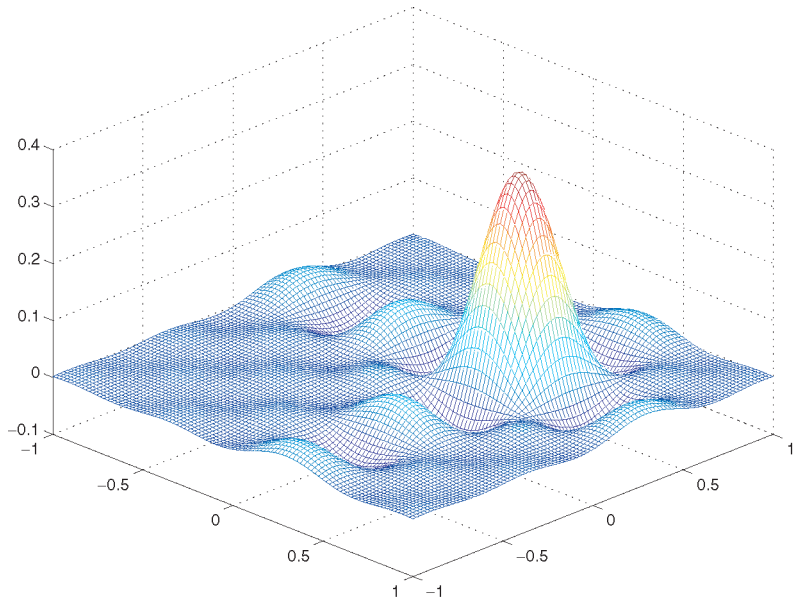

Fig. 4. Steerable approximation to a narrow Gaussian in 2D, steered to $x_0 = y_0 = 0.3$. Used to create images on Figure 8.

type used by Teo et al. [1997], namely $(1 + \cos)^n$, after transformation to the same coordinate system. The power n needed to achieve similar width is about 50 which corresponds to $(n + 1)^2 > 2500$ basis images if their set is to be used. Teo et al. [1997] present a method to reduce this basis set by using SVD to select only the most important images, but they must first compute all the basis images. Their light has the nice property of being always positive while our lights have significant negative lobes which

can potentially produce ringing artifacts in the final image. The results in the next section demonstrate that this effect is not objectionable. We believe this is a manifestation of limited sensitivity of human visual system to the fine details of lighting.

Similarly large number of basis functions is needed in an approach based on spherical harmonics [Dobashi et al. 1995] since the authors also approximate a $\cos^n$ light source and the order of Legendre polynomials involved can not be much lower than n. This approach also shares all our potential problems related to the oscillating tails of the light source. The authors use sigma factors [Hamming 1977] to solve the problem. This method can be used with our lights as well but this will broaden the lights somewhat. Since our target application is texture relighting, we believe it is better to have a sharper light and rely on imperfections of human visual system to hide the ripples. However, if our method is to be used for traditional lighting design where artifacts due to light oscillations are objectionable, Gibbs factors can be easily introduced in a straightforward manner, as described by Dobashi et al. [1995].

Our lights subtend a solid angle of about 0.25 steradians and are still much wider than some important light sources such as the Sun (less than one ten-thousandth of a steradian). However, if the object to be textured is stationary (which is true for many important objects, such as buildings or trees), the problem is reduced to one-dimension; all we need to consider is the Sun's movement along a single curve on the sky. In this case, the number of basis functions is reduced dramatically. For example, to steer a light of the width we discussed, one would need only seven (not 7^2) basis images. For most indoors scenes, the size of our lights is adequate to reproduce common illumination conditions, as clear from our examples.

6. IMPLEMENTATION

6.1 Basis Images

To use steerable lights, we first need to compute basis images. There are infinitely many potential basis sets possible. An advantage of the sampled set described in Section 3 (Eq. (3)) is that one can see how the real images will look and if our technique is useful for a particular application without rendering the complete set of basis images. We choose a sampled set as the set for which we compute basis images. We use 49 light axis positions on a 8-by-6 grid uniform in polar angle (we start from $\pi/3$ polar angle to avoid singularity of the coefficient matrix) and z coordinate, respectively. The direction of the last light coincides with the z axis. We chose these particular directions of the lights only for convenience. Any other set of 49 directions for which the coefficient matrix is nonsingular could have been chosen.

After the images are computed, we transform this set to the sinusoidal basis (Eq. (11)) by applying the inverse of the corresponding coefficient matrix (Section 3). Three example basis textures are shown in Figure 5. This is the basis used during rendering since steering coefficients in this basis are particularly simple (again given by Eq. (11)).

Of course, only some particular light intensity is captured in the basis set. To render an image lit with a different intensity light source, we need to be able to scale the result of steering process accordingly. Ideally, we would only need to multiply the images by the new light intensity, but the steered computation actually yields a small nonzero contribution at grazing angle due to the oscillating tails of the steerable light sources. Although this contribution is very small numerically, it is visually noticeable due to the sharp transition into the shadowed area where the normal of the base geometry faces away from the light. To fix this problem in a simple way, we use a linear map to scale the result of steered computation to enforce two boundary conditions. The first one is the value at normal light incidence, which we know from the simple analytic point light computation (for a normalized-to-one light intensity, it is just the albedo of the surface). The second condition is the value at grazing incidence,

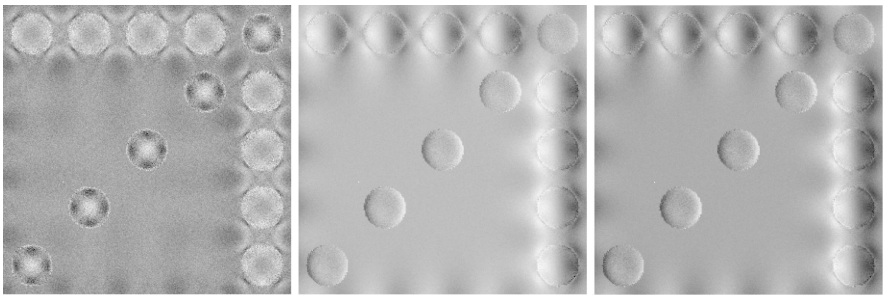

Fig. 5. Three basis textures derived from rendered textures. These textures contain both positive and negative values. They are stored as one-byte fixed-point images to reduce storage.

which we know should be zero. These two conditions are enough to precompute the coefficients of the linear map, which we apply to the result.

Basis image computation is a preprocessing step, so time is not very critical. We do, however, need a method able to deal with a *very* general light source. Due to both these factors, we use simple path tracing [Kajiya 1986] with large number of samples per pixel to compute the basis images. In this method, it is also easy to treat negative lobes in our lights: any path that originates in a negative lobe is treated as a positive one with the same absolute value but the result is subtracted from, instead of being added to, the pixel value. If the particular rendering method one wishes to use does not allow negative light values, it is easy to solve this problem at the cost of computing one extra image, the one with a perfectly diffuse light source. Then, due to linearity of rendering, one can add a constant to the basis light making it positive everywhere, compute the image with this new positive light and then subtract appropriately scaled diffuse light image. All steering computations are inherently floating point, so it is important to obtain the basis images in floating point format (in particular, without truncating negative values). We found, however, that we can uniformly quantize the basis images into interval $[0:255]$ and store only minimum and maximum values for each image as floating point numbers. This procedure does not affect visual quality (all textures in this paper were rendered using quantized basis images) but reduces storage requirements by a factor of four. We need to store 49*3 bytes per texture pixel. This number is comparable with other approaches based on aggressive precomputation [Dischler 1998; Heidrich et al. 2000]. For the grey textures such as those we use, only one color channel is needed.

6.2 Using Steerable Illumination Textures

Once the set of basis images is available, it constitutes a steerable illumination texture. Texture coordinates for a surface can be assigned through any standard technique. This can introduce distortions that, of course, will also distort the illumination information recorded in the steerable texture. Moreover, since we precomputed this information for a plane texture, it will be incorrect for highly curved surfaces. For the examples we worked with, this effect was not noticeable.

To use steerable textures during rendering, we do not need any information in addition to that already necessary for lighting calculation, that is, a coordinate frame on a surface and the direction to the light. In polygon-based rendering, if the normal and lighting direction stay constant for the entire surface (distant light source shining on a flat polygon), the set of coefficients is the same for the entire surface. If the normal direction changes, the basis images can be combined on either per-vertex or per-texel basis. The first method involves computing the set of coefficients at vertices and linearly interpolating them across the polygon. This is faster than computing a new set at every texel, but can break down if the normals at the vertices have substantially different directions (when compared with the half-width

of our light source). Similar considerations are valid for a change in light direction (a nearby light source). The advent of programmable shading hardware can make the direct implementation of both methods possible. One caveat is that the basis images must have linear intensities in them, that is, they should not have a nonlinear "gamma" value. Only the final image should be gamma corrected, and this is problematic for some graphics architectures. In a ray-tracing environment, it is easier to compute steering coefficients every time a ray hits a textured surface based on the local normal and light direction information.

7. RESULTS

7.1 General Scenes

Before we discuss our results for textures, we should note that there is nothing in our procedure described in Section 5, which relies on the fact that we want to use the lights we create to illuminate textures. This means that all previously explored applications of steerable light sources can benefit from the new method. One difference is that while all previous approaches ensured that the light has nonnegative intensity, our light sources have negative lobes and it is possible for the steering process to produce a negative number, which is clearly a nonphysical result. In such cases, we simply return zero. Note that we can truncate negatives in the result for particular direction *after* steering is done, but not in the basis images. Examples of a simple scene illuminated with our light source are shown in Figure 8. These images use the light position corresponding the intensity distribution shown on Figure 4. The mirror sphere shows the true intensity distribution of our light source. The remarkable fact is that for the both diffuse and moderately specular objects in the scene the lighting does not look that unusual. Under careful consideration, some artifacts due to oscillating parts of the light are noticeable in the shadowed region. The best way to alleviate this problem is to use sigma factors, as we discussed earlier. This example also demonstrates the limits of applicability of our method: highly specular objects positioned in such a way that they reflect the light source into the camera will exhibit strange reflection patterns.

Note that these results do not imply that one can use our technique for nondiffuse textures if the viewpoint changes. We only demonstrate the applicability of our light design method to previous applications of steerable light sources, all of which involve a fixed viewpoint.

7.2 Textures

We show our results on two representative textures: one with regularly spaced spherical bumps and dents and another created using Perlin noise. The regular texture presents worst-case scenario for our method since the shadows are clearly defined and the viewer can potentially see all the artifacts due to our unusual light sources. We choose a single-color bright diffuse textures and lit the samples with a single small light source to emphasize effects due to illumination. In a more diffusely lit environment, we expect our approach to perform even better since we already have to use rather broad lights which in this case will be more in-tune with the rest of illumination and will differ less from the exact result visually. Since the effects of global illumination usually change much more slowly than typical texture scale, we can simply treat the textured surface as flat and use only surface color information for the global illumination computation, as one would with ordinary textures.

If we do not have any method to compute visibility, steerable textures can be simply mapped onto the underlying surface as any other texture. The difference is that for each pixel its color is computed during runtime using linear combination of corresponding pixels in the basis set and coefficients computed through Eq. (11), see Section 6.2. Results of that process are shown in Figure 6. As expected, visibility

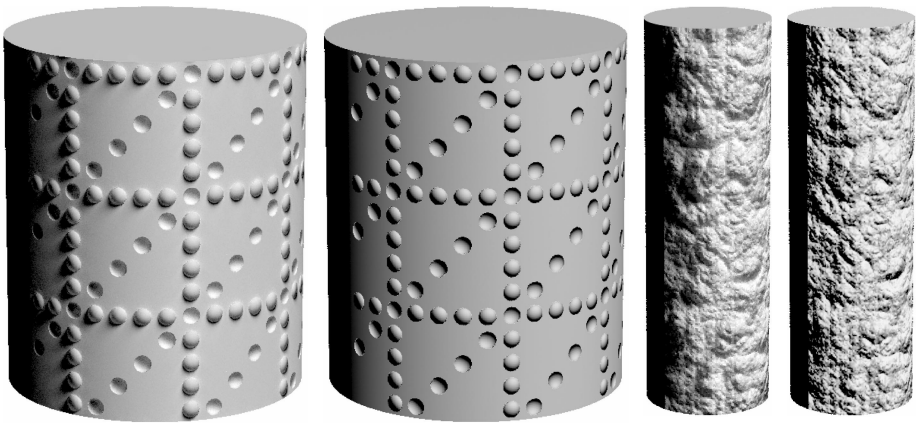

Fig. 6. Left to right: Steered, bump-mapped, steered, bump-mapped. All images lack correct bump visibility. Note that the steered images have both correct overall brightness due to interreflections (see Figure 7) and shadows.

artifacts are noticeable close to the silhouette, especially for the regular texture. For the random texture, the result is more acceptable, especially if a technique analogous to silhouette clipping [Sander et al. 2000] can be used. In both cases, however, lighting effects substantially improve the appearance of the surface by adding to the visual richness. These images are analogous to what would be computed using bump maps if shadows and interreflections could be included.

If we have the ability and the necessary information to compute visible surface point correctly, we can use texture coordinates for each pixel that are obtained taking into account visibility, rather than through the basic texture mapping process. The exact method used to compute visibility is not important. Our examples were rendered with a raytracer, but another technique such as relief textures could also be used with no change in the basic algorithm; only linear combinations of textures followed by standard texture mapping is needed. The results are shown in Figures 1 and 7 along with brute force renderings of the same geometries. The steered images are very close to the brute force "ground truth" images. These examples demonstrate that both negative lobes in our lights and their breadth do not produce visually objectionable artifacts and steerable textures can be used to approximate effects of illumination for diffuse bumpy surfaces.

7.3 Linear Interpolation Revisited

As we mentioned in Section 3, the steerable functions apparatus is a special case of general linear combination procedure. Another special case, which should be very familiar to the reader, is linear interpolation when the result is computed as a convex linear combination of nearest precomputed values. We can certainly use this approach for texture relighting with the advantage being that only a few basis images (at most four in the case of bilinear interpolation) are used to create the final image. This is smaller than the 49 coefficients which may be nonzero in our steering framework. Unfortunately, if a simple sharp light source is used to compute basis images for linear interpolation, the result is unsatisfactory due to cross-fading effect, as shown on Figure 9. To make a fair comparison among different approaches, we computed the same number of basis images (49) with the same light orientations for all the methods we studied.

In the other extreme case, ideal ambient lighting, there is no cross fading since there are no orientation-dependent effects and no distinct "position" of such a light. It is therefore interesting to ask a question of how well we can do with linear interpolation using a wider light source. Steerable

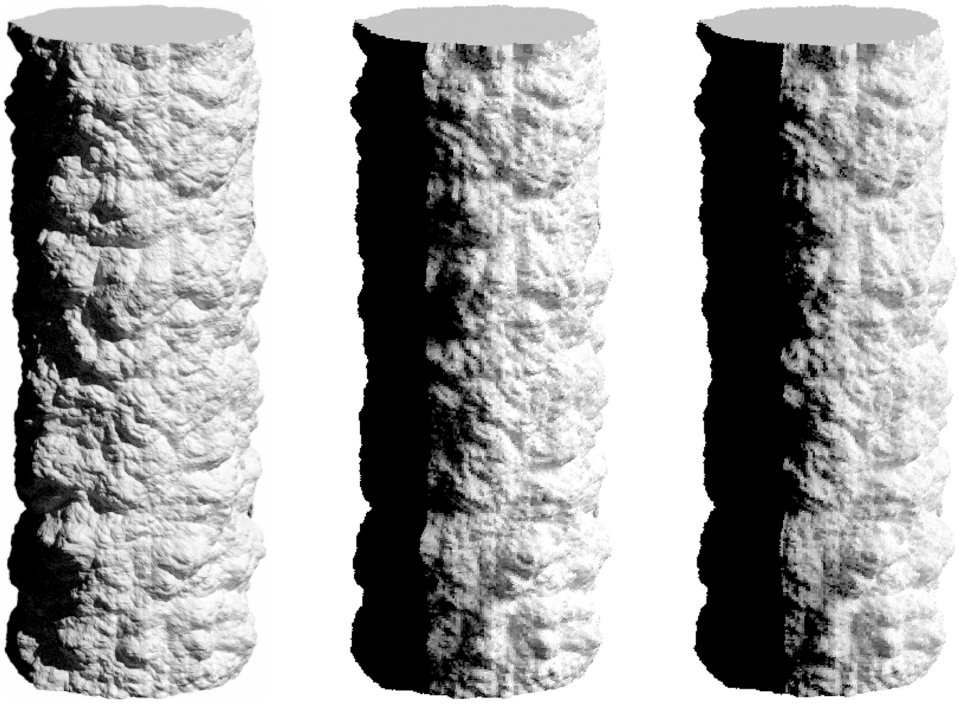

Fig. 7. Left: Full shadows and interreflections. Middle: A linear combination of 49 precomputed steerable textures. Right: A steerable light in a different direction. The same basis images were used for the middle and right figures.

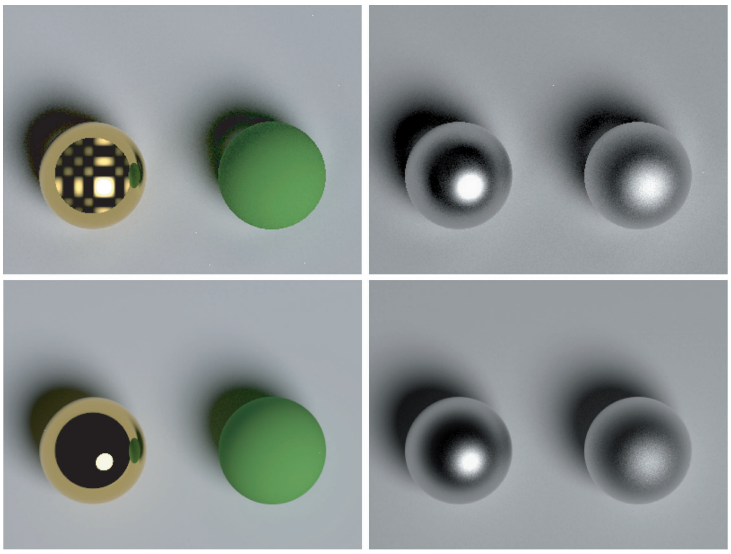

Fig. 8. Mirrored and diffuse and moderately specular spheres. Top: Steered light. Bottom: Real light with approximately the same width.

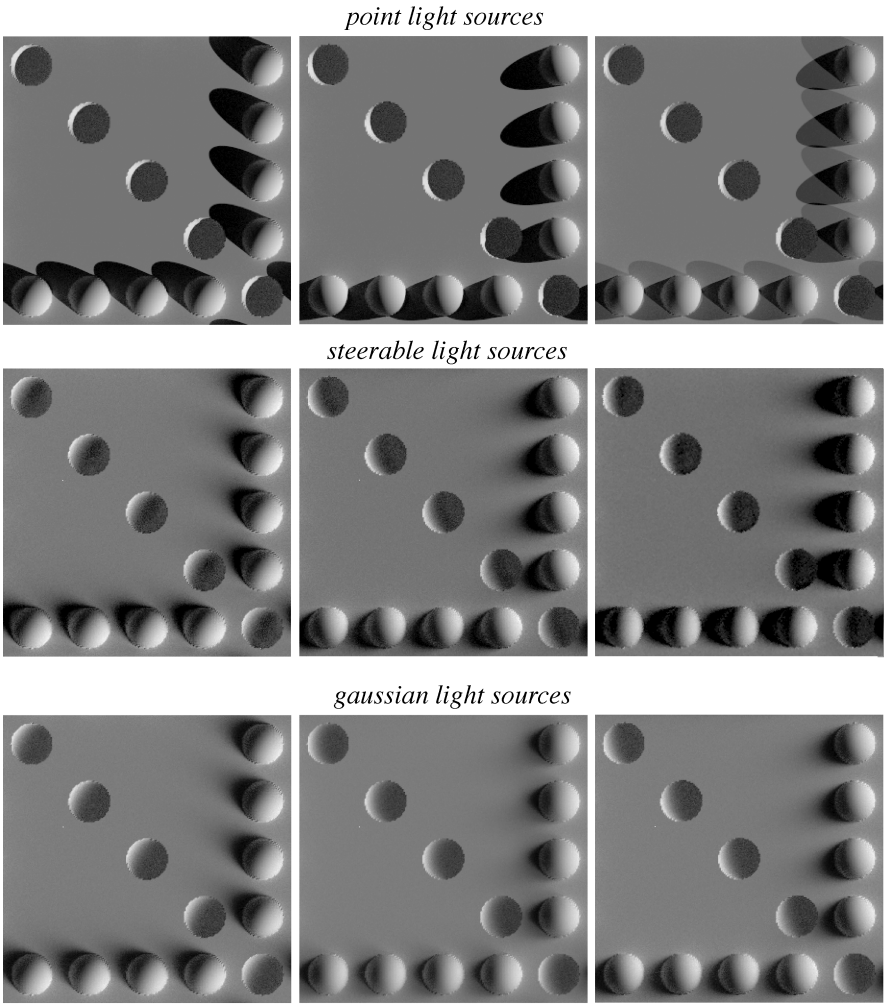

Fig. 9. Basis images (left, center) and the composite images for light shining from the right (right).

textures are, of course, free from this artifact and we designed the light to be as narrow as possible for a given number of basis functions. This leads to the idea of attempting a linear interpolation with a light, the width of which is equal to that of the main lobe of the steerable light. Of course, there is no reason for such a light to inherit the oscillatory behavior as well, so we use a simple 2D Gaussian:

$$f^p(x, y, x_0, y_0) = \exp(-29 * ((x - x_0)^2 + (y - y_0)^2)), \tag{12}$$

which we then map on the hemisphere using Eq. 7. Results of linear interpolation of such basis images are shown on Figure 9. One can see that while some shortening of shadows in interpolated images is noticeable, most of the cross-fading effect disappear. We believe that this simple method can be used in many cases for still images but in animations we saw it perform worse then truly steerable lights.

8. CONCLUSION

We have demonstrated that steerable illumination textures can provide visually plausible images with few basis functions. Because the steering textures are precomputed, they can include both shadows and interreflection. The runtime requirements (after preprocessing) of the framework discussed is only the linear combination of textures, so we believe this technique could be used in an interactive system such as a parallel ray tracer or a future hardware that includes a relief-texture extension and support for fast linear texture blends. The framework can also be used by batch renderers when the expense of computing "local" global illumination on the surface (i.e., among small bumps) is not practical. In that case, if more accuracy is needed, a larger basis set can be used without changing any part of our methodology. Larger basis sets allows the representation of narrower lights, and reduces artifacts due to oscillations of basis functions. The technique introduced could also be applied to any of the previous relighting applications that have used steerable lights. The resulting reduction in basis set size needed should improve the utility of that work.

REFERENCES

ARVO, J. 1995. *Analytic Methods for Simulated Light Transport*. Ph.D. dissertation, Yale University.

BALA, K., DORSEY, J., AND TELLER, S. 1999. Radiance interpolants for accelerated bounded-error ray tracing. *ACM Trans. Graphics 18*, 3 (July), 213–256.

BECKER, B. G. AND MAX, N. L. 1993. Smooth transitions between bump rendering algorithms. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '93*) (Anaheim, Calif., Aug. 1–6). ACM, New York, pp. 183–190.

BLINN, J. 1978. Simulation of wrinkled surfaces. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '78*). ACM, New York, pp. 286–292.

BLINN, J. AND NEWELL, M. 1976. Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (Oct.), 542–547.

COOK, R. L., CARPENTER, L., AND CATMULL, E. 1987. The Reyes image rendering architecture. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '87*) (July). ACM, New York, pp. 95–102.

DANA, K., VAN GINNEKEN, B., NAYAR, S., AND KOENDERINK, J. 1999. Reflectance and texture of real world surfaces. *ACM Trans. Graphics 18*, 1 (Jan.), 1–34.

DEBEVEC, P., HAWKINS, T., TCHOU, C., DUIKER, H.-P., SAROKIN, W., AND SAGAR, M. 2000. Acquiring the reflectance field of a human face. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '00*). ACM, New York, pp. 145–156.

DISCHLER, J.-M. 1998. Efficiently rendering macro geometric surface structures with bi-directional texture functions. In *Proceedings of the 9th Eurographics Workshop on Rendering* (July). pp. 169–180.

DOBASHI, Y., KANEDA, K., NAKATANI, H., AND YAMASHITA, H. 1995. A quick rendering method using basis functions for interactive lighting design. In *Proceedings of Eurographics*. pp. 229–240.

DORSEY, J., ARVO, J., AND GREENBERG, D. 1995. Interactive design of complex time dependent lighting. *IEEE Comput. Graph. Appl. 15*, 2 (Mar.), 26–36.

DORSEY, J. O., SILLION, F. X., AND GREENBERG, D. P. 1991. Design and simulation of opera lighting and projection effects. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '91*). ACM, New York, pp. 41–50.

FREEMAN, W. T. AND ADELSON, E. H. 1991. The design and use of steerable filters. *IEEE Trans. PAMI 13*, 6, 891–906.

GINNEKEN, B. V., KOENDERINK, J., AND DANA, K. J. 1999. Texture histograms as a function of irradiation and viewing direction. *Int. J. Comput. Vis. 31*, 2–3, 169–184.

GORTLER, S. J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M. F. 1996. The lumigraph. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '96*) (New Orleans, La., Aug. 4–9). ACM, New York, pp. 43–54.

GOTSMAN, C. 1994. Constant-time filtering by singular value decomposition. *Comput. Graph. Forum 13*, 2 (June), 153–163.

HAMMING, R. W. 1977. *Digital Filters*. Prentice-Hall, Englewood Cliffs, N.J.

HEEGER, D. J. AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '95*) (Los Angeles, Calif., Aug. 9–11). ACM, New York, pp. 229–238.

HEIDRICH, W., DAUBERT, K., KAUTZ, J., AND SEIDEL, H.-P. 2000. Illuminating micro geometry based on precomputed visibility. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '00*). ACM, New York, pp. 455–464.

HEIDRICH, W. AND SEIDEL, H.-P. 1998. Ray-tracing procedural displacement shaders. In *Proceedings of Graphics Interface*. pp. 8–16.

HORN, R. A. AND JOHNSON, C. R. 1990. *Matrix Analysis*. Cambridge University Press, Cambridge, Mass.

KAJIYA, J. T. 1986. The rendering equation. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '86*). ACM, New York, pp. 143–150.

LEVOY, M. AND HANRAHAN, P. 1996. Light field rendering. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '96*) (New Orleans, La., Aug. 4–9). ACM, New York, pp. 31–42.

MALZBENDER, T., GELB, D., AND WOLTERS, H. 2001. Polynomial texture maps. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '01*). ACM, New York, pp. 519–528.

MAX, N. 1988. Horizon mapping: shadows for bump mapped surfaces. *Vis. Comput. 4*, 109–117.

MICHAELIS, M. 1995. A lie goup approach to steerable filters. *Patt. Rec. Lett. 16*, 1165–1174.

NIMEROFF, J. S., SIMONCELLI, E., AND DORSEY, J. 1994. Efficient re-rendering of naturally illuminated environments. In *Proceedings of the Eurographics Workshop on Rendering*. pp. 359–373.

OLIVEIRA, M., BISHOP, G., AND MCALLISTER, D. 2000. Relief texture mapping. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '00*). ACM, New York, pp. 359–368.

PERONA, P. 1995. Deformable kernels for early vision. *IEEE Trans. PAMI 17*, 5, 488–499.

PHARR, M. AND HANRAHAN, P. 1996. Direct ray tracing of displacement mapped triangles. In *Proceedings of the Eurographics Workshop on Rendering*. pp. 31–40.

RAMAMOORTHI, R. AND HANRAHAN, P. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '01*). ACM, New York, pp. 497–500.

SANDER, P. V., GU, X., GORTLER, S. J., HOPPE, H., AND SNYDER, J. 2000. Silhouette clipping. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '00*). ACM, New York, pp. 327–334.

SILLION, F. X., ARVO, J., WESTIN, S., AND GREENBERG, D. 1991. A global illumination algorithm for general reflection distributions. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '91*). ACM, New York, pp. 187–196.

SIMONCELLI, E. P. AND FARID, H. 1995. Steerable wedge filters. In *Proceedings of the 5th International Conference on Computer Vision*. 186.

SIMONCELLI, E. P. AND FREEMAN, W. T. 1995. The steerable pyramid: A flexible architecture for multiscale derivative computation. *Int. Conf. Image Proc. 3*, 444.

SMITS, B., SHIRLEY, P., AND STARK, M. 2000. Direct ray tracing of displacement mapped triangles. In *Proceedings of the the Eurographics Workshop on Rendering*, pp. 307–318.

TEO, P. C. 1998. Theory and Applications of Steerable Functions. Ph.D. dissertation, Stanford Univ., Stanford, Calif.

TEO, P. C. AND HEL-OR, Y. 1996. Design of multi-parameter steerable functions using cascade basis reduction. Tech. Rep. CS-TN-96-32. Computer Science Dept., Stanford Univ., Stanford, Calif.

TEO, P. C., SIMONCELLI, E. P., AND HEEGER, D. J. 1997. Efficient linear re-rendering for interactive lighting design. Tech. Rep. CS-TN-97-60. Computer Science Dept., Stanford Univ., Stanford, Calif.

WILLAMS, L. 1983. Pyramidal parametrics. In *Proceedings of the ACM Conference on Computer Graphics* (*SIGGRAPH '83*). ACM, New York, pp. 1–11.

Received April 2001; revised October 2001; accepted October 2001