Photorealistic Surface Rendering with Microfacet Theory

Jonathan Dupuy

Université de Montréal
Université Claude Bernard de Lyon 1

Photorealistic Surface Rendering with Microfacet Theory

par
Jonathan Dupuy

Thèse présentée à la Faculté des études supérieures et postdoctorales
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique le 26 novembre 2015

Jury
Tamy Boubekeur, Professeur des universités, Telecom Paris (rapporteur)
Nicolas Holzschuch, Directeur de recherche à l'INRIA Grenoble Rhône-Alpes et LJK (rapporteur)
Jean-Michel Dischler, Professeur des universités à l'université de Strasbourg (examinateur)
Derek Nowrouzezahrai, Maître de conférences à l'université de Montréal (examinateur)
Christophe Schlick, Professeur des universités à l'université Bordeaux 2 (examinateur)
Victor Ostromoukhov, Professeur des universités à l'université de Lyon 1 (directeur)
Pierre Poulin, Professeur des universités à l'université de Montréal (directeur)
Jean-Claude Iehl, Maître de conférences à l'université de Lyon 1 (encadrant)

Résumé

La synthèse d'images dites photoréalistes nécessite d'évaluer numériquement la manière dont la lumière et la matière interagissent physiquement, ce qui, malgré la puissance de calcul impressionnante dont nous bénéficions aujourd'hui et qui ne cesse d'augmenter, est encore bien loin de devenir une tâche triviale pour nos ordinateurs. Ceci est dû en majeure partie à la manière dont nous représentons les objets : afin de reproduire les interactions subtiles qui mènent à la perception du détail, il est nécessaire de modéliser des quantités phénoménales de géométries. Au moment du rendu, cette complexité conduit inexorablement à de lourdes requêtes d'entrées-sorties, qui, couplées à des évaluations d'opérateurs de filtrage complexes, rendent les temps de calcul nécessaires à produire des images sans défaut totalement déraisonnables. Afin de pallier ces limitations sous les contraintes actuelles, il est nécessaire de dériver une représentation multiéchelle de la matière.

Dans cette thèse, nous construisons une telle représentation pour la matière dont l'interface correspond à une surface perturbée, une configuration qui se construit généralement via des cartes d'élévations en infographie. Nous dérivons notre représentation dans le contexte de la théorie des microfacettes (conçue à l'origine pour modéliser la réflectance de surfaces rugueuses), que nous présentons d'abord, puis augmentons en deux temps. Dans un premier temps, nous rendons la théorie applicable à travers plusieurs échelles d'observation en la généralisant aux statistiques de microfacettes décentrées. Dans l'autre, nous dérivons une procédure d'inversion capable de reconstruire les statistiques de microfacettes à partir de réponses de réflexion d'un matériau arbitraire dans les configurations de rétroréflexion. Nous montrons comment cette théorie augmentée peut être exploitée afin de dériver un opérateur général et efficace de rééchantillonnage approximatif de cartes d'élévations qui (a) préserve l'anisotropie du transport de la lumière pour n'importe quelle résolution, (b) peut être appliqué en amont du rendu et stocké dans des *MIP maps* afin de diminuer drastiquement le nombre de requêtes d'entrées-sorties, et (c) simplifie de manière considérable les opérations de filtrage par pixel, le tout conduisant à des temps de rendu plus courts. Afin de valider et démontrer l'efficacité de notre opérateur, nous synthétisons des images photoréalistes anticrenelées et les comparons à des images de référence. De plus, nous fournissons une implantation C++ complète tout au long de la dissertation afin de faciliter la reproduction des résultats obtenus. Nous concluons avec une discussion portant sur les limitations de notre approche, ainsi que sur les verrous restant à lever afin de dériver une représentation multiéchelle de la matière encore plus générale.

Mots clés : microfacettes, rendu, photoréalisme, anticrénelage, filtrage, MIP map

Abstract

Photorealistic rendering involves the numeric resolution of physically accurate light/matter interactions which, despite the tremendous and continuously increasing computational power that we now have at our disposal, is nowhere from becoming a quick and simple task for our computers. This is mainly due to the way that we represent objects: in order to reproduce the subtle interactions that create detail, tremendous amounts of geometry need to be queried. Hence, at render time, this complexity leads to heavy input/output operations which, combined with numerically complex filtering operators, require unreasonable amounts of computation times to guarantee artifact-free images. In order to alleviate such issues with today's constraints, a multiscale representation for matter must be derived.

In this thesis, we derive such a representation for matter whose interface can be modelled as a displaced surface, a configuration that is typically simulated with displacement texture mapping in computer graphics. Our representation is derived within the realm of microfacet theory (a framework originally designed to model reflection of rough surfaces), which we review and augment in two respects. First, we render the theory applicable across multiple scales by extending it to support noncentral microfacet statistics. Second, we derive an inversion procedure that retrieves microfacet statistics from backscattering reflection evaluations. We show how this augmented framework may be applied to derive a general and efficient (although approximate) downsampling operator for displacement texture maps that (a) preserves the anisotropy exhibited by light transport for any resolution, (b) can be applied prior to rendering and stored into MIP texture maps to drastically reduce the number of input/output operations, and (c) considerably simplifies per-pixel filtering operations, resulting overall in shorter rendering times. In order to validate and demonstrate the effectiveness of our operator, we render antialiased photorealistic images against ground truth. In addition, we provide C++ implementations all along the dissertation to facilitate the reproduction of the presented results. We conclude with a discussion on limitations of our approach, and avenues for a more general multiscale representation for matter.

Keywords: microfacet theory, rendering, photorealism, antialiasing, filtering, MIP map

Contents

List of Tables

List of Figures

Acknowledgements

This thesis concludes four invaluable years of research; I have learned so much and met so many great people. I first wish to thank my advisors Jean-Claude Iehl, Pierre Poulin, and Victor Ostromoukhov, who gave me the opportunity to experience this intense adventure in the first place. Thank you Jean-Claude for introducing me to the world of computer graphics, to OpenGL and the challenging problem of level-of-detail during my bachelor degree. Thank you Pierre for spending so much time on the deadlines with me, and for always being available for discussions during my stay at LIGUM; this year and a half spent in Montreal was unforgettable. Thank you Victor for sharing with me your fascinating stories on tilings and quasicrystals, and for always introducing me to the people you would invite over at LIRIS. My next thoughts go to my colleague and friend Eric Heitz, who has been by far my most important collaborator since the beginning on my PhD. To you, Eric, I must say that it has been both a pleasure and privilege to work with you; it really feels to me that no problem is too hard when we are working together. Next, I wish to thank several colleagues I have had the chance to meet during my PhD and who also had a great influence on my work. Stephen Hill: for giving me the opportunity to be a part of the fantastic physically based shading course you have been organizing along with Stephen McAuley so tremendously well for several years in a row now. Antoine Bouthors: for inviting me over at Weta Digital, and giving the chance to integrate some of my research in your rendering pipeline. Wenzel Jakob: for providing the amazing Mistuba renderer; the figures and source code snippets I included in my manuscript are inspired from the documentation you wrote for Mitsuba. Fabrice Neyret and Eric Bruneton: for sharing with me your philosophy of level-of-detail and scientific methodology during my master thesis. Brent Burley: for providing me with the T-Rex model and sharing your insights on BRDF authoring. Thiago Da Costa and Arno Zinke: for providing me with the LAGOA robot asset. Cyril Crassin: for perhaps being the only one I can talk to about OpenGL ARB extensions and politics at the same time. Guillaume Bouchard: for sharing with me your wise thoughts on rendering, always in a very unique, entertaining way. Lastly, I thank all my old friends and family: your continuous support throughout the years has been invaluable.

Financial Support Financial support was partly provided by grants from GRAND and the NSERC Discovery of Pierre Poulin, and the ANR AMCQMCSCGA of Victor Ostromoukhov. I also acknowledge an Explo'ra Doc grant from the Rhône-Alpes region.

Chapter 1

Introduction

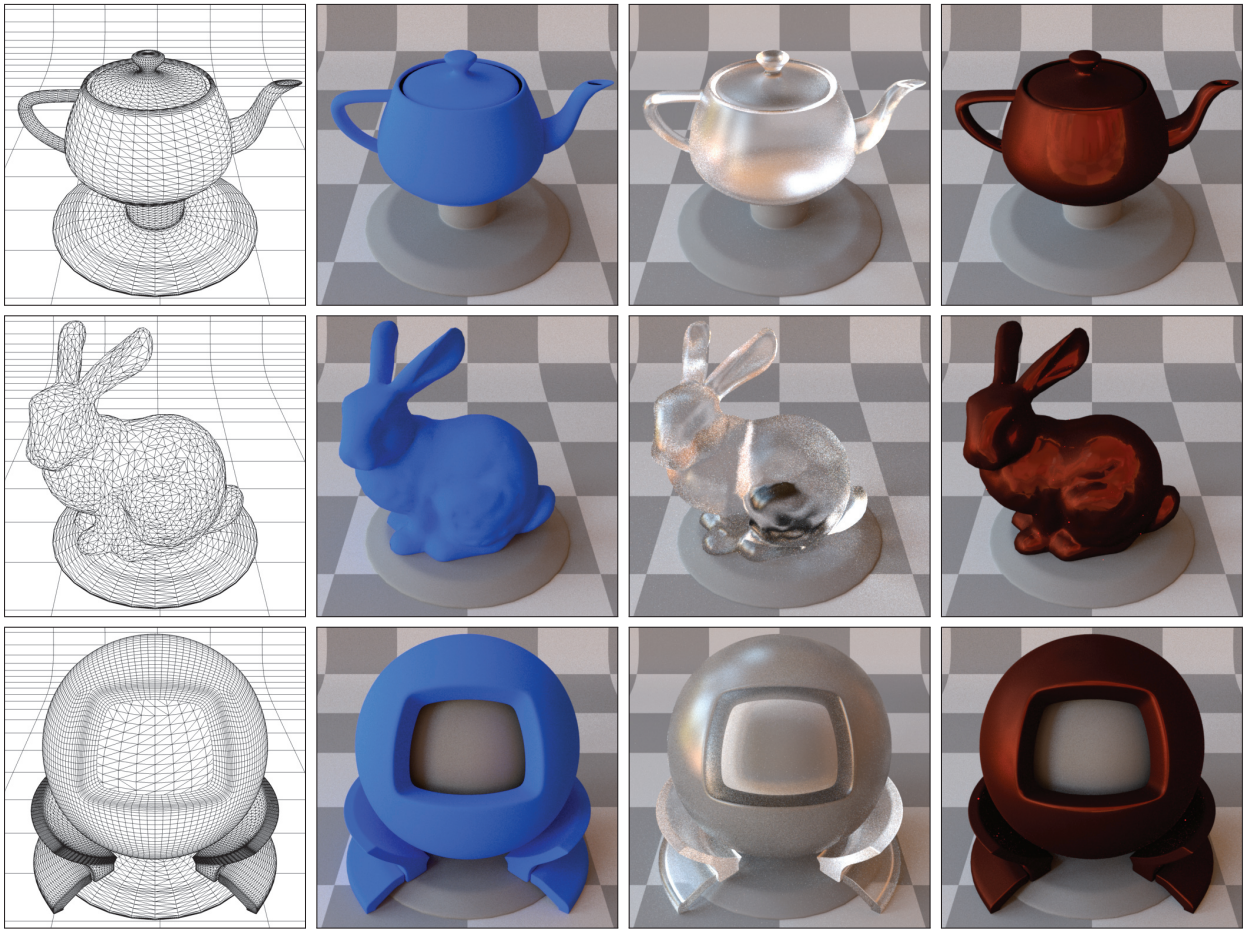
1.1 Motivation

Most objects we see in everyday life can be described in two steps: first by shape, then by appearance; Figure 1.1 illustrates their differences. Computer graphics applications tend to follow this approach, relying on polygon meshes for the former, and physically based shading for the latter; Figure 1.2 shows a few such combinations. When high-frequency detail is desired, these representations are further augmented with texture maps that either affect shape (e.g., displacement maps) or appearance (e.g., normal maps); Figure 1.3 shows a few common texture mapping techniques.

Despite being a remarkably effective solution for detail management, this combination of descriptors has led to a challenging problem for physically based rendering: we now have to somehow process thousands (or even more) of polygons and/or texels per pixel just to properly capture visibility and direct lighting effects. Since there usually are no analytic solutions to this problem, it must be solved numerically with, inexorably, a large number of samples (recall that the sampling rate should be at least twice the maximum frequency component of the scene according to the Nyquist sampling theorem). In most cases unfortunately, the computational cost induced by the Nyquist sampling theorem is too high to be practical. As a consequence, the integration ends up being undersampled for the sake of reasonable rendering times, ultimately leading in final images to unacceptable artifacts such as aliasing or noise.

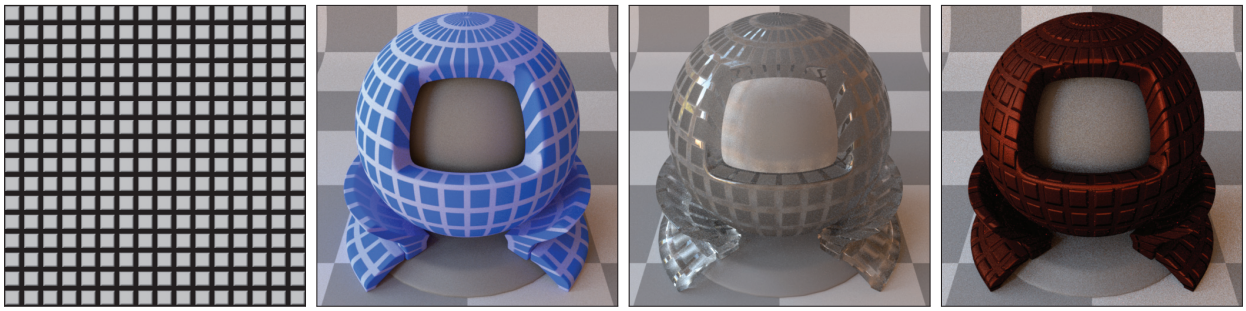
In order to deal with this specific issue, computer graphics applications commonly rely on more or less sophisticated screen-space filters, ranging from fully automatic (e.g., SMAA [JESG12]) to fully manual (e.g., an artist working overtime in Photoshop). As can be evidenced by the quality of current photorealistic renderings, this

Figure 1.1 – Shape versus appearance. **Left:** Objects with different shapes but the same appearance. **Right:** Objects with the same shape but different appearance. The plasma ball image was created by Soenke Rahn and is available at `http://commons.wikimedia.org/wiki/File:Plasmakugel_%28Plasma_Ball%29.JPG`



Shapes | Appearance 1 (diffuse) | Appearance 2 (dielectric) | Appearance 3 (conductor)

Figure 1.2 – Shape versus appearance in computer graphics.



Texture pattern | Reflectance | Roughness | Bump/normal

Figure 1.3 – Various texture mapping techniques.

approach works reasonably well in practice. However, it does tend to defeat some the main benefits of physically based rendering. In particular, screen-space filters do not guarantee shading consistency across pixels and time, thus requiring extensive tweaking to achieve satisfying results. Such limitations can quickly bottleneck a rendering pipeline, so it makes sense to look for alternative solutions.

Given the current evolution of computational power and the innovations in higher resolution / higher framerate displays, it seems that brute-force supersampling will remain out of reach for the foreseeable future (note: it will also be interesting to see how screen-space filters evolve, as their performance is intimately linked to display characteristics). In the meantime, rather than trying to adapt sampling rates to geometric complexity, there are potentially great benefits to be had by doing the opposite, i.e., adapting geometry to fixed amounts of samples; this can be achieved through the use of precomputed level-of-detail schemes.

Finding generic and lightweight object representations that can maintain physically based principles at all scales is quite difficult, however. The difficulty mainly resides in the mathematics involved, which are mostly nonlinear, highly dimensional, and sensitive to discretization. The main goal of this thesis is to motivate the conjecture that, despite difficulty, viable solutions for precomputed level-of-detail schemes can be derived. We demonstrate this by introducing an effective prefiltering scheme for displacement mapped surfaces of arbitrary appearance, which works similarly to MIP mapping [Wil83]. Our prefiltering scheme is derived within an augmented microfacet theory framework. Originally, microfacet theory is a mathematical framework designed for describing appearance through a bidirectional reflectance distribution function (BRDF) in a biscale approach; microfacet BRDFs are the object of Chapter 2. In order to make this framework applicable to displacement mapped surface filtering, we extend it in two ways. First, we generalize the theory to multiple scales; this is the purpose of Chapter 3. Second, we introduce an inverted theory, i.e., a framework to recover a microfacet configuration given an input material; this is the purpose of Chapter 4. Our extensions of microfacet theory can then be used to derive a multiscale representation for displaced surfaces that is general, yet simple, robust, and straightforward to implement in a physically based rendering pipeline. To emphasize this last point, a complete state-of-the-art C++ implementation of a multiscale microfacet BRDF library is presented progressively throughout the manuscript, along with numeric validation tests to help the interested reader in better understanding and reproducing the results of this thesis.

1.2 Thesis Overview

More formally, this thesis is structured as follows (note: each chapter starts with a visual summary of its contents).

- In the remainder of this chapter, we make a more rigorous mathematical treatment of the problem of filtering displacement mapped surfaces of arbitrary reflectance. We take advantage of this effort to recall some of the fundamental rules that apply to physically based rendering, to introduce mathematical notations, and to describe a general approach that we employ to solve numerical problems.

- In Chapter 2, we review previous and concurrent work on BRDF modelling with microfacet theory. While microfacet BRDFs can be found in the microfacet literature since the 1960s, some of its most powerful properties, e.g., importance sampling with the distribution of visible normals and roughness invariance, were introduced concurrently to the development of this thesis.

- In Chapter 3, we introduce noncentral microfacet theory, a generalization of microfacet theory to derive the BRDF of a rough specular microsurface at any scale. This contribution allows us to transpose the difficult problem of displacement map filtering into that of evaluating a microfacet BRDF. By approximating the microsurface with a Gaussian random process, we derive an analytic displacement filtering scheme that can be precomputed into MIP maps. We refer to this scheme as LEADR mapping.

- In Chapter 4, we introduce inverted microfacet theory, a framework to extract state-of-the-art microfacet BRDF components from arbitrary materials. This contribution allows us to convert any material into a microfacet configuration, which may then benefit from useful microfacet BRDF properties, e.g., roughness authoring and fast importance sampling. We also show how to approximate the extracted microfacets into Gaussian or GGX random processes, which are very popular in the industry.

- In Chapter 5, we show how to combine the contributions of the thesis to derive a general multiscale representation for displaced surfaces that is lightweight, efficient, and authorable. We discuss benefits and limitations of this representation with respect to previous work, and identify future extensions required to derive a more general multiscale representation for matter.

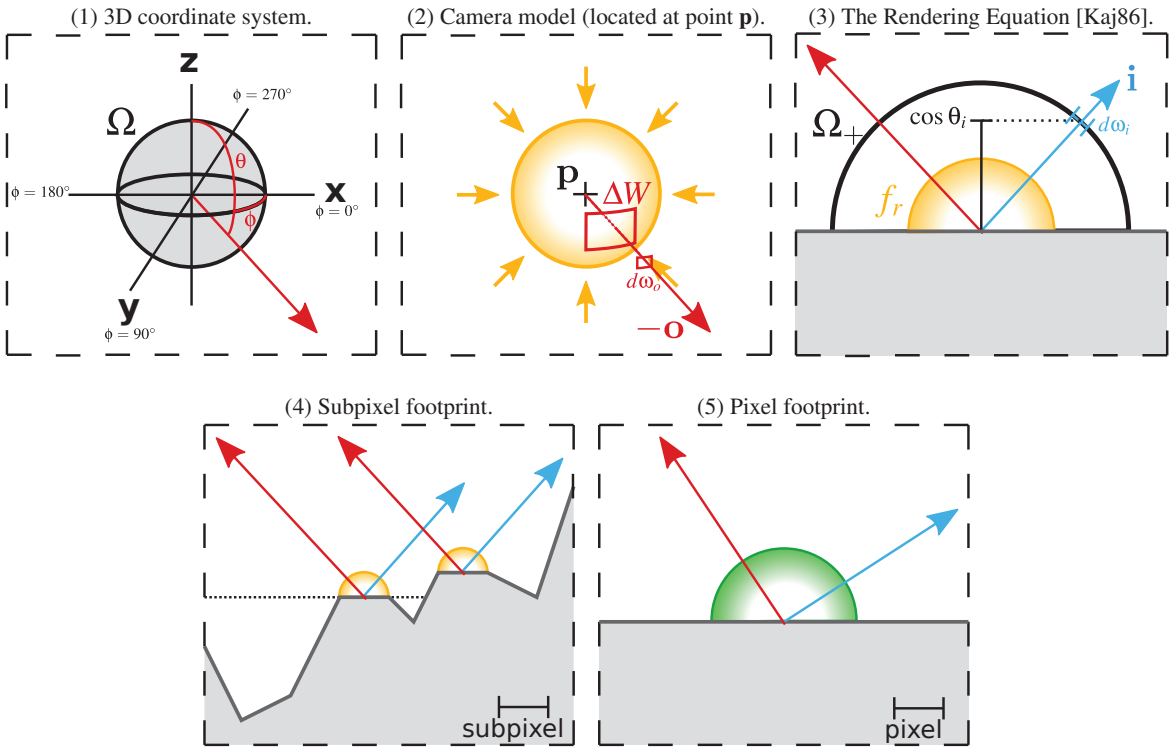


Figure 1.4 – Theoretical background illustrated.

1.3 Theoretical Background

Camera Model A photorealistic renderer is concerned with the problem of generating a purely synthetic image that is indistinguishable from a real photograph. In the simplest cases, this problem is approached by discretizing a continuous distribution of incident radiance $L > 0$ into a set of pixels that constitute the final image, thereby mimicking the behavior of a pinhole camera sensor with instantaneous exposure. We restrict our mathematical treatment of light transport to geometric optics, i.e., light propagates as straight rays that can be reflected or absorbed by geometry throughout the 3D scene; this is a common assumption in physically based rendering. We can determine the radiance incident to any 3D point $\mathbf{p} \in \mathbb{R}^3$ by measuring the amount of light rays that pass through this point. We can visualize such light rays by mapping the radiance incident to $\mathbf{p}$ on a unit sphere centered on $\mathbf{p}$; this is illustrated in Figure 1.4 (2). Any portion of this sphere can be used to define the image plane of a virtual pinhole camera.

Photorealistic Pixels Since we wish to generate a discrete image, the frequencies exhibited by the incident radiance distribution function L that exceed the Nyquist limit should be removed in order to avoid artifacts. This is why the discretization process of L is typically done after low-pass filtering. Mathematically, letting $W \in \mathbb{R}$ denote the low-pass filter (e.g., box or Gaussian) weight associated to a given screen pixel, the intensity $I > 0$ of the pixel is an inner product [Hec89, BN12]

$$I = \int_{\Delta W} L_o \cdot W(\mathbf{o}) \cdot d\omega_o, \tag{1.1}$$

where, letting Ω denote the unit 3D sphere, $\Delta W \subseteq \Omega$ is the filter's angular support on the sphere, $\mathbf{o}$ is the unit direction associated to the infinitesimal solid angle $d\omega_o$, and $L_o \geq 0$ denotes the value of L at direction $\mathbf{o}$, i.e., $L_o = L(\mathbf{o})$. Note that throughout this thesis, we manipulate unit directions either from their Cartesian components, or their spherical coordinates, which we link according to the following convention

$$\mathbf{o} = \begin{bmatrix} \sin\theta_o \cdot \cos\phi_o = x_o \\ \sin\theta_o \cdot \sin\phi_o = y_o \\ \cos\theta_o = z_o \end{bmatrix} \in \Omega. \tag{1.2}$$

The geometry of the coordinate system used throughout this thesis is illustrated in Figure 1.4 (1). We reserve the θ and ϕ notations to denote elevation and azimuthal angles, respectively.

The Rendering Equation In order to evaluate the incident radiance distribution function L during the low-pass filtering process, i.e., Equation (1.1), a physically based renderer computes the rendering equation [Kaj86]

$$L_o = \int_{\Omega_+} L_i \cdot f_r \cdot \cos\theta_i \cdot d\omega_i. \tag{1.3}$$

This equation gives the amount of light reflected towards $\mathbf{o}$ as a measure of the interactions occurring between an incident lighting distribution function $L_i \geq 0$ and a BRDF $f_r \geq 0$ over the set of unit directions defined on the hemisphere Ω_+ aligned with the tangent space of an infinitesimal surface element that projects into the pixel; the geometric configuration of the rendering equation is illustrated in Figure 1.4 (3). It is interesting to note that Equation (1.1) and Equation (1.3) look very similar; as such, the projected BRDF term of Equation (1.3), i.e., $f_r \cdot \cos\theta_i$, can be interpreted as a filter applied to the incident radiance distribution function L_i.

Displacement Mapped Surfaces The filter's angular support ΔW defines the footprint of the pixel in the 3D scene. Assuming that this footprint projects onto a displacement mapped surface (if not, split the footprint in two parts), then we know from the Nyquist sampling theorem that the amount of samples required to compute Equation (1.1) without artifacts should be at least twice the number of texels; the same logic applies to Equation (1.3). Hence, the further away we are from displacement mapped surfaces, the greater the rendering cost. The practical contribution of this thesis is to show that this costly filtering operation can be accelerated with precomputations. In particular, we show that in the case where displacement mapped surfaces fill the pixels' footprints, Equation (1.1) can be approximated by the rendering equation directly by using a multiscale microfacet BRDF, effectively reducing a 4D integration problem into a 2D integration problem, which, furthermore, can be measured with efficient importance sampling; these ideas are illustrated in Figure 1.4 (4, 5), and will be formulated mathematically throughout this thesis.

1.4 Practical Considerations

Numerical Integrations Throughout this thesis, we often rely on numerical integrations to solve and/or validate certain equations. Whenever we face an integral of the form

$$\mathscr{I} = \int_D f(x) \cdot dx, \tag{1.4}$$

where D denotes the domain of integration, we always proceed in two steps. First, we reformulate the integral in the unit segment $[0,1)$. If $D = [0,+\infty)$ for instance, then we can first perform the change of variable $x = \tan\theta$ ($dx = \sec^2\theta \cdot d\theta$) so that the integration domain becomes $[0,\pi/2)$

$$\mathscr{I} = \int_0^{\pi/2} f(\tan\theta) \cdot \sec^2\theta \cdot d\theta. \tag{1.5}$$

We apply next a simple scaling operation, i.e., $x = u\pi/2$ ($dx = \frac{\pi}{2} \cdot du$) to arrive at the unit segment

$$\mathscr{I} = \frac{\pi}{2} \int_0^1 f(\tan(u\pi/2)) \cdot \sec^2(u\pi/2) \cdot du. \tag{1.6}$$

Note that this technique is also known as transform sampling. Whenever the transformation involves quantile functions, we say that the integral is importance sampled according to the distributions from which the quantile functions originate. Once the transformation complete, we rely on Euler's method to compute the integration.

Recalling that computing the integral of a function boils down to finding the area under its curve, Euler's technique consists in discretizing the integral into a sum of rectangular areas. The width of the rectangles is constant and set to one over the number of rectangles, while their heights are given by the function evaluated at their lower-left corner, which defines the location of the samples. This gives

$$\mathscr{I} = \frac{1}{N}\frac{\pi}{2}\sum_{i=0}^{N-1} f(\tan(u_i\pi/2)) \cdot \sec^2(u_i\pi/2), \tag{1.7}$$

where $u_i = i/N$. This sum is particularly simple to implement on a computer, as illustrated in Listing 1.1, which provides a pseudocode to evaluate the above equation with $N = 100$ samples. Whenever it is of interest to compute an integral numerically, the series of substitutions that leads to such configurations will be provided.

```
double nint = 0;
const int cnt = 100;

for (int i = 0; i < cnt; ++i) {
    double u = (double)i / (double)cnt;
    double theta = u * M_PI * 0.5;
    double x = tan(theta);
    double cos_term = cos(theta);

    nint+= my_func(x) / (cos_term * cos_term);
}
nint*= M_PI * 0.5 / (double)cnt;
```

Listing 1.1 – Simple numeric integration in C++.

Implementing Bidirectional Reflectance Distribution Functions Due to its role in the rendering equation, i.e., Equation (1.3), the BRDF is a central component of a physically renderer. Thoroughly described by Nicodemus et al. [NRH+77], the BRDF implements the concept that a material can be described from the moment we know the fraction of light it reflects for any pair of unit directions involving an emitter and a receiver direction. We reserve the notation **i** to refer to the former and **o** for the latter. The emitter and receiver direction couple is used as the standard parameterization for the BRDF, and an implementation should be capable of returning its value for any **i** and **o** located on the hemisphere Ω_+. Alternatively, it is also convenient to evaluate the BRDF with the Rusinkiewicz parameterization [Rus98], which involves the halfway vector $\mathbf{h} \in \Omega_+$ and a difference vector $\mathbf{d} \in \Omega_+$. The Rusinkiewicz parameters are linked to the standard parameters through the relations

$$\begin{cases} \mathbf{h} = \dfrac{\mathbf{i}+\mathbf{o}}{\|\mathbf{i}+\mathbf{o}\|} \\ \mathbf{d} = \mathrm{Rot}_y(-\theta_h) \cdot \mathrm{Rot}_z(-\phi_h) \cdot \mathbf{i} \end{cases} \Leftrightarrow \begin{cases} \mathbf{i} = \mathrm{Rot}_z(+\phi_h) \cdot \mathrm{Rot}_y(+\theta_h) \cdot \mathbf{d} \\ \mathbf{o} = 2(\mathbf{i}\cdot\mathbf{h})\mathbf{h} - \mathbf{i} \end{cases} \tag{1.8}$$

Finally, a BRDF implementation should expose a transform sampling procedure. This ability is crucial for solving the rendering equation quickly, as the incident radiance function L_i is generally not known in advance. In general, the transform sampling procedure accounts for the cosine term too, so that we end up importance sampling the projected BRDF term $f_r \cdot \cos\theta_i$. All these features are present in the C++ interface of Listing 1.2, which we rely on to implement a multiscale microfacet BRDF library in the following chapters.

```
/* BRDF interface */
class brdf {
public:
    // evaluate f_r
    virtual vec3 eval(const vec3& i, const vec3& o,
                      const void *user_param = NULL) const = 0;
    virtual vec3 eval_hd(const vec3& h, const vec3& d,
                         const void *user_param = NULL) const;
    // evaluate f_r * cos
    virtual vec3 evalp(const vec3& i, const vec3& o,
                       const void *user_param = NULL) const;
    virtual vec3 evalp_hd(const vec3& h, const vec3& d,
                          const void *user_param = NULL) const;
    // evaluate f_r * cos / pdf
    virtual vec3 evalp_is(float_t u1, float_t u2,
                          const vec3& o,
                          vec3 *i, float_t *pdf,
                          const void *user_param = NULL) const;
    // importance sample f_r * cos using two uniform numbers
    virtual vec3 sample(float_t u1, float_t u2,
                        const vec3& o,
                        const void *user_param = NULL) const;
    // evaluate the PDF of a sample
    virtual float_t pdf(const vec3& i, const vec3& o,
                        const void *user_param = NULL) const;
    // utilities
    static void io_to_hd(const vec3& i, const vec3& o, vec3 *h, vec3 *d);
    static void hd_to_io(const vec3& h, const vec3& d, vec3 *i, vec3 *o);
    // dtor
    virtual ~brdf() {}
};
```

Listing 1.2 – C++ interface of a BRDF.

To make the C++ interface more intuitive to the reader, Listing 1.3 provides an implementation for a Lambertian material of (wavelength-dependent) reflectance parameter $k_d \in [0,1]$, i.e.,

$$f_r = \frac{k_d}{\pi}. \tag{1.9}$$

Note that such BRDFs are very limited in terms of artistic control and plausibility compared to microfacet BRDFs, which are the main focus of this thesis.

```
/* Lambertian BRDF */
class lambertian {
public:
    /* Lambertian Parameters */
    class params {
    public:
        params(const vec3& reflectance = vec3(1));
        vec3 m_reflectance;
    };
    /* Implementation */
    vec3 eval(const vec3& i, const vec3& o,
              const void *user_param = NULL) const;
};

/* Default BRDF importance sampling */
vec3
brdf::sample(
    float_t u1, float_t u2,
    const vec3& o,
    const void *user_param
) const {
    float_t x, y, z;

    uniform_to_concentric(u1, u2, &x, &y);
    z = sqrt(1.0 - x * x - y * y);
    DJB_ASSERT(z >= 0.0);

    return vec3(x, y, z);
}

/* Default BRDF PDF */
float_t brdf::pdf(const vec3& i, const vec3& o, const void *user_param) const
{
    return /* cos(theta_i) */i.z / M_PI;
}

/* Lambertian BRDF evaluation */
vec3
lambertian::eval(const vec3& i, const vec3& o, const void *user_param) const
{
    const lambertian::params params =
        user_param ? *reinterpret_cast<const lambertian::params *>(user_param)
                   : lambertian::params();

    return (params.m_reflectance / M_PI);
}
```

Listing 1.3 – C++ implementation of an ideal Lambertian BRDF.

Chapter 2

State-of-the-art Microfacet Theory

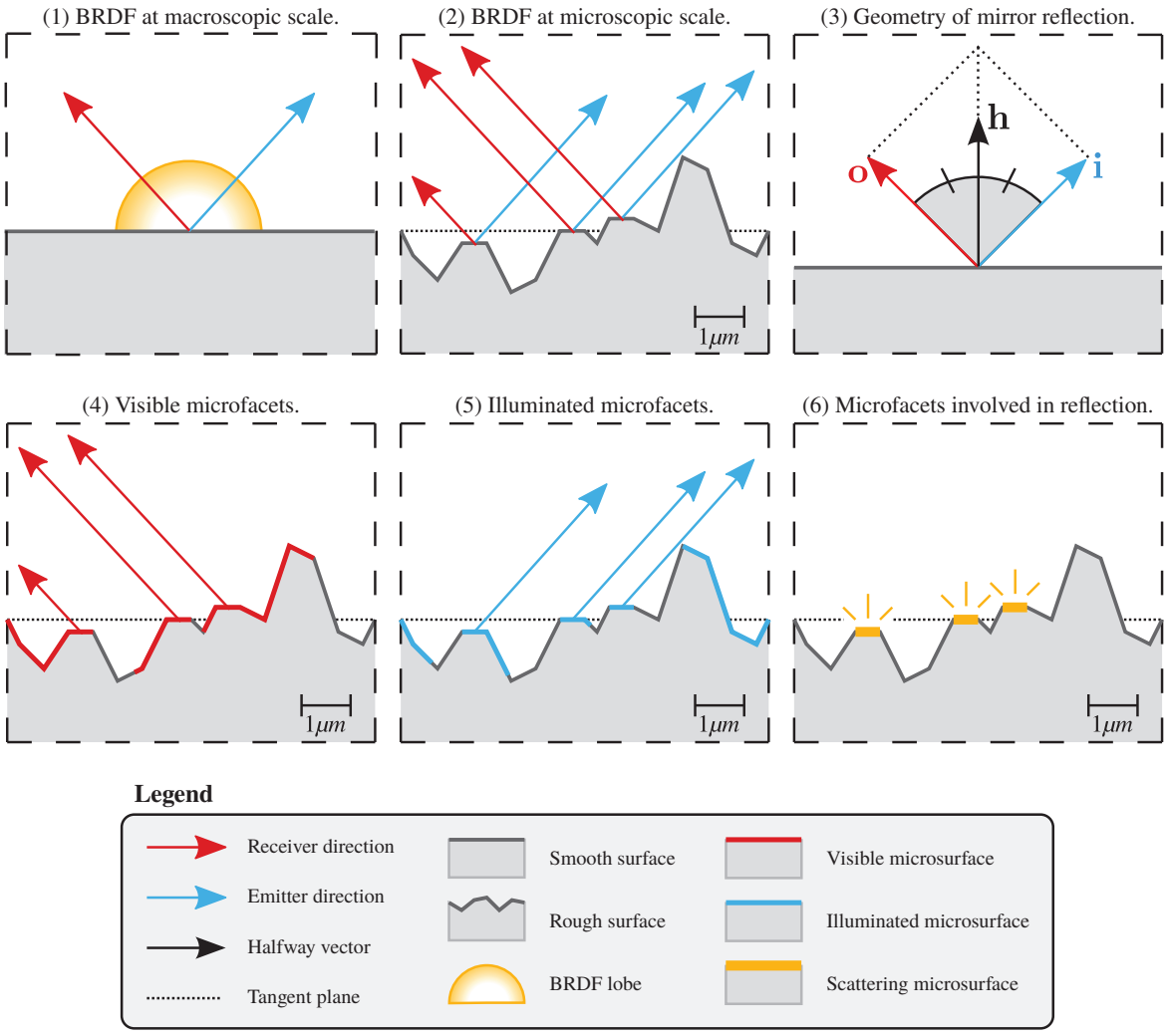


Figure 2.1 – Geometric overview of microfacet theory. Microfacet theory describes (1) the BRDF as the global response of (2) a microfaceted heightfield, whose microfacets act as (3) Fresnel mirrors. When considering single-bounce reflections, the fraction (with respect to the entire microsurface) of (4) visible and (5) illuminated mirror microfacets that face the halfway vector gives (6) the response of the BRDF.

```
/* Microfacet Public API */
class microfacet : public brdf {
public:
    enum {GAF_NMAP, GAF_VGROOVE, GAF_SMITH};
    /* microfacet parameters */
    class params;
    // Dtor
    virtual ~microfacet() {delete m_fresnel;}
    // BRDF interface
    vec3 eval(const vec3& i, const vec3& o,
              const void *user_param = NULL) const;
    vec3 evalp(const vec3& i, const vec3& o,
               const void *user_param = NULL) const;
    vec3 sample(float_t u1, float_t u2, const vec3& o,
                const void *user_param = NULL) const;
    float_t pdf(const vec3& i, const vec3& o,
                const void *user_param = NULL) const;
    vec3 evalp_is(float_t u1, float_t u2, const vec3& o,
                  vec3 *i, float_t *pdf, const void *user_param = NULL) const;
    // eval queries
    vec3 fresnel(float_t cos_theta_d) const {return m_fresnel->eval(cos_theta_d);}
    float_t ndf(const vec3& h,
                const params& params = params::standard()) const;
    float_t gaf(const vec3& h, const vec3& i, const vec3& o,
                const params& params = params::standard()) const;
    float_t g1(const vec3& h, const vec3& k,
               const params& params = params::standard()) const;
    float_t p22(float_t x, float_t y,
                const params& params = params::standard()) const;
    float_t vp22(float_t x, float_t y, const vec3& k,
                 const params& params = params::standard()) const;
    float_t vndf(const vec3& h, const vec3& k,
                 const params& params = params::standard()) const;
    // sampling queries
    virtual bool supports_smith_vndf_sampling() const = 0;
    virtual float_t qf2(float_t u, const vec3& k) const {return 0.0;}
    virtual float_t qf3(float_t u, float_t qf2) const {return 0.0;}
    // mutators
    void set_gaf(int gaf);
    void set_shadow(bool shadow) {m_shadow = shadow;}
    void set_fresnel(const fresnel::impl& f);
    // accessors
    int get_gaf() const {return m_gaf;}
    int get_shadow() const {return m_shadow;}
    const fresnel::impl& get_fresnel() const {return *m_fresnel;}
};
```

Listing 2.1 – C++ interface of microfacet BRDF.

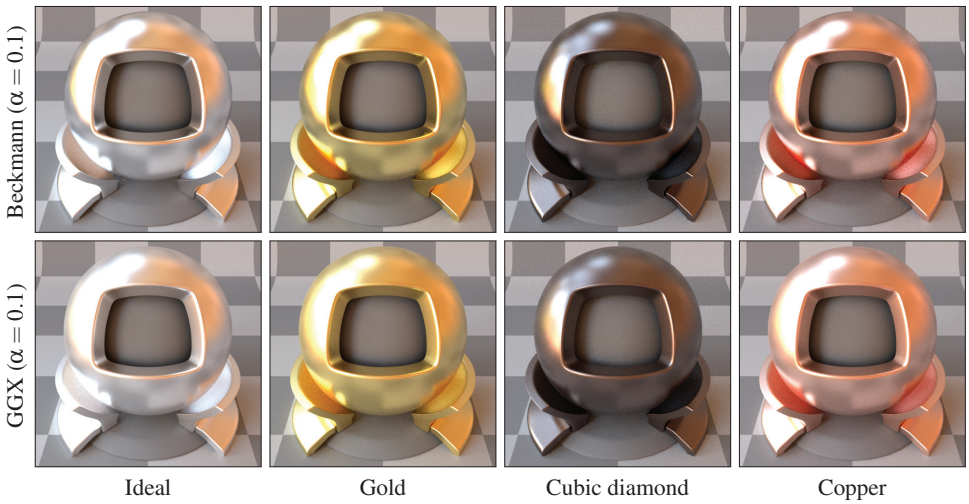


Figure 2.2 – Varying the Fresnel term on a microfacet BRDF.

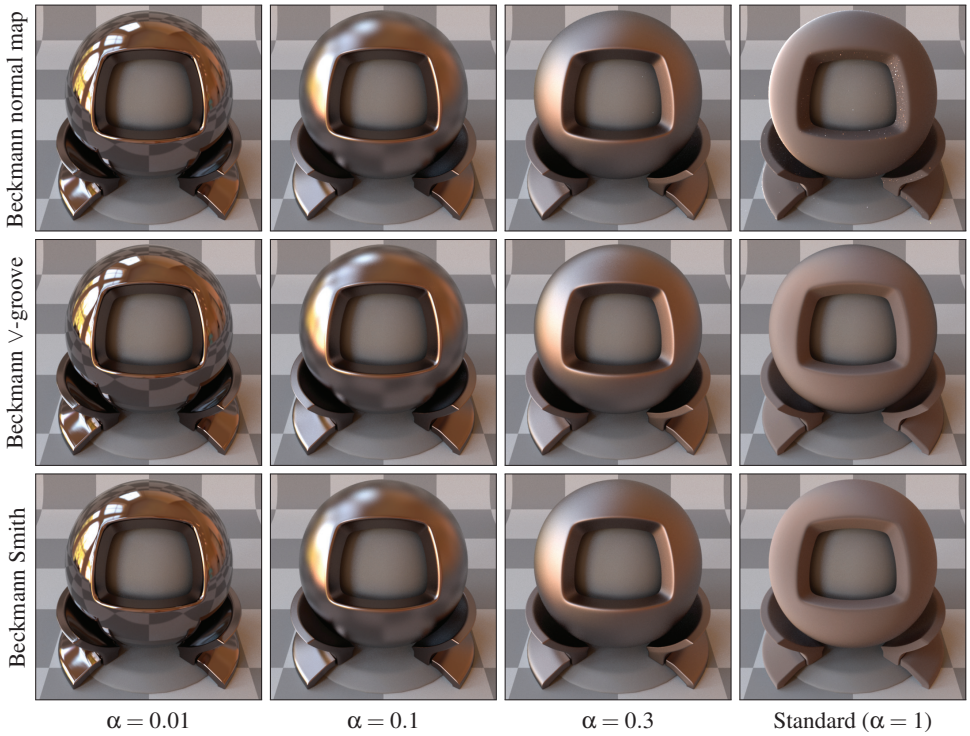


Figure 2.3 – Impact of the GAF term on a microfacet BRDF.

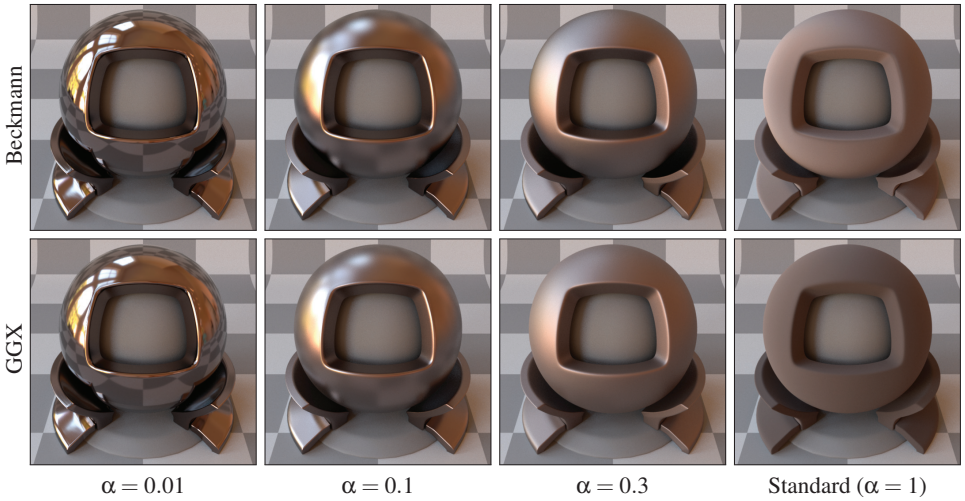


Figure 2.4 – Effect of isotropic roughness on a microfacet BRDF.

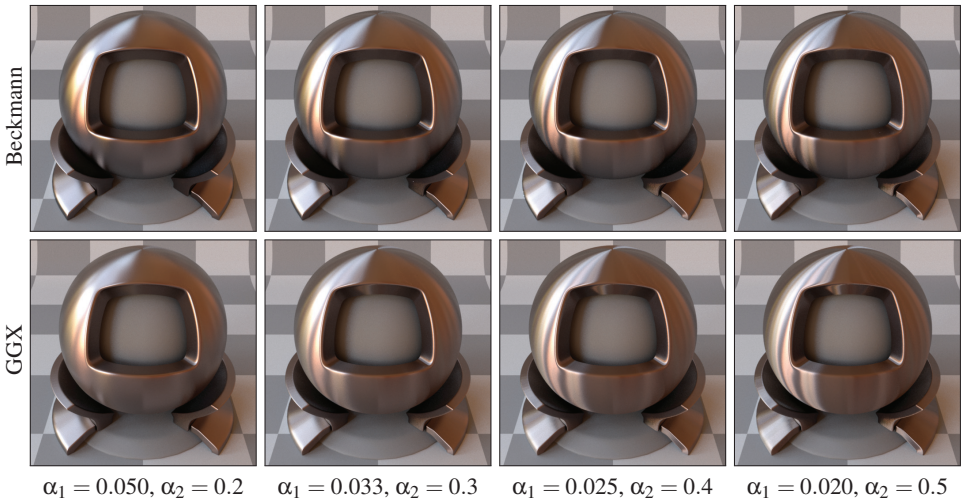


Figure 2.5 – Effect of anisotropic roughness on a microfacet BRDF.

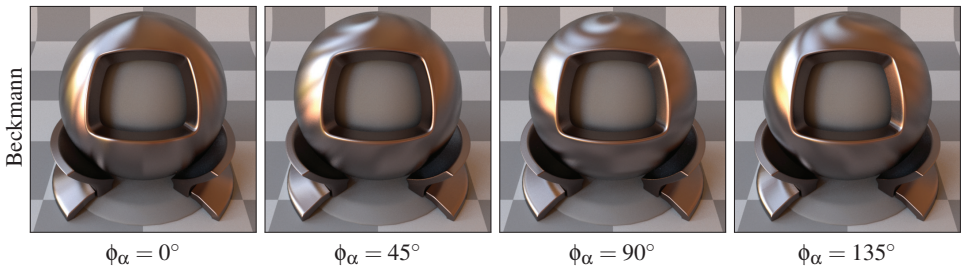


Figure 2.6 – Effect of anisotropy orientation on a microfacet BRDF.

2.1 Torrance-Sparrow Equation

Microfacet theory [TS67] is a framework that models the reflectance of rough materials. It defines the BRDF as a global response of a microscopic surface, referred hereafter as microsurface, composed of microfacets that act as Fresnel mirrors when radiated by light rays. What makes this approach particularly elegant is that, under the assumption that single-bounce mirror reflection dominates on the microsurface, it transposes the problem of evaluating the response of the BRDF into that of determinining the amount of visible and illuminated microfacets whose orientations form a mirror-like reflection between given viewing and illumination directions, which is mathematically tractable; Figure 2.1 (1, 2, 3) illustrates this idea. While determining the unoccluded areas of an arbitrary surface is a difficult problem in general, some solutions can be derived for certain types of surfaces. For such surfaces, the BRDF can be derived analytically from three main terms, according to the microfacet BRDF equation [WMLT07]

$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos\theta_i \cdot \cos\theta_o}. \tag{2.1}$$

We follow standard computer graphics notation for the various terms: $F \in [0,1]$ is the microfacet Fresnel term, $D \geq 0$ the microfacet normal distribution function (NDF) of the microsurface, and $G \in [0,1]$ the geometric attenuation factor (GAF) of the microsurface. We now discuss each of these terms in detail.

```
vec3
microfacet::evalp(const vec3& i, const vec3& o, const void *user_param) const
{
    const microfacet::params params =
        user_param ? *reinterpret_cast<const microfacet::params *>(user_param)
                   : microfacet::params::standard();
    vec3 h = normalize(i + o);
    float_t G = gaf(h, i, o, params);

    if (G > 0.0) {
        float_t cos_theta_d = dot(o, h);
        vec3 F = fresnel(cos_theta_d);
        float_t D = ndf(h, params);

        return (F * D * G) / (4.0 * o.z);
    }

    return vec3(0);
}

vec3
microfacet::eval(const vec3& i, const vec3& o, const void *user_param) const
{
    return (evalp(i, o, user_param) / i.z);
}
```

Listing 2.2 – Evaluating a microfacet BRDF in C++.

Microfacet Fresnel Term The microfacet Fresnel term F describes how microfacets act when a light ray hits them; it can be seen as the fundamental interaction between light and microsurfaces. In contrast to the other terms of Equation (2.1), F may vary across wavelengths and is thus responsible for explaining the color of the material. In general, F is parameterized by Rusinkiewicz's difference angle

$$F = F(\theta_d). \tag{2.2}$$

Various analytic expressions for F have been proposed in the computer graphics literature. Among the most popular models, we find the Fresnel equation for unpolarized light, which was introduced in computer graphics by Cook and Torrance [CT82], as well as its approximation derived by Schlick [Sch94b]; their expressions are given in Table 2.I. When computational performance is critical, such as for video games, Schlick's approximation is often preferred over the unpolarized formulation, due to the simpler arithmetic operations involved. Finally, note that if we set $F = 1$ at any light wavelength, then we can simulate a microsurface composed of ideal mirrors. In such cases, the microfacet BRDF takes on the analytic form

$$f_{r,id} = \frac{D \cdot G}{4 \cdot \cos\theta_i \cdot \cos\theta_o}. \tag{2.3}$$

Figure 2.2 shows materials generated with different F terms, while D and G are fixed.

Ideal	Unpolarized	Schlick [Sch94b]
$F(\theta_d) = 1$	$F(\theta_d) = \frac{1}{2}\frac{(g-c)^2}{(g+c)^2}\left[1 + \frac{(c(g+c)-1)^2}{(c(g-c)+1)^2}\right]$	$F(\theta_d) = F_0 + (1-F_0)(1-c)^5$

Notation η: index of refraction
θ_d: difference angle
$c = \cos\theta_d$
$g^2 = \eta^2 + c^2 - 1$
$F_0 = (\eta-1)^2/(\eta+1)^2$

Table 2.I – Popular Fresnel terms used in computer graphics.

```
/* Fresnel API */
namespace fresnel {

    /* Interface */
    class impl {
    public:
        virtual ~impl() {}
        virtual vec3 eval(float_t cos_theta_d) const = 0;
        virtual impl *copy() const = 0;
    };

} // namespace fresnel
```

Listing 2.3 – Fresnel interface in C++.

Microfacet Normal Distribution Function The microfacet NDF term D gives the proportion of microfacets oriented towards the halfway vector, i.e., the normals that connect the emitter and receiver directions through mirror reflection

$$\mathbf{i} = 2(\mathbf{o} \cdot \mathbf{h})\mathbf{h} - \mathbf{o} \tag{2.4}$$

$$\mathbf{h} = \frac{\mathbf{i} + \mathbf{o}}{\|\mathbf{i} + \mathbf{o}\|} \tag{2.5}$$

Note that these relations are illustrated in Figure 2.1 (3). The microfacet NDF is a directional distribution, defined on the hemisphere Ω_+ oriented towards the up direction of the tangent frame

$$D = D(\mathbf{h}). \tag{2.6}$$

The measure of D over Ω_+ gives the area of the microsurface [Hei14], and should be finite to be geometrically valid, i.e.,

$$\int_{\Omega_+} D(\mathbf{h}) \cdot d\omega_h \in [1, +\infty). \tag{2.7}$$

In addition, D should be normalized such that the projection of the microfacets onto the tangent plane has unit area [WMLT07, Hei14], i.e.,

$$\int_{\Omega_+} D(\mathbf{h}) \cdot \cos\theta_h \cdot d\omega_h = 1. \tag{2.8}$$

Various analytic expressions of D can be found in the computer graphics literature [WMLT07, BSH12, LKYU12]. In the general case, analytic directional distributions are not straightforward to manipulate or design. In contrast to unit directions, the slopes of the microsurface, i.e., its derivatives, are much easier to study, as they live in

the $\mathbb{R}^2$ plane. In the Ω_+ set, normals and slopes are linked through the bijection

$$\tilde{\mathbf{h}} = \begin{bmatrix} -\tan\theta_h \cdot \cos\phi_h = \tilde{x}_h \\ -\tan\theta_h \cdot \sin\phi_h = \tilde{y}_h \end{bmatrix}, \quad \tilde{\mathbf{h}} \in \mathbb{R}^2, \tag{2.9}$$

whose inverse is

$$\mathbf{h} = \frac{1}{\sqrt{\tilde{x}_h^2 + \tilde{y}_h^2 + 1}} \begin{bmatrix} -\tilde{x}_h \\ -\tilde{y}_h \\ 1 \end{bmatrix}, \quad \mathbf{h} \in \Omega_+. \tag{2.10}$$

The bijection between normals and slopes is illustrated geometrically in Figure 2.7. From this bijection, we may define a microfacet NDF from a bivariate slope probability distribution function (PDF) $P \geq 0$ such that

$$D(\mathbf{h}) = P(\tilde{\mathbf{h}}) \cdot \sec^4\theta_h. \tag{2.11}$$

```
float_t microfacet::ndf(const vec3& h, const microfacet::params& params) const
{
    float_t cos_theta_h_sqr = h.z * h.z;
    float_t cos_theta_h_sqr_sqr = cos_theta_h_sqr * cos_theta_h_sqr;
    float_t xslope = -h.x / h.z;
    float_t yslope = -h.y / h.z;

    return (p22(xslope, yslope, params) / cos_theta_h_sqr_sqr);
}
```

Listing 2.4 – Evaluating the NDF term in C++.

The secant term is the Jacobian that converts the measure of microfacet slope probability into a microfacet normal distribution [Hei14]. As long as P is a normalized PDF, i.e.,

$$\int_{\mathbb{R}^2} P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}} = 1, \tag{2.12}$$

Equation (2.8) holds. Among the most popular microfacet slope distributions, we find the Beckmann distribution [Bec65], and the GGX distribution [WMLT07]. The Beckmann distribution is simply a Gaussian PDF, while the GGX distribution emerges from the curvature of ellipsoids [TR75]; their respective expressions are provided in Table 2.II. Figures 2.2 and 2.4 show materials generated with different D terms, while F and G are fixed. Note that the GGX distribution is known as the Trowbridge-Reitz model in the physics literature [TR75], and the bivariate t distribution with two degrees of freedom in the mathematics literature [Jon02].

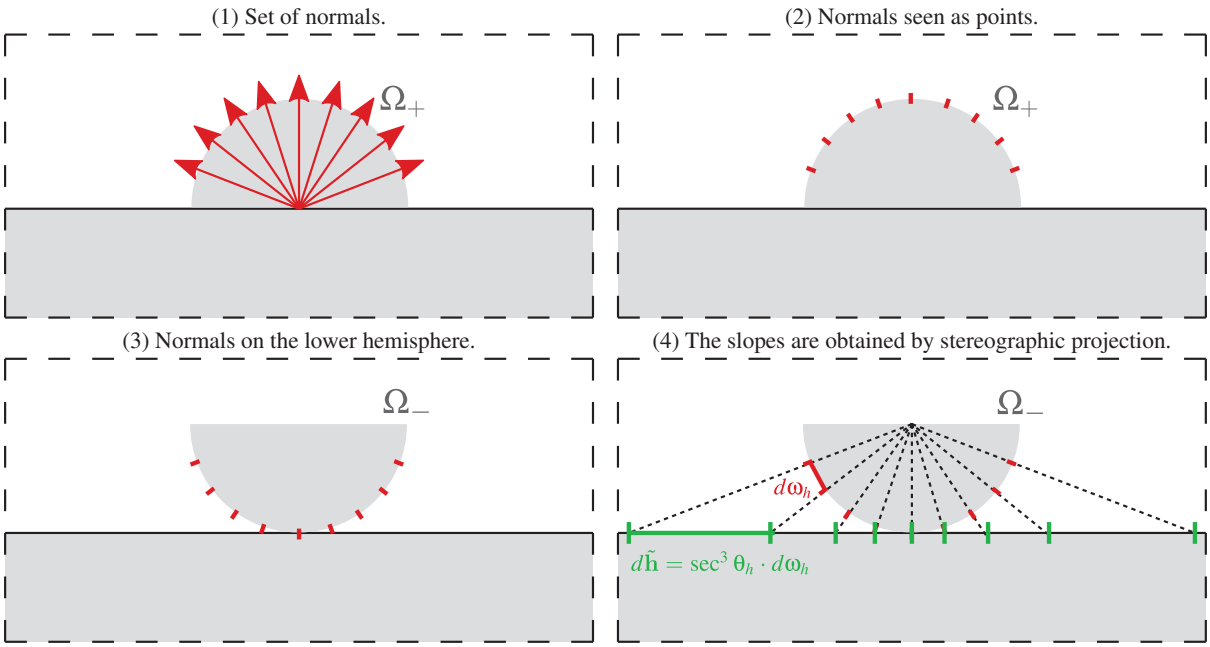


Figure 2.7 – Bijection between slopes and normals. (1) A set of normals defined within Ω_+ can also be seen as (2) a set of points lying on the boundary of Ω_+. In order to compute the slope associated to each normal, we first negate the z component of these points, which creates (3) a set of points that lie on the boundary of Ω_-. (4) The stereographic projection of each of these points onto the plane (shown in green) yields the associated slope; note that this transformation is nonlinear.

Microfacet Geometric Attenuation Factor The microfacet GAF term G is responsible for modelling occlusion effects occurring on the microsurface. Occlusions can be due to shadowing, i.e., the emitter direction, and to masking, i.e., the receiver direction; this idea is illustrated in Figure 2.1 (4, 5). Note that, although their appellation is different, computing the amount of either shadowing or masking boils down to determining the hidden portions of the microsurface from a specific direction. This problem is extremely difficult to solve for arbitrary surfaces, and solutions only exist for very specific surface profiles. In current state-of-the-art models, the GAF is given by the correlated bistatic Smith shadowing function [HMD+14]

$$G(\mathbf{i},\mathbf{o}) = \frac{G_1(\mathbf{i}) \cdot G_1(\mathbf{o})}{G_1(\mathbf{i}) + G_1(\mathbf{o}) - G_1(\mathbf{i}) \cdot G_1(\mathbf{o})}, \tag{2.13}$$

Beckmann	GGX
$P(\tilde{\mathbf{h}}) = \frac{1}{\pi}\exp\left(-\tilde{x}_h^2 - \tilde{y}_h^2\right)$	$P(\tilde{\mathbf{h}}) = \frac{1}{\pi}\frac{1}{(1+\tilde{x}_h^2+\tilde{y}_h^2)^2}$

Notation $\tilde{\mathbf{h}} \in \mathbb{R}^2$: slope of the halfway vector.

Table 2.II – Analytic microfacet slope probability density functions.

```
float_t
microfacet::gaf_smith(
    const vec3& h, const vec3& i, const vec3& o,
    const microfacet::params& params
) const {
    float_t G1_o = g1(h, o, params);
    float_t G1_i = g1(h, i, params);
    float_t tmp = G1_i * G1_o;

    if (tmp > 0.0)
        return (tmp / (G1_i + G1_o - tmp));

    return 0.0; // fully attenuated
}
```

Listing 2.5 – Evaluating the Smith GAF term in C++.

where $G_1 \in [0,1]$ denotes the Smith monostatic shadowing function [Smi67, Hei14]

$$G_1(\mathbf{k}) = \frac{\cos\theta_k}{\int_{\Omega_+} \underline{\mathbf{kh}} \cdot D(\mathbf{h}) \cdot d\omega_h}. \tag{2.14}$$

Note that we use the notation **kh** to express vector dot products, and the underlining to denote a quantity clamped to zero if it is negative, e.g., $\underline{\mathbf{kh}} = \max(0, \mathbf{k}\cdot\mathbf{h})$. The Smith model assumes that the microsurface structure is independent from the GAF term, which essentially means that the microsurface can be approximated by a collection of flakes [Hei14], as illustrated in Figure 2.8. While this approximation may appear to be quite crude, it matches experiments very well in practice [BSB00, BBS02, Hei14]. We also mention here two less realistic GAF models, namely the $\vee$-groove model of Torrance and Sparrow [TS67], and the normal map model. The former assumes that the microsurface is composed exclusively of symmetrical $\vee$-groove cavities, while the latter models the microsurface as a normal mapped surface, which is equivalent to neglecting occlusion effects; both models are illustrated in Figure 2.8, and their equations are given in Table 2.IV. Figure 2.3 shows materials generated with different G terms, while F and D are fixed.

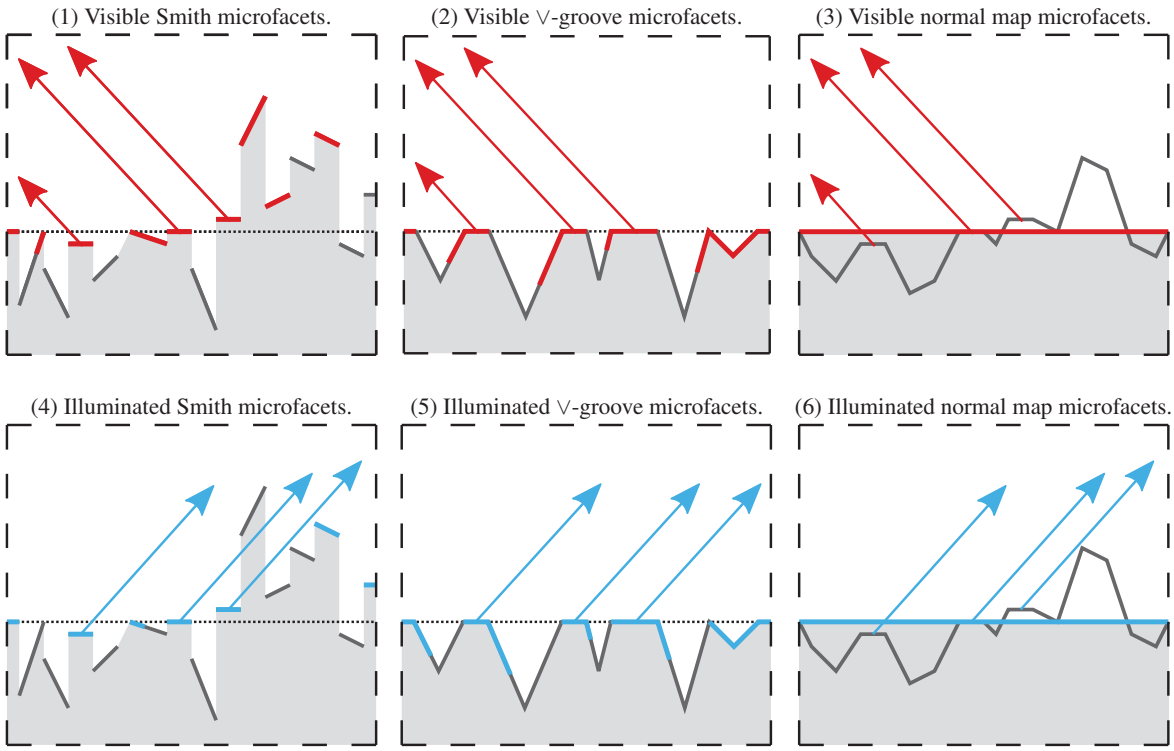


Figure 2.8 – Typical surface profiles handled by existing microfacet GAFs. (1, 4) The Smith GAF assumes that the microsurface structure is irrelevant for occlusion effects. (2, 5) The ∨-groove GAF assumes that the microsurface consists exclusively of symmetric ∨-groove cavities. (3, 6) The normal map GAF neglects all occlusion effects.

2.2 Importance Sampling

Distribution of Visible Slopes Physically based renderers employ BRDFs for shading. On such platforms, shading is typically computed by solving the rendering equation [Kaj86], which we already mentioned in the previous chapter (see Equation (1.3))

$$L_o = \int_{\Omega_+} L_i \cdot f_r \cdot \cos\theta_i \cdot d\omega_i.$$

In the general case, the integral is computed numerically because the incident radiance L_i is not known in advance. When relying on microfacet BRDFs, we can compute this integral efficiently by transforming uniformly distributed samples into the space of visible microfacet normals with a quantile transform [HD14]. The PDF

Beckmann	GGX
$G_1(\mathbf{k}) = \dfrac{2}{1+\mathrm{erf}(\nu)+\frac{\exp(-\nu^2)}{\nu\sqrt{\pi}}}$	$G_1(\mathbf{k}) = \dfrac{2}{1+\sqrt{1+\nu^{-2}}}$

Notation $\mathbf{k} \in \Omega_+$: incident direction, of elevation angle $\theta_k \in [0, \pi/2]$
$\nu = \cot\theta_k$

Table 2.III – The Smith masking term of analytic microfacet slope distributions.

Smith	$\vee$-groove	Normal map
$G = \dfrac{G_1(\mathbf{i}) \cdot G_1(\mathbf{o})}{G_1(\mathbf{i}) + G_1(\mathbf{o}) - G_1(\mathbf{i}) \cdot G_1(\mathbf{o})}$	$G = \min\left(1, 2\dfrac{z_i \cdot z_h}{(\mathbf{i}\cdot\mathbf{h})}, 2\dfrac{z_o \cdot z_h}{(\mathbf{o}\cdot\mathbf{h})}\right)$	$G = \dfrac{z_o \cdot z_h}{(\mathbf{o}\cdot\mathbf{h})}$

Notation $\mathbf{i} \in \Omega_+$: emitter direction, with elevation $z_i \geq 0$
$\mathbf{o} \in \Omega_+$: receiver direction, with elevation $z_o \geq 0$
$\mathbf{h} \in \Omega_+$: halfway direction, with elevation $z_h \geq 0$

Table 2.IV – Various microfacet GAF terms. The Smith model is considered as the most plausible.

$P_{\mathrm{vis}} > 0$ of the distribution of visible microfacet slopes is defined as [HD14]

$$P_{\mathrm{vis}}(\tilde{\mathbf{h}}; \mathbf{k}) = \frac{\underline{\mathbf{kh}} \cdot P(\tilde{\mathbf{h}}) \cdot G_1(\mathbf{k})}{\cos\theta_h \cdot \cos\theta_k}. \tag{2.15}$$

Note that in order to simplify notations, we will make the direction parameter $\mathbf{k}$ implicit, thus writing, e.g., $P_{\mathrm{vis}}(\tilde{\mathbf{h}})$ instead of $P_{\mathrm{vis}}(\tilde{\mathbf{h}}; \mathbf{k})$.

Quantile Functions Let $P_2 \geq 0$ denote the marginal PDF

$$P_2(\tilde{x}_h) = \int_{-\infty}^{\infty} P_{\mathrm{vis}}(\tilde{x}_h, \tilde{y}_h) \cdot d\tilde{y}_h. \tag{2.16}$$

The PDFs P_{vis} and P_2 are linked through the relation

$$P_{\mathrm{vis}}(\tilde{x}_h, \tilde{y}_h) = P_2(\tilde{x}_h) \cdot P_3(\tilde{y}_h | \tilde{x}_h) \Rightarrow P_3(\tilde{y}_h | \tilde{x}_h) = \frac{P_{\mathrm{vis}}(\tilde{x}_h, \tilde{y}_h)}{P_2(\tilde{x}_h)},$$

where $P_3 \geq 0$ is the PDF of $\tilde{y}_h$ conditioned on $\tilde{x}_h$. Let $F_2 \in [0,1]$ and $F_3 \in [0,1]$ respectively denote the cumulative distribution function of P_2 and P_3, i.e.,

$$F_2(\tilde{x}_h) = \int_{-\infty}^{\tilde{x}_h} P_2(\tilde{x}) \cdot d\tilde{x},$$
$$F_3(\tilde{y}_h|\tilde{x}_h) = \int_{-\infty}^{\tilde{y}_h} P_3(\tilde{y}|\tilde{x}_h) \cdot d\tilde{y},$$

and $Q_2 = F_2^{-1}$ and $Q_3 = F_3^{-1}$ their respective quantile functions. Given realizations $u_1, u_2 \in [0,1]$ of two independent uniform variates, the variates obtained by the quantile transformation

$$\tilde{\mathbf{h}} = \begin{bmatrix} Q_2(u_1) \\ Q_3(u_2|Q_2(u_1)) \end{bmatrix}$$

are distributed according to P_{vis} [HD14]. Note that if we transform these slopes into normals according to Equation (2.10), then the resulting normals are distributed according to the distribution of visible microfacet normals [HD14], which has the PDF

$$D_{\text{vis}}(\mathbf{h};\mathbf{k}) = \frac{\underline{\mathbf{kh}} \cdot D(\mathbf{h}) \cdot G_1(\mathbf{k})}{\cos\theta_k}. \tag{2.17}$$

```
float_t
microfacet::vndf(
    const vec3& h, const vec3& k,
    const microfacet::params& params
) const {
    float_t D = ndf(h, params);
    float_t G1 = g1(h, k, params);
    float_t kh = max((float_t)0.0, dot(k, h));

    return (kh * D * G1 / k.z);
}
```

Listing 2.6 – Evaluating the distribution of visible normals in C++.

Solving the Rendering Equation In order to compute the rendering equation numerically, we first reexpress the integral as a measure in terms of infinitesimal solid angles oriented towards halfway vectors. We accomplish this with the change of variable $\omega_i = 2(\omega_h \cdot \omega_o)\omega_h - \omega_o$ ($d\omega_i = 4 \cdot \mathbf{oh} \cdot d\omega_h$ [WMLT07]), which yields

$$L_o = \int_{\Omega_+} L_i \cdot f_r \cdot \cos\theta_i \cdot 4 \cdot \mathbf{oh} \cdot d\omega_h. \tag{2.18}$$

We then express this measure in slope space with the change of variable $\omega_h = \omega_h(\tilde{\mathbf{h}})$ ($d\omega_h = \cos^3\theta_h \cdot d\tilde{\mathbf{h}}$)

$$L_o = \int_{\mathbb{R}^2} L_i \cdot f_r \cdot \cos\theta_i \cdot 4 \cdot \mathbf{oh} \cdot \cos^3\theta_h \cdot d\tilde{\mathbf{h}} \tag{2.19}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} L_i \cdot f_r \cdot \cos\theta_i \cdot 4 \cdot \mathbf{oh} \cdot \cos^3\theta_h \cdot d\tilde{x}_h \cdot d\tilde{y}_h. \tag{2.20}$$

From this double integral, we can then apply quantile transformations. We start with the $\tilde{x}$ slope component. Letting $\tilde{x}_h = Q_2(u_1)$ ($d\tilde{x}_h = Q_2'(u_1) \cdot du_1 = \frac{1}{P_2(Q_2(u_1))} \cdot du_1$), we get

$$L_o = \int_{-\infty}^{\infty}\int_0^1 \frac{L_i \cdot f_r \cdot \cos\theta_i \cdot 4 \cdot \mathbf{oh} \cdot \cos^3\theta_h}{P_2(Q_2(u_1))} \cdot du_1 \cdot d\tilde{y}_h. \tag{2.21}$$

We then apply the second quantile substitution. Letting $\tilde{y}_h = Q_3(u_2|\tilde{x}_h)$ ($d\tilde{y}_h = Q_3'(u_2|\tilde{x}_h) \cdot du_2$, and so $d\tilde{y}_h = \frac{1}{P_3(Q_3(u_2)|Q_2(u_1))} \cdot du_2$), we get

$$L_o = \int_0^1\int_0^1 \frac{L_i \cdot f_r \cdot \cos\theta_i \cdot 4 \cdot \mathbf{oh} \cdot \cos^3\theta_h}{P_2(Q_2(u_1)) \cdot P_3(Q_3(u_2)|Q_2(u_1))} \cdot du_1 \cdot du_2, \tag{2.22}$$

which, using the fact that $P_{\text{vis}} = P_2 \cdot P_3$, as well as the definition of the microfacet BRDF, i.e., Equation (2.1), leads after simplifications to

$$L_o = \int_0^1\int_0^1 L_i \cdot F(\theta_d) \cdot \frac{G(\mathbf{i},\mathbf{o})}{G_1(\mathbf{o})} \cdot du_1 \cdot du_2. \tag{2.23}$$

We can then apply Euler's rule to compute the integral numerically.

2.3 Controlling Roughness

We can control the roughness of a microfacet BRDF intuitively by scaling the slopes of the microsurface, i.e., its derivatives. This feature is particularly relevant for artistic authoring. As scale increases, the surface becomes rougher. Conversely, as scale decreases, the surface becomes smoother. These ideas are illustrated in Figure 2.9.

Isotropic Roughness If we scale the microsurface isotropically by a factor $\alpha > 0$, then the distribution of microfacet slopes is also scaled. This means that we can retrieve the value of D by simply offsetting the slope distribution. Thus, we can define a microfacet NDF with explicit control over isotropic roughness as

$$D(\mathbf{h};\alpha) = P\left(\frac{\tilde{x}_h}{\alpha}, \frac{\tilde{y}_h}{\alpha}\right) \frac{\sec^4\theta_h}{\alpha^2}. \tag{2.24}$$

Beckmann	GGX
$F_3(\tilde{y}_h\|\tilde{x}_h) = \frac{1}{2} + \frac{1}{2}\operatorname{erf}(\tilde{x}_h)$	$F_3(\tilde{y}_h\|\tilde{x}_h) = \frac{1}{2} + \frac{\sigma\tilde{y}_h}{\pi\sigma + \pi\tilde{y}_h^2} + \frac{\arctan(\frac{\tilde{y}_h}{\sigma})}{\pi}$
$Q_3(u\|\tilde{x}_h) = \operatorname{erfinv}(2u-1)$	$Q_3(u\|\tilde{x}_h) = \sigma \begin{cases} -\sqrt{\frac{1}{\operatorname{ribinv}(2u, \frac{3}{2}, \frac{1}{2})} - 1} & \text{if } u < \frac{1}{2} \\ \sqrt{\frac{1}{\operatorname{ribinv}(2-2u, \frac{3}{2}, \frac{1}{2})} - 1} & \text{else.} \end{cases}$

Notation $\tilde{x}_h \in \mathbb{R}$: x component of the halfway slope
$u \in [0,1)$: uniform variate
erfinv: inverse of the error function
ribinv: inverse of the regularized incomplete beta function
$\sigma^2 = 1 + \tilde{x}_h$

Table 2.V – Conditional CDF and QF of analytic visible slope distributions.

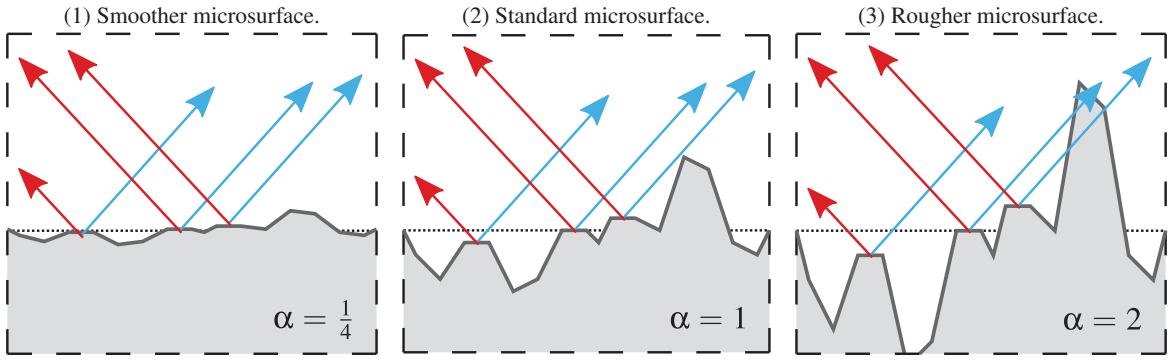


Figure 2.9 – Geometric interpretation of roughness. Roughness can be interpreted as an amount of vertical stretch that is applied to a standard microsurface, whose roughness is $\alpha = 1$ by convention.

Note that this expression of D still satisfies Equation (2.8) and thus remains physically sound. The visual impact of isotropic roughness on the microfacet BRDF is illustrated in Figure 2.4.

Elliptically Anisotropic Roughness We can also provide control over the anisotropy of materials. For this, we can stretch and rotate the microsurface along two major directions. These transformations can be controlled by tweaking the parameters of an ellipse, i.e., two radii $\alpha_1, \alpha_2 > 0$ and an angle $\phi_\alpha \in [-\pi/2, \pi/2]$. Thus, we

can define a microfacet NDF with explicit control over anisotropic roughness as

$$D(\mathbf{h};\alpha_1,\alpha_2,\phi_\alpha) = P\left(\frac{-\tan\theta_h\cdot\cos(\phi_h+\phi_\alpha)}{\alpha_1},\frac{-\tan\theta_h\cdot\sin(\phi_h+\phi_\alpha)}{\alpha_2}\right)\frac{\sec^4\theta_h}{\alpha_1\alpha_2}. \tag{2.25}$$

Note that this expression of D still satisfies Equation (2.8) and thus remains physically sound. Alternatively, we may write the transformation in terms of bivariate distribution parameters, which involve two scale parameters $\alpha_x,\alpha_y > 0$ and a correlation coefficient $\rho \in (-1,1)$

$$D(\mathbf{h};\alpha_1,\alpha_2,\phi_\alpha) = P\left(\frac{\tilde{x}_h}{\alpha_x},\frac{\alpha_x\cdot\tilde{y}_h-\rho\alpha_y\cdot\tilde{x}_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\right)\frac{\sec^4\theta_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}. \tag{2.26}$$

Note that in order to simplify notations, we shall use the notation

$$P(\tilde{\mathbf{h}};\alpha_1,\alpha_2,\phi_\alpha) = P_{\text{std}}\left(\frac{\tilde{x}_h}{\alpha_x},\frac{\alpha_x\cdot\tilde{y}_h-\rho\alpha_y\cdot\tilde{x}_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\right)\frac{1}{\alpha_x\alpha_y\sqrt{1-\rho^2}}, \tag{2.27}$$

so that we can write

$$D(\mathbf{h};\alpha_1,\alpha_2,\phi_\alpha) = P(\tilde{\mathbf{h}};\alpha_1,\alpha_2,\phi_\alpha)\sec^4\theta_h, \tag{2.28}$$

where $P_{\text{std}}(\tilde{\mathbf{h}}) = P(\tilde{\mathbf{h}};1,1,0)$ gives the microfacet slope PDF of the standard surface, i.e., free of transformations.

```
float_t microfacet::ndf(const vec3& h, const microfacet::params& params) const
{
    float_t cos_theta_h_sqr = h.z * h.z;
    float_t cos_theta_h_sqr_sqr = cos_theta_h_sqr * cos_theta_h_sqr;
    float_t xslope = -h.x / h.z;
    float_t yslope = -h.y / h.z;

    return (p22(xslope, yslope, params) / cos_theta_h_sqr_sqr);
}

float_t microfacet::p22(float_t x, float_t y, const params& params) const
{
    float_t nrm = params.m_a1 * params.m_a2;
    float_t x_ = x / params.m_ax;
    float_t tmp1 = params.m_ax * y - params.m_rho * params.m_ay * x;
    float_t tmp2 = params.m_ax * params.m_ay * params.m_rho;
    float_t y_ = tmp1 / tmp2;

    return p22_std(x_, y_) / nrm;
}
```

Listing 2.7 – Evaluating the distribution of visible normals in C++.

From an implementation standpoint, Equation (2.26) is preferable to Equation (2.25) whenever trigonometric functions induce performance issues. Note that the bivariate distribution parameters are linked to the ellipse

parameters through the relations

$$\alpha_x^2 = \frac{1}{2}\left(\alpha_1^2 + \alpha_2^2 + (\alpha_1^2 - \alpha_2^2)\cdot\cos(2\phi_\alpha)\right) \tag{2.29}$$

$$\alpha_y^2 = \frac{1}{2}\left(\alpha_1^2 + \alpha_2^2 - (\alpha_1^2 - \alpha_2^2)\cdot\cos(2\phi_\alpha)\right) \tag{2.30}$$

$$\rho = \frac{\alpha_1^2 - \alpha_2^2}{\alpha_x\alpha_y}\cdot\cos\phi_\alpha\cdot\sin\phi_\alpha, \tag{2.31}$$

whose inverses are

$$\alpha_1^2 = \frac{1}{2}\left(\alpha_x^2 + \alpha_y^2 + \sqrt{(\alpha_x^2 - \alpha_y^2)^2 + 4\rho^2\alpha_x^2\alpha_y^2}\right) \tag{2.32}$$

$$\alpha_2^2 = \frac{1}{2}\left(\alpha_x^2 + \alpha_y^2 - \sqrt{(\alpha_x^2 - \alpha_y^2)^2 + 4\rho^2\alpha_x^2\alpha_y^2}\right) \tag{2.33}$$

$$\phi = \arctan\left(\frac{\alpha_x^2 - \alpha_y^2 - \sqrt{(\alpha_x^2 - \alpha_y^2)^2 + 4\rho^2\alpha_x^2\alpha_y^2}}{2\rho\alpha_x\alpha_y}\right). \tag{2.34}$$

The visual impact of anisotropic roughness on a microfacet BRDF is illustrated in Figure 2.5.

Roughness Invariance of the Masking Term Defining roughness as a slope-space scaling operation makes it possible to define a wide variety of microfacet NDFs from a single slope distribution by using offset evaluations; we refer to this property as roughness invariance. The Smith masking term is also subject to this property [Hei14], which we prove here mathematically. We start with the definition of the Smith masking term, i.e., Equation (2.14), using an elliptically stretched NDF. We employ several substitutions and algebric

manipulations provided in the parentheses on the right-hand side for convenience

$$
\begin{aligned}
G_1(\mathbf{k};\alpha_1,\alpha_2,\phi_\alpha) &= \frac{z_k}{\int_{\Omega_+} D(\mathbf{h};\alpha_1,\alpha_2,\phi_\alpha)\cdot\underline{\mathbf{kh}}\cdot d\omega_h} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}\left(\frac{\tilde{x}_h}{\alpha_x},\frac{\alpha_x\cdot\tilde{y}_h-\rho\alpha_y\cdot\tilde{x}_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\right)\frac{\sec\theta_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\cdot\underline{\mathbf{kh}}\cdot d\tilde{\mathbf{h}}} && \begin{pmatrix}\omega_h=\omega_h(\tilde{\mathbf{h}})\\ \Rightarrow d\omega_h=\cos^3\theta_h\cdot d\tilde{\mathbf{h}}\end{pmatrix} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}\left(\frac{\tilde{x}_h}{\alpha_x},\frac{\alpha_x\cdot\tilde{y}_h-\rho\alpha_y\cdot\tilde{x}_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\right)\frac{1}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\cdot\underline{z_k-x_k\tilde{x}_h-y_k\tilde{y}_h}\cdot d\tilde{\mathbf{h}}} && \left(\sec\theta_h\cdot\mathbf{h}=\begin{bmatrix}-\tilde{x}_h\\-\tilde{y}_h\\1\end{bmatrix}\right) \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\cdot\underline{z_k-x_k\tilde{x}_h-y_k\tilde{y}_h}\cdot d\tilde{\mathbf{m}}} && \begin{pmatrix}\tilde{x}_h=\alpha_x\tilde{x}_m,\\ \tilde{y}_h=\alpha_y(\rho\tilde{x}_m+\sqrt{1-\rho^2}\tilde{y}_m)\\ \Rightarrow d\tilde{\mathbf{h}}=\alpha_x\alpha_y\sqrt{1-\rho^2}\cdot d\tilde{\mathbf{m}}\end{pmatrix} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\cdot\underline{z_k-a\tilde{x}_m-b\tilde{y}_m}\cdot d\tilde{\mathbf{m}}} && \begin{pmatrix}a=\alpha_x x_k+\rho\alpha_y y_k\\ b=\alpha_y y_k\sqrt{1-\rho^2}\end{pmatrix} \\
&= \frac{z_{k'}}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\sec\theta_m\cdot\underline{\mathbf{k}'\mathbf{m}}\cdot d\tilde{\mathbf{m}}} && \left(\mathbf{k}'=\frac{1}{\sqrt{a^2+b^2+z_k^2}}\begin{bmatrix}a\\b\\z_k\end{bmatrix}\right) \\
&= \frac{z_{k'}}{\int_{\Omega_+} D(\mathbf{m};1,1,0)\cdot\underline{\mathbf{k}'\mathbf{m}}\cdot d\omega_m} && \begin{pmatrix}\tilde{\mathbf{m}}=\tilde{\mathbf{m}}(\omega_m)\\ \Rightarrow d\tilde{\mathbf{m}}=\sec^3\theta_m\cdot d\omega_m\end{pmatrix} \\
&= G_1(\mathbf{k}';1,1,0) \\
&= G_{1,\text{std}}(\mathbf{k}').
\end{aligned}
$$

This demonstration shows that the Smith term of a scaled surface is equal to the Smith term of an unscaled Smith term evaluated with an offset parameter; the relation is illustrated in Figure 2.10.

```
float_t
microfacet::g1_smith(
    const vec3& h, const vec3& k,
    const microfacet::params& params
) const {
    float_t a = k.x * params.m_ax + k.y * params.m_ay * params.m_rho;
    float_t b = k.y * params.m_ay * params.m_sqrt_one_minus_rho_sqr;
    vec3 kprime = normalize(vec3(a, b, k.z));

    return g1_smith_std(h, kprime);
}
```

Listing 2.8 – Evaluating the Smith masking term with arbitrary roughness parameters in C++.

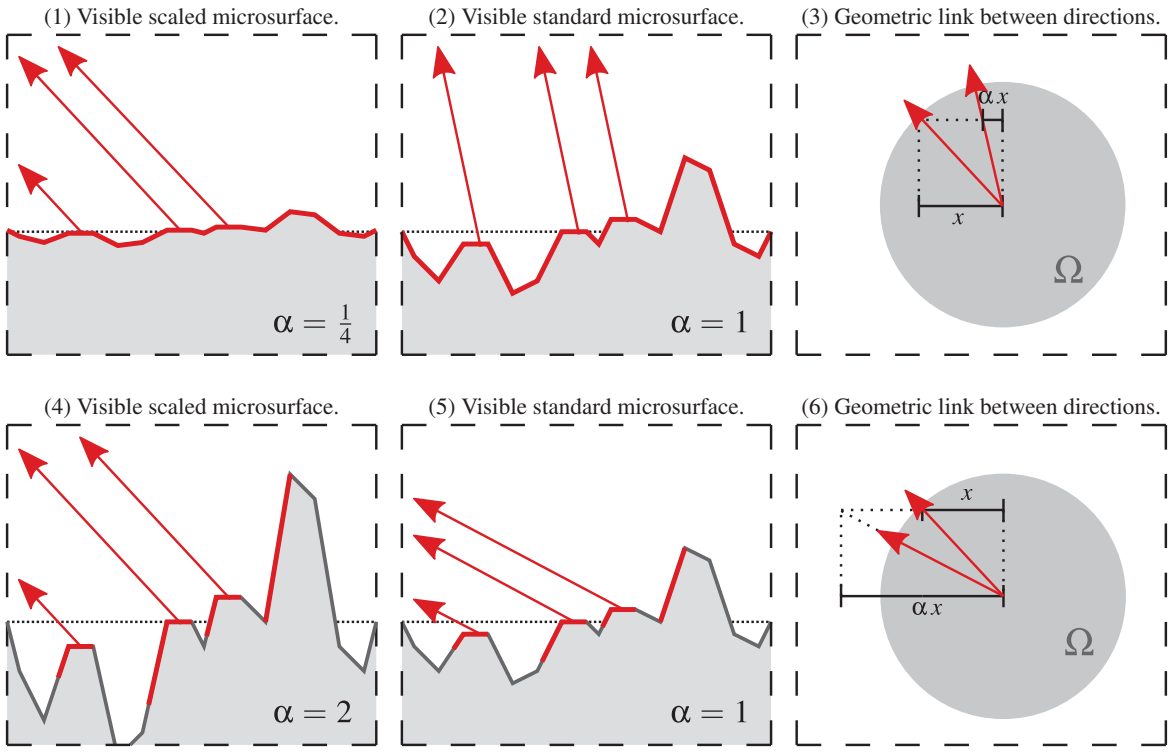


Figure 2.10 – Roughness invariance of the monostatic shadowing term. Shadowing from any direction on (1, 4) a scaled surface can be transposed into (2, 5) shadowing on the standard surface due to another direction. (3, 6) The incident directions are linked geometrically.

Importance Sampling with Roughness Parameters Since both NDF and GAF are roughness invariant, it follows by construction (see Equation (2.15)) that the distribution of visible slopes is also roughness invariant. We can also exploit roughness invariance to produce samples that are distributed according to a scaled microsurface from an unscaled microsurface. Indeed, if $\tilde{\mathbf{m}} \in \mathbb{R}^2$ constitutes a sample of an unscaled visible microfacet slope distribution from direction $\mathbf{k}'$, then the slope

$$\tilde{\mathbf{h}} = \begin{bmatrix} \alpha_x \cdot \tilde{x}_m \\ \alpha_y(\rho \cdot \tilde{x}_m + \sqrt{1-\rho^2} \cdot \tilde{y}_m) \end{bmatrix} \in \mathbb{R}^2 \tag{2.35}$$

constitutes a sample of the visible microfacet slope distribution of the scaled surface from direction $\mathbf{k}$; this operation is illustrated in Figure 2.11. We can thus exploit the quantile functions introduced in the previous section to produce visible slope samples of arbitrary roughness.

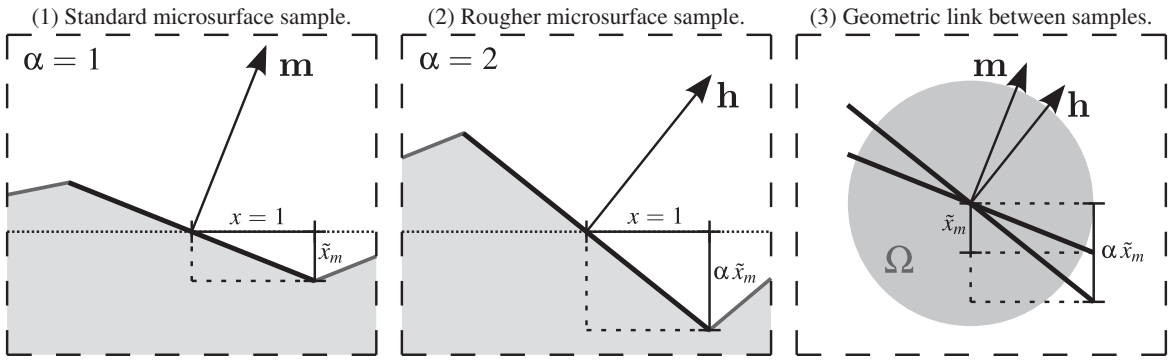


Figure 2.11 – Sampling a rough microsurface from a standard microsurface.

```
vec3
microfacet::sample(
    float_t u1, float_t u2,
    const vec3& o,
    const void *user_param
) const {
    const microfacet::params params =
        user_param ? *reinterpret_cast<const microfacet::params *>(user_param)
                   : microfacet::params::standard();
    float_t tx_h, ty_h, tx_m, ty_m;

    // create a standard variate
    switch (m_gaf) {
        case GAF_NONE: sample_nmap(u1, u2, o, &tx_m, &ty_m); break;
        case GAF_VGROOVE: sample_vgroove(u1, u2, o, &tx_m, &ty_m); break;
        case GAF_SMITH: sample_smith(u1, u2, o, &tx_m, &ty_m); break;
        default: abort(); // should never happen
    }

    // warp the variate with the microfacet parameters
    tx_h = params.m_ax * tx_m;
    float_t choleski = params.m_rho * tx_m
                     + params.m_sqrt_one_minus_rho_sqr * ty_m;
    ty_h = params.m_ay * choleski;

    // return the associated normal
    return normalize(vec3(-tx_h, -ty_h, 1));
}
```

Listing 2.9 – Importance sampling a microfacet BRDF with arbitrary roughness parameters in C++.

2.4 Implementation Details

```
/* Radial microfacets */
class radial : public microfacet {
public:
    radial(int gaf = microfacet::GAF_SMITH,
           const fresnel::impl& fresnel = fresnel::ideal(),
           bool shadow = true):
        microfacet(gaf, fresnel, shadow) {}

    // queries
    virtual float_t g1_radial(float_t cot_theta_k) const = 0;
    virtual float_t p22_radial(float_t r_sqr) const = 0;
private:
    float_t p22_std(float_t x, float_t y) const;
    float_t g1_smith_std(const vec3& k) const;
};
```

Listing 2.10 – C++ implementation for radial microfacet slope distributions.

Radial Slope Distributions Some slope distributions such as the Beckmann and GGX slope distributions are radial distributions, i.e., their PDF can be expressed as a 1D function

$$P(\tilde{\mathbf{h}}) = P_r(r_h), \tag{2.36}$$

where $r_h = \tan\theta_h = \sqrt{\tilde{x}_h^2 + \tilde{y}_h^2}$ is the radius of the slope in polar coordinates, and function $P_r > 0$ is normalized such that

$$\int_0^{2\pi} \int_0^{\infty} r_h \cdot P_r(r_h) \cdot dr_h \cdot d\phi_h = 1. \tag{2.37}$$

We can exploit Equation (2.36) to improve the implementation of microfacet BRDFs built from radial slope distributions. Indeed, for such distributions, the evaluation of the microfacet NDF becomes

$$D(\mathbf{h}; \alpha_1, \alpha_2, \phi_\alpha) = P_r\left(\sqrt{\left(\frac{\tilde{x}_h}{\alpha_x}\right)^2 + \left(\frac{\alpha_x \cdot \tilde{y}_h - \rho\alpha_y \cdot \tilde{x}_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\right)^2}\right) \frac{\sec^4\theta_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}. \tag{2.38}$$

```
float_t radial::p22_std(float_t x, float_t y) const
{
    return p22_radial(sqrt(x * x + y * y));
}
```

Listing 2.11 – C++ implementation for radial microfacet slope distributions.

The Smith masking term can also be simplified into a 1D function. To show this, we start from Equation (2.14) and reexpress the integral in slope space with the change of variables $\omega_h = \omega_h(\tilde{\mathbf{h}})$ ($d\omega_h = \sec^3\theta_h \cdot d\tilde{\mathbf{h}}$)

$$G_1(\mathbf{k}) = \frac{\cos\theta_k}{\int_{\mathbb{R}^2} \underline{z_k - x_k\tilde{x}_h - y_k\tilde{y}_h} \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}}.$$

Using the fact that $x_k = \sin\theta_k \cdot \cos\phi_k$, $y_k = \sin\theta_k \cdot \cos\phi_k$, and $z_k = \cos\theta_k$, we factorize the denominator by $\sin\theta_k$, which yields

$$G_1(\mathbf{k}) = \frac{\cot\theta_k}{\int_{\mathbb{R}^2} \underline{\cot\theta_k - (\cos\phi_k \cdot \tilde{x}_h + \sin\phi_k \cdot \tilde{y}_h)} \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}}.$$

We then reexpress the integral in polar coordinates, letting $\tilde{x}_h = r_h \cdot \cos\phi_h$ and $\tilde{y}_h = r_h \cdot \sin\phi_h$ ($d\tilde{\mathbf{h}} = r_h \cdot dr_h \cdot d\phi_h$)

$$G_1(\mathbf{k}) = \frac{\cot\theta_k}{\int_0^{2\pi}\int_0^{\infty} \underline{\cot\theta_k - r_h(\cos\phi_k \cdot \cos\phi_h + \sin\phi_k \cdot \sin\phi_h)} \cdot P_r(r_h) \cdot r_h \cdot dr_h \cdot d\phi_h},$$

which, using the relation $\cos(a-b) = \cos a \cdot \cos b + \sin a \cdot \sin b$, yields

$$G_1(\mathbf{k}) = \frac{\cot\theta_k}{\int_0^{2\pi}\int_0^{\infty} \underline{\cot\theta_k - r_h \cdot \cos(\phi_h - \phi_k))} \cdot P_r(r_h) \cdot r_h \cdot dr_h \cdot d\phi_h}.$$

Letting $\phi_h = \phi_m + \phi_k$ ($d\phi_h = d\phi_m$), and going back to slope space with $\tilde{x}_m = r_h \cdot \cos\phi_m$ and $\tilde{y}_m = r_h \cdot \sin\phi_m$ ($dr_h \cdot d\phi_m = \cot\theta_h \cdot d\tilde{\mathbf{m}}$) yields

$$\begin{aligned} G_1(\mathbf{k}) &= \frac{\cot\theta_k}{\int_0^{2\pi}\int_0^{\infty} \underline{\cot\theta_k - r_h \cdot \cos\phi_m} \cdot P_r(r_h) \cdot r_h \cdot dr_h \cdot d\phi_m} \\ &= \frac{\cot\theta_k}{\int_{\mathbb{R}^2} \underline{\cot\theta_k - \tilde{x}_h} \cdot P(\tilde{x}_h, \tilde{y}_h) \cdot d\tilde{\mathbf{h}}}, \end{aligned}$$

which may be written more concisely as

$$\begin{aligned} G_1(\mathbf{k}) &= \frac{\cot\theta_k}{\int_{-\infty}^{\cot\theta_k} (\cot\theta_k - \tilde{x}_h) \cdot P_1(\tilde{x}_h) \cdot d\tilde{x}_h} \\ &= G_{1\_radial}(\cot\theta_k), \end{aligned}$$

where P_1 is the marginal slope PDF

$$P_1(\tilde{x}_h) = \int_{-\infty}^{+\infty} P(\tilde{x}_h, \tilde{y}_h) \cdot d\tilde{y}_h. \tag{2.39}$$

```
float_t radial::g1_smith_std(const vec3& h, const vec3& k) const
{
    if (k.z <= DJB_EPSILON) return 0.0;
    if (k.z >= 1.0 - DJB_EPSILON) return 1.0;
    float_t cot_theta_k = k.z * inversesqrt(1.0 - k.z * k.z);

    return g1_radial(cot_theta_k);
}
```

Listing 2.12 – C++ implementation for radial microfacet slope distributions.

Validating the Evaluation API Two simple tests can be devised to validate the evaluation API of a microfacet BRDF. The first test consists in checking the normalization of the NDF, i.e., making sure that Equation (2.8) holds

$$\int_{\Omega_+} D(\mathbf{h}) \cdot \cos\theta_h \cdot d\omega_h = 1.$$

The second test consists in checking the normalization of the distribution of the visible NDF, i.e., making sure that the following equation holds for any $\mathbf{k} \in \Omega_+$

$$\int_{\Omega_+} D_{\text{vis}}(\mathbf{h};\mathbf{k}) \cdot d\omega_h = 1.$$

To compute these integrals, we first express the integrals in spherical coordinates with the change of variable $\omega_h = \omega_h(\theta_h, \phi_h)$ ($d\omega_h = \sin\theta_h \cdot d\theta_h \cdot d\phi_h$). We then make the changes of variable $\theta_h = u_1^2\pi/2$ ($d\theta_h = \pi u_1 \cdot du_1$) and $\phi_h = 2\pi u_2$ ($d\phi_h = 2\pi \cdot du_2$) so that we can apply Euler's rule. We thus have

$$\begin{aligned}
\int_{\Omega_+} \cos\theta_h \cdot D(\mathbf{h}) \cdot d\omega_h &= \int_0^{2\pi}\int_0^{\pi/2} \cos\theta_h \cdot D(\mathbf{h}) \cdot \sin\theta_h \cdot d\theta_h \cdot d\phi_h \\
&= 2\pi^2 \int_0^1 \int_0^1 \cos(u_1^2\pi/2) \cdot D(\mathbf{h}) \cdot \sin(u_1^2\pi/2) \cdot u_1 \cdot du_1 \cdot du_2 \\
&= 2\pi^2 \frac{1}{N_i}\frac{1}{N_j} \sum_{i=0}^{N_i}\sum_{j=0}^{N_j} \cos(u_i^2\pi/2) \cdot D(\mathbf{h}) \cdot \sin(u_i^2\pi/2) \cdot u_i,
\end{aligned}$$

and similarly

$$\int_{\Omega_+} D_{\text{vis}}(\mathbf{h};\mathbf{k}) \cdot d\omega_h = 2\pi^2 \frac{1}{N_i}\frac{1}{N_j} \sum_{i=0}^{N_i}\sum_{j=0}^{N_j} D_{\text{vis}}(\mathbf{h};\mathbf{k}) \cdot \sin(u_i^2\pi/2) \cdot u_i,$$

where $u_i = i/N_i$, $u_j = j/N_j$, $\theta_h = \pi u_i/2$, and $\phi_h = 2\pi u_j$. Note that the change of variable $\theta_h = u_1^2\pi/2$ produces more evaluations near the pole of Ω_+, which usually results in faster convergence than with uniform hemispherical sampling, i.e., with the change of variable $\theta_h = u_1\pi/2$.

```
bool test_ndf(const djb::microfacet& brdf)
{
    int N_i = 270;
    int N_j = 360;
    double nint = 0.0;

    for (int i = 0; i < N_i; ++i) {
        double u_i = (double)i / (double)N_i;
        double theta_h = u_i * u_i * M_PI / 2.0;

        for (int j = 0; j < N_j; ++j) {
            double u_j = (double)j / (double)N_j;
            double phi_h = u_j * 2.0 * M_PI;
            djb::vec3 h = djb::vec3(theta_h, phi_h);

            nint+= cos(theta_h) * brdf.ndf(h) * sin(theta_h) * u_i;
        }
    }
    nint*= 2.0 * M_PI * M_PI / (double)(N_i * N_j);

    return (fabs(nint - 1.0) < EPSILON);
}
```

Listing 2.13 – C++ microfacet NDF validation test.

```
bool test_vndf(const djb::microfacet& brdf, const djb::vec3& k)
{
    int N_i = 270;
    int N_j = 360;
    double nint = 0.0;

    for (int i = 0; i < N_i; ++i) {
        double u_i = (double)i / (double)N_i;
        double theta_h = u_i * u_i * M_PI / 2.0;

        for (int j = 0; j < N_j; ++j) {
            double u_j = (double)j / (double)N_j;
            double phi_h = u_j * 2.0 * M_PI;
            djb::vec3 h = djb::vec3(theta_h, phi_h);

            nint+= brdf.vndf(h, k) * sin(theta_h) * u_i;
        }
    }
    nint*= 2.0 * M_PI * M_PI / (double)(N_i * N_j);

    return (fabs(nint - 1.0) < EPSILON);
}
```

Listing 2.14 – C++ microfacet visible NDF validation test.

Chapter 3

Noncentral Microfacet Theory

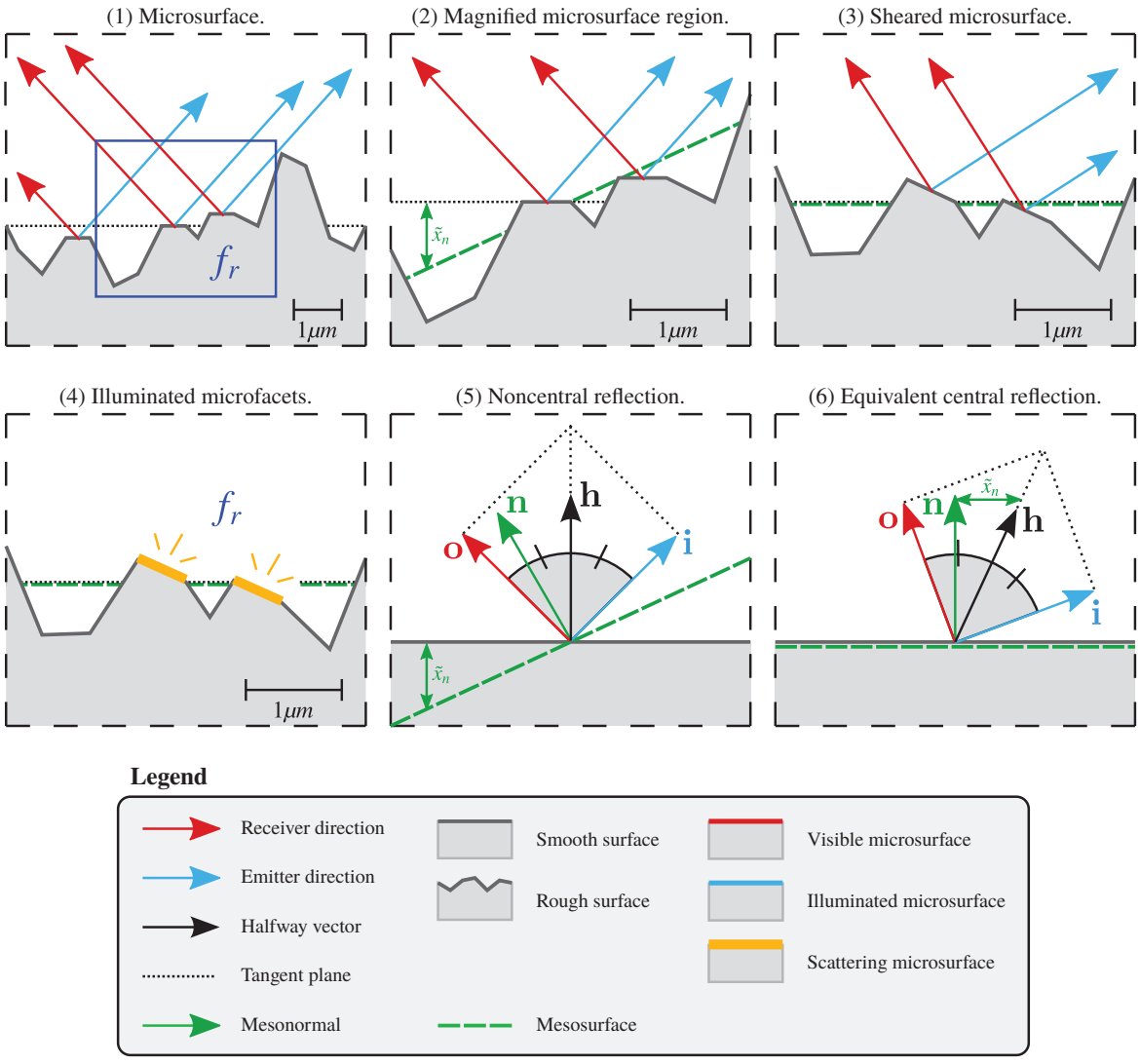


Figure 3.1 – Geometric overview of noncentral microfacet theory. Can microfacet theory be made applicable to multiple scales? (1) In order to derive the microfacet BRDF of a microsurface at any scale, microfacet theory must be extended to account for (2) configurations where the mean surface differs from the tangent plane. (3) By shearing such surfaces back to a standard configuration, we show that (4) the BRDF of such surfaces can be determined from the standard microsurface by (5, 6) transforming the illumination and observation directions.

3.1 Motivation

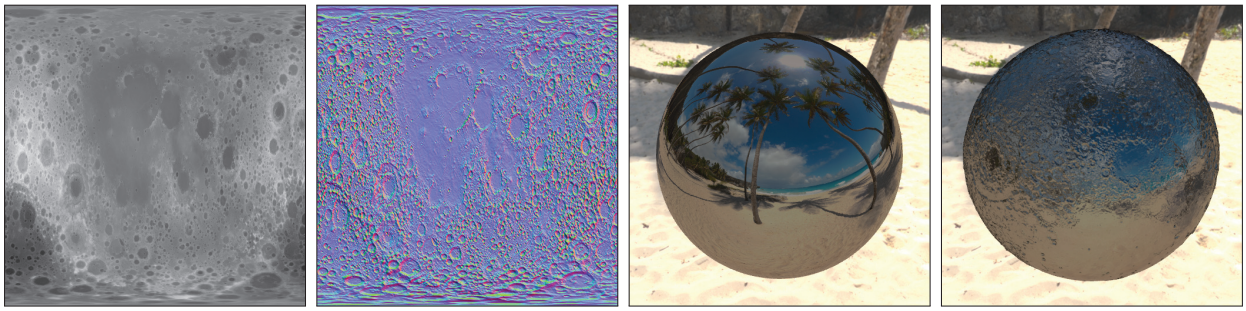
This chapter encompasses the main contributions of the LEADR mapping paper [DHI+13], which addresses the problem of accelerating displacement texture map filtering with MIP mapping [Wil83]. A displacement map [Coo84, CCC87] is a texture that displaces the surface it is mapped onto. In a physically based rendering context, the elements (texels) of such textures typically combine a scalar value and a normal. The former gives the magnitude of the displacement along the surface normal, while the latter provides the final normal of the surface, which should be used for shading. Although historically reserved to offline rendering contexts, displacement maps have progressively become ubiquitous in real-time rendering applications thanks to the availability of hardware accelerated subdivision surfaces [NLMD12, NL13]. As such, they are now a fundamental tool for rendering.

Similarly to, e.g., albedo texture maps, displacement maps must be filtered to match the resolution of the pixels they project into in order to guarantee consistent surface shading across scales [Hec89]. The simplest form of physically sound filtering is linear filtering, which consists in averaging the texels that contribute to the same pixel. But for this approach to work, the texels must interact linearly with shading [BN12]. Put more intuitively, the average shading of a texture mapped surface should be similar to the shading of the average texture mapped surface. Although this restriction holds most of the time for albedo texture maps (see, e.g., the work of Heitz et al. for counterexamples [HNPN13]), it does not for displacement maps. The reason why this is so is one of the main justifications for microfacet theory: the average BRDF of a displaced surface is not equal to the BRDF of its mean (hence smooth) surface.

The response of a proper displacement texture map filter should account for the BRDF of the surface, along with the view-dependent effects produced by the displacements. Since this is a complex task, we restrict for now the base surface BRDF to a Fresnel mirror, and postpone the treatment of the general case to Chapter 5. Such a restriction is useful here because the filtering configuration we are now considering corresponds to what is handled by microfacet BRDFs with only two notable differences, which we address in the remainder of this chapter. First, displacement mapped surfaces are typically defined in a frame where their mean surface differs from the tangent plane; in Section 3.2, we extend microfacet theory to support noncentral microfacet distributions that can account for such configurations. Second, because in a rendering context we expect to be able to visualize arbitrary portions of the displaced surfaces at multiple resolutions, we need an effective way to retrieve the NDF of the filtered texels; in Section 3.3, we leverage the linear representation introduced by Olano and Baker [OB10] to efficiently (though approximately) retrieve an anisotropic microfacet NDF at all

scales. The combination of these tools constitutes the LEADR mapping framework, which we discuss from a more practical point of view in Section 3.4.

3.2 Noncentral Microfacet BRDF Equations



Scalar texture map | Normal texture map | Base surface | Displaced surface

Figure 3.2 – Overview of the displacement mapping technique. Displacement mapping consists in displacing a surface along its geometric normal using a scalar texture map, and using a normal texture map to compute the final shading. Here, the scalar texture map is visualized as grey-scale values, while the normal texture map is color-coded.

The purpose of a displacement map is to add high-frequency details on top of a coarser surface. The perturbations produced by the displacements result in a new surface, whose shading can differ quite significantly from the base surface; Figure 3.2 shows such an example. In order to derive a microfacet BRDF for such a surface, we wish to account for configurations where the mean displaced surface differs from the plane tangent of the base surface, in which the displacement is defined. That way, we can rely on the same representation at any scale to evaluate shading. We can accomplish this by exploiting another invariance property of microfacet BRDFs, namely shear invariance: the microfacet BRDF of a sheared surface, whose mean surface normal $\mathbf{n} \in \Omega_+$ differs from the tangent up direction, can be retrieved from the components of the same unsheared surface, whose mean surface normal corresponds to the tangent up direction. Whenever the components of the mean surface slope $\tilde{\mathbf{n}} = (\tilde{x}_n, \tilde{y}_n) \in \mathbb{R}^2$ associated to the mean surface normal $\mathbf{n}$ differs from zero, we say that the microfacet BRDF associated to such a surface is noncentral, with shear parameters $\tilde{\mathbf{n}}$ (the mean slope can also be interpreted as the amount of shear that has been applied to the microsurface). We also introduce the notions of meso- and macrosurfaces to respectively denote the mean microsurface and the tangent plane.

Noncentral Microfacet NDF If $\tilde{\mathbf{h}}$ is a slope defined on the sheared microsurface, then it may be written as the offset slope of an unsheared microsurface $\tilde{\mathbf{h}} = \tilde{\mathbf{m}} + \tilde{\mathbf{n}}$, where $\tilde{\mathbf{n}} \in \mathbb{R}^2$ gives the shear applied to the microsurface (recall that shearing offsets the derivative). It follows that the NDF of any sheared surface may be retrieved from the NDF of the unsheared surface since $\tilde{\mathbf{m}} = \tilde{\mathbf{h}} - \tilde{\mathbf{n}}$ and hence

$$D(\mathbf{h}; \tilde{\mathbf{n}}) = P_{\text{std}}(\tilde{x}_h - \tilde{x}_n, \tilde{y}_h - \tilde{y}_n) \sec^4 \theta_h.$$

Note that shearing preserves the area of the surface projected onto the tangent plane, since for any $\tilde{\mathbf{n}}$ we have

$$\int_{\Omega_+} D(\mathbf{h}; \tilde{\mathbf{n}}) \cdot \cos\theta_h \cdot d\omega_h = 1. \tag{3.1}$$

Combining shear invariance with roughness invariance leads to the more general microfacet NDF expression

$$D(\mathbf{h}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) = P_{\text{std}}\left(\frac{\tilde{x}_h - \tilde{x}_n}{\alpha_x}, \frac{\alpha_x(\tilde{y}_h - \tilde{y}_n) - \rho\alpha_y(\tilde{x}_h - \tilde{x}_n)}{\alpha_x\alpha_y\sqrt{1-\rho^2}}\right) \frac{\sec^4\theta_h}{\alpha_x\alpha_y\sqrt{1-\rho^2}}, \tag{3.2}$$

which gives the microfacet NDF of a surface that has been arbitrarily sheared and scaled from a single microfacet slope PDF.

```
float_t microfacet::ndf(const vec3& h, const microfacet::params& params) const
{
    float_t cos_theta_h_sqr = h.z * h.z;
    float_t cos_theta_h_sqr_sqr = cos_theta_h_sqr * cos_theta_h_sqr;
    float_t xslope = -h.x / h.z - params.m_tx_n;
    float_t yslope = -h.y / h.z - params.m_ty_n;

    return (p22(xslope, yslope, params) / cos_theta_h_sqr_sqr);
}
```

Listing 3.1 – Evaluating the NDF term with arbitrary roughness and shear parameters in C++.

Noncentral Microfacet GAF Just like the microfacet NDF, the Smith masking term is also subject to shear invariance. We can prove this algebraically with a few manipulations provided in the parentheses on the right-

hand side for convenience

$$
\begin{aligned}
G_1(\mathbf{k};\tilde{\mathbf{n}}) &= \frac{z_k}{\int_{\Omega_+} D(\mathbf{h};\tilde{x}_n,\tilde{y}_n)\cdot\underline{\mathbf{kh}}\cdot d\omega_h} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_h-\tilde{x}_n,\tilde{y}_h-\tilde{y}_n)\sec\theta_h\cdot\underline{\mathbf{kh}}\cdot d\tilde{\mathbf{h}}} && \begin{pmatrix}\omega_h=\omega_h(\tilde{\mathbf{h}})\\ \Rightarrow d\omega_h=\cos^3\theta_h\cdot d\tilde{\mathbf{h}}\end{pmatrix} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_h-\tilde{x}_n,\tilde{y}_h-\tilde{y}_n)\cdot\underline{z_k-x_k\tilde{x}_h-y_k\tilde{y}_h}\cdot d\tilde{\mathbf{h}}} && \left(\sec\theta_h\cdot\mathbf{h}=\begin{bmatrix}-\tilde{x}_h\\-\tilde{y}_h\\1\end{bmatrix}\right) \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\cdot\underline{z_k-x_k\tilde{x}_h-y_k\tilde{y}_h}\cdot d\tilde{\mathbf{m}}} && \begin{pmatrix}\tilde{x}_h=\tilde{x}_m+\tilde{x}_n,\\ \tilde{y}_h=\tilde{y}_m+\tilde{y}_n\\ \Rightarrow d\tilde{\mathbf{h}}=d\tilde{\mathbf{m}}\end{pmatrix} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\cdot\underline{z_k-x_k\tilde{x}_n-y_k\tilde{y}_n-x_k\tilde{x}_m-y_k\tilde{y}_m}\cdot d\tilde{\mathbf{m}}} \\
&= \frac{z_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\cdot\underline{c-x_k\tilde{x}_m-y_k\tilde{y}_m}\cdot d\tilde{\mathbf{m}}} && \begin{pmatrix}c=z_k-x_k\tilde{x}_n-y_k\tilde{y}_n\\ =\underline{\mathbf{kn}}\cdot\sec\theta_n\end{pmatrix} \\
&= \frac{z_k}{c}\frac{c}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\cdot\underline{c-x_k\tilde{x}_m-y_k\tilde{y}_m}\cdot d\tilde{\mathbf{m}}} \\
&= \frac{z_k}{c}\frac{z'_k}{\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{x}_m,\tilde{y}_m)\sec\theta_m\cdot\underline{\mathbf{k}'\mathbf{m}}\cdot d\tilde{\mathbf{m}}} && \left(\mathbf{k}'=\frac{1}{\sqrt{x_k^2+y_k^2+c^2}}\begin{bmatrix}x_k\\y_k\\c\end{bmatrix}\right) \\
&= \frac{z_k}{c}\frac{z_{k'}}{\int_{\Omega_+} D(\mathbf{m};0,0)\cdot\underline{\mathbf{k}'\mathbf{m}}\cdot d\omega_m} && \begin{pmatrix}\tilde{\mathbf{m}}=\tilde{\mathbf{m}}(\omega_m)\\ \Rightarrow d\tilde{\mathbf{m}}=\sec^3\theta_m\cdot d\omega_m\end{pmatrix} \\
&= \frac{z_k}{c}G_1(\mathbf{k}';\mathbf{0}) \\
&= \frac{z_k}{c}G_{1,\text{std}}(\mathbf{k}'),
\end{aligned}
$$

which reveal that the G_1 term of any sheared microsurface can be determined from the unsheared microsurface. Combining shear invariance with roughness invariance leads to the more general microfacet Smith monostatic

shadowing expression

$$G_1(\mathbf{k};\alpha_1,\alpha_2,\phi_\alpha;\tilde{\mathbf{n}}) = G_{1,\text{std}}\left(\frac{1}{\sqrt{a^2+b^2+c^2}}\begin{bmatrix}a\\b\\c\end{bmatrix}\right)\frac{\cos\theta_k}{\underline{\mathbf{kn}}\cdot\sec\theta_n}, \tag{3.3}$$

where $a = \alpha_x x_k + \rho\alpha_y y_k$, $b = \alpha_y y_k\sqrt{1-\rho^2}$, and $c = z_k - x_k\tilde{x}_n - y_k\tilde{y}_n$. Since the G_1 term is shear invariant, it follows that the microfacet GAF term is also shear invariant. Note that both the $\vee$-groove and normal map GAFs are also shear invariant.

```
float_t
microfacet::g1_smith(
    const vec3& h, const vec3& k,
    const microfacet::params& params
) const {
    float_t a = k.x * params.m_ax + k.y * params.m_ay * params.m_rho;
    float_t b = k.y * params.m_ay * params.m_sqrt_one_minus_rho_sqr;
    float_t c = k.z - k.x * params.m_tx_n - k.y * params.m_ty_n;
    vec3 kprime = normalize(vec3(a, b, c));
    float_t g1 = g1_smith_std(h, kprime);

    if (g1 > 0.0)
        return g1 * (k.z / c);
    else
        return 0.0;
}
```

Listing 3.2 – Evaluating the Smith masking term with arbitrary roughness and shear parameters in C++.

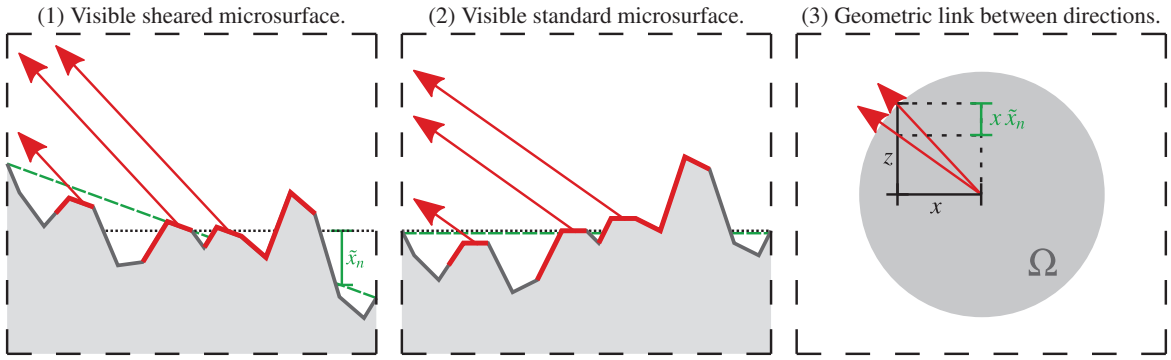


Figure 3.3 – Shear invariance of the monostatic shadowing term.

Importance Sampling Since the microfacet NDF and the microfacet GAF terms are shear invariant, it follows by construction (see Equation (2.15)) that the distribution of visible slopes is also shear invariant. Thus, we can also adapt the importance sampling scheme of a standard microsurface to that of an arbitrary sheared and scaled microsurface. Indeed, if $\tilde{\mathbf{m}}$ constitutes a sample of the untransformed visible slope PDF from direction

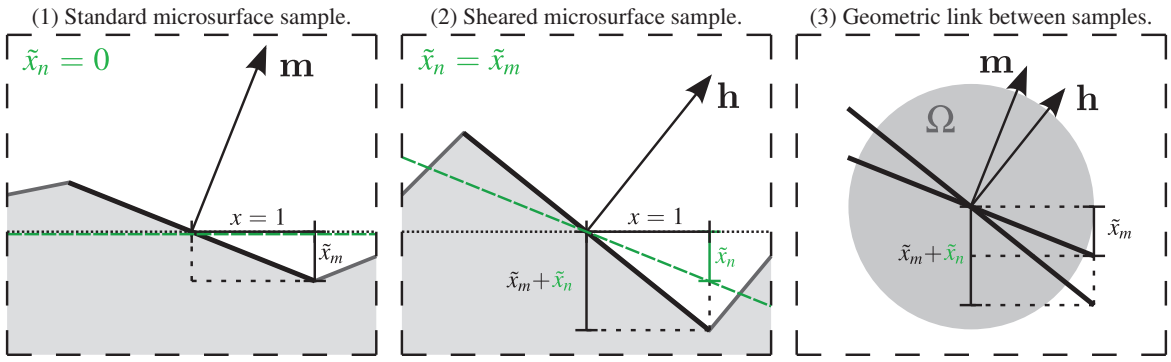


Figure 3.4 – Sampling a sheared microsurface from a standard microsurface.

$\mathbf{k}'$, then the slope

$$\tilde{\mathbf{h}} = \begin{bmatrix} \alpha_x \cdot \tilde{x}_m + \tilde{x}_n \\ \alpha_y(\rho \cdot \tilde{x}_m + \sqrt{1-\rho^2} \cdot \tilde{y}_m) + \tilde{y}_n \end{bmatrix} \tag{3.4}$$

constitutes a sample of the visible slope PDF from direction $\mathbf{k}$ of the sheared and scaled microsurface; this operation is illustrated in Figure 3.4.

```
vec3
microfacet::sample(
    float_t u1, float_t u2,
    const vec3& o,
    const void *user_param
) const {
    const microfacet::params params =
        user_param ? *reinterpret_cast<const microfacet::params *>(user_param)
                   : microfacet::params::standard();
    float_t tx_h, ty_h, tx_m, ty_m;

    // create a standard variate
    switch (m_gaf) {
        case GAF_NONE: sample_nmap(u1, u2, o, &tx_h, &ty_h); break;
        case GAF_VGROOVE: sample_vgroove(u1, u2, o, &tx_h, &ty_h); break;
        case GAF_SMITH: sample_smith(u1, u2, o, &tx_h, &ty_h); break;
        default: abort(); // should never happen
    }

    // warp the variate with the microfacet parameters
    tx_m = params.m_ax * tx_h + params.m_tx_n;
    float_t choleski = params.m_rho * tx_h
                     + params.m_sqrt_one_minus_rho_sqr * ty_h;
    ty_m = params.m_ay * choleski + params.m_ty_n;

    // return the associated normal
    return normalize(vec3(-tx_m, -ty_m, 1));
}
```

Listing 3.3 – Importance sampling a microfacet BRDF with arbitrary roughness and shear parameters in C++.

3.3 LEAN/LEADR Mapping

Because the constant increase in texture and screen resolutions has been counteracting the growth in computational power over the years, texture filtering remains an expensive operation. This is why prefiltered textures such as MIP mapped texture maps [Wil83, Hec89], which were introduced over thirty years ago, are still widely used and will probably remain popular for quite some time. MIP mapping consists in filtering an input texture at smaller resolutions prior to rendering and storing the output in texture maps of decreasing resolutions, called MIP maps. At render time, only the MIP texels that best match the on-screen resolution need to be evaluated, resulting in greater performance (due to the precomputations, and reduced input/output operations) and images with drastically reduced aliasing. Unfortunately, MIP mapping only works with linearly filterable textures, which means that in order to prefilter a displacement map, we first need to convert it into a linear representation. As a first step towards this direction, we can use the observation that the filtered response of a displaced Fresnel mirror can be approximated by the evaluation of a noncentral microfacet BRDF. Note that if the GAF can accurately predict the occlusion effects produced by the displacements, then the filtering operation is exact. In any case, this means that we can transpose the problem of representing a displacement map into that of representing a noncentral microfacet slope NDF, i.e., Equation (3.2), from which we can derive a microfacet BRDF; the remainder of this section is dedicated to this problem.

Beckmann Surfaces Finding a generic and linear representation for arbitrary noncentral microfacet NDFs is too difficult. Nevertheless, we can make significant progress if we assume that the microfacet NDF encompasses a Beckmann slope PDF at all scales, i.e.,

$$P_{\text{std}}(\tilde{\mathbf{h}}) \approx \frac{1}{\pi} \exp(-\tilde{x}_h^2 - \tilde{y}_h^2).$$

This assumption essentially restricts the behavior of the filtered surface to a specific distribution (Gaussian in this case), which simplifies the filtering problem to that of recovering roughness and shear parameters. For this purpose, Beckmann slope PDFs are extremely practical because their parameters can be retrieved from five

moments:

$$\mathscr{E}_1 := \int_{\mathbb{R}^2} \tilde{x}_h \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} = \tilde{x}_n \tag{3.5}$$

$$\mathscr{E}_2 := \int_{\mathbb{R}^2} \tilde{y}_h \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} = \tilde{y}_n \tag{3.6}$$

$$\mathscr{E}_3 := \int_{\mathbb{R}^2} \tilde{x}_h^2 \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} = \frac{\alpha_x^2}{2} + \tilde{x}_n^2 \tag{3.7}$$

$$\mathscr{E}_4 := \int_{\mathbb{R}^2} \tilde{y}_h^2 \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} = \frac{\alpha_y^2}{2} + \tilde{y}_n^2 \tag{3.8}$$

$$\mathscr{E}_5 := \int_{\mathbb{R}^2} \tilde{x}_h \cdot \tilde{y}_h \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} = \frac{\rho \alpha_x \alpha_y}{2} + \tilde{x}_n \tilde{y}_n. \tag{3.9}$$

Note that we provide the derivations that lead to these terms in the next paragraph. The reason why this is useful is that, by exploiting the summation form of these moments, we can retrieve the parameters from any set of $N > 0$ slope texels that project into a pixel

$$\mathscr{E}_1 = \frac{1}{N} \sum_{j=0}^{N-1} \tilde{x}_j \tag{3.10}$$

$$\mathscr{E}_2 = \frac{1}{N} \sum_{j=0}^{N-1} \tilde{y}_j \tag{3.11}$$

$$\mathscr{E}_3 = \frac{1}{N} \sum_{j=0}^{N-1} \tilde{x}_j^2 \tag{3.12}$$

$$\mathscr{E}_4 = \frac{1}{N} \sum_{j=0}^{N-1} \tilde{y}_j^2 \tag{3.13}$$

$$\mathscr{E}_5 = \frac{1}{N} \sum_{j=0}^{N-1} \tilde{x}_j \cdot \tilde{y}_j, \tag{3.14}$$

where $(\tilde{x}_j, \tilde{y}_j)$ denotes the j-th slope sample of the surface. Since the summation form of the moments is linear with respect to slopes, we can store slopes (rather than normals) in a texture map and precompute MIP maps texels as the average of their higher resolution counterparts. This linear representation was first introduced by Olano and Baker [OB10] under the name of LEAN mapping. It requires five floating-point numbers per MIP texel and is capable of capturing the roughness and main direction of anisotropy of a set of slopes, which is usually sufficient for most filtering cases.

```
// Linear Representation
class beckmann::lrep {
friend class beckmann;
public:
    // Ctor
    lrep(float_t E1 = 0.0, float_t E2 = 0.0,
         float_t E3 = 0.5, float_t E4 = 0.5,
         float_t E5 = 0.0);
private:
    // members
    float_t m_E1, m_E2; // first order slope moments
    float_t m_E3, m_E4; // second order slope moments
    float_t m_E5; // first order joint slope moment
};
```

Listing 3.4 – C++ implementation of a linear representation for Beckmann microfacet distributions.

```
void beckmann::params_to_lrep(const microfacet::params& params, lrep *lrep)
{
    float_t ax, ay, rho, tx_n, ty_n;
    params.get_pdfparams(&ax, &ay, &rho, &tx_n, &ty_n);
    (*lrep) = beckmann::lrep(tx_n, ty_n,
                             0.5 * ax * ax + tx_n * tx_n,
                             0.5 * ay * ay + ty_n * ty_n,
                             0.5 * rho * ax * ay + tx_n * ty_n);
}

void beckmann::lrep_to_params(const lrep& lrep, microfacet::params *params)
{
    float_t tx_n = lrep.m_E1;
    float_t ty_n = lrep.m_E2;
    float_t tmp1 = max((float_t)0.0, lrep.m_E3 - lrep.m_E1 * lrep.m_E1);
    float_t tmp2 = max((float_t)0.0, lrep.m_E4 - lrep.m_E2 * lrep.m_E2);
    /* clamp the parameters to valid values */
    float_t ax = max(1e-5, sqrt(2.0 * tmp1));
    float_t ay = max(1e-5, sqrt(2.0 * tmp2));
    float_t rho = 2.0 * (lrep.m_E5 - lrep.m_E1 * lrep.m_E2) / (ax * ay);
    rho = min((float_t)0.99, max((float_t)-0.99, rho));

    (*params) = microfacet::params::pdfparams(ax, ay, rho, tx_n, ty_n);
}
```

Listing 3.5 – C++ conversion routines for Beckmann microfacet distributions.

Beckmann Moments We can recover the scale and shear parameters of a Beckmann NDF from its first and second order slope moments. We prove here the results of Equations (3.5, 3.6, 3.7, 3.8, 3.9). First, we prove that the Beckmann slope PDF with expression

$$P_{\text{std}}(\tilde{\mathbf{h}}) = \frac{1}{\pi}\exp(-\tilde{x}_h^2 - \tilde{y}_h^2)$$

is properly normalized, as it integrates to one

$$\begin{aligned}
\int_{\mathbb{R}^2} P_{\text{std}}(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}} &= \frac{1}{\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \exp(-\tilde{x}_h^2 - \tilde{y}_h^2) \cdot d\tilde{x}_h \cdot d\tilde{y}_h \\
&= \frac{1}{\pi} \int_0^{2\pi} \int_0^{\infty} \exp(-r^2) \cdot r \cdot dr \cdot d\phi && \left(\begin{array}{c} \tilde{x}_h = r \cdot \cos\phi,\, \tilde{y}_h = r \cdot \sin\phi \\ \Rightarrow d\tilde{\mathbf{h}} = r \cdot dr \cdot d\phi \end{array} \right) \\
&= 2 \int_0^{\infty} \exp(-r^2) \cdot r \cdot dr && \left(\int_0^{2\pi} d\phi = 2\pi \right) \\
&= \int_0^{\infty} \exp(-x) \cdot dx && \left(\begin{array}{c} r = \sqrt{x} \\ \Rightarrow dr = \frac{1}{2\sqrt{x}} \cdot dx \end{array} \right) \\
&= 1. && (3.15)
\end{aligned}$$

Next, we prove that the first order moments of the standard PDF integrate to zero. We have

$$\begin{aligned}
\int_{\mathbb{R}^2} \tilde{x}_h \cdot P_{\text{std}}(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}} &= \frac{1}{\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{x}_h \cdot \exp(-\tilde{x}_h^2 - \tilde{y}_h^2) \cdot d\tilde{x}_h \cdot d\tilde{y}_h \\
&= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \tilde{x}_h \cdot \exp(-\tilde{x}_h^2) \cdot d\tilde{x}_h && \left(\frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \exp(-\tilde{y}_h^2) \cdot d\tilde{y}_h = 1 \right) \\
&= \frac{1}{\sqrt{\pi}} \int_0^{\infty} \tilde{x}_h \cdot \exp(-\tilde{x}_h^2) \cdot d\tilde{x}_h \\
&- \frac{1}{\sqrt{\pi}} \int_0^{\infty} \tilde{x}_h \cdot \exp(-\tilde{x}_h^2) \cdot d\tilde{x}_h \\
&= 0. && (3.16)
\end{aligned}$$

The same demonstration applies to the second first order moment, i.e., $\int_{\mathbb{R}^2} \tilde{y}_h \cdot P_{\text{std}}(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}} = 0$.

We also prove that the second order moments of the standard PDF integrate to $\frac{1}{2}$. We have

$$\begin{aligned}
\int_{\mathbb{R}^2} \tilde{x}_h^2 \cdot P_{\text{std}}(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}} &= \frac{1}{\pi} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \tilde{x}_h^2 \cdot \exp(-\tilde{x}_h^2 - \tilde{y}_h^2) \cdot d\tilde{x}_h \cdot d\tilde{y}_h \\
&= \frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \tilde{x}_h^2 \cdot \exp(-\tilde{x}_h^2) \cdot d\tilde{x}_h && \left(\frac{1}{\sqrt{\pi}} \int_{-\infty}^{\infty} \exp(-\tilde{y}_h^2) \cdot d\tilde{y}_h = 1 \right) \\
&= \frac{1}{2\sqrt{\pi}} \int_{-\infty}^{\infty} \tilde{x}_h \cdot f'(\tilde{x}_h) \cdot d\tilde{x}_h && \left(f'(\tilde{x}_h) = -2\tilde{x}_h \cdot \exp(-\tilde{x}_h^2) \right) \\
&= \frac{1}{2\sqrt{\pi}} \left(\underbrace{\tilde{x}_h \cdot f(\tilde{x}_h) \Big|_{-\infty}^{\infty}}_{=0} + \underbrace{\int_{-\infty}^{\infty} \exp(-\tilde{x}_h^2) \cdot d\tilde{x}_h}_{=\sqrt{\pi}} \right) && \text{(integration by parts)} \\
&= \frac{1}{2}. && (3.17)
\end{aligned}$$

The same demonstration applies to the second second order moment, i.e., $\int_{\mathbb{R}^2} \tilde{y}_h^2 \cdot P_{\text{std}}(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}} = \frac{1}{2}$.

Now we can proceed to prove that

$$\begin{aligned}
\mathscr{E}_1 &= \int_{\mathbb{R}^2} \tilde{x}_h \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} \\
&= \int_{\mathbb{R}^2} (\alpha_x \tilde{x}_m + \tilde{x}_n) \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} && \begin{pmatrix} \tilde{x}_h = \alpha_x \tilde{x}_m + \tilde{x}_n, \\ \tilde{y}_h = \alpha_y(\rho \tilde{x}_m + \sqrt{1-\rho^2}\tilde{y}_m) + \tilde{y}_n \\ \Rightarrow d\tilde{\mathbf{h}} = \alpha_x \alpha_y \sqrt{1-\rho^2} \cdot d\tilde{\mathbf{m}} \end{pmatrix} \\
&= \int_{\mathbb{R}^2} \tilde{x}_n \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} && \text{(Equation (3.16))} \\
&= \tilde{x}_n. && \text{(Equation (3.15))}
\end{aligned}$$

The proof for $\mathscr{E}_2 = \tilde{y}_n$ can be done similarly.

Next, we have

$$
\begin{aligned}
\mathscr{E}_3 &= \int_{\mathbb{R}^2} \tilde{x}_h^2 \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} \\
&= \int_{\mathbb{R}^2} (\alpha_x \tilde{x}_m + \tilde{x}_n)^2 \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} \qquad \begin{pmatrix} \tilde{x}_h = \alpha_x \tilde{x}_m + \tilde{x}_n, \\ \tilde{y}_h = \alpha_y(\rho \tilde{x}_m + \sqrt{1-\rho^2}\tilde{y}_m) + \tilde{y}_n \\ \Rightarrow d\tilde{\mathbf{h}} = \alpha_x \alpha_y \sqrt{1-\rho^2} \cdot d\tilde{\mathbf{m}} \end{pmatrix} \\
&= \int_{\mathbb{R}^2} (\alpha_x^2 \tilde{x}_m^2 + \tilde{x}_n^2 + \cancel{2\alpha_x \tilde{x}_n \tilde{x}_m}) \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} && \text{(Equation (3.16))} \\
&= \int_{\mathbb{R}^2} \alpha_x^2 \tilde{x}_m^2 \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} + \tilde{x}_n^2 && \text{(Equation (3.15))} \\
&= \frac{\alpha_x^2}{2} + \tilde{x}_n^2. && \text{(Equation (3.17))}
\end{aligned}
$$

The proof for $\mathscr{E}_4 = \frac{\alpha_y^2}{2} + \tilde{y}_n^2$ can be done similarly. Finally, we have

$$
\begin{aligned}
\mathscr{E}_5 &= \int_{\mathbb{R}^2} \tilde{x}_h \cdot \tilde{y}_h \cdot P(\tilde{\mathbf{h}}; \alpha_1, \alpha_2, \phi_\alpha; \tilde{\mathbf{n}}) \cdot d\tilde{\mathbf{h}} \\
&= \int_{\mathbb{R}^2} \tilde{x}_h \cdot \tilde{y}_h \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} \qquad \begin{pmatrix} \tilde{x}_h = \alpha_x \tilde{x}_m + \tilde{x}_n, \\ \tilde{y}_h = \alpha_y(\rho \tilde{x}_m + \sqrt{1-\rho^2}\tilde{y}_m) + \tilde{y}_n \\ \Rightarrow d\tilde{\mathbf{h}} = \alpha_x \alpha_y \sqrt{1-\rho^2} \cdot d\tilde{\mathbf{m}} \end{pmatrix} \\
&= \int_{\mathbb{R}^2} \alpha_x \alpha_y (\rho \tilde{x}_m^2 + \cancel{\sqrt{1-\rho^2} \tilde{x}_m \tilde{y}_m}) \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} && \text{(Equation (3.16))} \\
&+ \cancel{\int_{\mathbb{R}^2} \alpha_x \tilde{y}_n \tilde{x}_m \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}}} && \text{(Equation (3.16))} \\
&+ \cancel{\int_{\mathbb{R}^2} \tilde{x}_n \alpha_y (\rho \tilde{x}_m + \sqrt{1-\rho^2} \tilde{y}_m) \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}}} && \text{(Equation (3.16))} \\
&+ \int_{\mathbb{R}^2} \tilde{x}_n \tilde{y}_n \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} \\
&= \int_{\mathbb{R}^2} \alpha_x \alpha_y \rho \tilde{x}_m^2 \cdot P_{\text{std}}(\tilde{\mathbf{m}}) \cdot d\tilde{\mathbf{m}} + \tilde{x}_n \tilde{y}_n && \text{(Equation (3.15))} \\
&= \frac{\rho \alpha_x \alpha_y}{2} + \tilde{x}_n \tilde{y}_n. && \text{(Equation (3.17))}
\end{aligned}
$$

Experimental Validation Our filtering scheme predicts the appearance of Beckmann microgeometries. In order to validate this property, we can compare the appearance yielded by our multiscale microfacet BRDF against supersampled renderings of Gaussian microgeometries; Figure 3.5 shows the result of such an experiment based on displaced spheres under directional lighting, which clearly demonstrates a good match for various illumination directions and roughness configurations between our filtering scheme (computed with one

sample per pixel) and the reference images (computed using 256 samples per pixel). Note that more exhaustive tests are provided in the supplemental document of the original article [DHI+13].

3.4 Practical Considerations

Precomputations The first thing we need to do to use LEADR mapping is to extract a LEAN map from a displacement map. If the displacement map carries a normal map, then the conversion is straightforward [OB10]. Here, we show how to precompute LEAN maps from a displacement texture map that only consists of scalar displacement values. In this case, the texture can be interpreted as a discrete, scalar valued function $\xi : [0,1]^2 \to \mathbb{R}$, living in the unit parametric space $[0,1]^2$. To find the slopes produced by ξ at any location in $(u,v) \in [0,1]^2$, we need to compute the gradient $\nabla\xi \equiv [\partial\xi/\partial u, \partial\xi/\partial v]^t$. We can then express the slopes as

$$\tilde{x}(u,v) = \frac{\partial\xi(u,v)}{\partial u}, \quad \tilde{y}(u,v) = \frac{\partial\xi(u,v)}{\partial v}.$$

In order to compute these derivatives from texel space $[0,w] \times [0,h]$, where w and h respectively denote the width and height of the texture, the slopes must be scaled by the resolution of the texture

$$\tilde{x}(p,q) = w \cdot \frac{\partial\xi(p,q)}{\partial p}, \quad \tilde{y}(p,q) = h \cdot \frac{\partial\xi(p,q)}{\partial q}.$$

We can use this last equation to approximate the gradient with forward or backward differencing—depending on whether the texel coordinate is even or odd—just as GPUs compute screen-space derivatives. This yields

$$\tilde{x}(p,q) \approx w(\xi(p \pm 1, q) - \xi(p,q)), \quad \tilde{y}(p,q) \approx h(\xi(p, q \pm 1) - \xi(p,q)).$$

In practice, however, it is better to favor central differencing

$$\tilde{x}(p,q) \approx \frac{w}{2}(\xi(p+1,q) - \xi(p-1,q)), \quad \tilde{y}(p,q) \approx \frac{h}{2}(\xi(p,q+1) - \xi(p,q-1))$$

because it results in smoother shading; Figure 3.6 compares a rendering obtained with both approaches. For the sake of completeness, conversions of displacement maps into normal maps and normal maps into the first MIP of a LEAN/LEADR map are provided in Listings 3.6 and 3.7, respectively (the rest of the LEAN/LEADR MIP maps can be computed using a simple `GenerateMipmap()` routine).

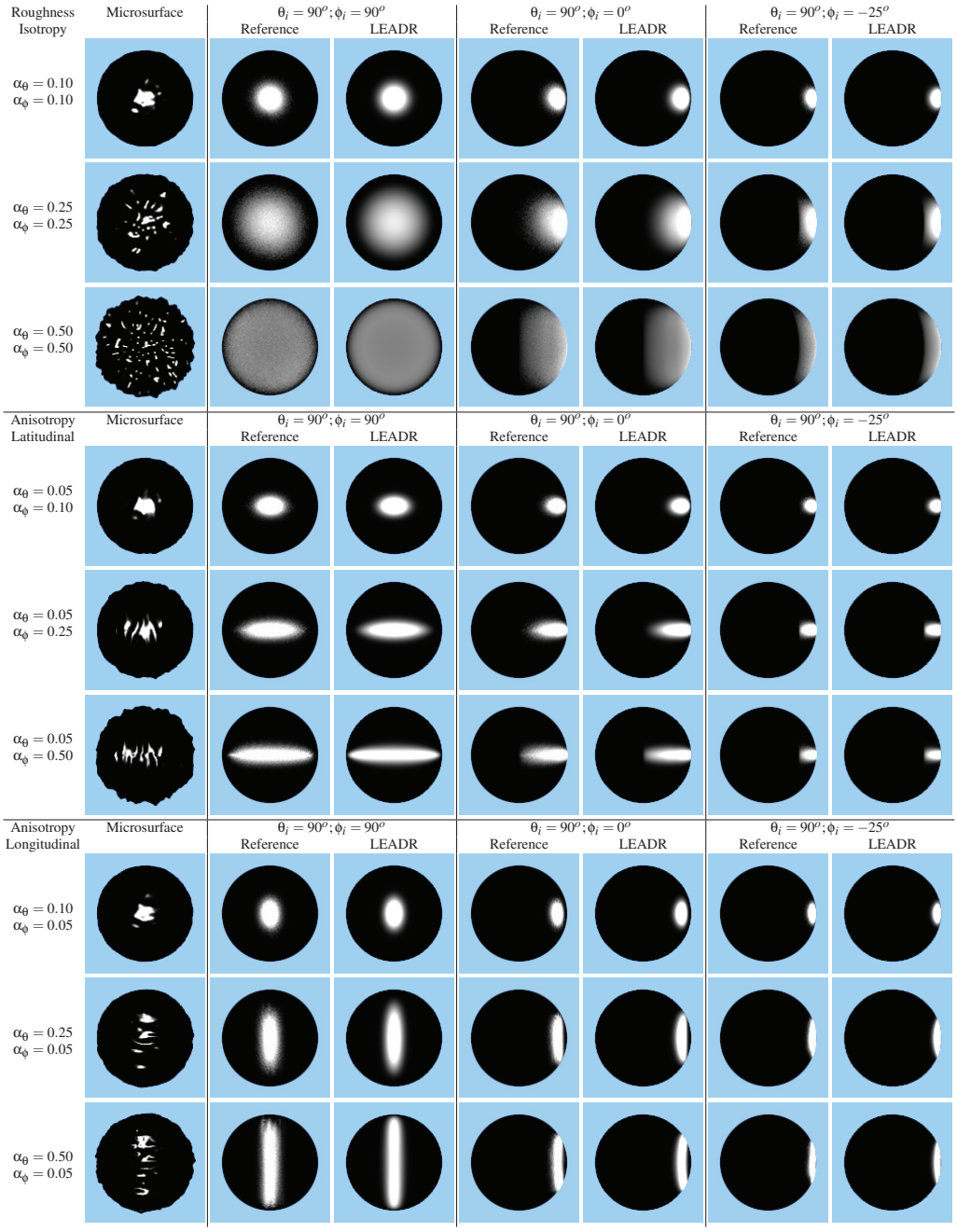
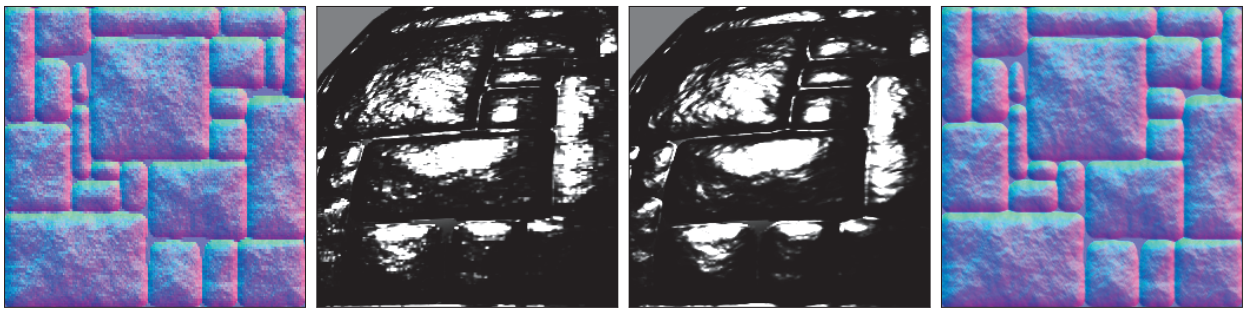


Figure 3.5 – Experimental validation of our filtering scheme. We compare the appearance of supersampled (256 samples per pixel) Beckmann microgeometries against the response of our multiscale microfacet BRDF for several roughnesses and lighting conditions. The *Microsurface* column is a preview of the Beckmann microgeometry.



Forward/backward differencing Central differencing

Figure 3.6 – Finite differencing scheme comparison for LEAN map precomputations. Central differencing results in smoother shading. The displacement map used as input comes from `http://www.gamedev.net/blog/33/entry-2198619-stone/`.

```
void
dmap2nmap(const CImg<uint8_t>& dmap, CImg<uint8_t>& nmap, float scale = 0.1f)
{
    int w = dmap.width();
    int h = dmap.height();

    nmap.resize(w, h, /*depth*/1, /*channels*/3);
    for (int i = 0; i < w; ++i)
    for (int j = 0; j < h; ++j) {
        uint8_t px_l = dmap.atXY(i - 1, j); // in [0,255]
        uint8_t px_r = dmap.atXY(i + 1, j); // in [0,255]
        uint8_t px_b = dmap.atXY(i, j + 1); // in [0,255]
        uint8_t px_t = dmap.atXY(i, j - 1); // in [0,255]
        float z_l = (float)px_l / 255.f; // in [0, 1]
        float z_r = (float)px_r / 255.f; // in [0, 1]
        float z_b = (float)px_b / 255.f; // in [0, 1]
        float z_t = (float)px_t / 255.f; // in [0, 1]
        float slope_x = (float)w * 0.5f * scale * (z_r - z_l);
        float slope_y = (float)h * 0.5f * scale * (z_t - z_b);
        float nrm_sqr = 1.f + slope_x * slope_x + slope_y * slope_y;
        float nrm_inv = 1.0 / sqrt(nrm_sqr);
        float nx = -slope_x * nrm_inv;
        float ny = -slope_y * nrm_inv;
        float nz = nrm_inv;
        float tmp1 = 0.5 * nx + 0.5; // in [0, 1]
        float tmp2 = 0.5 * ny + 0.5; // in [0, 1]

        nmap(i, j, 0, 0) = (uint8_t)(tmp1 * 255);
        nmap(i, j, 0, 1) = (uint8_t)(tmp2 * 255);
        nmap(i, j, 0, 2) = (uint8_t)(nz * 255);
    }
}
```

Listing 3.6 – C++ implementation of a routine to compute a normal map from a displacement map.

```
void
nmap2leanmap(
    const CImg<uint8_t>& nmap,
    CImg<float>& leanmap_1,
    CImg<float>& leanmap_2,
    float base_roughness = 1e-5
) {
    int w = nmap.width();
    int h = nmap.height();

    leanmap_1.resize(w, h, /*depth*/1, /*channels*/4);
    leanmap_2.resize(w, h, /*depth*/1, /*channels*/4);
    for (int i = 0; i < w; ++i)
    for (int j = 0; j < h; ++j) {
        uint8_t px_r = nmap.atXY(i, j, 0, 0); // in [0,255]
        uint8_t px_g = nmap.atXY(i, j, 0, 1); // in [0,255]
        uint8_t px_b = nmap.atXY(i, j, 0, 2); // in [0,255]
        float tmp1 = ((float)px_r / 255.f) * 2.0f - 1.0f; // in [-1, 1]
        float tmp2 = ((float)px_g / 255.f) * 2.0f - 1.0f; // in [-1, 1]
        float tmp3 = ((float)px_b / 255.f); // in [0, 1]
        float slope_x = -tmp1 / tmp3;
        float slope_y = -tmp2 / tmp3;
        float slope_x_sqr = slope_x * slope_x;
        float slope_y_sqr = slope_y * slope_y;
        float slope_xy = slope_x * slope_y;
        float base_roughness_sqr = 0.5f * base_roughness * base_roughness;

        leanmap_1(i, j, 0, 0) = slope_x; // E1
        leanmap_1(i, j, 0, 1) = slope_y; // E2
        leanmap_1(i, j, 0, 2) = 1.f;
        leanmap_1(i, j, 0, 3) = 1.f;
        leanmap_2(i, j, 0, 0) = slope_x_sqr + base_roughness_sqr; // E3
        leanmap_2(i, j, 0, 1) = slope_y_sqr + base_roughness_sqr; // E4
        leanmap_2(i, j, 0, 2) = slope_xy;    // E5
        leanmap_2(i, j, 0, 3) = 1.f;
    }
}
```

Listing 3.7 – C++ implementation of a routine to compute a LEAN/LEADR map from a normal map.

Surface Stretching and Shearing In practice, it is more convenient to store the moments of the unit parametric space slopes. We denote these moments as $\mathscr{E}'_1$, $\mathscr{E}'_2$, $\mathscr{E}'_3$, $\mathscr{E}'_4$, and $\mathscr{E}'_5$. The distribution must be expressed in the macrosurface tangent frame (x,y,z). Since meshes are rarely perfectly parameterized, the texture parameterization $u(x,y), v(x,y)$ usually produces distortions, which can be further accentuated in the case of animations.

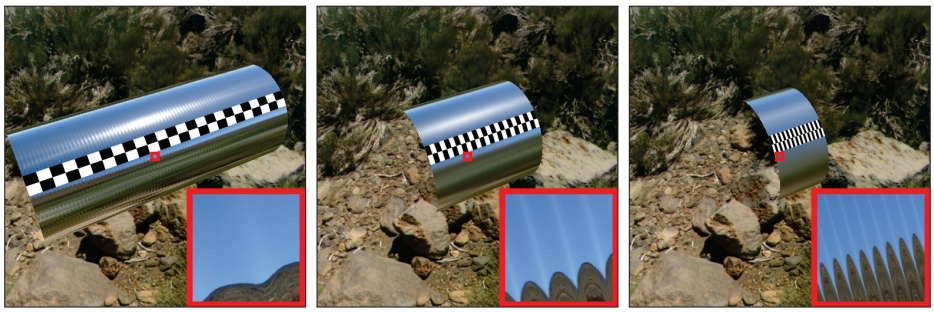


Figure 3.7 – Surface stretching.

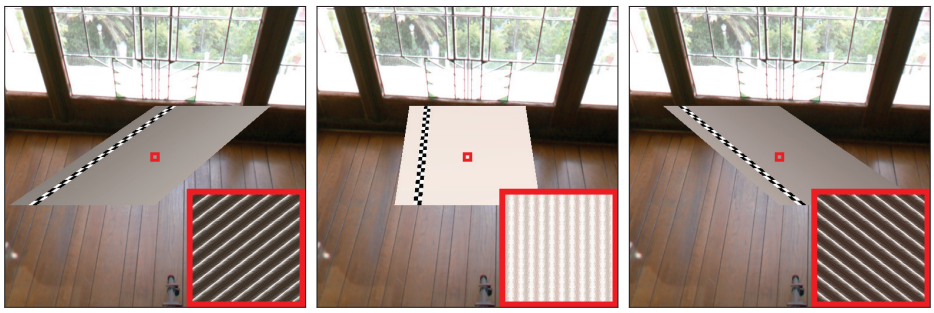


Figure 3.8 – Surface shearing.

The slopes in world space can be defined by

$$\left(\frac{\partial \xi}{\partial x}, \frac{\partial \xi}{\partial y}\right) = \left(\frac{\partial \xi}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial \xi}{\partial v}\frac{\partial v}{\partial x}, \frac{\partial \xi}{\partial u}\frac{\partial u}{\partial y} + \frac{\partial \xi}{\partial v}\frac{\partial v}{\partial y}\right).$$

If the distortions occurring on the macrosurface have low spatial variations, then we can reasonably assume that the mapping distortions represented by the terms $\frac{\partial u}{\partial x}$, $\frac{\partial v}{\partial x}$, $\frac{\partial u}{\partial y}$, and $\frac{\partial v}{\partial y}$ are locally constant. It follows that the appropriate slope distribution in world space gives

$$\begin{aligned}
\mathscr{E}_1 &= \frac{\partial u}{\partial x}\mathscr{E}'_1 + \frac{\partial v}{\partial x}\mathscr{E}'_2 \\
\mathscr{E}_2 &= \frac{\partial u}{\partial y}\mathscr{E}'_1 + \frac{\partial v}{\partial y}\mathscr{E}'_2 \\
\mathscr{E}_3 &= \left(\frac{\partial u}{\partial x}\right)^2 \mathscr{E}'_3 + \left(\frac{\partial v}{\partial x}\right)^2 \mathscr{E}'_4 + 2\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}\mathscr{E}'_5 \\
\mathscr{E}_4 &= \left(\frac{\partial u}{\partial y}\right)^2 \mathscr{E}'_3 + \left(\frac{\partial v}{\partial y}\right)^2 \mathscr{E}'_4 + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\mathscr{E}'_5 \\
\mathscr{E}_5 &= \frac{\partial u}{\partial x}\frac{\partial u}{\partial y}\mathscr{E}'_3 + \frac{\partial v}{\partial x}\frac{\partial v}{\partial y}\mathscr{E}'_4 + \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} + \frac{\partial v}{\partial x}\frac{\partial u}{\partial y}\right)\mathscr{E}'_5.
\end{aligned}$$

Computing the moments in this way allows support for animated geometry without having to update the LEADR MIP map hierarchy during animation: the distribution is adapted on the fly from the distortions, and adequately modulates the resulting appearance; this is illustrated in Figure 3.7 and Figure 3.8.

Height Scaling One may wish to apply a scaling factor $\beta \geq 0$ on displacements ξ to increase or reduce surface displacement. If β also exhibits low spatial variation across the surface, then it suffices to scale the first order moments $\mathscr{E}_1$ and $\mathscr{E}_2$ by β, and the second order moments $\mathscr{E}_3$, $\mathscr{E}_4$, and $\mathscr{E}_5$ by β^2. Note that this can be done

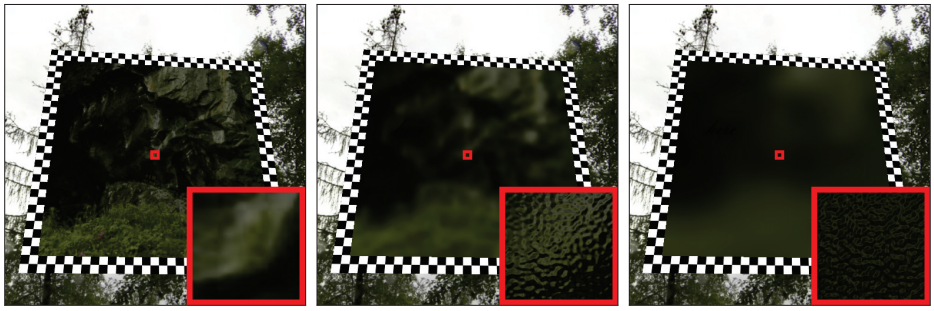


Figure 3.9 – Impact of scale on the displacement mapping technique.

at runtime without having to update the LEADR MIP map hierarchy. Figure 3.9 shows an example of scaled displacement map that uses the same LEAN map.

A Note on CLEAN Mapping Some applications may not be able to afford five floating-point numbers per texel for specular antialiasing purposes. For such scenarios, Baker introduces CLEAN maps [Bak11], which require only three floating-point numbers per texel at the cost of slightly less satisfying filtering quality (anisotropy is lost). When filtering with a CLEAN map, we are assuming that slope roughness is isotropic, i.e., $\alpha_x = \alpha_y$, and $\rho = 0$. The $\tilde{x}_n$ and $\tilde{y}_n$ parameters are determined similarly to a LEAN/LEADR map using the first slope moments in the x and y directions, which constitute the first two floating-point numbers of a CLEAN map. In order to retrieve the isotropic roughness parameter $\alpha = \alpha_x = \alpha_y$, Baker suggests to store and filter the squared magnitude of the second moments, i.e., $\tilde{x}_h^2 + \tilde{y}_h^2$. The roughness is then determined as follows

$$\alpha^2 = \underbrace{\int_{\mathbb{R}^2} (\tilde{x}_h^2 + \tilde{y}_h^2) \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}}_{\text{=CLEAN 3rd component}} - \tilde{x}_n^2 - \tilde{y}_n^2. \tag{3.18}$$

Depending on the textures however, this approach may result in aliasing because some directions may require higher roughness values. In order to guarantee antialiased highlights, the lobe of the slope distribution should be as large as the major radius of the ellipse produced by a LEAN/LEADR map, i.e., $\max(\alpha_1, \alpha_2)$. If we set the third component of the CLEAN map to yield the roughness value $\max(\alpha_1, \alpha_2)$, then the resulting highlights should always be large enough to avoid aliasing. Note that this is only a conjecture, since this approach has not been tested.

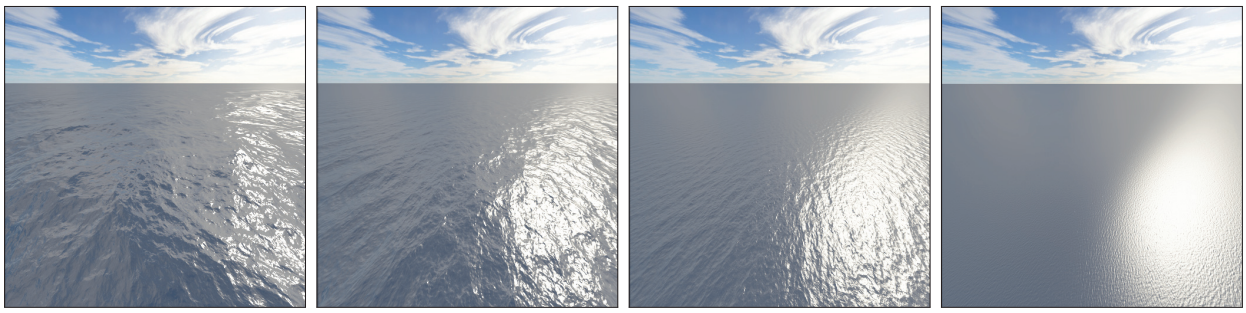


Figure 3.10 – A displacement mapped ocean rendered at different scales in real time with LEADR mapping.

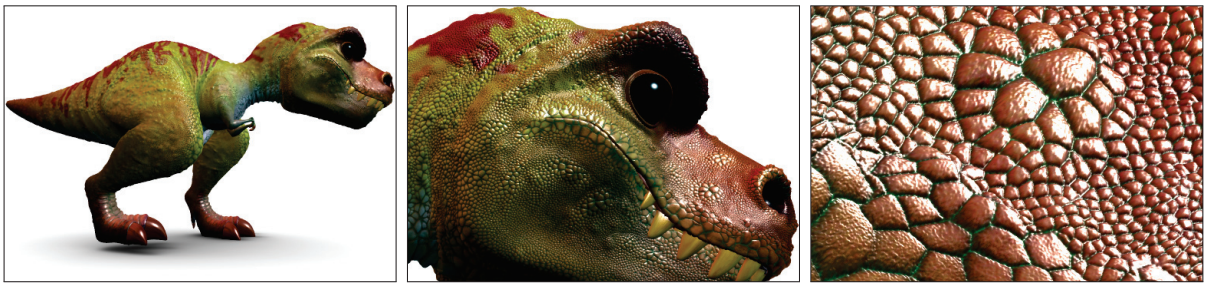


Figure 3.11 – A production asset (Ptex T-rex model © Walt Disney Animation Studios.) rendered in real time with LEADR mapping.

GPU Acceleration The LEADR mapping technique can be implemented on the GPU to produce high-quality renderings in real time with environment maps. Since a typical GPU rendering pipeline can only afford a limited number of shading evaluations per pixel, we can devise a filtered sampling approach that guarantees artifact-free images at the cost of slight bias, as follows. First, we recall from the previous chapter (see Equation (2.23)) that we can rewrite the rendering equation as

$$L_o = \frac{1}{N^2} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} L_i \cdot F(\theta_d) \cdot \frac{G(\mathbf{i},\mathbf{o})}{G_1(\mathbf{o})},$$

having

$$\mathbf{i} = 2(\mathbf{o}\cdot\mathbf{h})\mathbf{h} - \mathbf{o},$$

where $\mathbf{h}$ is the microfacet normal sample simulated with the uniform variates $u_1 = j/N$ and $u_2 = k/N$. By using prefiltered samples of L_i, we can emulate a smooth reconstruction filter that is guaranteed to produce antialiased results; this technique is also known as filtered importance sampling [CK07, KC08]. The problem is then to determine the size of the filter. A simple solution is to compute it according to the solid angle ω covered in the

incident direction $\mathbf{i}$

$$\omega \approx \cos\theta_o^3 \cdot A,$$

where $A \propto (\max(\alpha_1, \alpha_2)/N)^2$ is the surface area of a sample in slope space [DHI+13]. This guarantees that the solid angles overlap across neighboring samples and avoids banding artifacts for any value of $N \geq 1$. This technique was used to produce the images in, e.g., Figure 3.10 and Figure 3.11.

Chapter 4

Inverted Microfacet Theory

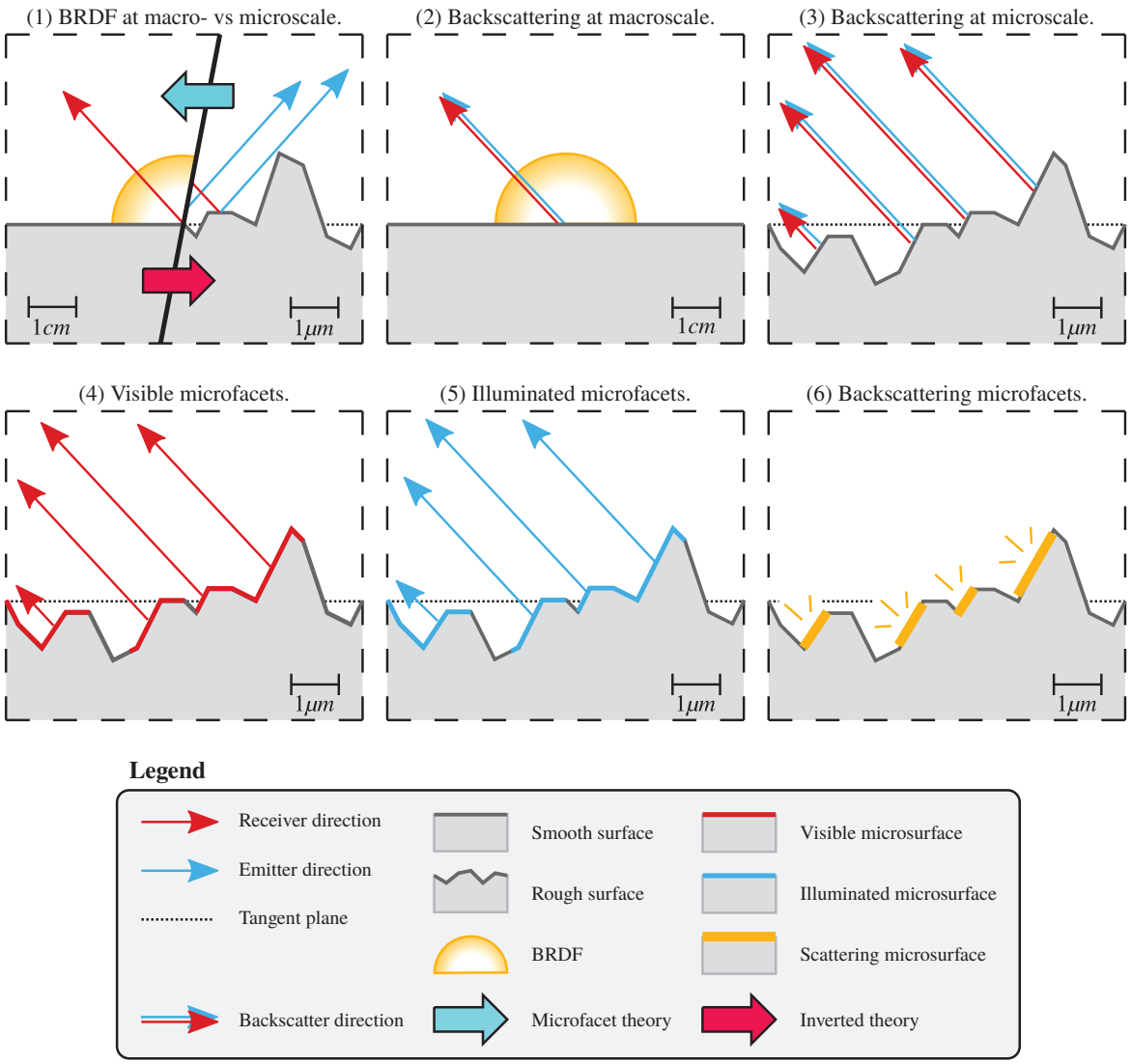


Figure 4.1 – Geometric overview of inverted microfacet theory. A microfacet BRDF extracts a BRDF from a microsurface. (1) Can we solve the inverse problem, i.e., is it possible to extract a microsurface from a BRDF? We can tackle this problem by considering the BRDF in (2, 3) backscattering configurations ($\mathbf{o} = \mathbf{i} = \mathbf{h}$). For such configurations, occlusions due to (4) masking and (5) shadowing become equal, and (6) the evaluation of the BRDF is directly proportional to the fraction (with respect to the entire microsurface) of microfacets that are visible from the backscattering direction. This observation is the key to solve the inverse problem.

4.1 Motivation

This chapter encompasses the main contributions of the paper entitled "Extracting Microfacet-based BRDF Parameters from Arbitrary Materials with Power Iterations" [DHI+15], which shows how to convert the reflectance of an arbitrary material into a microfacet BRDF. In order to alleviate the Fresnel mirror restriction that we imposed to displaced surfaces in the previous chapter, we will approximate the material of the undisplaced surface with a microfacet BRDF. But for this approach to be general (and hence practical), we need to design a procedure that can recover the components of a microfacet BRDF by solely evaluating the input material. Traditionally, this is done with a microfacet BRDF fitting procedure.

Existing microfacet-based fitting procedures may be classified in two categories. On the one hand, we find methods that typically introduce a new parametric microfacet normal distribution, and rely on sophisticated optimizations to fit its parameters [NDM05, LKYU12, BSH12]. Although this approach has been used successfully to fit measured materials, current methods suffer from two main problems. First, optimizations are not guaranteed to converge to a valid solution. Second, available parametric models are isotropic and single lobed, which limits the range of materials that they can fit accurately. On the other hand, we find methods that extract an arbitrary microfacet normal distribution into a table [AP07, WZT+08]. While this approach is considerably more general than the ones from the first category, as it naturally supports complex lobes and anisotropy, existing methods are not entirely satisfactory. Indeed, current methods either rely on crude approximations for visibility effects [AP07], or require a complex implementation involving sophisticated optimizations [WZT+08].

In the following sections, we introduce a new tabulation strategy that takes an arbitrary material as input, and extracts a microfacet normal distribution from backscattering configurations, and then a Fresnel function. Our approach combines the simplicity of the method of Ashikmnin and Premože [AP07], i.e., it is free from sophisticated optimizations, while retaining an accurate model for visibility effects. Specifically, we use the Smith microfacet visibility model, and also provide the derivations for the $\vee$-groove and normal map models, which we introduced in Chapter 2. In addition, we provide straightforward routines to convert our fits to Beckmann and GGX microfacet roughness parameters [WMLT07]. Finally, we discuss how implementations may take advantage of microfacet slope space to precompute importance sampling tables that support spatially varying roughness for our fitted materials.

4.2 Microfacet Terms Extraction

Backscattering Configurations In order to retrieve a microfacet slope distribution from an input material, we are going to use a simplified form of Equation (2.1). Specifically, we focus on backscattering configurations, which reduces the dimensionality of the BRDF from 4D to 2D, and simplifies the Fresnel term to a constant; Figure 4.1 (2, 3) shows the geometry of a backscattering configuration. Indeed, in such configurations, we have $\mathbf{i} = \mathbf{o} = \mathbf{h}$, as well as $\theta_d = 0$; for convenience, we write $F_0 = F(0)$. Equation (2.1) predicts that the BRDF takes the form

$$f_r(\mathbf{o},\mathbf{o}) = \frac{F_0 \cdot D(\mathbf{o}) \cdot G(\mathbf{o},\mathbf{o})}{4 \cdot \cos^2 \theta_o}. \tag{4.1}$$

However, this expression is not entirely accurate because the GAF term G overestimates occlusion effects. When $\mathbf{i} = \mathbf{o}$, shadowing and masking are fully correlated, and the GAF should degenerate into the monostatic form [Hei14]

$$G(\mathbf{o},\mathbf{o}) = G_1(\mathbf{o}). \tag{4.2}$$

The geometry of shadowing and masking in a backscattering configuration is illustrated in Figure 4.1 (4, 5). The accurate form of the GAF term leads to the microfacet backscattering equation

$$f_r(\mathbf{o},\mathbf{o}) = \frac{F_0 \cdot D(\mathbf{o}) \cdot G_1(\mathbf{o})}{4 \cdot \cos^2 \theta_o}. \tag{4.3}$$

With this equation at hand, we may now proceed to the extraction of a microfacet slope distribution P, given an input material f_r. Note that the inability of Equation (2.13) to degenerate into Equation (4.2) is known as the "hotspot" problem, and remains an open problem. In practice, we use Equation (4.2) for fitting purposes and Equation (2.13) for rendering. Our results thus overestimate occlusion effects in hotspot configurations by the factor $1/(2 - G_1) \in [0.5, 1]$.

Eigensystem Construction In our microfacet BRDF fitting problem, we are given an input material f_r and asked to extract a microfacet Fresnel term and a microfacet slope distribution. As a first step towards this direction, we swap the product $F_0 \cdot D$ and f_r to the other side of the equality in the microfacet backscattering equation, i.e., Equation (4.3). This yields

$$F_0 \cdot D(\mathbf{o}) = \frac{4 \cdot f_r(\mathbf{o},\mathbf{o}) \cdot \cos^2 \theta_o}{G_1(\mathbf{o})}. \tag{4.4}$$

Next, by replacing D and G_1 in Equation (4.4) by their definitions, i.e., Equation (2.11) and Equation (2.14) respectively, we get an equation for the microfacet slope PDF

$$F_0 \cdot P(\tilde{\mathbf{o}}) = \int_{\Omega_+} K(\mathbf{o},\mathbf{h}) \cdot P(\tilde{\mathbf{h}}) \cdot d\omega_h, \tag{4.5}$$

where

$$K(\mathbf{o},\mathbf{h}) = 4 \cdot f_r(\mathbf{o},\mathbf{o}) \cdot \cos^5\theta_o \cdot \underline{\mathbf{oh}} \cdot \sec^4\theta_h. \tag{4.6}$$

Equation (4.5) is not trivial to solve: in the mathematics literature, it belongs to the family of multivariate Fredholm equations of the second kind with kernel K [PM12]. In our case, the form of the kernel is not known in advance due to the input material f_r, so we solve this equation numerically by discretizing Equation (4.5) with a quadrature rule. Letting w_j denote the j-th quadrature rule weight and $i = 1, \cdots, N$, we obtain the new relation

$$F_0 \cdot P(\tilde{\mathbf{o}}_i) = \sum_{j=1}^{N} w_j \cdot K(\mathbf{o}_i,\mathbf{h}_j) \cdot P(\tilde{\mathbf{h}}_j), \tag{4.7}$$

where $\tilde{\mathbf{o}}_1 = \tilde{\mathbf{h}}_1, \cdots, \tilde{\mathbf{o}}_N = \tilde{\mathbf{h}}_N$ are the (slope) quadrature points, located in $\mathbb{R}^2$. Now, letting $\mathbf{p}$ denote the discretized PDF vector $\mathbf{p} = (P(\tilde{\mathbf{o}}_1), \cdots, P(\tilde{\mathbf{o}}_N))^t$, and $\mathbf{K}$ the matrix

$$\mathbf{K} = \begin{bmatrix} w_1 K(\mathbf{o}_1,\mathbf{h}_1) & \cdots & w_N K(\mathbf{o}_1,\mathbf{h}_N) \\ \vdots & \ddots & \vdots \\ w_1 K(\mathbf{o}_N,\mathbf{h}_1) & \cdots & w_N K(\mathbf{o}_N,\mathbf{h}_N) \end{bmatrix}, \tag{4.8}$$

Equation (4.7) rewrites as the eigenvalue problem

$$F_0 \cdot \mathbf{p} = \mathbf{K} \cdot \mathbf{p}. \tag{4.9}$$

This result shows that the problem of retrieving a microfacet slope PDF given an input material BRDF f_r translates into finding an eigenvector $\mathbf{p}$ whose components are all nonnegative.

Resolution via Power Iterations Since the entries of matrix $\mathbf{K}$ are all nonnegative, the Perron-Frobenius theorem [Per07, Fro12] states that it always has a unique eigenvector $\mathbf{p}$ whose values are all nonnegative. It is thus a valid solution for the microfacet slope distribution P. By solving Equation (4.9), we therefore have the **guaranteed** ability to compute a valid microfacet slope PDF from an input material BRDF. Furthermore, the theorem states that this eigenvector is associated to the largest eigenvalue of the matrix. As such, we can

compute it straightforwardly with the power iteration method.

The power iteration method is based on the property that the eigenvector associated to the largest eigenvalue of a matrix emerges after successive multiplications with a vector. In our implementation, we initialize $\mathbf{p} = (1, \cdots, 1)^t$, and successively multiply it by $\mathbf{K}$. In practice, we use only four successive multiplications, which has turned out to be sufficient for all our test cases. Once the vector has been determined, we build a continuous PDF by linearly interpolating the values of $\mathbf{p}$, and normalize it to satisfy Equation (2.12). We store the result in a table, which completes the extraction of P. Algorithm 1 provides pseudocode for our method.

Algorithm 1 Extract P

function EXTRACT_$P(f_r, N)$
  for each $i, j \in [1, N]$ **do** ▷ Build kernel matrix
    $K_{i,j} \leftarrow w_j \, 4 f_r(\mathbf{o}_i, \mathbf{o}_i) \cos^5 \theta_{o_i} \, \underline{\mathbf{o}_i \mathbf{h}_j} \sec^4 \theta_{h_j}$
  $\mathbf{p} \leftarrow (1, \cdots, 1)^t$
  for $0 \leq i < M$ **do** ▷ Power iterations (we set $M = 4$)
    $\mathbf{p} \leftarrow \mathbf{K} \cdot \mathbf{p}$
  $P \leftarrow \text{normalize}(\mathbf{p})$

With the extraction process of P complete, we can now evaluate the microfacet NDF D extracted from the input material thanks to Equation (2.11). It follows from Equation (2.14) that we can also compute the Smith term G_1. At this point, we can already create a fully functional microfacet BRDF whose microfacets act as "ideal" mirrors, i.e.,

$$f_{r,id}(\mathbf{i}, \mathbf{o}) = \frac{D(\mathbf{h}) \cdot G(\mathbf{i}, \mathbf{o})}{4 \cdot \cos \theta_i \cdot \cos \theta_o}. \tag{4.10}$$

Note that this equation is a special case of Equation (2.1), where $F(\theta_d) = 1$ for any θ_d. We can thus turn to the problem of retrieving the microfacet Fresnel term in order to complete the fitting process.

Fresnel Extraction In order to extract the Fresnel term, we compute for each color channel the average ratio between the input material BRDF and Equation (4.10) over all possible θ_d configurations, i.e.,

$$F(\theta_d) = \mathbb{E}\left[\frac{f_r(\mathbf{i}, \mathbf{o})}{f_{r,id}(\mathbf{i}, \mathbf{o})} \,|\, \mathbf{ih} = \cos \theta_d \right], \tag{4.11}$$

where we use the notation $\mathbb{E}[x|y]$ to denote the expectation of variable x over the domain that satisfies condition y. Algorithm 2 provides pseudocode for our method.

The main advantage of our approach is that it is fully automatic. It is very accurate when the behavior of

Algorithm 2 Extract F

function EXTRACT_$F(f_r, f_{r,id})$
  for $\theta_d \in [0, \pi/2]$ **do**
    $F(\theta_d) \leftarrow 0$
    $N \leftarrow 0$
    for $\phi_d, \phi_h \in [0, 2\pi], \theta_h \in [0, \pi/2]$ **do**
      $\mathbf{i} \leftarrow$ from_half_diff$(\mathbf{h}, \mathbf{d})$
      $\mathbf{o} \leftarrow$ reflect$(\mathbf{i}, \mathbf{h})$
      $F(\theta_d) \leftarrow F(\theta_d) + f_r(\mathbf{i}, \mathbf{o}) / f_{r,id}(\mathbf{i}, \mathbf{o})$
      $N \leftarrow N + 1$
    $F(\theta_d) \leftarrow F(\theta_d)/N$

the input material accords to microfacet theory, i.e., when f_r is roughly equal to Equation (2.1). When this condition does not hold, our algorithm will propagate the fitting errors into the Fresnel term. Such situations will arise for, e.g., materials with directionally dependent albedos, such as certain fabrics or car paints. In such cases, the extracted behavior of F differs significantly from what is predicted by the Fresnel law, and the term should be regarded as a residual function instead of an actual Fresnel function.

Optimization: Eigensystem Construction for Isotropic Materials Although our fitting process is already complete, we introduce here an optimization for the extraction of P that works for isotropic materials. If the input material is isotropic, then the microfacet NDF D depends only on the elevation angle θ_h. It follows from Equation (2.11) that the microfacet slope PDF is also isotropic, which implies that it may be expressed as a 1D radial function

$$P(\tilde{\mathbf{h}}) = g(\theta_h). \tag{4.12}$$

In such cases, the problem of retrieving the 2D function P, i.e., Equation (4.5), simplifies to that of finding the 1D function g. This problem also translates into a (univariate) Fredholm equation of the second kind

$$F_0 \cdot g(\theta_o) = \int_0^{\pi/2} K'(\theta_o, \theta_h) \cdot g(\theta_h) \cdot d\theta_h, \tag{4.13}$$

with kernel K'

$$K'(\theta_o, \theta_h) = \int_0^{2\pi} K(\mathbf{o}, \mathbf{h}) \cdot \sin\theta_h \cdot d\phi_h. \tag{4.14}$$

To arrive at this particular result, we start from Equation (4.5) and apply Equation (4.12)

$$F_0 \cdot g(\theta_o) = \int_{\Omega_+} K(\mathbf{o},\mathbf{h}) \cdot g(\theta_h) \cdot d\omega_h.$$

We then proceed with a few simple algebric manipulations, and Equation (4.13) naturally emerges

$$\begin{aligned} F_0 \cdot g(\theta_o) &= \int_0^{2\pi} \int_0^{\pi/2} K(\mathbf{o},\mathbf{h}) \cdot g(\theta_h) \cdot \sin\theta_h \cdot d\theta_h \cdot d\phi_h \\ &= \int_0^{\pi/2} \left[\int_0^{2\pi} K(\mathbf{o},\mathbf{h}) \cdot \sin\theta_h \cdot d\phi_h \right] \cdot g(\theta_h) \cdot d\theta_h \\ &= \int_0^{\pi/2} K'(\theta_o,\theta_h) \cdot g(\theta_h) \cdot d\theta_h. \end{aligned}$$

Note that the choice of the azimuthal angle ϕ_o to fully define $\mathbf{o}$ is arbitrary in Equation (4.14). As for the general case, Equation (4.13) may be expressed as an eigenvalue problem of the form

$$F_0 \cdot \mathbf{p}' = \mathbf{K}' \cdot \mathbf{p}', \tag{4.15}$$

where $\mathbf{p}' = (g(\theta_{o_1}), \cdots, g(\theta_{o_N}))^t$. Since the entries of matrix $\mathbf{K}'$ are all nonnegative, we can also solve Equation (4.15) with the power iteration method. In practice, we also use four successive multiplications to recover the solution $\mathbf{p}'$. Algorithm 3 provides pseudocode for this specialized method.

Algorithm 3 Extract Isotropic P

function EXTRACT_P_ISOTROPIC(f_r, N)
  $\phi_o \leftarrow 0$ ▷ The choice is arbitrary here
  for each $i, j \in [1, N]$ **do** ▷ Build kernel matrix
    $K'_{i,j} \leftarrow \int_0^{2\pi} w_j \cdot K(\mathbf{o}_i, \mathbf{h}_j) \cdot \sin\theta_{h_j} \cdot d\phi_h$
  $\mathbf{p}' \leftarrow (1, \cdots, 1)^t$
  for $0 \leq i < M$ **do** ▷ Power iterations (we set $M = 4$)
    $\mathbf{p}' \leftarrow \mathbf{K}' \cdot \mathbf{p}'$
  $P \leftarrow$ normalize($\mathbf{p}'$)

```
void tabular::compute_p22_smith(const brdf& brdf, int res)
{
    int cnt = res - 1;
    float_t dtheta = sqrt(M_PI * 0.5) / (float_t)cnt;
    matrix km(cnt);

    for (int i = 0; i < cnt; ++i) {
        float_t tmp = (float_t)i / (float_t)cnt;
        float_t theta = tmp * sqrt(M_PI * 0.5);
        float_t theta_o = theta * theta;
        float_t cos_theta_o = cos(theta_o);
        float_t tan_theta_o = tan(theta_o);
        vec3 fr = brdf.eval(vec3(theta_o, 0.0), vec3(theta_o, 0.0));
        float_t fr_i = fr.intensity();
        float_t kji_tmp = (dtheta * pow(cos_theta_o, 6.0)) * (8.0 * fr_i);

        for (int j = 0; j < cnt; ++j) {
            const float_t dphi_h = M_PI / 180.0;
            float_t tmp = (float_t)j / (float_t)cnt;
            float_t theta = tmp * sqrt(M_PI * 0.5);
            float_t theta_h = theta * theta;
            float_t cos_theta_h = cos(theta_h);
            float_t tan_theta_h = tan(theta_h);
            float_t tan_product = tan_theta_h * tan_theta_o;
            float_t nint = 0.0;

            for (float_t phi_h = 0.0; phi_h < 2.0 * M_PI; phi_h+= dphi_h)
                nint+= max(1.0, tan_product * cos(phi_h));
            nint*= dphi_h;

            km(j, i) = theta * kji_tmp * nint * tan_theta_h
                     / (cos_theta_h * cos_theta_h);
        }
    }

    // compute slope pdf
    const std::vector<double> v = km.eigenvector(4);
    for (int i = 0; i < (int)v.size(); ++i)
        m_p22.push_back(1e-2 * v[i]);
    m_p22.push_back(0);
}
```

Listing 4.1 – Extracting a microfacet slope PDF assuming a Smith GAF in C++.

Inversion with the $\vee$-Groove GAF We give here the procedure to extract a microfacet slope PDF assuming a $\vee$-groove GAF. In this case, the inversion procedure is much simpler than for the Smith GAF and only requires a few algebric manipulations

$$P(\tilde{\mathbf{h}}) = \frac{4 \cdot \cos^6 \theta_h \cdot f_r(\mathbf{h},\mathbf{h})}{F_0 \cdot \min\left(1, 2 \cdot \cos^2 \theta_h\right)}, \tag{4.16}$$

which may also be rewritten in the numerically stable form

$$P(\tilde{\mathbf{h}}) = \frac{4}{F_0} \cdot \cos^4 \theta_h \cdot f_r(\mathbf{h},\mathbf{h}) \cdot \max\left(\cos^2 \theta_h, \frac{1}{2}\right). \tag{4.17}$$

```
void tabular::compute_p22_vgroove(const brdf& brdf, int res)
{
    int cnt = res - 1;

    for (int i = 0; i < cnt; ++i) {
        float_t u = (float_t)i / (float_t)cnt; // in [0, 1)
        float_t theta_h = u * u * M_PI * 0.5; // in [0, pi/2)
        float_t cos_theta_h = cos(theta_h);
        float_t cos_theta_h_sqr = cos_theta_h * cos_theta_h;
        float_t gaf = max((float_t)0.5, cos_theta_h_sqr);
        vec3 fr = brdf.eval(vec3(theta_h, 0), vec3(theta_h, 0));
        float_t fr_i = fr.intensity();
        float_t pdf = gaf * (cos_theta_h_sqr * cos_theta_h_sqr) * (4.0 * fr_i);

        m_p22.push_back(pdf);
    }
    m_p22.push_back(0);
}
```

Listing 4.2 – Extracting a microfacet slope PDF assuming a $\vee$ groove GAF in C++.

Inversion with the Normal Map GAF We give here the procedure to extract a microfacet slope PDF assuming a normal map GAF. In this case, the inversion procedure is even simpler than for the $\vee$-groove GAF, and only requires a few algebric manipulations

$$P(\tilde{\mathbf{h}}) = \frac{4}{F_0} \cdot \cos^4 \theta_h \cdot f_r(\mathbf{h}, \mathbf{h}). \tag{4.18}$$

```
void tabular::compute_p22_nmap(const brdf& brdf, int res)
{
    int cnt = res - 1;

    for (int i = 0; i < cnt; ++i) {
        float_t u = (float_t)i / (float_t)cnt; // in [0, 1)
        float_t theta_h = u * u * M_PI * 0.5; // in [0, pi/2)
        float_t cos_theta_h = cos(theta_h);
        float_t cos_theta_h_sqr = cos_theta_h * cos_theta_h;
        vec3 fr = brdf.eval(vec3(theta_h, 0), vec3(theta_h, 0));
        float_t fr_i = fr.intensity();
        float_t pdf = (cos_theta_h_sqr * cos_theta_h_sqr) * (4.0 * fr_i);

        m_p22.push_back(pdf);
    }
    m_p22.push_back(0);
}
```

Listing 4.3 – Extracting a microfacet slope PDF assuming a normal map GAF in C++.

Conversion to Beckmann Roughness We retrieve the parameters by computing 2nd order moments like in LEAN/LEADR mapping [DHI+13]

$$\alpha_x^2 = 2\int_{\mathbb{R}^2} \tilde{x}_h^2 \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}$$
$$\alpha_y^2 = 2\int_{\mathbb{R}^2} \tilde{y}_h^2 \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}$$
$$\rho\alpha_x\alpha_y = 2\int_{\mathbb{R}^2} \tilde{x}_h \cdot \tilde{y}_h \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}.$$

Note that if P is a Beckmann slope PDF, then our conversion is exact.

```
microfacet::params tabular::fit_beckmann_parameters(const tabular& tab)
{
    const int ntheta = 128;
    float_t dtheta = M_PI / (float_t)ntheta;
    float_t nint = 0.0;
    float_t alpha;

    for (int i = 0; i < ntheta; ++i) {
        float_t u = (float_t)i / (float_t)ntheta; // in [0,1)
        float_t theta_h = u * u * M_PI * 0.5;
        float_t cos_theta_h = cos(theta_h);
        float_t r_h = tan(theta_h);
        float_t r_h_sqr = r_h * r_h;
        float_t p22_r = tab.p22_radial(r_h_sqr);

        nint+= (u * r_h_sqr * r_h * p22_r) / (cos_theta_h * cos_theta_h);
    }
    nint*= dtheta * M_PI; /* M_PI = int_0^2pi cos^2 phi dphi */
    alpha = sqrt(2.0 * nint);

    return microfacet::params::isotropic(alpha);
}
```

Listing 4.4 – Converting a radial tabulated slope PDF into a Beckmann roughness parameter in C++.

Conversion to GGX Roughness Note that the 2nd order moments diverge with GGX. We propose an alternative estimation to retrieve the parameters of the scale matrix

$$\alpha_x = \int_{\mathbb{R}^2} |\tilde{x}_h| \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}$$
$$\alpha_y = \int_{\mathbb{R}^2} |\tilde{y}_h| \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}$$
$$\beta_1 = \int_{\mathbb{R}^2} \frac{\tilde{x}_h \tilde{y}_h}{\tilde{x}_h^2 + \tilde{y}_h^2} \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}$$
$$\beta_2 = \int_{\mathbb{R}^2} \frac{\tilde{y}_h^2}{\tilde{x}_h^2 + \tilde{y}_h^2} \cdot P(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}$$
$$\rho = \frac{\alpha_y}{\alpha_x} \frac{\beta_1}{\beta_1^2 + \beta_2^2}.$$

Note that if P is a GGX slope PDF, then our conversion is exact.

```
microfacet::params tabular::fit_ggx_parameters(const tabular& tab)
{
    const int ntheta = 128;
    float_t dtheta = M_PI / (float_t)ntheta;
    float_t nint = 0.0;
    float_t alpha;

    for (int i = 0; i < ntheta; ++i) {
        float_t u = (float_t)i / (float_t)ntheta; // in [0,1)
        float_t theta_h = u * u * M_PI * 0.5;
        float_t cos_theta_h = cos(theta_h);
        float_t r_h = tan(theta_h);
        float_t r_h_sqr = r_h * r_h;
        float_t p22_r = tab.p22_radial(r_h_sqr);

        nint+= (u * r_h_sqr * p22_r) / (cos_theta_h * cos_theta_h);
    }
    nint*= dtheta * 4.0; /* 4.0 = int_0^2pi fabs(cos(phi)) dphi */
    alpha = nint;

    return microfacet::params::isotropic(alpha);
}
```

Listing 4.5 – Converting a radial tabulated slope PDF into a GGX roughness parameter in C++.

4.3 Experiments

A Note on Precomputations In our implementation, we rely exclusively on rectangle quadrature rules to compute our integrals. This includes the microfacet slope PDF, i.e., $w_1 = \cdots = w_N = 1/N$. We store one RGB 1D table for the microfacet Fresnel, and three scalar-valued 1D (resp. 2D) tables for each of P, G_1, and the quantile function for isotropic (resp. anisotropic) materials. The dimensions of the tables depend on the number of samples that we evaluate from the material in the elevation and azimuthal directions. In the isotropic configuration, the number of samples is equal to N. In this case, our representation stores (including Fresnel and the other tables) $3N + 3N = 6N$ scalar values. In the anisotropic configuration, the number of samples is equal to $N = N_\theta \times N_\phi$, where N_θ and N_ϕ respectively denote the number of elevation and azimuthal samples used to evaluate the reflectance of the input material. In this case, our representation stores (including Fresnel and the other tables) $3N_\theta \times (1 + N_\phi)$ scalar values.

Unit Testing We can validate our microfacet slope PDF extraction by testing our algorithms against an analytic model of Equation (4.3) based on a Beckmann distribution [WMLT07] with varying roughness. Table 4.I shows the maximum relative errors that we measured during our experiment in the isotropic case, using $N = 360$ BRDF evaluations. As can be seen from the reported numbers, the error is small (less than 3%). We attribute

this error to the quadrature rules that we use to solve integral Equations (4.5) and (4.13), and consider it as the minimum error produced by our algorithms. The anisotropic case produces the same amount of relative error.

α	0.01	0.02	0.05	0.15
δ_{max}	0.03	0.004	0.002	0.0005

Table 4.I – Maximum relative error in backscattering between Beckmann BRDFs with varying roughness and their respective fits computed with our algorithm.

Fitting Isotropic Materials We proceeded to an extensive fitting comparison against the state-of-the-art parametric model, referred hereafter as SGD, of Bagher et al. [BSH12] using the MERL material database of Matusik et al. [MPBM03]. To compute our fits, we initialized our algorithm with $N = 90$ and made sure that the entire backscattering data was sampled only once. At 32-bit floating-point precision, each of our fits requires 2.1 KB of memory. The exhaustive tests are provided in the supplemental document of the original article [DHI+15], which also includes more detailed numerical analyses as well as delta-E difference images. Figure 4.3 shows some comparative renderings of both methods as well as with the ABC microfacet model [LKYU12] against the reference for a few materials, using 512 samples per pixel. Note that for the *two-layer-gold* and *changing-paint1* materials, the SGD fitting optimization failed and resulted in flawed images. This example emphasizes one of the strengths of our fitting method over optimization techniques, since, as we showed in the previous section, our fits can not result in such failures. Note that these materials were not the only ones affected by this issue in the database. In general, we believe our method is qualitatively superior to SGD and on par with ABC for metallic materials. Differences with SGD and ABC are most visible in Figure 4.3 at grazing angles. For most other materials in the supplemental document of the original article [DHI+15], our fits are either on par or slightly below the SGD model. While our observation is mainly qualitative, it is also in agreement with the average delta-E difference image of our supplemental document, which is illustrated in Figure 4.2.

Worse Isotropic Fitting For certain isotropic materials of the MERL database, we noticed that our method could produce fits that were qualitatively less satisfying than previous work. Figure 4.3 carries a few such materials, which can also be found in our supplemental document. In this particular figure, our fit of the *alumina-oxide* material is worse than those of the ABC and SGD models. We see two reasons why our method would produce less satisfying fits. The first reason is due to the input material itself: since our method extracts the microfacet NDF from backscattering data exclusively, it is highly sensitive to the quality of such configurations.

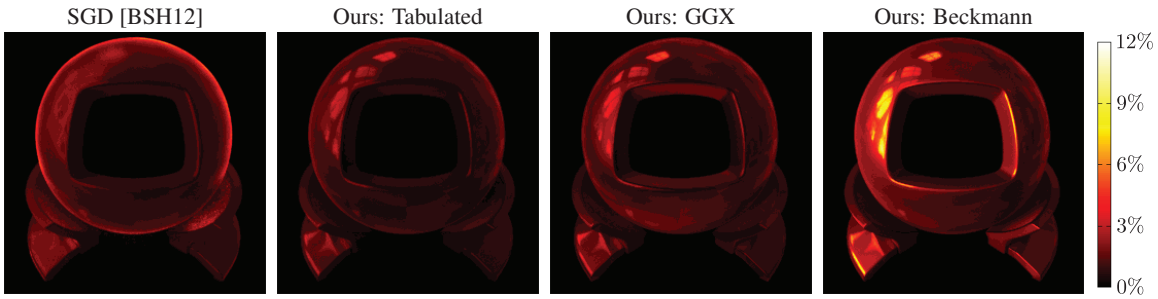


Figure 4.2 – Mean delta-E difference image on the entire MERL database [MPBM03].

Thus, if backscattering is poorly acquired, then our method will fail at reproducing the input BRDF faithfully. The second reason is due to our BRDF model: our model is based on a microfacet BRDF model alone and, as such, is limited to material behaviors that are close to what is predicted by the equations. Layered and/or composite materials (e.g., *alumina-oxide* in Figure 4.3) as well as strong Lambertian component (e.g., some paints and acrylics) tend to be qualitatively less satisfying than the SGD fits in our supplemental document. For the specific case of isotropic materials with poor backscattering data and/or strong Lambertian components, fitting methods based on optimizations [NDM05, LKYU12, BSH12, WZT$^+$08] should also perform better than our method in general. As for layered and color-changing materials (e.g., *changing-paint1* in Figure 4.3), they remain a challenging open problem.

Fitting Anisotropic Materials Our method also supports anisotropic materials. To review its performance, we tested some highly anisotropic materials from the recent database of Filip and Vavra [FV14]. To compute our fits, we initialized our algorithm with $\theta_N = 90$ and $\phi_N = 90$. At 32-bit floating-point precision, each of our fits requires 6 KB of memory. Our results were computed using 512 samples per pixel and are illustrated in Figure 4.4. For each material, the lobe of the BRDF (and hence the microfacet NDF) is captured accurately. Note however that, as we predicted in the previous section, our fit of the *fabric106* material failed at reproducing the directionally dependent albedo exhibited by the reference.

Speed Our fitting algorithms are very fast: it takes us less than 1 second to fit an isotropic material from the MERL database of Matusik et al. [MPBM03], and less than 20 seconds for an anisotropic material of Filip and Vavra [FV14]. Naturally, fitting performance depends on the resolution of the tables, i.e., on N. We measured the impact of such a factor for both isotropic and anisotropic algorithms. Results are plotted in Figure 4.5, where the timings include the computations of the slope PDF, the Smith term, the quantile functions, and the

Figure 4.3 – Side-by-side fitting comparisons of a few representative isotropic materials from the MERL database [MPBM03].

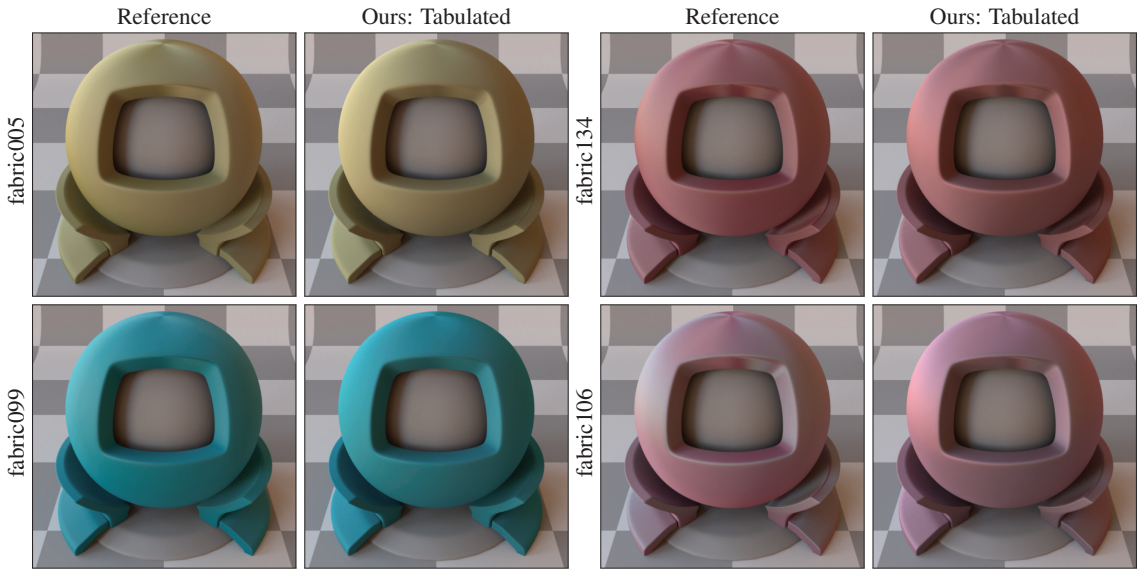


Figure 4.4 – Side-by-side fitting comparisons of a few representative anisotropic materials from the UTIA database [FV14].

Fresnel term on an Intel 2.4GHz Core i5 CPU. We believe such timings make our method much faster than previous work.

Memory Because our representation is roughness invariant, it allows us to create a multitude of materials at constant memory costs. As an example, we rendered the scene illustrated in Figure 4.6 using 512 samples per pixel and a few KB of memory. Such rendering configurations are only possible with slope-space tables. Otherwise, the per-pixel sampling rate and/or memory consumption to store importance sampling tables should be increased. For roughness mapped models such as ones shown in Figure 4.6, where the number of different materials is very large, such approaches would have been particularly impractical. Alongside analytic microfacet BRDF models, we believe our tabulation strategy is the first to support such rendering configurations trivially.

Conversion to Analytic BRDFs Although our memory footprint is constant per fitted material, some applications might not be able to afford the storage of precomputed tables. This is typically the case in real-time rendering contexts, where analytic microfacet BRDF models such as the Beckmann or GGX models are a necessity. We showed that our tabulated microfacet slope PDF can be converted straightforwardly to either Beckmann or GGX roughness parameters using slope moments. With such analytic microfacet models, only the Fresnel table needs to be stored. In order to quantify the loss in fitting quality compared to our tabu-

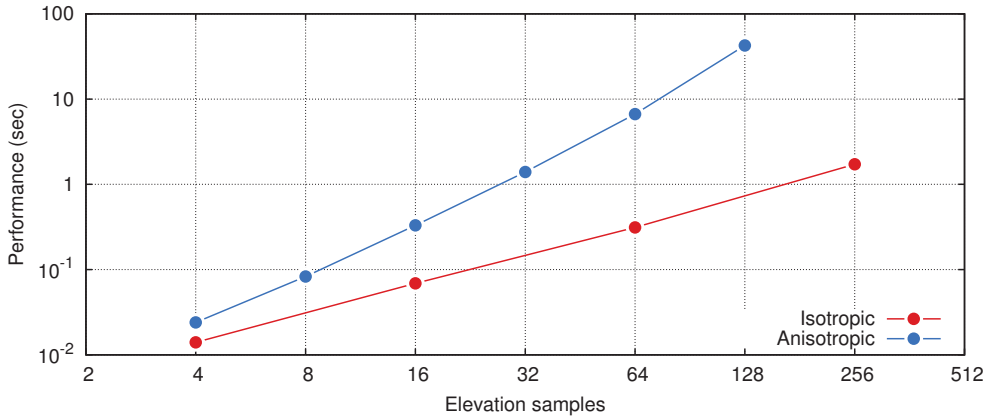
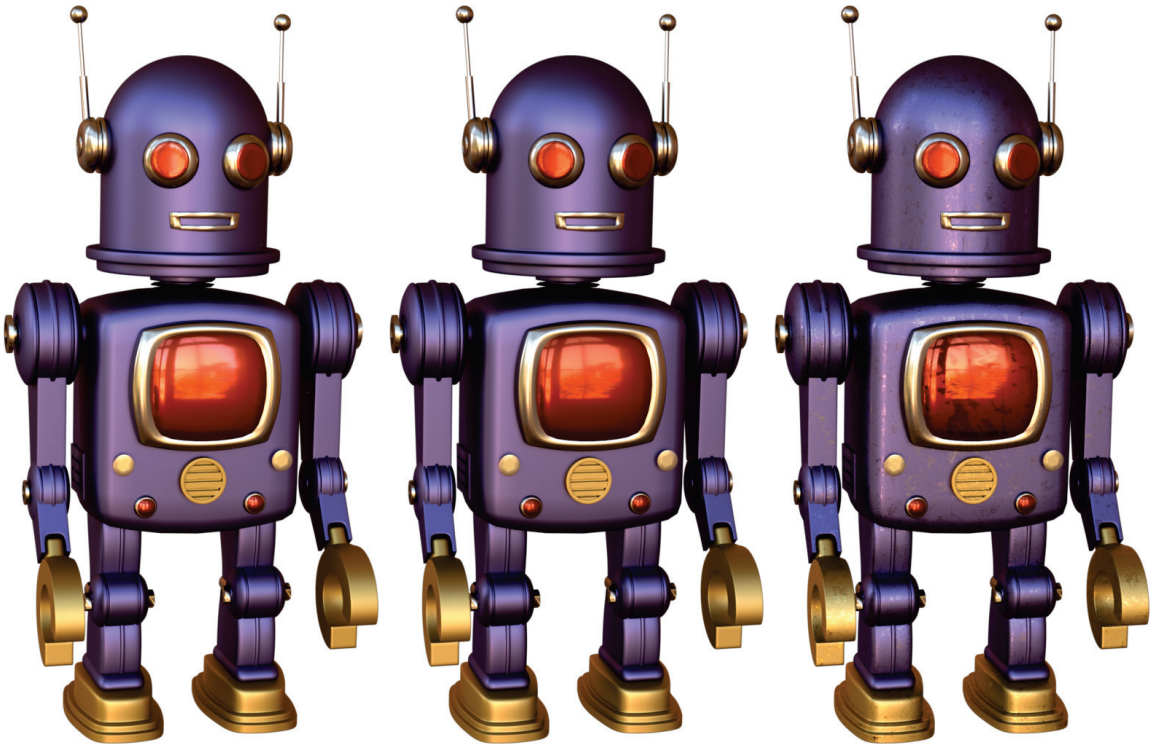


Figure 4.5 – Fitting timings (in seconds) of our algorithms as a function of the number of input material evaluations along the elevation direction.

lated representation, we placed some renderings obtained with the analytic models next to our tabulated fits in Figure 4.3. We noticed that the GGX model performs generally better than the Beckmann model (see also Figure 4.2), which is in agreement with previous observations [TR75, WMLT07, Bur12]. Note that the analytic models are also incorporated in the detailed fitting analysis provided in the supplemental document of the original article [DHI+15].

4.4 Implementation Details

Online Computations We implemented BSDF plugins in the Mitsuba renderer [Jak10] to use our microfacet model. Our plugins implement the functions *eval sample* and *pdf*, which are called by Mitsuba's Monte-Carlo integrator. Because our tables are roughness invariant, we produce a wide variety of roughness effects on the fly by scaling the lookup parameters by α_x, α_y, and ρ.



(a) (b) (c)

Figure 4.6 – Multiple material design on a production asset. (a) Measured BRDFs. (b) Our fitted microfacet BRDFs. (c) Our fitted microfacet BRDFs controled by a roughness texture map. Model courtesy of LAGOA.

Slope PDF Normalization We store the tabulated radial slope PDF. Once we have retrieved the coefficients with the power iteration method, we normalize the PDF by dividing each coefficient of the table by the integral

$$
\begin{aligned}
k &= \int_0^{2\pi}\int_0^{+\infty} r_h \cdot P_r(r_h)\cdot dr_h \\
&= 2\pi\int_0^{\pi/2} \tan\theta_h \cdot P_r(\tan\theta_h)\cdot \sec^2\theta_h \cdot d\theta_h && \left(r_h = \tan\theta_h \Rightarrow dr_h = \sec^2\theta_h\cdot d\theta_h\right) \\
&= 2\pi^2\int_0^{1} \tan(u^2\pi/2)\cdot P_r(\tan(u^2\pi/2))\cdot \sec^2(u^2\pi/2)\cdot u\cdot du && \left(\theta_h = u^2\pi/2 \Rightarrow d\theta_h = \pi u\cdot du\right) \\
&= \frac{2\pi^2}{N}\sum_{i=0}^{N} \tan(u_i^2\pi/2)\cdot P_r(\tan(u_i^2\pi/2))\cdot \sec^2(u_i^2\pi/2)\cdot u_i. && (u_i = i/N)
\end{aligned}
$$

```
void tabular::normalize_p22()
{
    const int ntheta = 128;
    const float_t dphi = 2.0 * M_PI;
    const float_t dtheta = M_PI / (float_t)ntheta;
    float_t k, nint = 0.0;

    for (int i = 0; i < ntheta; ++i) {
        float_t u = (float_t)i / (float_t)ntheta; // in [0,1)
        float_t theta_h = u * u * M_PI * 0.5;
        float_t r_h = tan(theta_h);
        float_t cos_theta_h = cos(theta_h);
        float_t p22_r = p22_radial(r_h * r_h);

        nint+= (u * p22_r * r_h) / (cos_theta_h * cos_theta_h);
    }
    nint*= dtheta * dphi;

    // normalize the slope pdf
    k = 1.0 / nint;
    for (int i = 0; i < (int)m_p22.size(); ++i)
        m_p22[i]*= k;
}
```

Listing 4.6 – Slope PDF normalization in C++.

Smith Masking Term We compute the Smith masking term with the following integral

$$
\begin{aligned}
G_1(\mathbf{k}) &= \frac{\cos\theta_k}{\int_{\Omega_+} \underline{\mathbf{kh}} \cdot D(\mathbf{h}) \cdot d\omega_h} \\
&= \frac{\cos\theta_k}{\int_0^{2\pi}\int_0^{\pi/2} \underline{\mathbf{kh}} \cdot D(\mathbf{h}) \cdot \sin\theta_h \cdot d\theta_h \cdot d\phi_h} && \begin{pmatrix} \omega_h = \omega_h(\theta_h, \phi_h) \\ \Rightarrow d\omega_h = \sin\theta_h \cdot d\theta_h \cdot d\phi_h \end{pmatrix} \\
&= \frac{\cos\theta_k}{2\pi^2 \int_0^1\int_0^1 \underline{\mathbf{kh}} \cdot D(\mathbf{h}) \cdot \sin(u_1^2\pi/2) \cdot du_1 \cdot du_2} && \begin{pmatrix} \theta_h = u_1^2\pi/2,\ \phi_h = 2\pi u_2 \\ \Rightarrow d\theta_h = \pi u_1 \cdot du_1,\ d\phi_h = 2\pi \cdot du_2 \end{pmatrix} \\
&= \frac{N_i N_j \cdot \cos\theta_k}{2\pi^2 \sum_{i=0}^{N_i}\sum_{j=0}^{N_j} \underline{\mathbf{kh}} \cdot D(\mathbf{h}) \cdot \sin(u_i^2\pi/2)}. && (u_i = i/N_i,\ u_j = j/N_j)
\end{aligned}
$$

Note that for isotropic materials, the Smith term is independent from the azimuthal component of the incident vector ϕ_k. So in practice, we only store the results for the case $\phi_k = 0$.

```
void tabular::compute_g1()
{
    const int ntheta = 90;
    const int nphi   = 180;
    float_t dtheta = M_PI / (float_t)ntheta;
    float_t dphi   = 2.0 * M_PI / (float_t)nphi;
    int cnt = m_p22.size() - 1;

    for (int i = 0; i < cnt; ++i) {
        float_t tmp = (float_t)i / (float_t)cnt; // in [0, 1)
        float_t theta_k = tmp * 0.5 * M_PI; // in [0, pi/2)
        float_t cos_theta_k = cos(theta_k);
        float_t sin_theta_k = sin(theta_k);
        float_t nint = 0.0;

        for (int j2 = 0; j2 < nphi; ++j2) {
            float_t u_j = (float_t)j2 / (float_t)nphi; // in [0, 1)
            float_t phi_h = u_j * 2.0 * M_PI;          // in [0, 2pi)
            for (int j1 = 0; j1 < ntheta; ++j1) {
                float_t u_i = (float_t)j1 / (float_t)ntheta; // in [0, 1)
                float_t theta_h = u_i * u_i * M_PI * 0.5; // in [0, sqrt(pi/2))
                float_t sin_theta_h = sin(theta_h);
                float_t kh = sin_theta_k * sin_theta_h * cos(phi_h)
                           + cos_theta_k * cos(theta_h);

                nint+= max((float_t)0.0, kh)
                     * ndf(vec3(theta_h, phi_h))
                     * u_i * sin_theta_h;
            }
        }
        nint*= dtheta * dphi;
        m_g1.push_back(min((float_t)1.0, cos_theta_k / nint));
    }
    m_g1.push_back(0);
}
```

Listing 4.7 – Computation of the Smith monostatic shadowing term in C++.

Importance Sampling Table Our tabulated representation does not use the distribution of visible normals for importance sampling. Rather, it relies on the less effective scheme of Walter et al. [WMLT07], which consists in sampling the distribution of slopes. We describe here the procedure to build the table in the case of radial slope distributions. In this case, we produce a sample in polar coordinates. The angle is taken uniformly, while the radius is sampled from the quantile function associated to the radial component of the distribution P_r. First,

we compute the radial CDF $F_r \in [0,1]$ associated to P_r

$$
\begin{aligned}
F_r(r_h) &= 2\pi \int_0^{r_h} r \cdot P_r(r) \cdot dr \\
&= 2\pi \int_0^{r_h} \tan\theta \cdot P_r(\tan\theta) \cdot \sec^2\theta \cdot d\theta && \begin{pmatrix} r = \tan\theta \\ \Rightarrow dr = \sec^2\theta \cdot d\theta \end{pmatrix} \\
&= 2\pi^2 \int_0^{r_h} \tan(u^2\pi/2) \cdot P_r(\tan(u^2\pi/2)) \cdot \sec^2(u^2\pi/2) \cdot d\theta && \begin{pmatrix} \theta = u^2\pi/2 \\ \Rightarrow d\theta = u\pi \cdot du \end{pmatrix} \\
&= \frac{2\pi^2}{N} \sum_{i=0}^{r_i < r_h} \tan(u_i^2\pi/2) \cdot P_r(\tan(u_i^2\pi/2)) \cdot \sec^2(u_i^2\pi/2) \cdot d\theta. && \begin{pmatrix} u_i = i/N \\ r_i = \tan(u_i^2\pi/2) \end{pmatrix}
\end{aligned}
$$

Once the CDF is computed, we invert it numerically to get the quantile function $Q_r = F_r^{-1}$.

```
void tabular::compute_cdf()
{
    int cnt = (int)m_p22.size() - 1;
    float_t dtheta = M_PI / (float_t)cnt;
    float_t nint = 0.0;

    m_cdf.resize(0);
    for (int i = 0; i < cnt; ++i) {
        float_t u = (float_t)i / (float_t)cnt;
        float_t theta_h = u * u * M_PI * 0.5;
        float_t cos_theta_h = cos(theta_h);
        float_t r_h = tan(theta_h);
        float_t p22_r = p22_radial(r_h * r_h);

        nint+= (u * r_h * p22_r) / (cos_theta_h * cos_theta_h);
        m_cdf.push_back(nint * dtheta * /* normalize */(2.0 * M_PI));
    }
    m_cdf.push_back(1);
}
```

Listing 4.8 – Computation of the radial CDF of a tabulated radial microfacet slope PDF in C++.

```
void tabular::compute_qf()
{
    int cnt = (int)m_p22.size() - 1;
    int res = cnt * 8; // resolution of inversion

    m_qf.resize(0);
    m_qf.push_back(0);
    for (int i = 1; i < cnt; ++i) {
        float_t cdf = (float_t)i / (float_t)cnt;

        for (int j = 0; j < res; ++j) {
            float_t u = (float_t)j / (float_t)res;
            float_t theta_h = u * M_PI * 0.5;
            float_t qf = cdf_radial(tan(theta_h));

            // lerp lookup
            if (qf >= cdf) {
                m_qf.push_back(u);
                break;
            } else if (j == res) {
                m_qf.push_back(1.0);
            }
        }
    }
    m_qf.push_back(1.0);
}
```

Listing 4.9 – Computation of the radial QF of a tabulated radial microfacet slope PDF in C++.

In the following paragraphs, we describe a way to precompute the tables for importance sampling with a tabulated distribution of visible normals.

Marginal Slope Quantile Function Before building the quantile functions for importance sampling with the distribution of visible slopes, we need to compute the quantile function associated to the marginal slope PDF $P_1 > 0$, which is

$$\begin{aligned}
P_1(\tilde{x}_h) &= \int_{-\infty}^{\infty} P(\tilde{x}_h, \tilde{y}_h) \cdot d\tilde{y}_h \\
&= \int_{-\pi/2}^{\pi/2} P(\tilde{x}_h, \tan\theta_h) \cdot \sec^2\theta_h \cdot d\theta_h && \left(\tilde{y}_h = \tan\theta_h \Rightarrow d\tilde{y}_h = \sec^2\theta_h \cdot d\theta_h\right) \\
&= 2\int_{0}^{\pi/2} P(\tilde{x}_h, \tan\theta_h) \cdot \sec^2\theta_h \cdot d\theta_h && (P(x,y) = P(x,-y)) \\
&= 2\pi\int_{0}^{1} P(\tilde{x}_h, \tan(u^2\pi/2)) \cdot \sec^2(u^2\pi/2) \cdot u \cdot du && \left(\theta_h = u^2\pi/2 \Rightarrow d\theta_h = u\pi \cdot du\right) \\
&= \frac{2\pi}{N}\sum_{i=0}^{N} P(\tilde{x}_h, \tan(u_i^2\pi/2)) \cdot \sec^2(u_i^2\pi/2) \cdot u_i. && (u_i = i/N)
\end{aligned}$$

Note that the property $P(x,y) = P(x,-y)$ is only true for isotropic materials, as P_1 is even, i.e., $P_1(x) = P_1(-x)$. We take advantage of this property to precompute the PDF for $x \in [0, +\infty)$, rather than $x \in (-\infty, +\infty)$. The CDF

$F_1 \in [0,1]$ associated to P_1 can also benefit from symmetry. Indeed, we can write it as

$$F_1(x) = \frac{1}{2} + \text{sign}(x) \cdot \int_0^{|x|} P_1(y) \cdot dy.$$

So in practice, we precompute for $x \in [0, +\infty)$ the integral

$$\begin{aligned}
\int_0^x P_1(y) \cdot dy &= \int_0^{\arctan x} P_1(\tan\theta) \cdot \sec^2\theta \cdot d\theta && \left(y = \tan\theta \Rightarrow dy = \sec^2\theta \cdot d\theta\right) \\
&= \pi \int_0^{\frac{2}{\pi}\arctan x} P_1(\tan(u^2\pi/2)) \cdot \sec^2(u^2\pi/2) \cdot u \cdot du && \left(\theta = u^2\pi/2 \Rightarrow d\theta = \pi u \cdot du\right) \\
&= \frac{\pi}{N} \sum_{i=0}^{j} P_1(\tan(u_i^2\pi/2)) \cdot \sec^2(u_i^2\pi/2) \cdot u_i. && \left(u_i = i/N,\ j = \frac{2N}{\pi}\arctan x\right)
\end{aligned}$$

Once we have computed the CDF, we can compute the quantile function $Q_1 = F_1^{-1}$. Since F_1 is symmetric, we only need to compute and store half of the quantile function.

Marginal Visible Slope Quantile Function We exploit a property of isotropic materials to make the precomputation of the quantile function of the marginal visible slope PDF practical. For isotropic materials, the distribution of visible slopes is independent of the azimuthal angle of the incident vector. Hence, by choosing $\phi_k = 0$, we can express the marginal visible slope PDF as

$$P_{\text{vis}}(\tilde{\mathbf{h}}; \mathbf{k}) = \max(0, 1 - \tan\theta_k \cdot \tilde{x}_h) \cdot P(\tilde{\mathbf{h}}) \cdot G_1(\mathbf{k}).$$

From this equation, it follows that the marginal visible slope PDF $P_2 > 0$ is

$$\begin{aligned}
P_2(\tilde{x}_h; \mathbf{k}) &= \int_{-\infty}^{\infty} P_{\text{vis}}(\tilde{x}_h, \tilde{y}; \mathbf{k}) \cdot d\tilde{y} \\
&= \max(0,\ 1 - \tan\theta_k \cdot \tilde{x}_h) \cdot P_1(\tilde{x}_h) \cdot G_1(\mathbf{k}).
\end{aligned}$$

Assuming that $\tilde{x}_h < \cot\theta_k$, the marginal visible slope CDF is defined as

$$\begin{aligned}
F_2(\tilde{x}_h;\mathbf{k}) &= \int_{-\infty}^{\tilde{x}_h} P_2(\tilde{x};\mathbf{k})\cdot d\tilde{x} \\
&= G_1(\mathbf{k})\int_{-\infty}^{\tilde{x}_h}(1-\tan\theta_k\cdot\tilde{x})\cdot P_1(\tilde{x})\cdot d\tilde{x} \\
&= G_1(\mathbf{k})\int_0^{F_1(\tilde{x}_h)}(1-\tan\theta_k\cdot Q_1(p))\cdot dp && \left(\tilde{x}=Q_1(p)\Rightarrow d\tilde{x}=\frac{1}{P_1(Q_1(p))}\cdot dp\right) \\
&= F_1(\tilde{x}_h)\cdot G_1(\mathbf{k})\left[1-\tan\theta_k\cdot\int_0^1 Q_1(F_1(\tilde{x}_h)\cdot u)\cdot du\right] && (p=F_1(\tilde{x}_h)\cdot u\Rightarrow dp=F_1(\tilde{x}_h)\cdot du) \\
&= F_1(\tilde{x}_h)\cdot G_1(\mathbf{k})\left[1-\frac{\tan\theta_k}{N}\sum_{i=0}^{N} Q_1(F_1(\tilde{x}_h)\cdot u_i)\right]. && (u_i=i/N)
\end{aligned}$$

In order to invert F_2, we use the fact that the CDF may be written as a composition

$$F_2(\tilde{x}_h;\mathbf{k}) = g(F_1(\tilde{x}_h);\mathbf{k}) \Rightarrow Q_2(u) = Q_1(g^{-1}(u;\mathbf{k})),$$

where

$$g(x;\mathbf{k}) = x\cdot G_1(\mathbf{k})\left[1-\tan\theta_k\cdot\int_0^1 Q_1(x\cdot u)\cdot du\right].$$

Conditional Visible Slope Quantile Function The conditional visible slope PDF is defined as

$$\begin{aligned}
P_3(\tilde{y}_h|\tilde{x}_h;\mathbf{k}) &= \frac{P_{\text{vis}}(\tilde{x}_h,\tilde{y}_h;\mathbf{k})}{P_2(\tilde{x}_h;\mathbf{k})} \\
&= \frac{P(\tilde{x}_h,\tilde{y}_h)}{P_1(\tilde{x}_h)}.
\end{aligned}$$

For isotropic materials, P_3 is a symmetric PDF. Therefore, we can apply some of the treatments we used to compute F_1 and Q_1 for F_3 and Q_3, respectively. Note that computing F_3 by integrating the ratio of P and P_1 becomes numerically unstable as soon as we consider large $\tilde{x}_h$ values. To avoid this issue, we proceed as follows. Let $F\in[0,1]$ denote the bivariate CDF associated to P. We have

$$\begin{aligned}
F(\tilde{x}_h,\tilde{y}_h) &= \int_{-\infty}^{\tilde{x}_h}\int_{-\infty}^{\tilde{y}_h} P(\tilde{x},\tilde{y})\cdot d\tilde{x}\cdot d\tilde{y} \\
&= \int_0^{F_1(\tilde{x}_h)}\int_0^{F_1(\tilde{y}_h)} \frac{P(Q_1(u_1),Q_1(u_2))}{P_1(Q_1(u_1))\cdot P_1(Q_1(u_2))}\cdot du_1\cdot du_2 && \left(\begin{array}{c}\tilde{x}=Q_1(u_1),\ \tilde{y}=Q_1(u_2)\\ \Rightarrow d\tilde{x}=\frac{1}{P_1(Q_1(u_1))}\cdot du_1,\ d\tilde{y}=\frac{1}{P_1(Q_1(u_2))}\cdot du_2\end{array}\right) \\
&= C(F_1(\tilde{x}_h),F_1(\tilde{y}_h)),
\end{aligned}$$

where $C \in [0,1]$ is the copula CDF [Sch07] associated to the microfacet slope distribution. We can express the PDF of the microfacet slope distribution as the derivative of the copula

$$\begin{aligned} P(\tilde{x}_h, \tilde{y}_h) &= \frac{\partial^2 C(F_1(\tilde{x}_h), F_1(\tilde{y}_h))}{\partial \tilde{x}_h \cdot \partial \tilde{y}_h} \\ &= c(F_1(\tilde{x}_h), F_1(\tilde{y}_h)) \cdot P_1(\tilde{x}_h) \cdot P_1(\tilde{y}_h), \end{aligned}$$

where $c = \frac{\partial^2 C(u_1,u_2)}{\partial u_1 \cdot \partial u_2} \geq 0$ is the copula PDF associated to the microfacet slope distribution. It follows that we can write

$$\begin{aligned} F_3(\tilde{y}_h|\tilde{x}_h) &= \int_{-\infty}^{\tilde{y}_h} \frac{P(\tilde{x}_h, \tilde{y})}{P_1(\tilde{x}_h)} \cdot d\tilde{y} \\ &= \int_{-\infty}^{\tilde{y}_h} \frac{c(F_1(\tilde{x}_h), F_1(\tilde{y})) \cdot \cancel{P_1(\tilde{x}_h)} \cdot P_1(\tilde{y})}{\cancel{P_1(\tilde{x}_h)}} \cdot d\tilde{y} \\ &= \int_0^{F_1(\tilde{y}_h)} \frac{c(F_1(\tilde{x}_h), u) \cdot \cancel{P_1(Q_1(u))}}{\cancel{P_1(Q_1(u))}} \cdot du \qquad \begin{pmatrix} \tilde{y} = Q_1(u) \\ \Rightarrow d\tilde{y} = \frac{1}{P_1(Q_1(u))} \cdot du \end{pmatrix} \\ &= \frac{\partial C(F_1(\tilde{x}_h), F_1(\tilde{y}_h))}{\partial u_1}. \qquad \left(\frac{\partial C(F_1(\tilde{x}_h), 0)}{\partial u_1} = 0 \right) \end{aligned}$$

Hence, we can compute and store the partial derivative of the copula in practice, which avoids numerical issues since no divisions are involved. The quantile function Q_3 is computed and stored similarly to Q_1 for multiple $\tilde{x}_h$ values.

Chapter 5

Conclusion

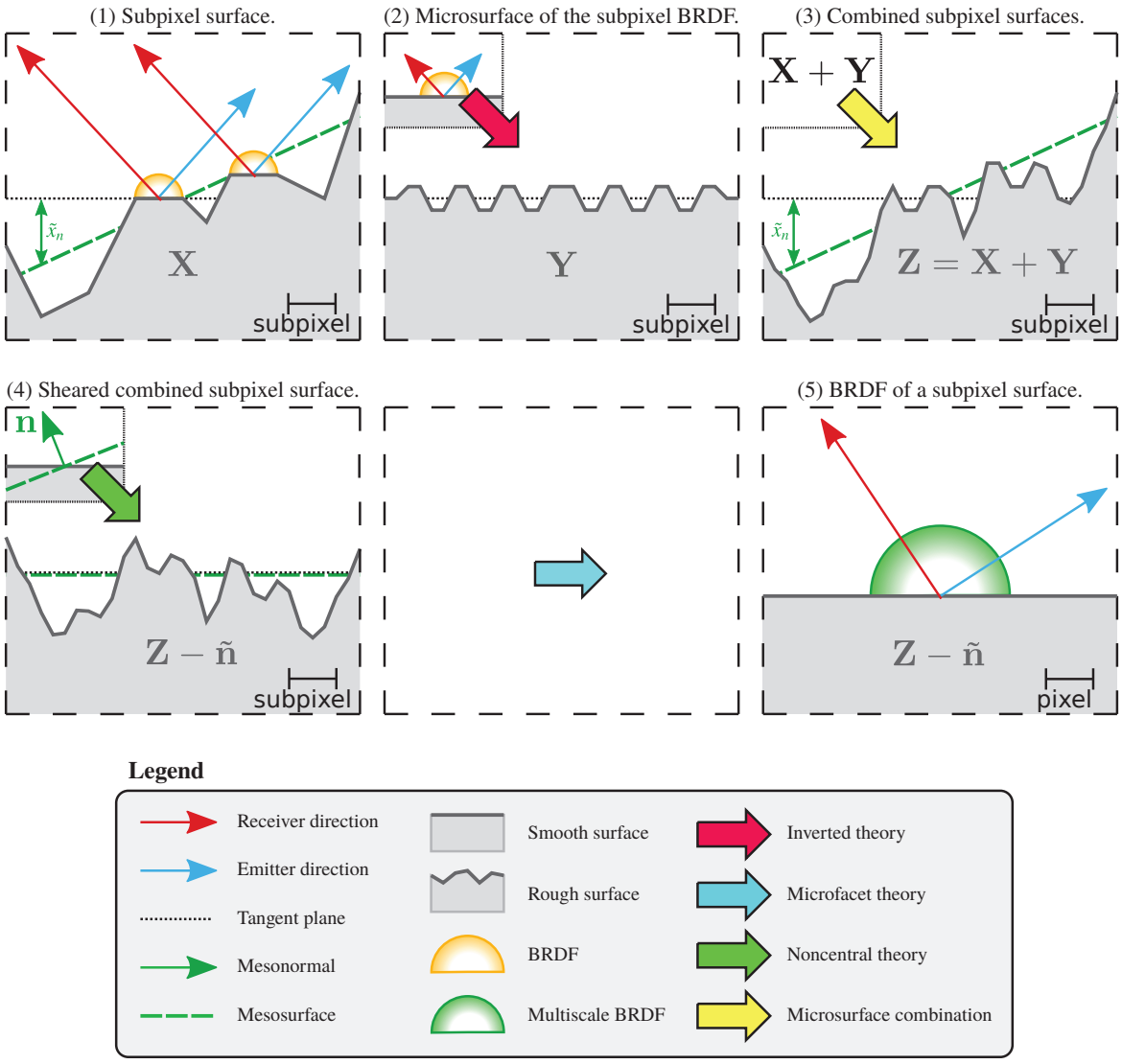


Figure 5.1 – Geometric overview of our displacement map filtering algorithm. In order to handle (1) displaced surfaces of arbitrary reflectance, we first (2) extract a microsurface from the BRDF of the base surface. (3) We show how to combine the microsurface and the displaced surface together in order to produce a noncentral microsurface, which (4) can be handled with noncentral microfacet theory. (5) The filtered response of a displaced surface can thus be computed from a multiscale microfacet BRDF.

5.1 Downsampling Operator

In the previous chapters, we saw how to filter displaced Fresnel mirrors at any scale with noncentral microfacet theory, and how to convert any material into a microfacet BRDF with inverted microfacet theory. With both these tools at hand, we can formulate a more general filtering scheme for displaced surfaces of arbitrary reflectance as the problem of deriving a microfacet BRDF of a surface displaced by the sum of two random processes: one that comes from the displacement map itself, and another that comes from the base material of the surface (if the base material of the surface is a microfacet BRDF, then the displacement is already known; otherwise, we convert it using the algorithms we presented in Chapter 4). It follows that if we can retrieve the microfacet NDF that emerges from the combination of both processes, then we solve our filtering problem; the entire idea is illustrated in Figure 5.1.

Combining Displacements By noting that summing displacement processes also implies summing their derivatives, i.e., their slopes, we can express the problem of retrieving the slope PDF (and hence, the NDF) of the combined displacement process more formally as follows: If we denote by $\mathbf{X}$ and $\mathbf{Y}$ the random slope processes yielded by the displacement map, and the base material with PDFs $P_{\mathbf{X}} \geq 0$ and $P_{\mathbf{Y}} \geq 0$ respectively, then what is the PDF $P_{\mathbf{Z}} \geq 0$ of the random process $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$? We can solve this problem by using the characteristic function [Luk72, GA02]. The characteristic function $C_{\mathbf{Z}} \geq 0$ of a bivariate random process $\mathbf{Z}$ is defined as the Fourier transform of its PDF, i.e.,

$$C_{\mathbf{Z}}(\mathbf{t}) = \int_{\mathbb{R}^2} e^{i\mathbf{t}\tilde{\mathbf{h}}} \cdot P_{\mathbf{Z}}(\tilde{\mathbf{h}}) \cdot d\tilde{\mathbf{h}}. \tag{5.1}$$

Equivalently, Equation (5.1) can be seen as a complex moment function

$$C_{\mathbf{Z}}(\mathbf{t}) = \mathbb{E}[e^{i\mathbf{Z}\mathbf{t}}]. \tag{5.2}$$

Since $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$, we can also write $C_{\mathbf{Z}}(\mathbf{t}) = \mathbb{E}[e^{i(\mathbf{X}+\mathbf{Y})\mathbf{t}}]$, so that the characteristic function of $\mathbf{Z}$ becomes a function of the PDFs of $\mathbf{X}$ and $\mathbf{Y}$

$$C_{\mathbf{Z}}(\mathbf{t}) = \int_{\mathbb{R}^2} \int_{\mathbb{R}^2} e^{i\mathbf{t}(\tilde{\mathbf{x}}+\tilde{\mathbf{y}})} \cdot P_{\mathbf{X}}(\tilde{\mathbf{x}}) \cdot P_{\mathbf{Y}}(\tilde{\mathbf{y}}) \cdot d\tilde{\mathbf{x}} \cdot d\tilde{\mathbf{y}}. \tag{5.3}$$

The PDF of $\mathbf{Z}$ is then the inverse Fourier transform of Equation (5.3), which, after a few manipulations and using the definition of the Dirac delta function, expresses $P_{\mathbf{Z}}$ as the convolution $P_{\mathbf{X}} * P_{\mathbf{Y}}$ between $P_{\mathbf{X}}$ and $P_{\mathbf{Y}}$

$$
\begin{aligned}
P_{\mathbf{Z}}(\tilde{\mathbf{h}}) &= \frac{1}{4\pi^2}\int_{\mathbb{R}^2} e^{-i\tilde{\mathbf{h}}\mathbf{t}} \cdot C_{\mathbf{Z}}(\mathbf{t}) \cdot d\mathbf{t} \\
&= \frac{1}{4\pi^2}\int_{\mathbb{R}^2} e^{-i\tilde{\mathbf{h}}\mathbf{t}} \cdot \left[\int_{\mathbb{R}^2}\int_{\mathbb{R}^2} e^{i(\tilde{\mathbf{x}}+\tilde{\mathbf{y}})\mathbf{t}} \cdot P_{\mathbf{X}}(\tilde{\mathbf{x}}) \cdot P_{\mathbf{Y}}(\tilde{\mathbf{y}}) \cdot d\tilde{\mathbf{x}} \cdot d\tilde{\mathbf{y}}\right] \cdot d\mathbf{t} && \text{(Equation (5.3))} \\
&= \frac{1}{4\pi^2}\int_{\mathbb{R}^2}\int_{\mathbb{R}^2}\int_{\mathbb{R}^2} e^{i(\tilde{\mathbf{x}}+\tilde{\mathbf{y}}-\tilde{\mathbf{h}})\mathbf{t}} \cdot P_{\mathbf{X}}(\tilde{\mathbf{x}}) \cdot P_{\mathbf{Y}}(\tilde{\mathbf{y}}) \cdot d\tilde{\mathbf{x}} \cdot d\tilde{\mathbf{y}} \cdot d\mathbf{t} \\
&= \int_{\mathbb{R}^2}\int_{\mathbb{R}^2} \delta(\tilde{\mathbf{x}}+\tilde{\mathbf{y}}-\tilde{\mathbf{h}}) \cdot P_{\mathbf{X}}(\tilde{\mathbf{x}}) \cdot P_{\mathbf{Y}}(\tilde{\mathbf{y}}) \cdot d\tilde{\mathbf{x}} \cdot d\tilde{\mathbf{y}} && \left(\delta(\mathbf{x}) \overset{\text{def}}{=} \frac{1}{4\pi^2}\int_{\mathbb{R}^2} e^{i\mathbf{x}\mathbf{t}} \cdot d\mathbf{t}\right) \\
&= \int_{\mathbb{R}^2} P_{\mathbf{X}}(\tilde{\mathbf{x}}) \cdot P_{\mathbf{Y}}(\tilde{\mathbf{h}}-\tilde{\mathbf{x}}) \cdot d\tilde{\mathbf{x}} && \left(\int_{\mathbb{R}^2} \delta(\tilde{\mathbf{x}}+\tilde{\mathbf{y}}-\tilde{\mathbf{h}}) \cdot P_{\mathbf{Y}}(\tilde{\mathbf{y}}) \cdot d\tilde{\mathbf{y}} = P_{\mathbf{Y}}(\tilde{\mathbf{h}}-\tilde{\mathbf{x}})\right) \\
&= (P_{\mathbf{X}} * P_{\mathbf{Y}})(\tilde{\mathbf{h}}).
\end{aligned}
$$

We can thus retrieve the microfacet NDF of the combined process by computing a convolution of microfacet slope PDFs. Note that this property only works if $\mathbf{X}$ and $\mathbf{Y}$ are statistically independent, which is a valid assumption in the case of merging a displacement map into a microfacet BRDF process.

Beckmann Distributions In a physically based rendering context, numerical convolutions are too expensive to be practical. We can avoid numerical convolutions by relying on Beckmann slope distributions. Indeed, Beckmann slope distributions belong to the family of stable distributions [Nol13] (as they are equivalent to Gaussian distributions, up to a scale factor), which have the convenient particularity of being invariant to convolution, i.e., the convolution of two Beckmann PDFs is also a Beckmann PDF. More formally, if $\mathbf{X}$ and $\mathbf{Y}$ follow independent Beckmann distributions with respective PDF parameters $\{\alpha_{x,\mathbf{X}}, \alpha_{y,\mathbf{X}}, \rho_{\mathbf{X}}, \tilde{\mathbf{n}}_{\mathbf{X}}\}$ and $\{\alpha_{x,\mathbf{Y}}, \alpha_{x,\mathbf{Y}}, \rho_{\mathbf{Y}}, \tilde{\mathbf{n}}_{\mathbf{Y}}\}$, then the random process $\mathbf{Z} = \mathbf{X} + \mathbf{Y}$ follows a Beckmann distribution with parameters $\{\alpha_x, \alpha_y, \rho, \tilde{\mathbf{n}}\}$, where

$$
\begin{aligned}
\alpha_x^2 &= \alpha_{x,\mathbf{X}}^2 + \alpha_{x,\mathbf{Y}}^2 \\
\alpha_y^2 &= \alpha_{y,\mathbf{X}}^2 + \alpha_{y,\mathbf{Y}}^2 \\
\rho &= \frac{\rho_{\mathbf{X}}\alpha_{x,\mathbf{X}}\alpha_{y,\mathbf{X}} + \rho_{\mathbf{Y}}\alpha_{x,\mathbf{Y}}\alpha_{y,\mathbf{Y}}}{(\alpha_{x,\mathbf{X}}^2 + \alpha_{y,\mathbf{X}}^2)(\alpha_{x,\mathbf{Y}}^2 + \alpha_{y,\mathbf{Y}}^2)} \\
\tilde{\mathbf{n}} &= \tilde{\mathbf{n}}_{\mathbf{X}} + \tilde{\mathbf{n}}_{\mathbf{Y}}.
\end{aligned}
$$

These parameters can be computed from the linear representation of Beckmann distributions, since we have [OB10]

$$\begin{aligned}
\mathbb{E}[\mathbf{Z}] &= \mathbb{E}[\mathbf{X}] + \mathbb{E}[\mathbf{Y}] \\
\mathbb{E}[\mathbf{Z}^2] &= \mathbb{E}[\mathbf{X}^2] + \mathbb{E}[\mathbf{Y}^2] + 2\mathbb{E}[\mathbf{X}]\mathbb{E}[\mathbf{Y}] \\
\mathbb{E}[Z_x Z_y] &= \mathbb{E}[X_x X_y] + \mathbb{E}[Y_x Y_y] + \mathbb{E}[X_x]\mathbb{E}[Y_y] + \mathbb{E}[X_y]\mathbb{E}[Y_x],
\end{aligned}$$

which implies that the convolution of Beckmann PDFs can be computed by simply summing their linear representation.

```
beckmann::lrep beckmann::lrep::operator+(const lrep& r) const
{
    return beckmann::lrep(m_E1 + r.m_E1,
                          m_E2 + r.m_E2,
                          m_E3 + r.m_E3 + 2.0 * m_E1 * r.m_E1,
                          m_E4 + r.m_E4 + 2.0 * m_E2 * r.m_E2,
                          m_E5 + r.m_E5 + m_E1 * r.m_E2 + m_E2 * r.m_E1);
}

beckmann::lrep& beckmann::lrep::operator+=(const lrep& rep)
{
    m_E1+= rep.m_E1;
    m_E2+= rep.m_E2;
    m_E3+= rep.m_E3 + 2.0 * m_E1 * rep.m_E1;
    m_E4+= rep.m_E4 + 2.0 * m_E2 * rep.m_E2;
    m_E5+= rep.m_E5 + m_E1 * rep.m_E2 + m_E2 * rep.m_E1;

    return (*this);
}
```

Listing 5.1 – Summing the linear representation of a Beckmann distribution in C++.

Implementation Details The implementation of our displacement map downsampling operator is divided into two stages: a precomputation stage and a runtime stage. The precomputation stage consists in converting the input displacement texture maps into LEAN/LEADR maps using Listing 3.6 and Listing 3.7, and computing the associated MIP maps. Furthermore, if the input BRDF of the displaced surface is not based on a Beckmann microfacet BRDF, then it must also be converted into Beckmann roughness parameters using the procedure we described in Listing 4.4. This completes the precomputation stage. At runtime, we incorporate the displacements that are too fine to be perceived at the pixel level into shading by evaluating a noncentral microfacet BRDF, i.e., the equations we introduced in Chapter 3. In order to retrieve the parameters of the BRDF, we evaluate a MIP texture lookup of the LEAN/LEADR map and merge the retrieved linear data with the base roughness of the surface BRDF using Listing 5.1 and Listing 3.5. Figure 5.2 illustrates the GPU pipeline of our implementation.

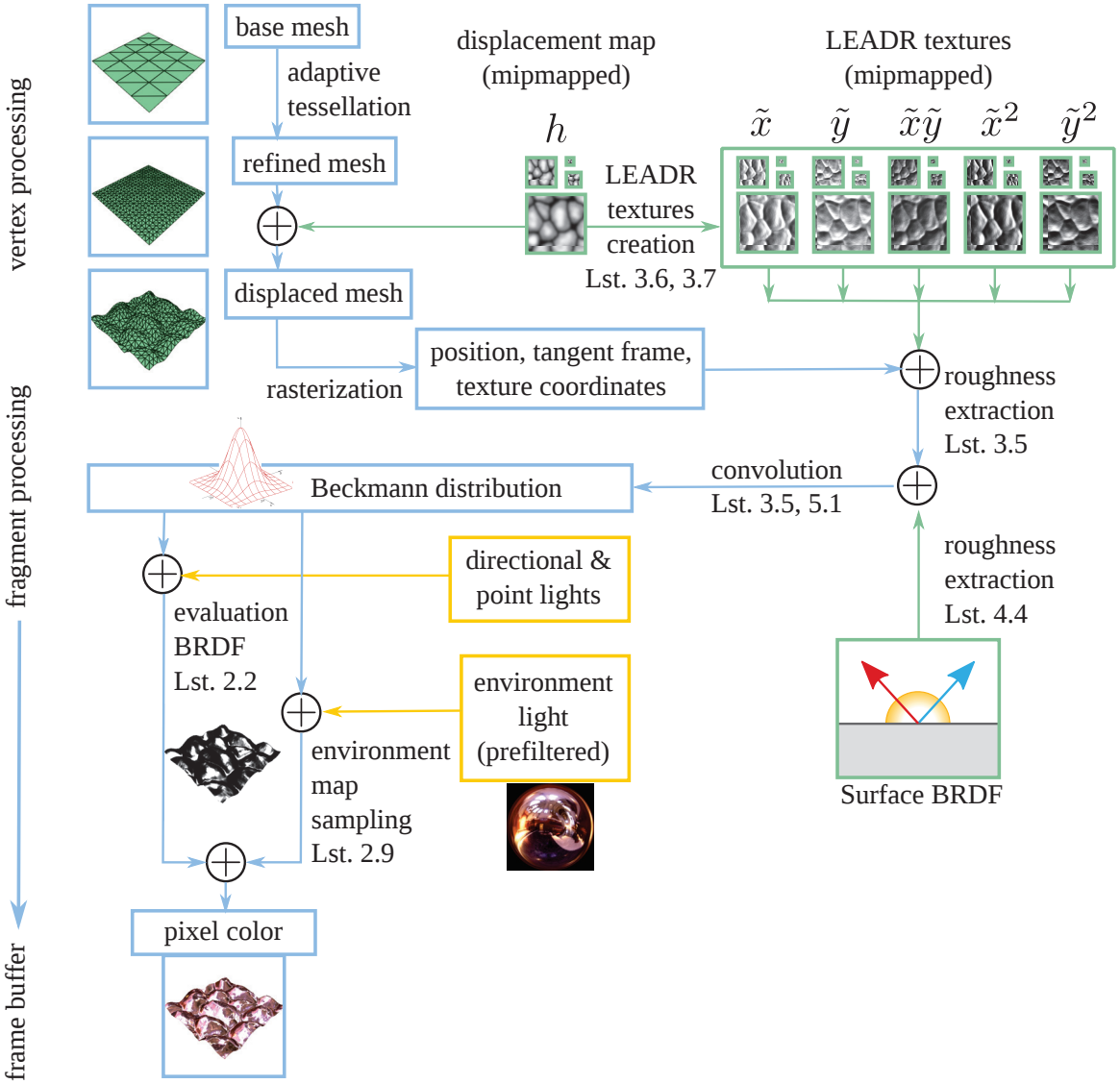


Figure 5.2 – GPU pipeline of our displacement map filtering technique.

Downsampling Results Our downsampling operator for displaced surfaces of arbitrary reflectance takes a displacement map and a BRDF as input, and produces a MIP map hierarchy that can preserve the anisotropy of light transport at any resolution. Figure 5.3 illustrates the importance of this property, where a displacement map is progressively rendered at lower resolutions with the *nickel* material from the MERL database. Notice how the overall appearance is preserved across resolutions with our method; in contrast, naive MIP mapping results in a smoother appearance. Figures 5.4 and 5.5 show the effect of our downsampling operator on two different high resolution displacement maps, applied on surfaces with a base material from the MERL database. The

leftmost image provides a supersampled rendering (using 256^2 samples per pixel) of the original material and displacement map, and serves as a reference image. Thanks to our representation, physically based renderers can load lower texture resolutions for distant objects without biasing the final image, thus reducing input-output operations as well as variance.

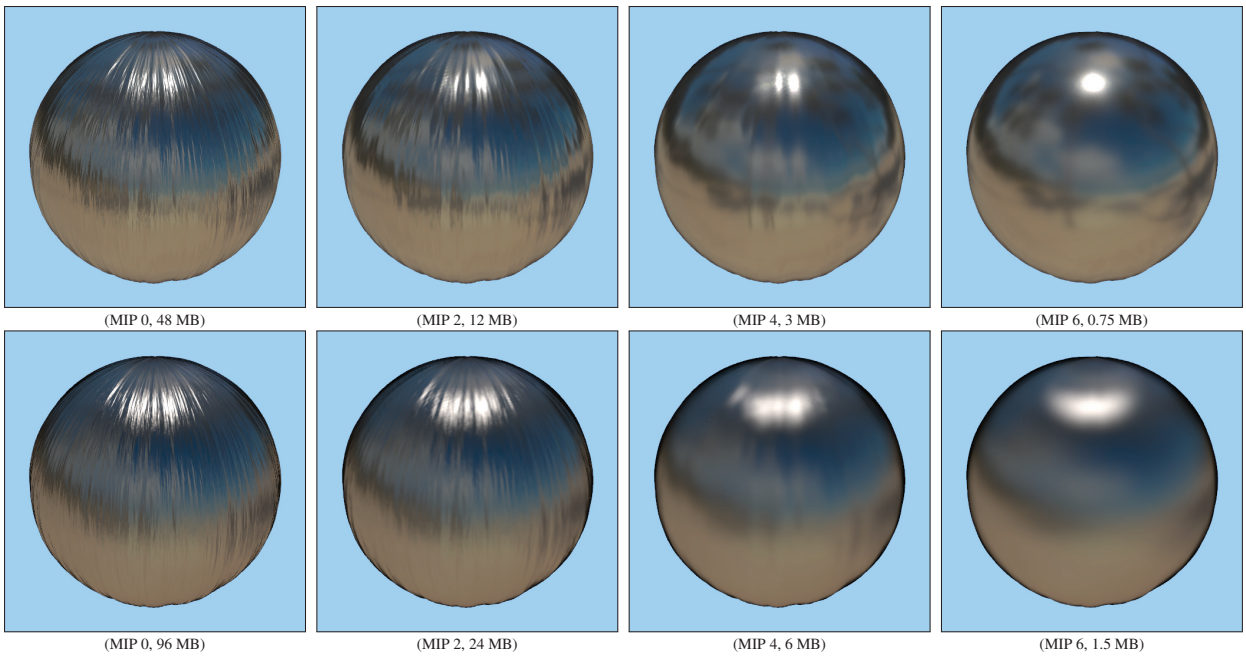


Figure 5.3 – Comparison between (top) naive MIP mapping and (bottom) our downsampling operator. The base material is *nickel* from the MERL database.

5.2 Positioning and Discussions

The problem of appearance modeling and filtering is not new in the computer graphics literature. Here, we position and discuss some properties and aspects of the contributions of this thesis with respect to existing solutions.

Computer Graphics BRDFs Reflectance modeling has a long history in computer graphics, and entire surveys are dedicated to this only topic [Sch94a, KE09, MSUA12]. According to Blinn [Bli77], the very first computer graphics BRDF models acted as ideal Lambertian reflectors. While matte surfaces tend to be well approximated by such BRDFs, they quickly turned out to be inadequate to mimic glossy apperances. In or-

Figure 5.4 – Effect of our downsampling operator on scratch-like displaced surfaces with a few representative isotropic materials from the MERL database [MPBM03].

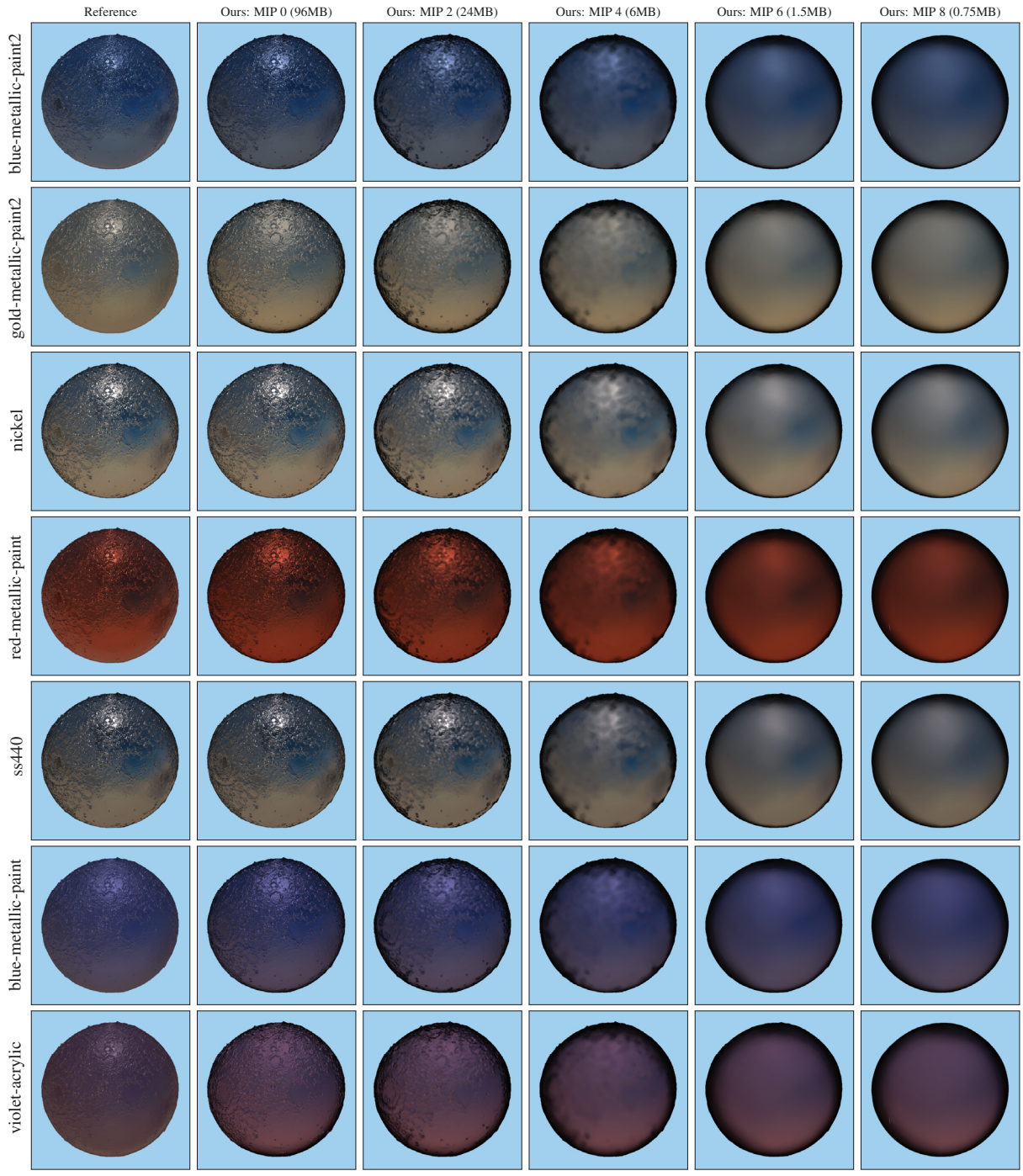


Figure 5.5 – Effect of our downsampling operator on rugged displaced surfaces with a few representative isotropic materials from the MERL database [MPBM03].

der to support such behaviors, Phong [Pho75] introduced an empirical model, known as the Phong BRDF, that would produce sharp isotropic BRDF lobes around the direction of ideal specular reflection. A few years later, Blinn [Bli77] suggested to reparameterize the Phong BRDF with the halfway direction, while also introducing microfacet theory to the computer graphics community; the newly parameterized Phong BRDF would later become the Blinn-Phong BRDF and the subject of various extensions to account for proper normalization [Lew94], and anisotropy [AS00]. It is worth mentioning that in this same 1977 article, Blinn introduced the Trowbridge-Reitz distribution [TR75], which would later be rediscovered by Walter et al. [WMLT07] under the name of GGX distribution [Bur12]. After the introduction of microfacet theory, other empirical models were introduced to account for complex scattering effects such as multiple scattering [ON94, Sch94b], anisotropy [PF90, War92, Sch94b], and/or polarization [HTSG91]. These models are empirical in the sense that some of their mathematical components do not have a geometric interpretation. In contrast, our microfacet BRDF model, which is built upon and extends the formulation of Walter et al. [WMLT07], has a clear geometric interpretation. Geometry is the key property that allows our appearance model to trivially support such features as anisotropy, noncentrality, and inversion in an actual implementation.

A Note on the Diffuse Term of Some Microfacet BRDFs The microfacet BRDF equation may sometimes come with an additional Lambertian term under the form

$$f_r = \frac{F \cdot D \cdot G}{4 \cdot \cos\theta_i \cdot \cos\theta_o} + k_d \left(\frac{1}{\pi} - \frac{F \cdot D \cdot G}{4 \cdot \cos\theta_i \cdot \cos\theta_o} \right). \tag{5.4}$$

Mathematically, this equation can be interpreted as a linear blend between our microfacet BRDF equation, i.e., Equation (2.1), and a diffuse BRDF, with parameter $k_d \in [0,1]$; notice how when $k_d = 0$ and $k_d = 1$, the BRDF acts respectively as Equation (2.1), and as an ideal Lambertian BRDF. Originally, the diffuse term was introduced by Torrance and Sparrow [TS67] as a purely empirical term for multiple and/or internal scattering effects occurring on the microsurface. Indeed, they write at the beginning of the formulation section of their paper: "The diffuse component may originate either from multiple reflections among the facets and/or from internal scattering". While incorporating a Lambertian term could have only resulted in improved BRDF fits, as it increases the degrees of freedom of the BRDF, this thesis neglects it completely. This is due to the lack of a clear geometric interpretation of Equation (5.4), which was the key property to introduce both the multiscale microfacet theory of Chapter 3 and the inverse microfacet theory of Chapter 4. As we shall discuss in the next section, deriving a microfacet BRDF model that accounts for the geometry scattering effects other than single bounces remains a challenging problem. Nevertheless, if such a model could be derived, it seems plausible that

it would inherit the properties of the current theory.

Appearance Filtering Techniques The most accurate form of appearance filtering techniques relies on precomputing the complete BRDF at any location and for any scale, and store it in a bidirectional texture function (BTF) [CMS87, BM93, MCT+05, WDR09]. While such methods are capable of better capturing the view-dependent effects exhibited by the displacement and the base BRDF of the surface than our technique, they carry two important limitations that strongly reduce their practicality. First, BTFs are very hard to manipulate for artists. Second, BTFs require dissuasive amounts of memory, as they typically store 6D functions, including 4D for the BRDF, and 2D for the spatial parameters. In the context of multiscale rendering, an additional dimension should be used, which overall adds considerable overhead on input/output operations. In contrast, our representation is lightweight and linear, and can be manipulated by artists to achieve a wide range of appearances.

Normal Map Filtering Techniques The problem of filtering arbitrary appearance was also studied with normal mapping. Convolution-based normal map filtering methods [Fou92, Tok05, HSRG07] use the fact that, at any scale, the BRDF is the convolution of a base BRDF and an NDF. Since masking, shadowing, and projection weighting are nonlinear functions of the view, light, and normal directions, incorporating these effects into the convolution is difficult. Tan et al. [TLQ+05, TLQ+08] use several Gaussian lobes with a masking-shadowing term, but omit the important view-dependent projection weighting effect and do not normalize their BRDF. Note that if we employ the normal map GAF in our BRDF evaluations, then our framework acts as a normal map filtering algorithm that supports arbitrary reflectance, thus generalizing some previous work in this area.

LEAN Mapping Olano and Baker [OB10, Bak11] introduced the lightweight representation we use in our framework in the context of normal map filtering. We adopted their representation for the two following reasons. First, their representation allows for linear filtering of the data in a manner that properly captures filtered reflectance. Second, their representation supports anisotropy and multilayer superposition. The main limitation of LEAN mapping in the context of our work comes from the non-physically based BRDF employed by the model: It lacks proper normalization, as the Jacobian of the reflection operation is missing, and is not capable of reproducing the important masking, shadowing, and projection weighting effects. Moreover, it is only designed for specular microfacets under point and directional lighting. This thesis showed how to account for occlusion effects, as well as arbitrary lighting conditions.

Multilobe Representations Some previous work uses multiple lobes to represent and store normal distributions [Fou92, TLQ+05, HSRG07, TLQ+08]. While multiple lobes can represent more complex distributions than what we can, their precomputations are problematic. Indeed, in order to make interpolation possible, every lobe in a texel must match the same lobe in the neighboring texels [TLQ+05, HSRG07, TLQ+08]. Solving this problem requires heavy nonlinear optimizations, and matching failures may result in visual artifacts when two nonmatching lobes are interpolated. In contrast, our single anisotropic lobe representation allows for lightweight memory storage, simple and fast precomputations, as well as no-failure interpolation.

5.3 Future Research

The main purpose of this thesis was to introduce a fully functional filtering framework for arbitrary complex displacement mapped surfaces. Our framework builds upon microfacet theory, which we augmented in the previous chapters, as well as upon the convenient properties of Beckmann distributions. While the presented results can offer significant improvements in terms of rendering quality and speed, there is still much room for improvement; the following paragraphs are dedicated to identifying current limitations and avenues for future work.

Beckmann Distributions Despite their very practical properties (linear representation, stability under convolution), Beckmann distributions are probably not the best parametric model to use for filtering for two reasons. First, their short (Gaussian) tails make them highly sensitive to outliers, which tends to yield overblurry highlights as filtering footprints increase. Second, they may fit measured data quite poorly, which introduces significant bias when fitting some materials. This effect is particularly visible for the *gold-metallic-paint2* and *violet-acrylic* materials in Figures 5.4 and 5.5. In order to improve on this second point, a mixture of Beckmann distributions can be employed; an extensive filtering comparison between a single Beckmann lobe and a mixture of eight lobes is available in the supplemental document of this thesis. Note however that mixture models require more arithmetic operations, as well as an additional random number to determine the lobe to sample. In contrast, GGX distributions [WMLT07] seem much more promising, because they fit measured data much better, and their heavy tails should (note that this is a conjecture) result in sharper highlights in the presence of outliers. Deriving a complete filtering framework for displaced surfaces of arbitrary reflectance using GGX distributions remains an open problem.

BRDFs Despite its effectiveness to faithfully describe many real-world appearances, microfacet BRDFs remain applicable to a limited set of materials because effects such as transmission, multibounce scattering, diffraction, or layers are not supported by the extended theory we have presented. While it has been shown that certain effects could be incorporated, e.g., transmission [WMLT07], diffraction [Sta99], or layers [JDJM14], most of the remaining effects are still open problems in the context of deriving a multiscale representation for matter, and should be addressed in order to make the theory even more general.

Curvature Our displaced surface filtering model and its derivations are based on the fundamental assumption that the displacement map is applied over a locally planar patch. In theory, this means that the method is not valid when the pixel footprint covers a curved macrosurface since the curvature must be filtered along with the displacement map. For instance, the claws of the *T-rex* model in Figure 3.11 are small, smooth, curved, and highly specular. As such, they exhibit aliasing that cannot be filtered by our scheme alone. While, in practice, we could deal with this problem by offsetting roughness heuristically due to curvature [DHI+13], the problem of properly filtering displacement maps along with large pieces of macrosurfaces is complex and remains an open problem [BN12].

Multiscale Representation for Matter Our filtering framework makes the per-pixel rendering complexity proportional to the number of displaced surfaces that project into a pixel, where previous algorithms would have been proportional to the total number of texels that project into a pixel. While this shift in complexity is undoubtedly a step forward towards more reasonable rendering times, the ability to handle multiple surfaces inside a pixel remains an open problem. Such scenarios arise for, e.g., trees and fur, whose geometry clearly can not be solely described with displaced surfaces at all scales. A first attempt to deal with such configurations was introduced concurrently to this thesis in the paper entitled "The SGGX Microflake Distribution" [HDCD15], which addresses the problem of downsampling directional data represented as volumes (rather than surfaces), using voxels. The paper shows how the GGX distribution can be extended to work in the spherical (rather than hemispherical) domain, and plugged into a microflake phase function (the 3D analogue of a microfacet BRDF [JAM+10]), to derive an efficient downsampling operator for 3D directional data. While downsampling voxelized directional data is an important contribution, it only constitutes a small brick in the elaboration of a more general multiscale representation for matter, which should be capable of accounting for correlations between the way matter is oriented, structured, and how it interacts with light. Fulfilling these goals requires to extend our current understanding of how light and matter interact, which should be worth the effort.

5.4 List of Publications

International Journals with Reviewing Committee

- [HDCD15] *The SGGX Microflake Distribution*.
  Eric Heitz, Jonathan Dupuy, Cyril Crassin, Carsten Dachsbacher.
  ACM SIGGRAPH 2015.

- [DHI+15] *Extracting Microfacet-based BRDF Parameters from Arbitrary Materials with Power Iterations*.
  Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Victor Ostromoukhov.
  Eurographics Symposium on Rendering EGSR 2015.

- [DHI+13] *Linear Efficient Antialiased Displacement and Reflectance Mapping*.
  Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, Victor Ostromoukhov.
  ACM SIGGRAPH Asia 2013.

- [DB12] *Real-time Animation and Rendering of Ocean Whitecaps*.
  Jonathan Dupuy, Eric Bruneton.
  ACM SIGGRAPH Asia 2012 (Technical Briefs).

Chapters in Books

- [DIP14] *Quadtrees on the GPU*.
  Jonathan Dupuy, Jean-Claude Iehl, Pierre Poulin.
  GPU Pro 5: Advanced Rendering Techniques.

Courses

- [HMD+14] *Physically based Shading in Theory and Practice*.
  Stephen Hill, Stephen McAuley, Jonathan Dupuy, Yoshiharu Gotanda, Eric Heitz, Naty Hoffman, Sébastien Lagarde, Anders Langlands, Ian Megibben, Farhez Rayani, Charles de Rousiers.
  ACM SIGGRAPH Course 2014.

Technical Reports

- [HD15] *Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation*.
  Eric Heitz, Jonathan Dupuy.

- [Dup14] *Phong Tessellation for Quads*.
  Jonathan Dupuy.

Bibliography

[AP07] Michael Ashikhmin and Simon Premože. Distribution-based BRDFs. *Technical Report, University of Utah*, 2007.

[AS00] Michael Ashikhmin and Peter Shirley. An Anisotropic Phong BRDF Model. *Journal of Graphics Tools*, 5(2):25–32, 2000.

[Bak11] Dan Baker. Spectacular Specular – LEAN and CLEAN Specular Highlights. In *Proc. Game Developer Conference 2011*, 2011.

[BBS02] Christophe Bourlier, Gerard Berginc, and Joseph Saillard. One- and Two-dimensional Shadowing Functions for Any Height and Slope Stationary Uncorrelated Surface in the Monostatic and Bistatic Configurations. *IEEE Trans. Antennas and Propagation*, 50(3):312–324, 2002.

[Bec65] Petr Beckmann. Shadowing of Random Rough Surfaces. *IEEE Trans. Antennas and Propagation*, 13(3):384–388, May 1965.

[Bli77] James F. Blinn. Models of Light Reflection for Computer Synthesized Pictures. In *Proc. SIGGRAPH '77*, pages 192–198. ACM, 1977.

[BM93] Barry G. Becker and Nelson L. Max. Smooth Transitions Between Bump Rendering Algorithms. In *Proc. SIGGRAPH '93*, pages 183–190, 1993.

[BN12] Eric Bruneton and Fabrice Neyret. A Survey of Non-linear Pre-filtering Methods for Efficient and Accurate Surface Shading. *IEEE Trans. Vis. Comput. Graph.*, 18(2):242–260, 2012.

[BSB00] Christophe Bourlier, Joseph Saillard, and Gerard Berginc. Effect of Correlation Between Shadowing and Shadowed Points on the Wagner and Smith Monostatic One-dimensional Shadowing Functions. *IEEE Trans. Antennas and Propagation*, 48(3):437–446, 2000.

[BSH12] Mahdi Bagher, Cyril Soler, and Nicolas Holzschuch. Accurate Fitting of Measured Reflectances using a Shifted Gamma Micro-facet Distribution. *Computer Graphics Forum*, 31(4):1509–1518, 2012.

[Bur12] Brent Burley. Physically-Based Shading at Disney. In *SIGGRAPH 2012 Courses: Practical physically-based shading in film and game production*, 2012.

[CCC87] Robert L. Cook, Loren Carpenter, and Edwin Catmull. The Reyes Image Rendering Architecture. In *Proc. SIGGRAPH '87*, pages 95–102. ACM, 1987.

[CK07] Mark Colbert and Jaroslav Křivánek. GPU-based importance sampling. In *GPU Gems 3*, chapter 20. Addison-Wesley, 2007.

[CMS87] Brian Cabral, Nelson Max, and Rebecca Springmeyer. Bidirectional Reflection Functions from Surface Bump Maps. In *Proc. SIGGRAPH '87*, pages 273–281, 1987.

[Coo84] Robert L. Cook. Shade trees. In *Proc. SIGGRAPH '84*, pages 223–231. ACM, 1984.

[CT82] Robert L. Cook and Kenneth E. Torrance. A Reflectance Model for Computer Graphics. *ACM Trans. on Graphics*, 1(1):7–24, January 1982.

[DB12] Jonathan Dupuy and Eric Bruneton. Real-time Animation and Rendering of Ocean Whitecaps. In *ACM SIGGRAPH Asia 2012 (Technical Briefs)*, pages 15:1–3. ACM, 2012.

[DHI+13] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, Fabrice Neyret, and Victor Ostromoukhov. Linear Efficient Antialiased Displacement and Reflectance Mapping. *ACM Trans. on Graphics*, 32(6):211:1–11, November 2013.

[DHI+15] Jonathan Dupuy, Eric Heitz, Jean-Claude Iehl, Pierre Poulin, and Victor Ostromoukhov. Extracting Microfacet-based BRDF Parameters from Arbitrary Materials with Power Iterations. *Computer Graphics Forum*, 34(4):21–30, 2015.

[DIP14] Jonathan Dupuy, Jean-Claude Iehl, and Pierre Poulin. *GPU Pro 5: Advanced Rendering Techniques*, chapter Quadtrees on the GPU. CRC Press, March 2014.

[Dup14] Jonathan Dupuy. Phong Tessellation for Quads. Technical report, 2014.

[Fou92] Alain Fournier. Normal Distribution Functions and Multiple Surfaces. In *Proc. Graphics Interface '92 Workshop on Local Illumination*, pages 45–52, 11 May 1992.

[Fro12] Georg Frobenius. Über Matrizen aus nicht Negativen Elementen, S. *B. Preuss. Akad. Wiss. Berlin*, pages 456–477, 1912.

[FV14] Jiri Filip and Radomir Vavra. Template-Based Sampling of Anisotropic BRDFs. *Computer Graphics Forum (Pacific Graphics)*, 33(7):91–99, 2014.

[GA02] John E. Gray and Stephen R. Addison. Characteristic Functions in Radar and Sonar. In *Proc. Southeastern Symposium on System Theory, 2002*, pages 31–35, 2002.

[HD14] Eric Heitz and Eugene D'Eon. Importance Sampling Microfacet-Based BSDFs using the Distribution of Visible Normals. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, EGSR, pages 103–112, 2014.

[HD15] Eric Heitz and Jonathan Dupuy. Implementing a Simple Anisotropic Rough Diffuse Material with Stochastic Evaluation. Technical report, 2015.

[HDCD15] Eric Heitz, Jonathan Dupuy, Cyril Crassin, and Carsten Dachsbacher. The SGGX Microflake Distribution. *ACM Trans. on Graphics*, 34(4):48:1–11, July 2015.

[Hec89] Paul S. Heckbert. Fundamentals of Texture Mapping and Image Warping. Technical report, 1989.

[Hei14] Eric Heitz. Understanding the Masking-Shadowing Function in Microfacet-Based BRDFs. *Journal of Computer Graphics Techniques (JCGT)*, 3(2):32–91, 2014.

[HMD+14] Stephen Hill, Stephen McAuley, Jonathan Dupuy, Yoshiharu Gotanda, Eric Heitz, Naty Hoffman, Sébastien Lagarde, Anders Langlands, Ian Megibben, Farhez Rayani, and Charles de Rousiers. Physically based Shading in Theory and Practice. In *SIGGRAPH 2014 Courses*, pages 23:1–8. ACM, 2014.

[HNPN13] Eric Heitz, Derek Nowrouzezahrai, Pierre Poulin, and Fabrice Neyret. Filtering Color Mapped Textures and Surfaces. In *Proc. Symp. on Interactive 3D Graphics and Games*. ACM, 2013.

[HSRG07] Charles Han, Bo Sun, Ravi Ramamoorthi, and Eitan Grinspun. Frequency Domain Normal Map Filtering. *ACM Trans. on Graphics*, 26(3):28:1–11, July 2007.

[HTSG91] Xiao D. He, Kenneth E. Torrance, François X. Sillion, and Donald P. Greenberg. A Comprehensive Physical Model for Light Reflection. In *Proc. SIGGRAPH '91*, pages 175–186. ACM, 1991.

[Jak10] Wenzel Jakob. Mitsuba Renderer, 2010. http://www.mitsuba-renderer.org.

[JAM+10] Wenzel Jakob, Adam Arbree, Jonathan T. Moon, Kavita Bala, and Steve Marschner. A Radiative Transfer Framework for Rendering Materials with Anisotropic Structure. In *ACM Trans. on Graphics*, SIGGRAPH '10, pages 53:1–13. ACM, 2010.

[JDJM14] Wenzel Jakob, Eugene D'Eon, Otto Jakob, and Steve Marschner. A Comprehensive Framework for Rendering Layered Materials. *ACM Trans. on Graphics (Proceedings of SIGGRAPH 2014)*, 33(4), 2014.

[JESG12] Jorge Jimenez, Jose I. Echevarria, Tiago Sousa, and Diego Gutierrez. SMAA: Enhanced Morphological Antialiasing. *Computer Graphics Forum (Proc. EUROGRAPHICS 2012)*, 31(2), 2012.

[Jon02] M. C. Jones. Student's Simplest Distribution. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 51(1):41–49, 2002.

[Kaj86] James T. Kajiya. The Rendering Equation. In *Proc. of ACM SIGGRAPH '86*, pages 143–150, 1986.

[KC08] Jaroslav Křivánek and Mark Colbert. Real-time Shading with Filtered Importance Sampling. *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, 27(4):1147–1154, 2008.

[KE09] Murat Kurt and Dave Edwards. A Survey of BRDF Models for Computer Graphics. *ACM SIGGRAPH Comput. Graph.*, 43(2):4:1–7, May 2009.

[Lew94] Robert R. Lewis. Making Shaders More Physically Plausible. *Computer Graphics Forum*, 13(2):109–120, 1994.

[LKYU12] Joakim Löw, Joel Kronander, Anders Ynnerman, and Jonas Unger. BRDF Models for Accurate and Efficient Rendering of Glossy Surfaces. *ACM Trans. on Graphics*, 31(1):9:1–14, 2012.

[Luk72] Eugene Lukacs. A Survey of the Theory of Characteristic Functions. *Advances in Applied Probability*, 4(1):1–38, 1972.

[MCT+05] Wan-Chun Ma, Sung-Hsiang Chao, Yu-Ting Tseng, Yung-Yu Chuang, Chun-Fa Chang, Bing-Yu Chen, and Ming Ouhyoung. Level-of-detail Representation of Bidirectional Texture Functions for Real-time Rendering. In *Proc. Symp. on Interactive 3D Graphics and Games*, pages 187–194. ACM, 2005.

[MPBM03] Wojciech Matusik, Hanspeter Pfister, Matt Brand, and Leonard McMillan. A Data-Driven Reflectance Model. *ACM Trans. on Graphics*, 22(3):759–769, 2003.

[MSUA12] Rosana Montes Soldado and Carlos Ureña Almagro. An Overview of BRDF Models. Technical Report LSI-2012-001, Universidad de Granada, 2012.

[NDM05] Addy Ngan, Frédo Durand, and Wojciech Matusik. Experimental Analysis of BRDF Models. In *Proc. Eurographics Symposium on Rendering*, EGSR, pages 117–226, 2005.

[NL13] Matthias Niessner and Charles Loop. Analytic Displacement Mapping Using Hardware Tessellation. *ACM Trans. on Graphics*, 32(3):26:1–9, July 2013.

[NLMD12] Matthias Niessner, Charles Loop, Mark Meyer, and Tony Derose. Feature-adaptive GPU Rendering of Catmull-Clark Subdivision Surfaces. *ACM Trans. on Graphics*, 31(1):6:1–11, February 2012.

[Nol13] John. P. Nolan. *Stable Distributions - Models for Heavy Tailed Data*. Birkhauser, Boston, 2013. In progress, Chapter 1 online at http://academic2.american.edu/∼jpnolan/stable/stable.html.

[NRH+77] Fred E. Nicodemus, Joseph C. Richmond, J.J. Hsia, W.I. Ginsberg, and T. Limperis. Geometrical Considerations and Nomenclature for Reflectance. *Applied Optics*, 9:1474–1475, 1977.

[OB10] Marc Olano and Dan Baker. LEAN Mapping. In *Proc. Symp. on Interactive 3D Graphics and Games*, pages 181–188. ACM, 2010.

[ON94] Michael Oren and Shree K. Nayar. Generalization of Lambert's Reflectance Model. In *Proc. SIGGRAPH '94*, pages 239–246, 1994.

[Per07] Oskar Perron. Zur Theorie der Matrices. *Mathematische Annalen*, 64(2):248–263, 1907.

[PF90] Pierre Poulin and Alain Fournier. A model for anisotropic reflection. In *Proc. SIGGRAPH '90*, pages 273–282. ACM, 1990.

[Pho75] Bui Tuong Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

[PM12] Andrei D Polyanin and Alexander V Manzhirov. *Handbook of Integral Equations*. CRC Press, 2012.

[Rus98] Szymon M Rusinkiewicz. A New Change of Variables for Efficient BRDF Representation. In *Rendering Techniques '98*, pages 11–22. Springer, 1998.

[Sch94a] Christophe Schlick. A Survey of Shading and Reflectance Models. *Computer Graphics Forum*, 13(2):121–131, 1994.

[Sch94b] Christophe Schlick. An Inexpensive BRDF Model for Physically-based Rendering. *Computer Graphics Forum*, 13:233–246, 1994.

[Sch07] Thorsten Schmidt. Coping with Copulas. *Copulas–From Theory to Application in Finance*, pages 3–34, 2007.

[Smi67] B. Smith. Geometrical Shadowing of a Random Rough Surface. *IEEE Trans. Antennas and Propagation*, 15(5):668–671, 1967.

[Sta99] Jos Stam. Diffraction Shaders. In *Proc. SIGGRAPH '99*, pages 101–110. ACM, 1999.

[TLQ+05] Ping Tan, Stephen Lin, Long Quan, Baining Guo, and Heung-Yeung Shum. Multiresolution Reflectance Filtering. In *Proc. Eurographics Symposium on Rendering*, EGSR'05, pages 111–116, 2005.

[TLQ+08] Ping Tan, Stephen Lin, Long Quan, Baining Guo, and Harry Shum. Filtering and Rendering of Resolution-Dependent Reflectance Models. *IEEE Trans. Vis. Comput. Graph.*, 14(2):412–425, 2008.

[Tok05] Michael Toksvig. Mipmapping Normal Maps. *Journal of Graphics, GPU, and Game Tools*, 10(3):65–71, 2005.

[TR75] T. S. Trowbridge and K. P. Reitz. Average Irregularity Representation of a Rough Surface for Ray Reflection. *J. Opt. Soc. Am.*, 65(5):531–536, May 1975.

[TS67] K. E. Torrance and E. M. Sparrow. Theory for Off-Specular Reflection From Roughened Surfaces. *J. Opt. Soc. Am.*, 57(9):1105–1112, September 1967.

[War92] Gregory J Ward. Measuring and Modeling Anisotropic Reflection. In *Proc. SIGGRAPH '92*, pages 265–272. ACM, 1992.

[WDR09] Hongzhi Wu, Julie Dorsey, and Holly Rushmeier. Characteristic Point Maps. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 28(4):1227–1236, 2009.

[Wil83] Lance Williams. Pyramidal Parametrics. In *Proc. SIGGRAPH '83*, pages 1–11. ACM, 1983.

[WMLT07] Bruce Walter, Stephen R. Marschner, Hongsong Li, and Kenneth E. Torrance. Microfacet Models for Refraction Through Rough Surfaces. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)*, EGSR, pages 195–206, 2007.

[WZT+08] Jiaping Wang, Shuang Zhao, Xin Tong, John Snyder, and Baining Guo. Modeling Anisotropic Surface Reflectance with Example-based Microfacet Synthesis. *ACM Trans. on Graphics*, 27(3):41:1–9, August 2008.