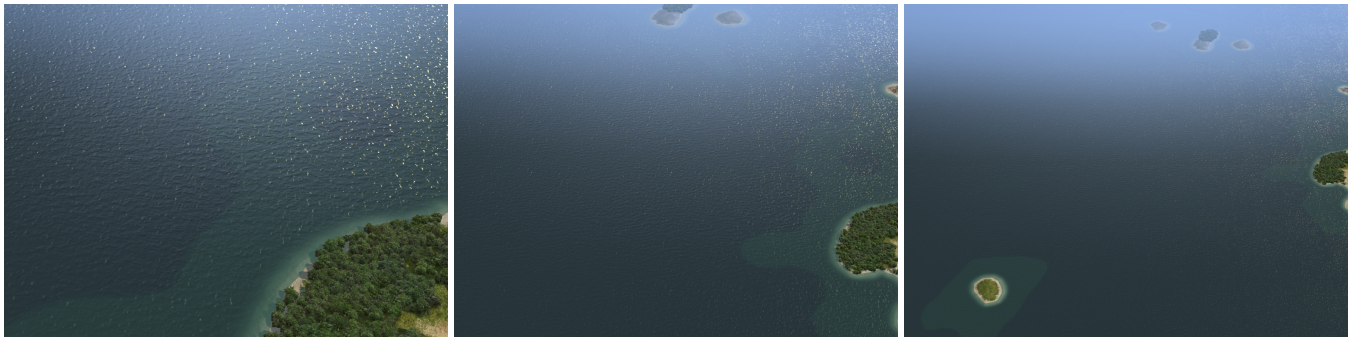
LEAN Mapping

Marc Olano*
Firaxis Games

Dan Baker†
Firaxis Games



(a) (b) (c)

Figure 1: *In-game views of a two-layer LEAN map ocean with sun just off screen to the right, and artist-selected shininess equivalent to a Blinn-Phong specular exponent of 13,777: (a) near, (b) mid, and (c) far. Note the lack of aliasing, even with an extremely high power.*

Abstract

We introduce Linear Efficient Antialiased Normal (LEAN) Mapping, a method for real-time filtering of specular highlights in bump and normal maps. The method evaluates bumps as part of a shading computation in the tangent space of the polygonal surface rather than in the tangent space of the individual bumps. By operating in a common tangent space, we are able to store information on the distribution of bump normals in a linearly-filterable form compatible with standard MIP and anisotropic filtering hardware. The necessary textures can be computed in a preprocess or generated in real-time on the GPU for time-varying normal maps. The method effectively captures the bloom in highlight shape as bumps become too small to see, and will even transform bump ridges into anisotropic shading. Unlike even more expensive methods, several layers can be combined cheaply during surface rendering, with per-pixel blending. Though the method is based on a modified Ward shading model, we show how to map between its parameters and those of a standard Blinn-Phong model for compatibility with existing art assets and pipelines, and demonstrate that both models produce equivalent results at the largest MIP levels.

CR Categories: I.3.7 [Three-Dimensional Graphics and Realism]: Surface Shading; I.3.3 [Picture/Image Generation]; G.3 [Probability and Statistics]

Keywords: bump maps, texture filtering, MIP mapping, shading models

*email: olano@umbc.edu (work done while on sabbatical from UMBC)
†email: dbaker@firaxis.com

1 Introduction

For over thirty years, bump mapping has been an effective method for adding apparent detail to a surface [Blinn 1978]. We use the term *bump mapping* to refer to both the original height texture that defines surface normal perturbation for shading, and the more common and general *normal mapping*, where the texture holds the actual surface normal. These methods are extremely common in video games, where the additional surface detail allows a rich visual experience without complex high-polygon models.

Unfortunately, bump mapping has serious drawbacks with filtering and antialiasing. When viewed at a distance, standard MIP mapping of a bump map can work for diffuse shading [Kilgard 2000], but fails to capture changes in specularity. A shiny but bumpy surface, seen far enough away that the bumps are no longer visible, should appear as if it were a duller surface, with formerly visible bumps becoming part of the surface microstructure. Bump mapping will instead produce a surface with the correct average normal but the original shininess (Figure 2(a-c)), which can lead to significant aliasing.

The problem is even worse for bumps with any repeated directional pattern. Bump directionality should result in anisotropic shading when the bumps are no longer individually discernible, much as with geometrically derived anisotropic shading models [Poulin and Fournier 1990]. Traditional bump maps instead revert to a symmetric highlight (Figure 2(d-f)).

Existing approaches either require precomputation too expensive to compute on the fly [Cabral et al. 1987; Fournier 1992; Westin et al. 1992; Schilling 1997; Han et al. 2007], large per-texel runtime data [Fournier 1992; Han et al. 2007], or significant approximations to the shading model [Olano and North 1997; Toksvig 2005]. Many use representations that do not combine linearly, violating a core assumption of standard texture filtering [Cabral et al. 1987; Westin et al. 1992; Schilling 1997]. We instead desire an approach that is fast, compatible with existing texture filtering hardware, and requires minimal precomputation to allow live changes to bump shapes. It should allow even extremely shiny surfaces without aliasing artifacts. As a further constraint, the method should work well with existing Blinn-Phong based lighting [Blinn 1977],

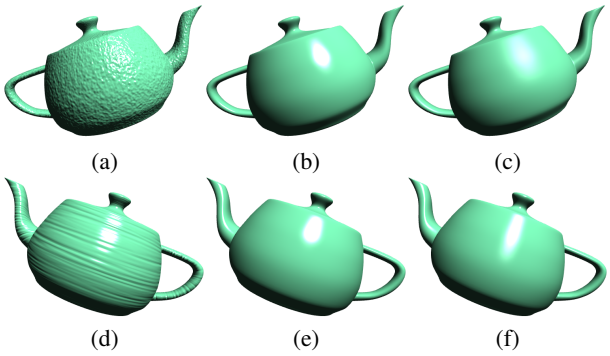


Figure 2: *Comparison of traditional bump mapping and LEAN mapping. (a) Finest MIP level of 512x512 bump map. Either method works at this level. (b) Coarsest MIP level of bump map with traditional bump filtering and Blinn-Phong shading. (c) Coarsest MIP level using LEAN mapping. (d) Finest MIP level of a ridged bump map. (e) Coarsest MIP level with traditional bump filtering. (f) Coarsest MIP level with LEAN mapping.*

to avoid heavy changes to existing game assets and art tools. Finally, we would like to be able to combine several layers of bumps, for time-varying effects, high-resolution decals, and detail textures.

We have developed LEAN Mapping (Linear Efficient Antialiased Normal Mapping) as a modification of the Ward [1992] shading model that computes bump shading as off-center probability distributions in the overall surface tangent space. This combined bump and shading model uses the linear mixing property of the mean and second moment of these distributions to allow filtered recombination of bumps. The new model requires just one additional MIP texture lookup per shading evaluation, and effectively captures antialiasing of highlight shape and transition of anisotropic bumps into an anisotropic highlight. Further, it is easy to generate the necessary textures on the GPU from an existing height or normal map, requiring just one rendering pass into two texture render targets plus standard MIP level generation. We have also developed three methods for combining multiple filtered bump maps at render-time, with varying tradeoffs depending on the per-frame rendering budget.

2 Related Work

Blinn [1978] invented bump mapping as a means of adding surface detail, using a scalar texture to define a displacement of the underlying surface. His bump maps have the advantage of being easily extended to true displacement maps, but require computing the partial derivatives at shading time. As a result, most recent implementations instead use a normal map [Cohen et al. 1998], commonly defined in the surface tangent space [Peercy et al. 1997].

Williams [1983] first noticed the problem of highlight aliasing with texture filtering, which Kajiya [1985] identified as part of a hierarchy of scales from surface changes to bumps to BRDF. Many approaches to bridge one or more stages of this hierarchy exceed our constraints for precomputation, for example using Monte-Carlo tracing of bump microstructure [Cabral et al. 1987; Westin et al. 1992; Becker and Max 1993].

Filtering of diffuse-shaded bumps can just use the un-normalized normals from linear texture filtering [Kilgard 2000]. The combination of filtered bumps and specular reflection is not so straightforward. One of the most active areas of research in specular normal filtering surrounds what representation to use for the combination of bumps and shading. Some operate in the tangent plane of the surface: Fournier [1992] numerically fits up to seven "Phong lobes" per MIP texel to the distribution of normals in the base level, totaling 56 per trilinear MIP lookup, while Schilling [1997] stores the variance of a 2D Gaussian distribution of normals, Toksvig [2005] estimates a 2D Gaussian based on normal length after MIP lookup, and Kautz and Seidel [2000] assume a fractal self-similar bump and BRDF distribution. Others assume a full 3D distribution of normals, either as a 3D Gaussian distribution [Olano and North 1997], or a mixture of Von Mises-Fischer distributions fit with spherical harmonics [Han et al. 2007]. None of these existing methods handle combination of bump layers.

$\vec{v}$	arbitrary vector
$\vec{v}.xy, \vec{v}.z$	subsets of components of $\vec{v}$
$\hat{v}$	$\vec{v}$ normalized to unit length
$\tilde{v}$	$\vec{v}$ projected onto its $z = 1$ plane: $\vec{v}.xy/\vec{v}.z$
$\vec{v}_n$	$\vec{v}$ in a space with the z axis aligned with $\vec{n}$
$\vec{n}$	base surface normal
$\vec{b}$	bump normal
$\vec{v}$	vector from surface toward viewer
$\vec{l}$	vector from surface toward light
$\vec{h}$	vector half way between $\vec{v}$ and $\vec{l}$: $(\hat{v} + \hat{l})$
s	Blinn-Phong specular exponent
Σ	2D covariance matrix

Figure 3: *Notation*

The most promising for our purposes are those that support a simple MIP-based reconstruction. Of these, Olano and North [1997] and Toksvig [2005] do not have the reconstruction accuracy we desire, and Han et al. [2007] is too expensive in precomputation and run-time use. Like our method, Schilling [1997] represents bumps as Gaussian variance, but in a form that does not truly combine linearly. Donnelly and Lauritzen [2006] showed linear texture filtering of Gaussian mean and second moments for Variance Shadow Mapping, a representation we adopt for bump filtering.

Microfacet shading models are particularly amenable to a combined bump-shading model, since they already assume a statistical distribution of normal facets. Many realistic shading models are based on a Beckmann distribution [Beckmann and Spizzichino 1963] over the hemisphere of normal directions above the surface [Cook and Torrance 1981; Ward 1992; He et al. 1991]. We use a Ward model with an added Fresnel term. This is essentially the same as the Cook-Torrance model without shadowing and masking terms used by Hara et al. [2005] and Han et al. [2007].

3 Method

We develop LEAN mapping as a simple model that is compatible with existing diffuse bump filtering, has low precomputation cost, low run-time cost, can be used alongside existing Blinn-Phong or Beckmann-distribution based shading models, and allows several approaches to the combination of multiple bump layers.

Notation used in this section is summarized in Figure 3.

3.1 Blinn-Phong/Beckmann Equivalence

Our bump specularity is a modification of the Ward model [Ward 1992]. The Ward model assumes perfectly reflective microfacets, randomly distributed around the overall surface normal. The Beckmann distribution, a Gaussian distribution of normal slopes, is evaluated at the half vector, $\vec{h}$, to give the expected number of facets that

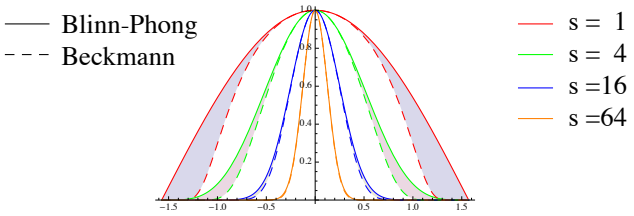


Figure 4: *Comparison of Blinn-Phong and Beckmann highlight shape as a function of angle between $\vec{n}$ and $\vec{h}$ (in radians). Fit improves dramatically with increasing s.*

the distribution predicts will be perfectly oriented to reflect $\vec{v}$ to $\vec{l}$.

To use it with existing Blinn-Phong-based game assets, we first show the equivalence of the Blinn-Phong model and a symmetric Ward model based on the Beckmann distribution. Lyon [1993] observed that Blinn-Phong approximates a Gaussian as the specular exponent s increases. From this, we observe that it also well approximates an isotropic Beckmann distribution with variance $1/s$ (See Figure 4). In terms of the angle θ between $\hat{n}$ and $\hat{h}$, we have

$$\cos(\theta)^s \approx e^{-\frac{s}{2}\tan^2\theta}.$$

The Beckmann distribution should be normalized by multiplying by $s/\sqrt{2\pi}$. The entire normalization factor for the largest MIP level could be incorporated into the specular coefficient. Instead, we include the multiplication by s in all instances of the Blinn-Phong model (but still fold the constant term into the specular coefficient). In fact, for our current game title, we were already using the s normalization of Blinn-Phong to avoid loss of apparent total brightness as the highlight tightens. Figure 5(a) and (b) show that these two models produce visually equivalent results.

3.2 Surface Beckmann

The Beckmann distribution is a 2D Gaussian distribution on the projection of the microfacet normals onto a plane one unit above the surface (Figure 6). Since we will be dealing with many such projections, we introduce the notation $\tilde{h}$ for the 2D projection of $\vec{h}$ onto the $z = 1$ plane. Beckmann-based shading models use a Gaussian centered at the origin on this plane. When applied to bump maps, each bump normal defines its own tangent plane for projection. One problem for previous attempts to MIP a combined bump and specular model is the difficulty in combining these distinct planes (Figure 7). Previous approaches have resolved the projection problem with distributions on a sphere [Olano and North 1997; Han et al. 2007] or by folding bump contribution into roughness at each level of the MIP pyramid [Schilling 1997]. We solve it by incorporating the bump normal direction into the Beckmann shading model. We use the normal of the underlying surface as the common projection plane, and represent the specular highlight on each bump as a Gaussian centered on the bump normal's projection onto this plane (Figure 8). Rather than the standard Beckmann distribution

$$\frac{1}{\sqrt{2\pi}|\Sigma|}e^{-\frac{1}{2}\tilde{h}_b^T\Sigma^{-1}\tilde{h}_b},$$

we use

$$\frac{1}{\sqrt{2\pi}|\Sigma|}e^{-\frac{1}{2}(\tilde{h}_n-\tilde{b}_n)^T\Sigma^{-1}(\tilde{h}_n-\tilde{b}_n)}. \quad (1)$$

Figure 5(c) shows that this model is visually equivalent to the standard Blinn-Phong and Beckmann models.

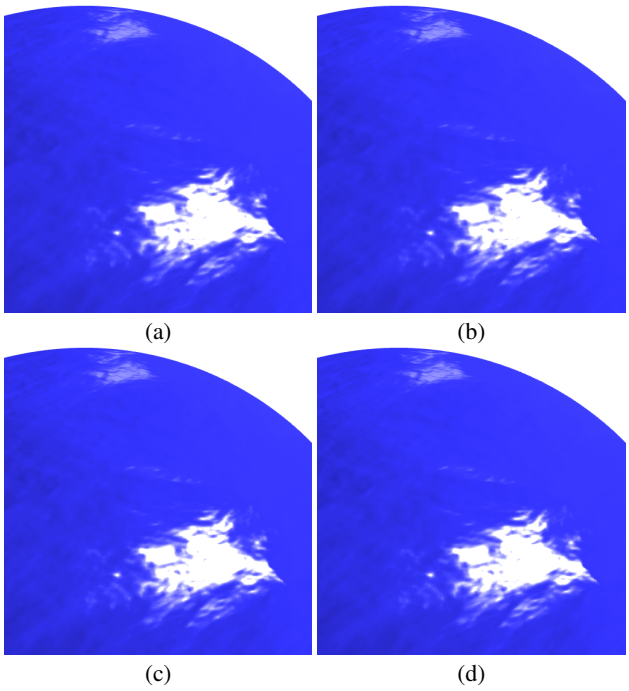


Figure 5: *Visual equivalence of shading models at a scale that avoids filtering, showing a sphere with both a frontal and glancing highlight: (a) Blinn-Phong applied to bump normal; (b) Beckmann in the bump tangent frame; (c) Beckmann in the surface tangent frame with bumps as off-center distributions; (d) LEAN mapping, mathematically equivalent to (c) at this scale.*

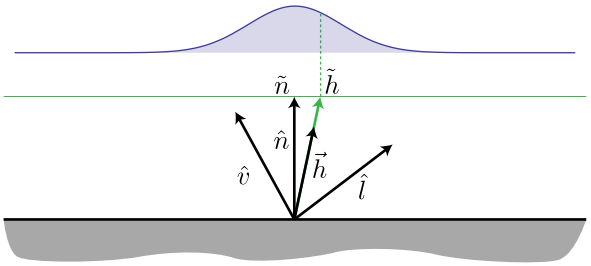


Figure 6: *2D illustration of a Beckmann distribution-based lighting model. Unit vectors $\hat{v}$ and $\hat{l}$ are used to compute $\vec{h}$ half way between them. $\vec{h}$ is projected onto a plane perpendicular to $\hat{n}$, and a Gaussian centered at 0 on this plane gives the specular reflection.*

3.3 Normal, Mean, Covariance, and Moments

Once we represent the microfacet distribution of each bump in a common plane, combining bumps from two texels into a new collective distribution is straightforward. Assume two texture samples in the MIP map have mean bump directions $\bar{b}$ and $\bar{b}'$. If each is an average of n base-map normals, we have

$$\bar{b} = \frac{1}{n}\sum_{1}^{n}\tilde{b}_i; \quad \bar{b}' = \frac{1}{n}\sum_{n+1}^{2n}\tilde{b}_i.$$

Combining these two to get a new mean bump direction over the joint $2n$ base-map normals, we have

$$\frac{1}{2n}\sum_{1}^{2n}\tilde{b}_i = \frac{1}{2}\bar{b} + \frac{1}{2}\bar{b}'.$$

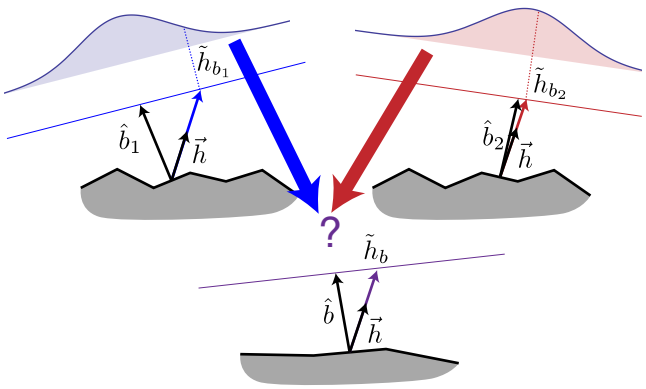


Figure 7: *Problem combining surface shading from distinct bumps. Each bump defines its own tangent plane, so combining shading contributions into a new plane is not well defined.*

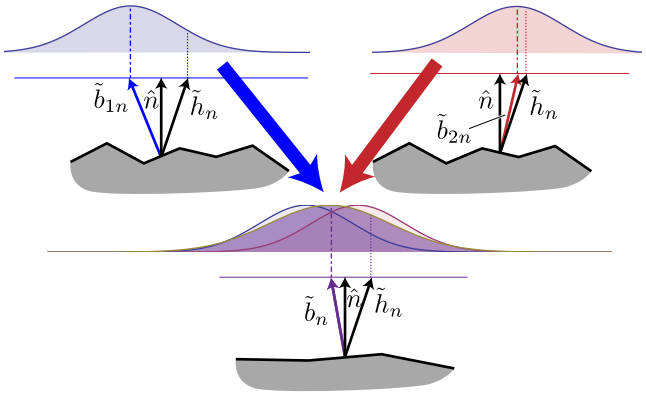


Figure 8: *New shading model incorporating bump direction as off-center Beckmann distributions in the surface tangent plane.*

The projected $\tilde{b}$ bump directions combine linearly, so any standard linear filtering kernel can be used to find the average bump normal at the center of a new microfacet distribution. This is true if we use box filtering, summed area tables [Crow 1984], elliptically weighted averaging [Heckbert 1989], Feline [McCormack et al. 1999], or any other linear filtering method. In particular, standard MIP level generation and hardware trilinear MIP or anisotropic texture lookups will all work.

This gives the highlight center, but does not affect its size or shape, which is controlled by the covariance matrix Σ in Equation (1),

$$\Sigma = \begin{bmatrix} (\frac{1}{n}\sum \tilde{b}.x^2) - \bar{b}.x^2 & (\frac{1}{n}\sum \tilde{b}.x\,\tilde{b}.y) - \bar{b}.x\,\bar{b}.y \\ (\frac{1}{n}\sum \tilde{b}.x\,\tilde{b}.y) - \bar{b}.x\,\bar{b}.y & (\frac{1}{n}\sum \tilde{b}.y^2) - \bar{b}.y^2 \end{bmatrix} \quad (2)$$

Schilling [1997] also used a covariance matrix based model, but stored an upper-triangular decomposition of the matrix. Neither the covariance matrix, nor upper-triangular decomposition combine linearly, but the second moments do, and can be used to reconstruct the elements of Equation (2) [Olano and North 1997; Donnelly and Lauritzen 2006]:

$$\frac{1}{n}\sum_1^n \tilde{b}.x^2; \quad \frac{1}{n}\sum_1^n \tilde{b}.x\,\tilde{b}.y; \quad \frac{1}{n}\sum_1^n \tilde{b}.y^2.$$

Since these take the same sum-over-microfacets form as $\tilde{b}$, they can also be stored in a texture and combined with any linear filter kernel. This combination of bump covariances in a common space are the key to transitioning large scale bump behavior into microfacet shading behavior.

To summarize, given a bump normal,

$$N = (\vec{b}_n.x,\ \vec{b}_n.y,\ \vec{b}_n.z),$$

the top level of a LEAN map texture is seeded with

$$B = (\tilde{b}_n.x,\ \tilde{b}_n.y) \quad (3)$$

$$M = (\tilde{b}_n.x^2,\ \tilde{b}_n.x\,\tilde{b}_n.y,\ \tilde{b}_n.y^2). \quad (4)$$

Recall that in our notation $\tilde{b}$ is a division by $\vec{b}.z$. Standard filtered texture sampling of these five values will give an antialiased and filtered blend of bumps and specular shading for any view. We can reconstruct the bump normal for diffuse shading from $(\tilde{b}.xy, 1)$. To save computation and improve the quality of the diffuse filtering, we can instead store the bump normal, N, directly in the three empty texture slots (assuming two four-element textures). This also allows diffuse filtering using the un-normalized normal after texture filtering [Kilgard 2000].

Any method can generate MIP levels (e.g. the driver's MIP filter chain), either as a preprocess or per-frame texture generation. During shading, trilinear or anisotropic sampling computes the correct filtering. Given filtered texture values, B and M (from maps seeded according to Equations ((3) and (4)), we reconstruct Σ

$$\Sigma = \begin{bmatrix} M.x - B.x * B.x & M.y - B.x * B.y \\ M.y - B.x * B.y & M.z - B.y * B.y \end{bmatrix}, \quad (5)$$

and use it in Equation (1). Though Equation (1) calls for Σ^{-1}, this is relatively trivial to compute since Σ is only a 2x2 matrix.

3.4 Base Surface Roughness

As with microfacet shading models, the previous section assumes the top level of the bump map consists of perfect mirror reflectors. This may be acceptable for environment mapping perfectly shiny surfaces, but generally we have surfaces with some base roughness. Han et al. [2007] show that an existing BRDF can be combined with a normal distribution by convolution. Intuitively, each facet of the normal distribution contributes its underlying BRDF kernel to the whole combined distribution. Han et al. use frequency space for this convolution. Fortunately, the Fourier transform of a Gaussian is another Gaussian with the inverse variance. Thus (ignoring normalization factors here for compactness) the convolution of a normal distribution with covariance Σ and shading with Blinn-Phong exponent s is

$$e^{-\frac{1}{2}(\tilde{h}_n-\tilde{b}_n)^T\Sigma^{-1}(\tilde{h}_n-\tilde{b}_n)} \otimes e^{-\frac{s}{2}(\tilde{h}_n-\tilde{b}_n)^T(\tilde{h}_n-\tilde{b}_n)}.$$

In frequency space this becomes

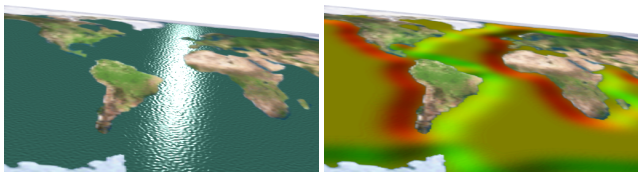
$$e^{-\frac{1}{2}(\tilde{h}_n-\tilde{b}_n)^T\Sigma(\tilde{h}_n-\tilde{b}_n)} e^{-\frac{1}{2s}(\tilde{h}_n-\tilde{b}_n)^T(\tilde{h}_n-\tilde{b}_n)}$$

$$= e^{-\frac{1}{2}(\tilde{h}_n-\tilde{b}_n)^T(\Sigma+\frac{1}{s}I)(\tilde{h}_n-\tilde{b}_n)}.$$

We don't actually need to do any computations in frequency space. We can compute the results of the convolution by just adding $1/s$ to the x^2 and y^2 terms of Σ when computing M in Equation (4):

$$M = (\tilde{b}_n.x^2 + 1/s,\ \tilde{b}_n.x\,\tilde{b}_n.y,\ \tilde{b}_n.y^2 + 1/s). \quad (6)$$

This has the effect of baking the Blinn-Phong specularity into the texture. Figure 5(d) shows that this is visually equivalent to the Blinn-Phong model. Alternately, we can add $1/s$ during final shading, when reconstructing Σ by Equation (5).



(a) (b)

Figure 11: *Ocean current simulation combining four moving layers of Gabor noise [Lagae et al. 2009]: (a) Still shot of the water. Ocean waves follow a clockwise flow around the Atlantic, with stable antialiasing as we zoom in and out; (b) Flow map used to weight moving water layers, computed as the curl of a distance map from the continents. Red and green channels give a flow vector, and layers are weighted in proportion to their movement direction.*

3.5 Combining Layers

In some cases, we want to combine multiple layers of bumps, either as an overlayed bump decal, or to support some moving bump element (ocean waves, shock waves, impact effects, etc.). Since the texture data is fast enough to create per frame, most situations where multiple layers of bumps are combined can just do so at the texture generation time (Figure 9(d),10(d)).

However, there are some cases when it is preferable to combine layers of bumps at render time. For example, we can create an ocean water surface with several layers of wave bumps moving in different directions. The entire ocean is too big to create a single texture at sufficient resolution to resolve the waves, but we can repeat a smaller wave texture across the surface and use a lower-resolution full ocean current texture to guide the mixing of layers. This allows global ocean currents (Figure 11), and helps disguise any texture repeat artifacts with differing mixing ratios (Figure 1).

Given bumps defined by a height field, $f(x, y)$, we can directly compute the projected bump normal, $\widetilde{bf}$ [Blinn 1978]:

$$\widetilde{bf} = (-\frac{\partial f}{\partial x}, -\frac{\partial f}{\partial y}).$$

For a linear combination of two height fields, $f(x, y)$ and $g(x, y)$ according to barycentric weights t and u, $t + u = 1$, this becomes:

$$\begin{aligned}\tilde{b} &= (-\frac{\partial(tf + ug)}{\partial x}, -\frac{\partial(tf + ug)}{\partial y}) \\ &= t\,\widetilde{bf} + u\,\widetilde{bg}\end{aligned}$$

So to find $\tilde{b}$ for a mix of height fields we just mix the projected normals. Since we don't need the height field for the mixing, we can use the same mixture model for bumps given as normal maps. Given a mix of $\tilde{b}$'s, either from original height fields or mixed normal maps, the mixing of second moments is slightly more complex. The second moments are

$$\begin{aligned}\tilde{b}.x^2 &= (t\,\widetilde{bf}.x + u\,\widetilde{bg}.x)^2 \\ &= t^2(\widetilde{bf}.x)^2 + u^2(\widetilde{bg}.x)^2 + 2\,t\,u\,(\widetilde{bf}.x\,\widetilde{bg}.x) \qquad (7)\\ \tilde{b}.y^2 &= (t\,\widetilde{bf}.y + u\,\widetilde{bg}.y)^2 \\ &= t^2(\widetilde{bf}.y)^2 + u^2(\widetilde{bg}.y)^2 + 2\,t\,u\,(\widetilde{bf}.y\,\widetilde{bg}.y) \qquad (8)\\ \tilde{b}.x\,\tilde{b}.y &= (t\,\widetilde{bf}.x + u\,\widetilde{bg}.x)(t\,\widetilde{bf}.y + u\,\widetilde{bg}.y) \\ &= t^2\,(\widetilde{bf}.x\,\widetilde{bf}.y) + u^2\,(\widetilde{bg}.x\,\widetilde{bg}.y) \\ &\quad + t\,u\,(\widetilde{bf}.x\,\widetilde{bg}.y) + t\,u\,(\widetilde{bg}.x\,\widetilde{bf}.y), \qquad (9)\end{aligned}$$

The blue terms in Equations (7), (8), and (9) are dependent on f or g alone, and are already stored as part of the standard LEAN map. The red terms involve both f and g, for a total of four extra fg cross terms. Note that these are independent of the mixing ratio between f and g. They can be computed and stored in a third texture, and just multiplied by the appropriate t u factor for rendering (Figure 9(e),10(e)). If the layers are moving relative to each other, the mixing textures will need to be recomputed each frame, but the mixing factors can be varied per pixel during rendering. For more layers, it is necessary to create these mixing textures for all pairwise combinations of the base layers, so while two layers need one mixing texture, three would require three mixing textures, four would need six, etc.

Rather than create and store all of these mixing textures, we can approximate them from the components of the f and g's B textures at rendering time (Equation (3)). This has the advantage of not requiring any additional textures or texture lookups, and eliminating all need for per-frame texture construction (Figure 9(f)). It will, however, produce incorrect results if the layers are coherent and reinforce or cancel each other enough to affect the overall filtered bump shading. For example, if f is a set of ridges and $g = 1 - f$, an equal mixing of both should reduce to a smooth surface. Using this approximation, the near view will still be correct. However, the distant shading will be anisotropic, since the mixing approximation cannot distinguish a mixing of two coherent anisotropic bump textures that reinforce each other from a mixture that cancel each other out (Figure 10(f)).

This is only a problem for highly correlated bump fields. In our applications, the cross terms are nowhere near as coherent, nor as stable, and we are able to use the mixing approximation with no visible artifacts.

These three solutions cover most bump layer mixing situations. Direct computation during LEAN map generation has the least run-time overhead, but requires either static maps or per-frame map generation, and needs sufficient LEAN map resolution (somewhat defeating the purpose of detail textures or high-resolution decals). Mixing textures allow per-pixel run-time remixing of textures, but also need a LEAN map resolution sufficient to cover the mix of all source bump maps. Finally, the mixing approximation does not need any additional data, even when mixing LEAN maps of differing resolutions, but can occasionally suffer artifacts, most commonly seen if using the same map (or its inverse) for more than one layer.

4 Results

LEAN maps work well on a mix of hardware and software. We have run versions of the LEAN map on a wide variety of platforms, including an Apple MacBook Pro with OpenGL on an NVIDIA 9600M GT, Windows Vista 64-bit with DirectX 11 on an AMD Radeon 5870, and DirectX 9 on a range of NVIDIA and ATI cards. Figure 12 shows some performance data. We compare regular Blinn-Phong bump mapping, LEAN mapping with a single precomputed LEAN map layer, LEAN mapping with the map textures generated each frame with automatic MIP generation, a mix of two-layers using precomputed mixture textures, and a mix of two layers using the mixture approximation. In all cases, the shaders included ambient and diffuse terms (but no diffuse color texture), and a Fresnel modulation of the specular term using the Schlick [1993] approximation. Frame rates were recorded at 1600x1200 with a single object filling the entire screen. Our frame rates of almost 1000 frames per second to over 1500 FPS on modern hardware clearly show that this is an efficient method, but since it is just one part of a game shader, frame rates are not too indicative of total perfor-

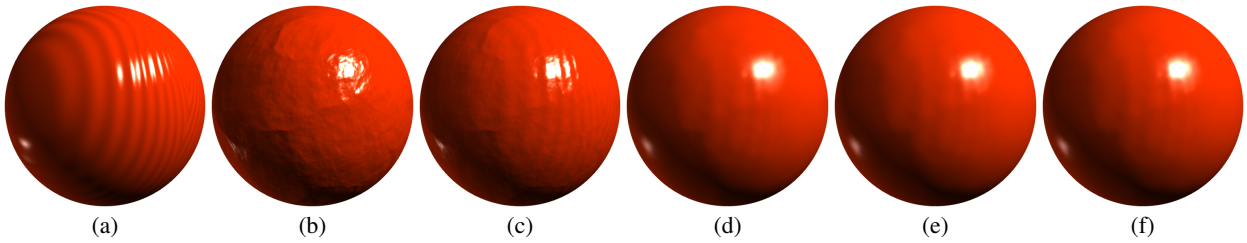


Figure 9: *Mixing of bump layers. (a,b) Each bump layer alone; (c) equal mixture of a and b at the finest MIP level; (d-f) comparison at a coarser MIP level (using MIP bias to emphasize differences): (d) mixed as height fields when generating a single LEAN map; (e) using mixture textures; (f) using mixture approximation. The mixture approximation works well when the layers are not coherent.*

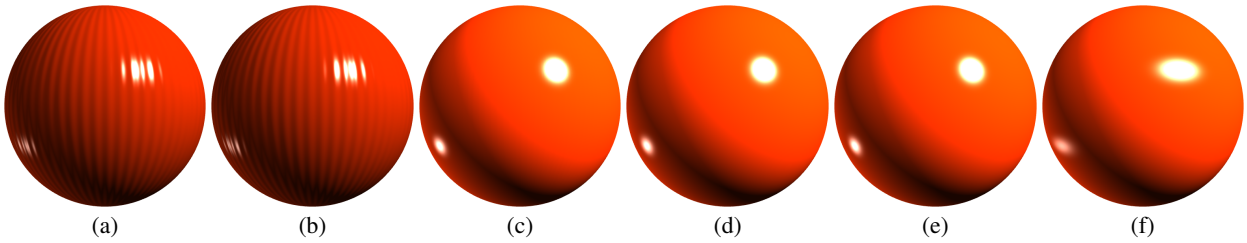


Figure 10: *Mixing of coherent bump layers,* $(\sin x)$ *and* $(-\sin x)$*: (a,b) Each bump layer alone; (c) equal mixture of a and b at the finest MIP level: bumps cancel out to a smooth surface; (d-f) comparison at the coarsest MIP level (using MIP bias to emphasize differences): (d) mixed as height fields when generating a single LEAN map; (e) using mixture textures; (f) using mixture approximation. The mixture approximation gets an elongated highlight from the anisotropy of the underlying layers.*

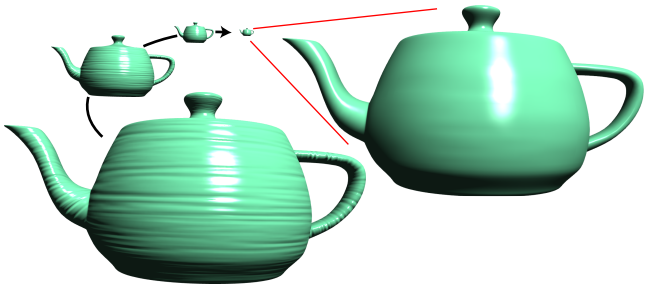


Figure 13: *Anisotropic bump pattern as a model moves away.*

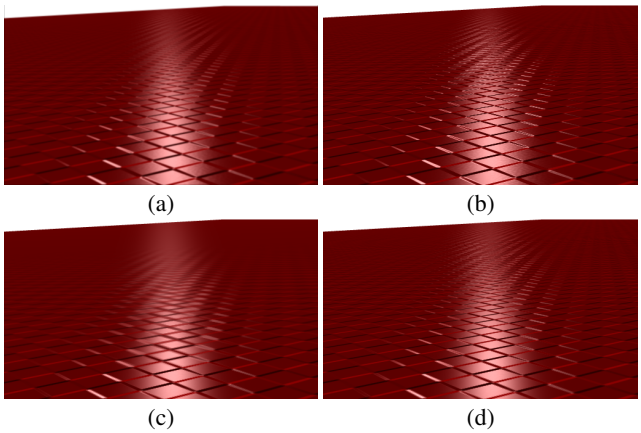


Figure 14: *Bump antialiasing on a raised checker grid: (a) "ground truth" computed with 64x supersampling and a Gaussian pixel reconstruction filter; (b) ordinary bump maps, exhibiting significant highlight aliasing; (c) LEAN mapping with trilinear MIP filtering, fixes the bump aliasing and exhibits checker-aligned anisotropic shading; (d) LEAN mapping with hardware anisotropic texturing, fixes some over-blurring of the primary highlight.*

mance. We also provide Direct3D instruction counts, in ALU and texture instructions for each. These correspond to between 6 and 12 cycles for each shader.

All images in this paper have an isotropic base roughness, with specular exponents ranging 128 to about $14,000$. Any shading anisotropy is derived from the bumps themselves. Figure 2(d-f) show an isotropic bump distribution turning into a more diffuse highlight. Figure 2(a-c) show the same model with an anisotropic bump pattern. Figure 13 shows the same bump pattern at natural scale, showing antialiasing and filtering.

Figure 14 shows antialiasing of a raised checker pattern. Notice the significant distance aliasing in the unfiltered version. In the trilinear MIP-filtered version, the aliasing is gone, and there is some checker-aligned distant highlight spread, but the elongated main highlight diffuses out a little early. The anisotropic-filtered version fixes the over-blurring of the main highlight while retaining the other filtering advantages.

Figure 1 shows a two-layer LEAN map ocean in a game currently under development. The ocean consists of two bump layers moving in different directions. The main highlight is just off-screen to the right, but peripheral highlights are visible as glints on the water. The artist-selected base specular power is 13,777, added to the covariance at run-time rather than folded directly into the LEAN map. The 16-bit LEAN maps are generated at load time from source height-maps with render-to-texture passes using an 8-tap Sobel filter to estimate the per-layer height-field gradients.

	Blinn-Phong	Single LEAN map	Per-frame generation	Mixture textures	Mixture approx.
ATI Radeon HD 5870	1570 FPS	1540 FPS	917 FPS	1450 FPS	1458 FPS
D3D instructions	30 ALU + 1 tex	42 ALU + 2 tex	gen: 8 ALU + 1 tex use: 42 ALU + 2 tex	54 ALU + 5 tex	54 ALU + 4 tex

Figure 12: *Performance comparison. Frames per second were recorded at 1600x1200 full screen with all pixels covered. Instruction counts are as reported by the Direct3D HLSL compiler with the pixel shader 3.0 profile.*

```
// unpack normal
float3 tn = tex2D(normalMap, coord);
float3 N = float3(2*tn.xy-1,tn.z);

// compute B and M by Equations (3) and (6)
// sc is a scale factor mapping B to -1 to 1
float2 B = N.xy/(sc*N.z);
float3 M = float3(B.x*B.x+1/s,B.y*B.y+1/s,B.x*B.y);

// pack into LEAN map render targets
output.lean1 = float4(tn,.5*M.z+.5);
output.lean2 = float4(.5*B+.5, M.xy);
```

Figure 15: *Code fragment to render top-level LEAN map textures from an existing normal map.*

```
// unpack normal
float4 t1 = tex2D(lean1, texcoord);
float3 N = float3(2*t1.xy-1,t1.z);

// unpack B and M
float4 t2 = tex2D(lean2, texcoord);
float2 B = (2*t2.xy-1)*sc;
float3 M = float3(t2.zw,2*t1.w-1)*sc*sc;

// convert M to Σ by Equation (5)
float3 Σ = M - float3(B*B, B.x*B.y);
float3 Det = Σ.x*Σ.y - Σ.z*Σ.z;

// compute specular by Equation (1)
float2 h̃ = h⃗.xy/h⃗.z - B;
float e = (h̃.x*h̃.x*Σ.y + h̃.y*h̃.y*Σ.x - 2.*h̃.x*h̃.y*Σ.z);
float spec = (Det<=0) ? 0. : exp(-.5*e/Det)/sqrt(Det);
```

Figure 16: *Run-time LEAN mapping pseudo-code fragment.*

5 Implementation

Figure 15 shows a code fragment to generate a LEAN map from a source normal map. The code includes a scale factor, `sc`, that can be used to avoid overflow and loss of precision. It should be chosen so B will fill, but not overflow, the -1 to 1 range. For 8-bit texture components, the choice of `sc` is critical to avoid numerical problems, but for 16-bit it is sufficient to just choose a factor that avoids texture overflow. This code fragment is most of the complete shader, missing only input and output declarations and function declaration. This code is used on a quad to generates two texture render targets that are the top level of the LEAN map. After running this pass, either a standard user or driver-provided MIP generation fills in the remaining levels of the maps.

Figure 16 shows a code fragment to use the resulting LEAN maps. This code uses mathematical symbols to match notation from Figure 3, but otherwise is stock HLSL. The code fragment shown only computes the specular term for one light. Additional lights would just repeat the final three lines of the fragment. In use, this fragment should to be combined with base surface color textures, diffuse shading and a Fresnel term.

6 Limitations

Like other Beckmann distribution-based shading models, LEAN mapping can only capture one direction of anistropy. It cannot capture the bi-modal distribution of a set of sharp V groves, or the two principal directions of anistropy aligned with the threads of woven materials like particularly shiny cloth. For such models, a method like Frequency Domain Normal Map Filtering might be better suited [Han et al. 2007]. We find this limitation acceptable in exchange for the speed and simplicty of LEAN mapping, the large variety of materials it does support, and the compatibility with the existing art pipeline.

None of the options for combining bump layers is ideal, though the three methods we present together handle most cases. For fastest operation when the layers' bumps are not highly correlated with each other, the mixing approximation works well. For layers, including coherent layers, that do not move relative to each other but may have changing per-pixel mixing ratios, the mixing textures work well. Neither of these options need to update the LEAN maps every frame, so maps can be generated in a preprocess or on level load. If neither case applies, or if the layer blending function contains high enough frequencies to introduce its own bumps, we can still generate a fully mixed LEAN map every frame.

We often use specular powers, s, in the order of 256 up to $10 - 20,000$. It is rare to use powers over 64 for ordinary bump mapping due to the severe aliasing problem, but LEAN mapping allows even extremely high specular powers. Since one bit change in an 8-bit texture component is $1/256$, we require 16-bit components to keep sufficient precision. With careful normalization into a 0-1 texture range, it is still possible use 8-bit textures and incorporate the constant $1/s$ factor at shading time. In this case, there will be some change in highlight shape at the base level due to rounding errors in the covariance reconstruction. Figures 2, 9(c,d) and 10(c,d), and 13 all use 8-bit LEAN maps, all other figures use 16-bit. In general, 8-bit textures only make sense if absolutely needed for speed or space.

7 Conclusion

We have presented LEAN mapping as a unified shading model representing both bump orientation and BRDF, with textures containing the 2D mean and second moment matrix of a Gaussian distribution in a projection plane defined by the underlying surface. This representation is easy enough to construct on the fly, or as a preprocess from either height-field or normal based bump maps. The terms combine linearly, so work with any standard texture filtering approach, including hardware-accelerated MIP pyramid construction, trilinear MIP sampling and anisotropic filtering. We show how existing Blinn-Phong or Beckmann-distribution based shading models can be adapted to the new model, to produce visually equivalent results when filtering is not necessary, but correct filtering and antialiasing when it is needed. Effective bump antialiasing allows high specular powers of 10,000 or more. The low texture and computational overhead, along with the Blinn-Phong compatibility significantly eases adoption in an existing game art pipeline. Fi-

nally, we show three methods for mixing layers of bumps, allowing time-varying or spatially-varying flow, decals, and detail texture.

8 Acknowledgments

Special thanks to Steve Egrie for artistic work on the in-game water system. Figure 1 looks good thanks to him. Thanks also to Josh Barczak, John Kloetzli and Kiran Sudhakara for their helpful comments on drafts of this paper.

References

BECKER, B. G., AND MAX, N. L. 1993. Smooth transitions between bump rendering algorithms. In *Proceedings of SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, 183–190. ISBN 0-201-58889-7.

BECKMANN, P., AND SPIZZICHINO, A. 1963. *The Scattering of Electromagnetic Waves from Rough Surfaces*. Pergamon Press.

BLINN, J. F. 1977. Models of light reflection for computer synthesized pictures. In *Computer Graphics (SIGGRAPH '77 Proceedings)*, 192–198.

BLINN, J. F. 1978. Simulation of wrinkled surfaces. In *Proceedings of SIGGRAPH 78: The 5th annual conference on comptuer graphics and interactive techniques*, 286–292.

CABRAL, B., MAX, N., AND SPRINGMEYER, R. 1987. Bidirectional reflection functions from surface bump maps. In *Proceedings of SIGGRAPH 87: The 14th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 273–281.

COHEN, J., OLANO, M., AND MANOCHA, D. 1998. Appearance-preserving simplification. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM Press, 115–122.

COOK, R. L., AND TORRANCE, K. E. 1981. A reflectance model for computer graphics. In *Computer graphics (Proceedings of SIGGRAPH 81)*, ACM SIGGRAPH, vol. 15, 307–316.

CROW, F. C. 1984. Summed-area tables for texture mapping. In *Proceedings of SIGGRAPH 84*, H. Christiansen, Ed., vol. 18 of *Computer Graphics*, 207–212.

DONNELLY, W., AND LAURITZEN, A. 2006. Variance shadow maps. In *I3D 06: Proceedings of the 2006 Symposium on Interactive 3D Graphics and Games*, ACM, 161–165.

FOURNIER, A. 1992. Normal distribution functions and multiple surfaces. In *Graphics Interface '92 Workshop on Local Illumination*, Canadian Information Processing Society, 45–52.

HAN, C., SUN, B., RAMAMOORTHI, R., AND GRINSPUN, E. 2007. Frequency domain normal map filtering. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26*, 3 (July), 28–1–28–11.

HARA, K., NISHINO, K., AND IKEUCHI, K. 2005. Multiple light srouces and reflectance property estimation based on a mixture of spherical distributions. In *Proceedings of ICCV 05*, vol. 2, IEEE.

HE, X., TORRANCE, K., SILLION, F., AND GREENBERG, D. 1991. A comprehensive physical model for light reflection. In *Computer Graphics (Proceedings of ACM SIGGRAPH 91)*, T. W. Sederberg, Ed. ISBN 0-201-56291-X.

HECKBERT, P. S. 1989. *Fundamentals of Texture Mapping and Image Warping*. Master's thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

KAJIYA, J. T. 1985. Anisotropic reflection models. In *Computer Graphics (Proceedings of SIGGRAPH 85)*, vol. 19, 15–21.

KAUTZ, J., AND SEIDEL, H.-P. 2000. Towards interactive bump mapping with anisotropic shift-variant BRDFs. In *GH '00: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, ACM Press, New York, NY, USA, 51–58.

KILGARD, M. J. 2000. A practical and robust bump-mapping technique for today's GPU's. In *GDC 2000*. Available at http://www.nvidia.com/.

LAGAE, A., LEFEBVRE, S., DRETTAKIS, G., AND DUTRÉ, P. 2009. Procedural noise using sparse Gabor convolution. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2009) 28*, 3 (July), 28–1–28–10.

LYON, R. 1993. Phong shading reformulation for hardware rendering. Tech. Rep. 43, Apple.

MCCORMACK, J., PERRY, R., FARKAS, K. I., AND JOUPPI, N. P. 1999. Feline: Fast elliptical lines for anisotropic texture mapping. In *Proceedings of SIGGRAPH 99*, Addison Wesley Longman, A. Rockwood, Ed., Computer Graphics Proceedings, Annual Conference Series, 243–250.

OLANO, M., AND NORTH, M. 1997. Normal distribution mapping. Tech. Rep. TR97-041, Department of Computer Science, University of North Carolina, Chapel Hill, North Carolina.

PEERCY, M., AIREY, J., AND CABRAL, B. 1997. Efficient bump mapping hardware. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, 303–306.

POULIN, P., AND FOURNIER, A. 1990. A model for anisotropic reflection. In *Computer Graphics (Proceedings of SIGGRAPH 90)*, F. Baskett, Ed., 273–282.

SCHILLING, A. 1997. Toward real-time photorealistic rendering: challenges and solutions. In *HWWS 97: Proceedings of the ACM SIGGRAPH/Eurographics Workshop on Graphics Hardware*, ACM, 7–15.

SCHLICK, C. 1993. A customizable reflectance model for everyday rendering. In *Eurographics Workshop on Rendering*, 73–84.

TOKSVIG, M. 2005. Mipmapping normal maps. *Journal of Graphics Tools 10*, 3, 65–71.

WARD, G. 1992. Measuring and modeling anisotropic reflection. In *Proceedings of SIGGRAPH 92*, ACM Press, Computer Graphics Proceedings, Annual Conference Series, 265–272.

WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. 1992. Predicting reflectance functions from complex surfaces. In *Proceedings of SIGGRAPH 92*, ACM Press, Computer Graphics Proceedings, Annual Conference Series, 255–264.

WILLIAMS, L. 1983. Pyramidal parametrics. In *SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, vol. 17, 1–11.