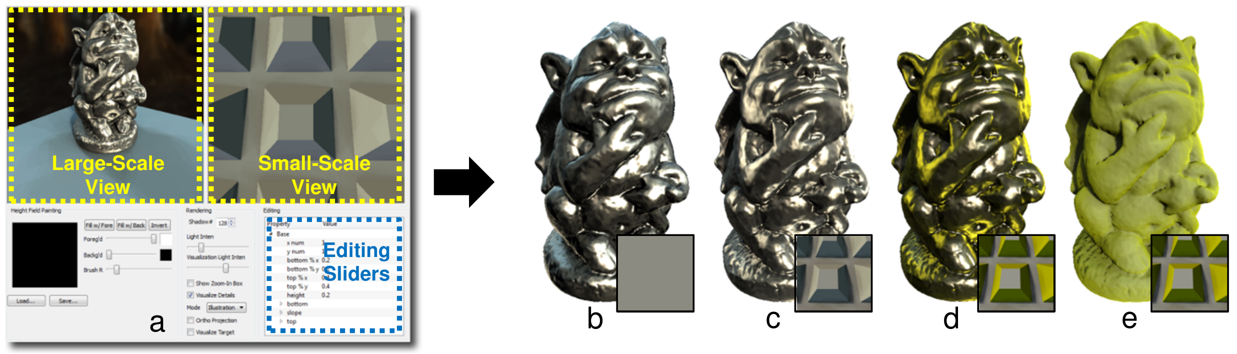
Physically-Based Interactive Bi-Scale Material Design

Hongzhi Wu Julie Dorsey Holly Rushmeier
Computer Graphics Group, Yale University



Shadow # 128
☐ Show Zoom-In Box
☑ Visualize Details
Mode Illustration
☐ Ortho Projection
☐ Visualize Target

Property	Value
Base	
x num	1
y num	1
bottom % x	0.2
bottom % y	0.2
top % x	0.4
top % y	0.4
height	0.2
bottom	
slope	
top	

Figure 1: *Bi-scale material design: large-scale appearance changes produced by editing of small-scale geometry (b & c), color (d) and BRDFs (e), using our interactive system (a), which quickly updates the appearance of the large-scale object after small-scale edits. The small-scale details (rendered with a Lambertian BRDF for a better visualization) are shown in the bottom right corner of (b-e). In (d), the color of the small-scale side faces is adjusted to yellow. In (e), the small-scale material is changed from a measured silver-metallic-paint2 BRDF to a Lambertian model.*

Abstract

We present the first physically-based interactive system to facilitate the appearance design at *different scales* consistently, through manipulations of both small-scale geometry and materials. The core of our system is a novel reflectance filtering algorithm, which rapidly computes the large-scale appearance from small-scale details, by exploiting the low-rank structures of the Bidirectional Visible Normal Distribution Function and pre-rotated BRDFs in the matrix formulation of our rendering problem. Our algorithm is three orders of magnitude faster than a ground-truth method. We demonstrate various editing results of different small-scale geometry with analytical and measured BRDFs. In addition, we show the applications of our system to physical realization of appearance, as well as modeling of real-world materials using very sparse measurements.

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: bi-scale, material editing, reflectance filtering, low-rank matrix

Links: DL PDF WEB VIDEO

1 Introduction

The appearance of materials can vary considerably when viewed at different scales. For example, individual grains of sand or fabric threads that are visible on close view merge into a material described by a single reflectance function when viewed from a distance. Physically speaking, the large-scale appearance is uniquely determined by averaging the look of small-scale details [Bruneton and Neyret 2011]. Therefore, it would be desirable to build an editing system for interactive appearance design at *different scales*, by manipulating small-scale structures. This could be useful in applications like building exterior design, where the user edits the looks of a building at different view distances.

Existing interactive material editing systems (e.g. [Ben-Artzi et al. 2006; Pellacini and Lawrence 2007]) focus on adjusting material appearance only at a *single scale*. On the other hand, previous work [Westin et al. 1992; Gondek et al. 1994], which computes realistic large-scale appearance by simulating light interactions in small-scale details, is too slow to provide interactive feedback. Although converting small-scale structures to large-scale appearance is essentially performing reflectance filtering, related techniques [Bruneton and Neyret 2011] are not suitable for our purpose, due to the lack of support for general geometry and materials [Han et al. 2007], or costly computational overhead [Wu et al. 2009].

This paper presents, to our knowledge, the first physically-based interactive bi-scale material editing system, which manipulates small-scale geometry and Bidirectional Reflectance Distribution Functions (BRDFs), to facilitate appearance design at two different scales consistently. The user can freely change both small-scale geometry and materials, then our system quickly computes the large-scale appearance to provide interactive visual feedback. As illustrated in Fig. 1, various small-scale edits can have dramatic effects on appearance. We achieve an acceleration rate of over 5000:1, when compared with a ground-truth method similar to [Westin et al. 1992], implemented on modern hardware. The key to the performance of our system is a novel reflectance filtering algorithm, which efficiently processes the *Bidirectional Visible Normal Distribution Function* (BVNDF) and pre-rotated BRDFs, derived from the changing small-scale details. We observe and exploit the low-rank structures in both quantities to accelerate the large-scale appearance computation, using Singular Value Decomposition (SVD) combined with the random projection method [Vempala 2004].

Our system can also guide the physical realization of bi-scale appearance, since the small-scale details are explicitly modeled. In addition, real-world materials can be approximately modeled by mimicking the small-scale details and then fine-tuning the large-scale appearance, using very few photographs. We believe that our system can be useful in many applications, including building exterior design, outdoor advertisements, physical realization of appearance as well as rapid material modeling in visual effects industry.

In summary, the major contributions of this paper are:

- We propose a novel form of interactive material design at two scales, through the manipulation of both small-scale geometry and materials. Our method is physically based so that the large-scale appearance is consistent with the small-scale details.
- We bridge the gap between interactive editing and previous work on bi-scale material modeling [Westin et al. 1992], by using a novel reflectance filtering algorithm, that rapidly computes large-scale appearance from changing small-scale details.
- Our system facilitates the design of physically realizable materials, since we explicitly model small-scale details.
- We propose a new modeling method for real-world materials with very sparse measurements using bi-scale constraints.

Terminology. We use the terms "small/large-scale" instead of "micro/milli-scale" as in some previous work, because we do not want to impose tight limits on the absolute sizes of both scales. Our method will work as long as the ratio between the large and the small scale is big, and the small scale is greater than the wavelength of light.

2 Previous Work

Interactive Material Editing. There has been much research effort devoted to interactive material editing in the past decade (e.g. [Ben-Artzi et al. 2006; Pellacini and Lawrence 2007]). Sophisticated methods are developed to achieve interactive or even real-time frame rates. However, no existing work considers the editing of appearance at different scales. In comparison, our method handles two scales of the appearance, and we allow editing to the small-scale geometry, in addition to the material reflectance, since both will affect the large-scale appearance. Our system can also be used to model complex materials from scratch by constructing small-scale details, which is not possible in other editing systems.

Predicting Appearance from Small-Scale Details. Microfacet-based BRDFs representing the macro-scale appearance [Cook and Torrance 1982; Oren and Nayar 1994; Ashikmin et al. 2000] are generated by modeling the statistical distributions of the orientations and shadowing of perfectly specular or Lambertian micro-scale facets, assuming decorrelation between the two factors. Although discussed in [Ashikmin et al. 2000], it is unknown how to extend the idea to efficiently handle facets of general materials. In addition, the focus of these papers is on analysis, not interactive material editing. On the other hand, since we explicitly model small-scale details, rather than use microfacet distributions, we no longer require the decorrelation assumption, and our bi-scale design can be physically realized directly. Ershov et al. [2004] edit an analytical BRDF for the large-scale appearance of a physically realizable paint. The method, however, does not generalize to arbitrary small-scale details.

The methods described in [Westin et al. 1992; Gondek et al. 1994] explicitly model the small-scale surface structures and then compute the large-scale appearance by an expensive simulation of light interactions. Heidrich et al. [2000] computes a large-scale BRDF, which takes indirect illumination into account, using pre-computed small-scale visibility. Recently, Zhao et al. [2011] produces highly realistic large-scale appearance of fabric by modeling detailed small-scale structures from micro CT imaging. The above methods are not suitable for editing, where the small-scale details are changed interactively.

Reflectance Filtering. Normal map filtering techniques [Tan et al. 2005; Han et al. 2007] do not consider shadowing and masking effects, and they only support limited types of analytical BRDFs. Wu et al. [2009] proposed a reflectance filtering algorithm which handles general geometry and materials. However, their method tightly couples the processing of the small-scale geometry with materials: whenever either one changes, an expensive precomputation must be performed, which can take as long as several hours.

Physical Realization of Appearance. Weyrich et al. [2009] fabricate a height-field, whose effective large-scale appearance approximates a custom reflectance function. Both Hasan et al. [2010] and Dong et al. [2010] extend the idea to incorporate custom subsurface scattering properties. All these methods require a costly inverse computation. There is no guarantee that a physical realization exists within the capability of fabrication equipment, which closely matches a designed appearance. By contrast, our system facilitates material design in the space of physically realizable structures; fabrication constraints can be incorporated. The edited large-scale appearance is guaranteed to be physically consistent with small-scale structures. Furthermore, our system allows the user to interactively explore the space of all possible materials under the constraints of the small-scale details, which is not possible in previous work. Our method can be viewed as an alternative approach for appearance fabrication, complementary to existing work.

Symbol	Description
A	a surface patch
Ω	the upper hemisphere
S^2	the domain of normals
n	a normal
$V(\boldsymbol{\omega})$	visibility along direction $\boldsymbol{\omega}$ at point x
$a_v(\boldsymbol{\omega}_o)$	visible projected area of A along $\boldsymbol{\omega}_o$
$L(\boldsymbol{\omega}_o)$	reflected radiance along $\boldsymbol{\omega}_o$
$\overline{L}(\boldsymbol{\omega}_o)$	average reflected radiance along $\boldsymbol{\omega}_o$
$f_r(n, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$	a small-scale BRDF
$f(n, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$	a cosine-weighted rotated BRDF
$\gamma(n, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$	BVNDF
$\overline{f}_r$	an effective large-scale BRDF
ΔA	the area of a finite-sized facet
N	a matrix of discretized BVNDF
M	a matrix of pre-rotated BRDFs
U_N, V_N^T or U_M, V_M^T	SVD factorization of N or M
j	an index for discretized n
k	an index for the discretized bidirectional domain $\Omega \times \Omega$
s	an index for discretized A

Table 1: *Summary of the notation used in the paper.*

3 The Rendering Pipeline

Providing interactive visual feedback is critical to a material editing system [Kerr and Pellacini 2010]. Therefore, we introduce a high-performance rendering pipeline, to efficiently convert edited small-scale details into effective large-scale appearance. The pipeline takes in small-scale details as input, discretizes the geometry into

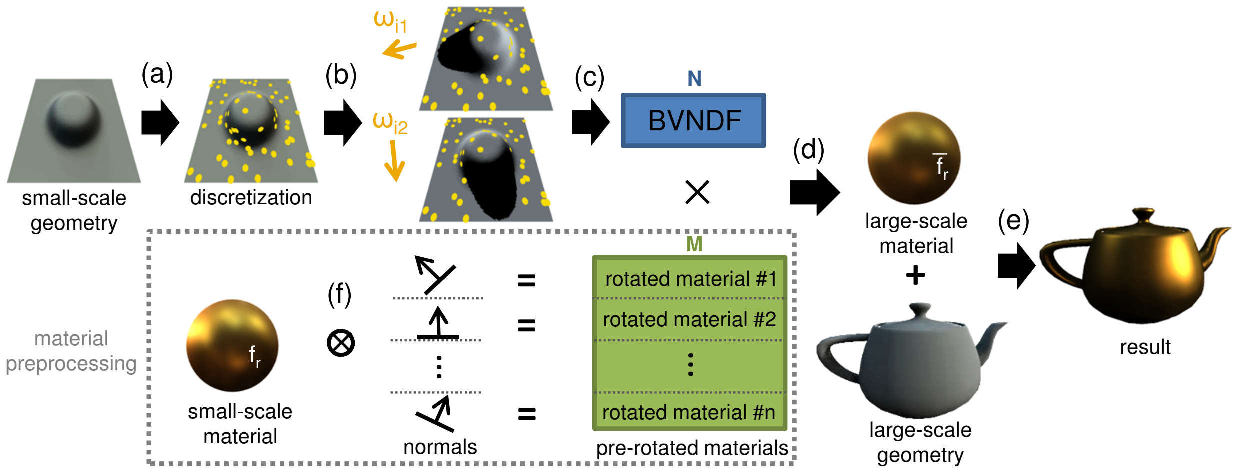


Figure 2: *Our rendering pipeline: starting from a small-scale structure, we discretize its geometry into small facets, the centers of which are shown as yellow dots above (a); next, we sample the visibility of these facets for different lighting directions (b); we further convert the spatial visibility information into a BVNDF (c), and combine with pre-rotated BRDFs (f) to get the effective large-scale material $\overline{f}_r$ (d); finally, we render a large-scale object using $\overline{f}_r$ (e).*

facet samples (Sec. 3.2), computes their directional visibility information (Sec. 3.3), and then converts it into a BVNDF (Sec. 3.4). Combining the BVNDF with rotated small-scale BRDFs obtained in a precomputation process (Sec. 3.5), we get the effective BRDF and perform the final large-scale appearance rendering (Sec. 3.6). Please see Fig. 2 for an illustration.

3.1 Preliminaries

We first derive the formulation for the effective large-scale appearance from small-scale details, using the rendering equation [Kajiya 1986]. We assume isotropic small-scale BRDFs and direct illumination throughout the paper. Extensions to more general cases would be an interesting future avenue. Note that the first few steps of the derivation are common in related literature [Ashikmin et al. 2000; Wu et al. 2009]. We include them here for completeness.

We start from the reflected radiance L at point x along a view direction $\boldsymbol{\omega}_o$:

$$L(\boldsymbol{x},\boldsymbol{\omega}_o) = \int_\Omega L_i(\boldsymbol{x},\boldsymbol{\omega}_i)V(\boldsymbol{x},\boldsymbol{\omega}_i)f_r(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)(n\cdot\boldsymbol{\omega}_i)d\boldsymbol{\omega}_i.$$

Here Ω is the upper hemisphere, $\boldsymbol{\omega}_i$ is the lighting direction, L_i is the incident radiance, and V is the visibility function, which returns 1 if $\boldsymbol{x}$ is not blocked along a direction and 0 otherwise. $\boldsymbol{n}$ is the normal at $\boldsymbol{x}$, and f_r is the BRDF. Note that our definition of f_r is slightly different from the standard convention, as we include n as the input of f_r to represent the rotated BRDF: $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ are transformed from the global coordinate system into the local frame defined by n when evaluating f_r. In addition, $(\cdot)$ is the cosine of the angle between the two vectors, which is clamped to zero if it is negative.

If we look at a surface patch A from a distance, the spatially averaged reflected radiance $\overline{L}$ along $\boldsymbol{\omega}_o$ is the average of all reflected radiance from visible parts of A (see Fig. 3 for an illustration):

$$\begin{aligned}\overline{L}(\boldsymbol{\omega}_o) =& \frac{1}{a_v(\boldsymbol{\omega}_o)}\int_{A_v(\boldsymbol{\omega}_o)} L(\boldsymbol{\omega}_o)dA_v(\boldsymbol{\omega}_o)\\ =& \frac{1}{a_v(\boldsymbol{\omega}_o)}\int_{A_v(\boldsymbol{\omega}_o)}\int_\Omega L_i(\boldsymbol{\omega}_i)V(\boldsymbol{\omega}_i)\\ & f_r(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)(\boldsymbol{n}\cdot\boldsymbol{\omega}_i)\,d\boldsymbol{\omega}_i\,dA_v(\boldsymbol{\omega}_o),\end{aligned} \tag{1}$$

where $a_v(\boldsymbol{\omega}_o)$ is the visible projected area of A along $\boldsymbol{\omega}_o$, and $A_v(\boldsymbol{\omega}_o)$ is the visible part of A, when viewed from $\boldsymbol{\omega}_o$. Note that we drop $\boldsymbol{x}$ from here for clarity.

One may think of A as being composed of infinite number of infinitesimal planar facets. dA is the area of one facet, then $dA_v(\boldsymbol{\omega}_o)$ is just the visible projected area of the facet along $\boldsymbol{\omega}_o$:

$$dA_v(\boldsymbol{\omega}_o) = V(\boldsymbol{\omega}_o)(\boldsymbol{n}\cdot\boldsymbol{\omega}_o)dA. \tag{2}$$

Then $a_v(\boldsymbol{\omega}_o)$ can be computed as:

$$a_v(\boldsymbol{\omega}_o) = \int dA_v(\boldsymbol{\omega}_o) = \int V(\boldsymbol{\omega}_o)(\boldsymbol{n}\cdot\boldsymbol{\omega}_o)dA. \tag{3}$$

Next, we define the cosine-weighted rotated BRDF as:

$$f(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) = f_r(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)(\boldsymbol{n}\cdot\boldsymbol{\omega}_i)(\boldsymbol{n}\cdot\boldsymbol{\omega}_o). \tag{4}$$

Substituting Eq. 2 and 4 back into Eq. 1 gives

$$\begin{aligned}\overline{L}(\boldsymbol{\omega}_o) &= \frac{1}{a_v(\boldsymbol{\omega}_o)}\int_\Omega L_i(\boldsymbol{\omega}_i)\int_A f(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)V(\boldsymbol{\omega}_i)V(\boldsymbol{\omega}_o)dAd\boldsymbol{\omega}_i\\ &= \int_\Omega L_i(\boldsymbol{\omega}_i)\left(\frac{1}{a_v(\boldsymbol{\omega}_o)}\int_{S^2} f(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\gamma(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)d\boldsymbol{n}\right)d\boldsymbol{\omega}_i\\ &= \int_\Omega L_i(\boldsymbol{\omega}_i)\overline{f}_r(\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)d\boldsymbol{\omega}_i,\end{aligned} \tag{5}$$

where

$$\gamma(\hat{\boldsymbol{n}},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) = \int_A V(\boldsymbol{\omega}_i)V(\boldsymbol{\omega}_o)\delta(\hat{\boldsymbol{n}}-\boldsymbol{n})dA, \tag{6}$$

$$\boxed{\overline{f}_r(\boldsymbol{\omega}_i,\boldsymbol{\omega}_o) = \frac{1}{a_v(\boldsymbol{\omega}_o)}\int_{S^2} f(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)\gamma(\boldsymbol{n},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)d\boldsymbol{n}.} \tag{7}$$

Note that δ is the Dirac function, S^2 is the surface of unit sphere since f_r with $\boldsymbol{n}$ pointing to the lower hemisphere may also contribute to the reflected radiance.

In Eq. 6, we have defined $\gamma(\hat{\boldsymbol{n}},\boldsymbol{\omega}_i,\boldsymbol{\omega}_o)$, which is the BVNDF that describes the visible normal distribution of A when viewed from $\boldsymbol{\omega}_o$ under a directional light from $\boldsymbol{\omega}_i$. Here we use $\hat{\boldsymbol{n}}$ as the parameter

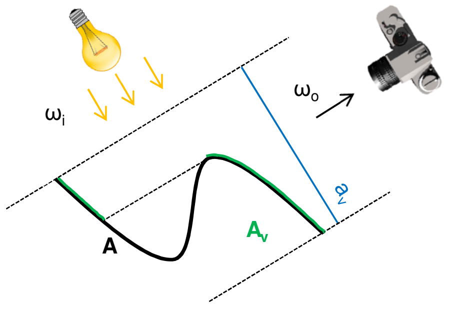


Figure 3: *An illustration of A, A_v and a_v used in our derivation (a reproduction of Fig. 3 in [Wu et al. 2009]).*

of γ to distinguish from $\boldsymbol{n}$, which is the normal associated with dA. Han et al. [2007] define a similar term called Normal Distribution Function. The key difference is that theirs does not include the bidirectional visibility, thus they do not take into account important shadowing and masking effects that are non-negligible for general geometry. Please refer to [Wu et al. 2009] for more details on the impact of shadowing and masking effects over appearance.

In Eq. 7, we reach the final goal of our derivation, the equation for computing effective large-scale reflectance distribution function, $\overline{f}_r$. Unfortunately, brute-force computation of Eq. 7 could be prohibitively expensive, because we need to compute γ by testing visibility for all facets of A (Eq. 6), and rotate f_r for every $\boldsymbol{n}$ on A, both of which are non-trivial tasks.

We further discretize Eq. 7 in order to compute numerically:

$$\overline{f}_r(\boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k}) \approx \frac{\sum_j f(\boldsymbol{n}_j, \boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k})\gamma(\boldsymbol{n}_j, \boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k})}{a_v(\boldsymbol{\omega}_{o,k})} = \frac{1}{a_v(\boldsymbol{\omega}_{o,k})}\sum_j N_{kj} M_{jk}^T, \tag{8}$$

where $(\boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k})$ is a discretization of the bidirectional domain $\Omega \times \Omega$, and $\boldsymbol{n}_j$ is a discretization of the unit sphere S^2. N represents the BVNDF and M stores rotated small-scale BRDFs for different $\boldsymbol{n}_j$, as follows:

$$N_{kj} = \gamma(\boldsymbol{n}_j, \boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k}) \quad \text{and} \quad M_{jk}^T = f(\boldsymbol{n}_j, \boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k}). \tag{9}$$

In the following sections, we will focus on efficiently computing and representing N and M, as well as quickly computing Eq. 8, which are most crucial to the performance of our editing system.

3.2 Processing Geometry

Given edited small-scale details as input, the first step in our pipeline is to randomly sample planar facets (see Sec. 3.1) on the geometry for subsequent processing. Without loss of generality, we assume that the input geometry is a triangular mesh. Our sampling method first randomly picks a triangle in the mesh with a probability proportional to its area. Next, we uniformly sample a point in this triangle as the center of a facet. The associated normal is then computed by interpolating the normals from three vertices of the triangle using barycentric coordinates. After all facet centers and normals are generated, we estimate the area of each facet, by sampling many points over the mesh, assigning each point to the closest facet center using a kd-tree, and computing the area of each facet proportional to the number of points that are closest to its center. We perform the sampling in the object space to avoid resampling when the view direction changes as in the post-projection image space. The sampled facets are a good approximation to the original geometry as long as the number of samples is sufficiently large.

3.3 Processing Visibility

After we obtained facets approximating the original small-scale geometry, we need to determine their visibility V along different directions to facilitate the computation of $\overline{f}_r$ (Sec. 3.1). Similar to sampling facets, the spatial visibility also changes with the edited geometry, so it is not suitable to use a precomputation-based method. We introduce a shadow-mapping-based algorithm that is not only robust to handle the variety of small-scale geometry in all our experiments, but also efficient to run on GPU. The detailed steps are as follows:

1. Similar to conventional shadow mapping, generate depth maps for different directions by rendering the triangular mesh representing the original geometry.
2. Pack the coordinates of the centers of all facets into a regular RGB texture.
3. Render the texture as a screen-space quad using a special pixel shader, which fetches the coordinates and performs conventional shadow determination by comparing with the depth maps.
4. Read back the visibility results to CPU from GPU.

We have found that relatively low sampling rate of directions is sufficient in all our experiments. Note that we duplicate the small-scale geometry around itself when generating depth maps to compute the visibility as if the geometry is tiled over the large-scale surface.

Once we obtained the visibility information V, the visible area function $a_v(\boldsymbol{\omega}_o)$ in $\overline{f}_r$ can be computed by discretizing the integration in Eq. 3 into a summation as:

$$a_v(\boldsymbol{\omega}_o) \approx \sum_s V(\boldsymbol{\omega}_o)(\boldsymbol{n}_s \cdot \boldsymbol{\omega}_o)\Delta A_s, \tag{10}$$

where ΔA_s is the estimated facet area (Sec. 3.2).

3.4 Processing BVNDF

In this section, we describe the computation of N from previous visibility results V. Recall from Eq. 9 that N is essentially a tabulation of γ, the BVNDF. Each row of N stores the distribution of normals of parts of A that are visible from both $\boldsymbol{\omega}_{i,k}$ and $\boldsymbol{\omega}_{o,k}$. We compute N using the following equation, derived by discretizing Eq. 6 and then substituting into Eq. 9:

$$N_{kj} = \sum_s V(\boldsymbol{\omega}_{i,k})V(\boldsymbol{\omega}_{o,k})\tilde{\delta}(\boldsymbol{n}_s, \boldsymbol{n}_j)\Delta A_s. \tag{11}$$

Here $\tilde{\delta}(\boldsymbol{n}_s, \boldsymbol{n}_j)$ is a discretized δ function, which returns 1 if $\boldsymbol{n}_s$ is closest to $\boldsymbol{n}_j$ among all other discretized normals, and 0 otherwise.

One key observation in our initial experiments is that N is of low-rank. This suggests applying SVD to keep only a low-rank approximation for acceleration in the computation of $\overline{f}_r$ in Eq. 8. However, directly computing SVD on N takes relatively long time, which outweighs its potential speedup. Thanks to the low-rank structure, we can perform random projection to accelerate the SVD process and still use the low-rank approximation for later stages. The idea of random projection is that, as long as a matrix is of low-rank, one could perform SVD on its reduced-sized version more quickly without much loss in quality. Note that we apply $U_N \Leftarrow U_N \Sigma_N$ after SVD so that $N = U_N V_N^T$ to save the storage for Σ_N and the

```
$P$ : a $m \times n$ matrix; $R$ : a $m \times l$ matrix, each element is drawn i.i.d from $N(0, 1)$.
1. Compute $Q = \frac{1}{\sqrt{l}} R^T P$
2. Compute the SVD of $Q$ as $Q = U_Q \Sigma_Q V_Q^T$
3. Approximate $P$ as $P \approx (P V_Q) V_Q^T$
```

Table 2: *Pseudo-code of random-projection accelerated SVD. Pleaser refer to [Vempala 2004] for more details.*

time to reconstruct N in rendering. We list the pseudo-code of the random-projection accelerated SVD algorithm in Tab. 2. It is interesting to note that in our experiments, the bidirectional visibility in the spatial domain has much higher rank than its counter part in the S^2 domain of normals. We believe the main reason is that by transforming the bidirectional visibility into the S^2 domain, we discard the spatial information in the matrix, and thus essentially decrease its rank.

3.5 Processing Materials

The only quantity in Eq. 8 that we have not calculated is M. Observe that each column of M represents a tabulation of the cosine-weighted f_r (Eq. 4), whose local frame is aligned with n_j. In practice, we have found it very challenging to efficiently process M: the matrix is computed from costly evaluations of f_r rotated to different n, at different $(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$, whose total number could be huge for specular materials (e.g. 83GB using double-precision floats for a Cook-Torrance model with $\alpha = 0.2$). Moreover, the large size of M even makes it impossible to store the whole matrix in memory.

Fortunately, we have also observed the low-rank property in M. Therefore, we are able to apply the aforementioned random-projection accelerated SVD on M. Note that here using random-projection not only accelerates the computation, more importantly, it makes the SVD *tractable*, as to our knowledge, no current workstation can perform standard SVD on a matrix as large as 83GB. After SVD, we have $M = U_M V_M^T$, where V_M^T contains basis cosine-weighted rotated f_r, and U_M stores the coefficients for expressing the original rotated f_r using the basis functions in V_M^T.

Even with the help of random projection, the SVD algorithm is still not fast enough for interactive editing, as the entire matrix M must be generated from f_r first, and the random projection takes time to reduce the huge-sized matrix. Nevertheless, we can compress M for each material that will be used in the editing in an offline precomputation process. During editing, we just need U_M and V_M^T, whose total size fits in memory thanks to the low-rank property, for computing $\overline{f}_r$, and we do not need to evaluate f_r anymore. We believe our compression results could be improved using more involved methods such as Clustered Principal Components Analysis [Sloan et al. 2003]. We do not use non-linear basis functions, such as spherical Gaussians by Green et al. [2006], because the coherence of different rotated small-scale BRDFs cannot be exploited.

3.6 Rendering the Large-Scale Appearance

Finally, we are ready to efficiently compute $\overline{f}_r$ as follows. After substituting the SVD results of N and M into Eq. 8, we get:

$$\overline{f}_r(\boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k}) \approx \frac{1}{a_v(\boldsymbol{\omega}_{o,k})} \sum_j (U_N)_{kj} (V_N^T V_M U_M^T)_{jk}. \quad (12)$$

Compared to Eq. 8, here we are also performing a dot product of corresponding row/column vectors from two matrices. But the key difference is that by exploiting the low-rank structure in both N and M, we not only save the footprint by storing U_N, V_N^T, V_M and U_M^T instead of N and M, but also accelerate the computation of Eq. 12: we perform a dot product of two vectors of a length equals the number of basis BVNDFs in V_N^T (less than 35 in our experiments), which is usually much smaller than the number of n samples in the original Eq. 9, especially for specular materials (e.g. 6,442 for the Cook-Torrance model with $\alpha = 0.2$). Note that due to the low-rank property, the matrix multiplication $V_N^T V_M U_M^T$ can be quickly performed without slowing down the pipeline too much.

After $\overline{f}_r$ is computed, we use it as an effective data-driven BRDF to render large-scale objects in a deferred shading fashion: we render on GPU the normals, tangents and visibility with respect to light sources of large-scale objects into buffers, then read them back and evaluate Eq. 5 on CPU for the final rendering result.

3.7 Implementation Details

Determining Sampling Rates. We use paraboloid maps [Brabec et al. 2002] to map n, $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$ onto a regular 2D image. Then we map non-empty pixels in the 2D image into a 1D domain. We do not adopt the half-angle parameterization [Rusinkiewicz 1998] for M, since it changes with the normal, and thus cannot be shared by all rotated versions of the same BRDF. We employ a binary-search like method to determine the sampling rate of the bidirectional domain of M (i.e. the column number): we start with an interval of minimum and maximum possible sampling rates; then the interval is reduced to half each time, based on whether the original rotated BRDFs can be represented above a specified quality, using the midpoint of the interval as the sampling rate; the process stops when there is only one number in the interval, which is selected as the optimal sampling rate. We apply the same procedure to compute the optimal row number of M. For N, we manually specify the number of facet samples used in Sec. 3.2, which should be large enough to cover all important visual features.

Computing and Using N and M. To alleviate the paraboloid mapping distortion, we scale N and M with proper weights before performing SVD (Sec. 3.4 and 3.5) and scale the results back afterwards. Before each SVD, we perform mean subtraction according to the standard PCA procedure. We omit the mean in all related formulas throughout the paper for clarity. When precomputing M, sub-matrices of M and R are sampled/computed on the fly due to their tremendous size. We use the same seed for our random number generator to ensure that we get the same R all the time. Additionally, we exploit the reciprocity in the bidirectional domain for both N and M to avoid redundant computation and storage. In rendering, linear interpolation is applied to N and M to avoid blockiness in the result. We use Intel MKL and SSE instructions for efficient processing of matrices in our pipeline.

Color Channels. We extend our previous algorithm on single-channel data to handle RGB channels. For M, before precomputation, we first tabulate the current BRDF to form a matrix, whose column vector represents the reflectance values for three color channels. Next, we apply SVD to this matrix and keep the first principal direction $\boldsymbol{d}$. For each evaluation of f_r, we project the result along $\boldsymbol{d}$ to transform into a 1D space. During rendering, after we obtained $\overline{f}_r$ in the transformed 1D space, we multiply it with $\boldsymbol{d}$ to get the result back in RGB space. The above procedure is similar to [Heeger and Bergen 1995].

In addition, we would like to allow interactive editing to the color/intensity of f_r across different parts of the small-scale geometry. To efficiently implement this feature, we introduce spatially-varying color weights $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_r, \boldsymbol{\alpha}_g, \boldsymbol{\alpha}_b)$ to the small-scale details. When constructing N from facets (Sec. 3.4), we use $({\boldsymbol{\alpha}_r}_s \Delta A_s, {\boldsymbol{\alpha}_g}_s \Delta A_s, {\boldsymbol{\alpha}_b}_s \Delta A_s)$ instead of ΔA_s as the weights for the bidirectional visibility vector. This results in an N that is three times the original size described previously, due to the extension to RGB channels. The rest of the pipeline remains unchanged.

Multiple Materials. We separately handle multiple materials in the small-scale details. After the visibility computation (Sec. 3.2), we group facets according to their materials, process each group through the rest of the pipeline, and sum the results as the overall effective reflectance value. Note that for facets with different $\boldsymbol{\alpha}$ but the same f_r, we group them together since they share the same underlying BRDF (i.e. M).

4 Bi-Scale Material Editing

Based on the rendering pipeline introduced in the previous section, we develop our bi-scale material editor, as shown in Fig. 1-a. We implement a simple slider-based interface, which allows manipulations to both small-scale geometry and materials. The small-scale geometry is created either by setting the parameters of a procedural model, or by specifying a height-field. The procedural models we use for examples in this paper include pyramids, grooves, rods and woven threads. The shape or the distribution of shapes can be adjusted using pre-defined parameters. The height-field can be edited with a 2D image painting interface. We would like to emphasize that, our system is not limited to handle the types of geometry listed above. In fact, any geometric editing method that outputs a triangular mesh can be used for the small-scale geometry. Pre-rotated small-scale materials (Sec. 3.5) are loaded at the start of our system. Different materials can be assigned to different parts of the small-scale geometry. Our system also allows changing the spatially varying color weights of materials.

4.1 Small-Scale Material Editing

We support small-scale material editing similar to that in existing single-scale material editing systems. One simple edit is to change the color/intensity of materials by adjusting the color weights (introduced in Sec. 3.7). Fig. 4 shows an example. In Fig. 4-a, a cup is coated with a grooved material with a measured *pearl-paint* BRDF. In Fig. 4-b, the color of the bottom of the small-scale grooves is changed to green, while the sides are made yellow, resulting in a bi-color cup. Our system also allows changing the small-scale BRDFs. As Fig. 4-c shows, the small-scale material of the bi-color cup is further changed from *pearl-paint* to an analytical Ward BRDF ($\alpha = 0.3$), which produces a more specular large-scale appearance.

Note that we do not support direct editing on existing precomputed representations of materials (e.g. freely changing the specular exponent of a Blinn-Phong model), due to the cost for tabulating and processing M. To bypass this limit, we discretize the range of possible parameters and precompute corresponding variations of one material (e.g. discretize the specular exponent of a Blinn-Phong BRDF). In runtime, we can change to one precomputed variation of the material to achieve an equivalent edit.

4.2 Small-Scale Geometry Editing

We demonstrate small-scale geometry editing on procedural models and height-fields. For a procedural model, we directly manipulate its parameters and regenerate the corresponding triangular mesh. In Fig. 5, we achieve anisotropic metallic appearance (5-b) from isotropic one (5-a) by raising the height of a pyramid-like structure. We use an isotropic Cook-Torrance model ($\alpha = 0.2$) as the small-scale BRDF. To non-uniformly change the anisotropic reflection along different directions, we alter the ratio of the small-scale

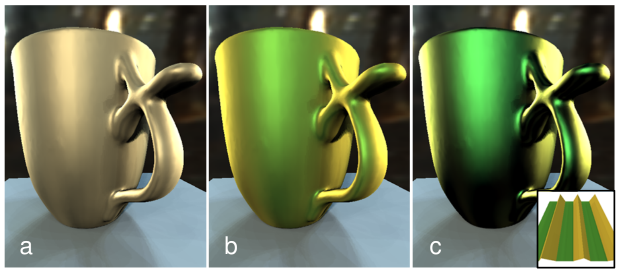


Figure 4: *Small-scale material editing. (a): a grooved cup with a measured pearl-paint BRDF. (b): the color of the bottom of the grooves changed to green, and the rest to yellow. (c): the small-scale material changed to a Ward BRDF. Note that small-scale structures are visualized in the bottom right corner.*

side faces, as shown in Fig. 5-c. We can also achieve unusual 5-way anisotropic reflection by changing the small-scale geometry to a pentagonal pyramid (Fig. 5-d). To our knowledge, no existing BRDF model can represent this kind of reflectance with physical plausibility, unless the small-scale geometry is explicitly modeled, like in our system.

We mimic the appearance of velvet in Fig. 6 using small-scale rods, and a measured *silver-paint* BRDF, whose color weights are changed to red (Sec. 3.7). In our system, we can manipulate the randomness that the rods are perturbed, or orient the rods toward different directions, to create various interesting looks. The consistency between the small and large-scale appearance can be seen in the zoom-in sequence at the start of our accompanying video.

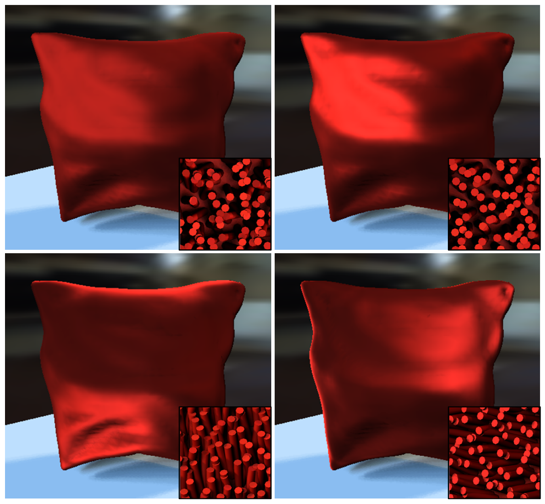


Figure 6: *Velvet-like appearance using small-scale rods and a measured silver-paint BRDF. Top left: randomly-spread rods. Top right: less randomly-spread, more concentrated rods. Bottom left: random rods oriented upwards the sky. Bottom right: random rods oriented to the left.*

Fig. 7-a & b show examples of a two-hue shot silk appearance, created using a small-scale woven structure with a measured *blue-metallic-paint* BRDF. The warp and weft threads are assigned different color weights. We adjust the sizes of the threads to create different looks. This two hue effect cannot be achieved in material editors using standard BRDF models. To demonstrate this, we numerically compute the best fit of a Lambertian plus a Cook-Torrance model to the materials designed in our system, and the

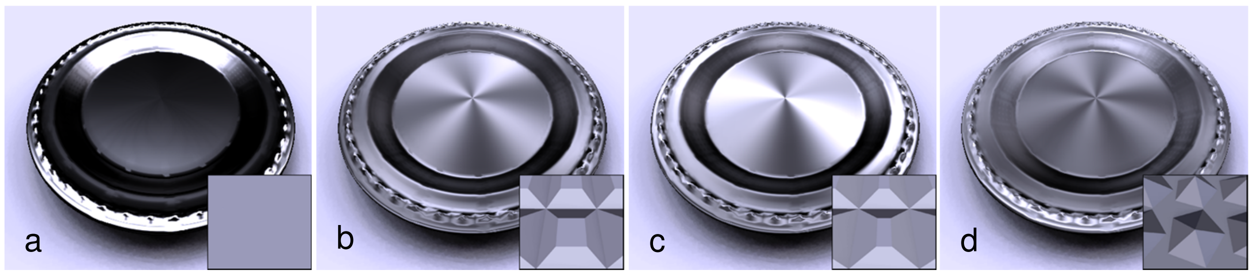


Figure 5: *Editing anisotropic metallic appearance by altering a small-scale pyramid-like geometry. From left to right: isotropic reflection, anisotropic reflection, varying anisotropic reflection intensities, and unusual anisotropic reflection along five directions.*

results are shown in Fig. 7-c & d. It is clear that the original two-hue appearance, which is typical for this type of fabric, is missing.

In addition to procedural geometry, we support small-scale geometry generated from a height-field image. The user can load an existing image, or directly paint/modify the height-field using our user interface. Fig. 8 shows large-scale appearance generated from a variety of height-fields. Please see our video for an interactive height-field editing demo.

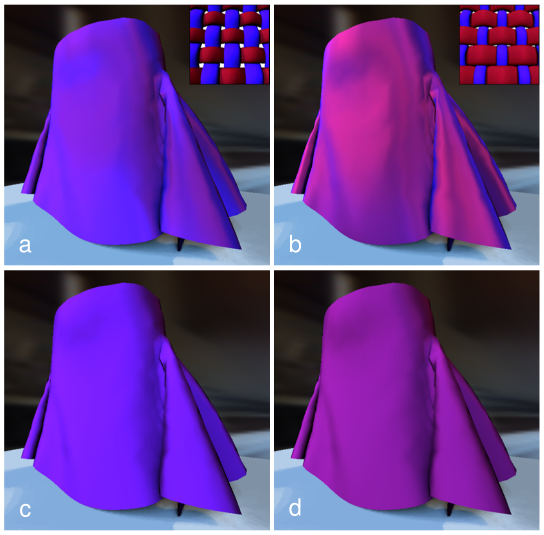


Figure 7: *Cloth-like appearance using a small-scale woven structure. Top row: the sizes of the threads are adjusted to create a different look. Bottom row: fitting the materials in the top row using a Lambertian plus a Cook-Torrance model. The original two-hue characteristic is completely lost.*

5 Physical Realization and Modeling of Materials

In this section, we introduce two additional applications that closely link our system with reality: physical realization of designed appearance, and modeling of real-world materials using bi-scale constraints. First of all, we can realize the designed bi-scale appearance in the real-world by physically constructing the small-scale details, since they are explicitly modeled in our system. Unlike previous work (e.g. [Weyrich et al. 2009]), we do not need a sophisticated optimization process to figure out the exact small-scale details. Fig. 9 illustrates an example. We would like to realize two types of bi-scale appearance, designed using small-scale grooves or a planar structure with green and yellow Lambertian materials (similar to Fig. 4). To do that, we first flatten the small-scale geometry onto a 2D plane and print out the result using a color printer (Fig. 9-a), while keeping the relative sizes among different facets. Here we assume the BRDF of our printed pattern is also Lambertian. Next, we fold the printed pattern into grooves (Fig. 9-b), which essentially becomes the small-scale details, and glue them onto a cylinder served as the large-scale geometry. The cylinder is pre-wrapped with a paper of printed markers to help keep the spacing beneath grooves in accordance with the design in our system. Photographs of the cylinders taken using a calibrated camera and a directional light are shown in Fig. 9-b,c,e & f. Note that due to the limitation of our crafting precision, we are unable to manually make very small structures compared to the size of the cylinder. Therefore, the small-scale structures are still visible in the distant-view photographs (Fig. 9-c & f). Nevertheless, when comparing with our rendering results using the same camera and lighting parameters (Fig. 9-d & g), the major visual features of the large-scale material are well kept. Moreover, considering that interreflections happen in Fig. 9-c, our direct-illumination-only assumption still produces a useful prediction that matches the physical realization.

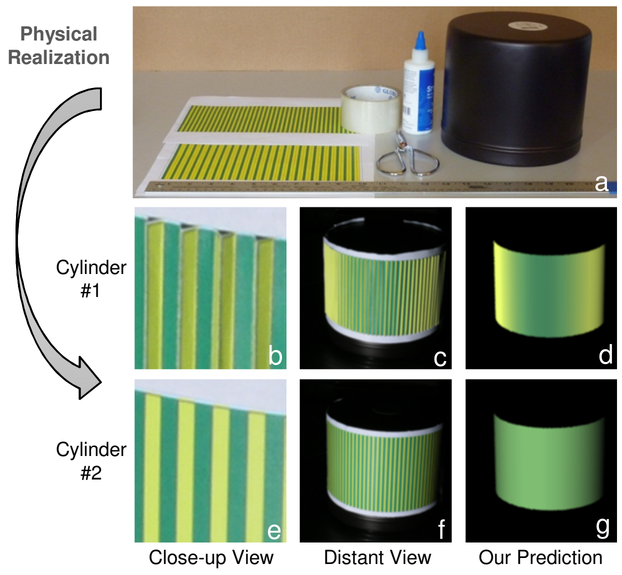


Figure 9: *Physical realization of appearance. We design bi-scale appearance using our system, which is then realized by building the explicitly modeled small-scale structures, with conventional office supplies (a). Cylinder #1 has small-scale grooves (b) similar to Fig. 4, and #2 has a planar structure (e). The results (c & f) are compared with our rendering simulations (d & g).*

In addition to the aforementioned application, we can apply our system in the reverse direction (i.e. from the real world to a digital

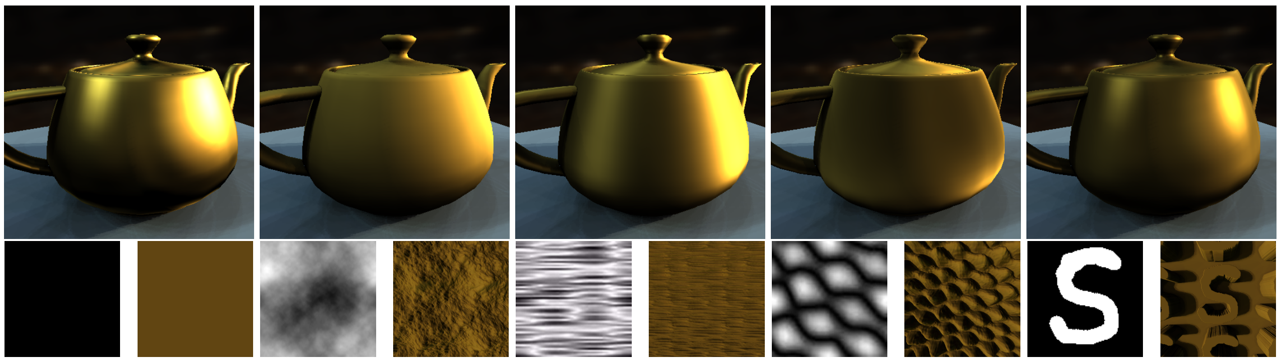


Figure 8: *Generating different large-scale appearance using various small-scale height-field geometries with a measured gold-metallic-paint BRDF. Top row: large-scale appearance. Bottom row: for each pair of images, the left one is a height-field and the right one is the small-scale geometry generated from it.*

appearance representation). In fact, we can approximately model a real-world material in a bi-scale fashion using as few as two photographs taken at different scales. The idea is that despite the 4D nature of a BRDF describing a material, we can exploit the small-scale details as a priori to roughly reconstruct a BRDF from very sparse measurements. We illustrate the process in Fig. 10 to model two different examples of woven fabric. To be specific, we first approximately model the small-scale details (Fig. 10-d) by observing a close-up-view photograph (Fig. 10-a). Then, we fine-tune the small-scale geometry and materials in our system, to match our large-scale rendering (Fig. 10-e) with the other photograph (Fig. 10-b). Here we take the photograph with the same camera and lighting settings as described before, and render the large-scale appearance using the same parameters. To validate our result, we compare an additional photograph under a different lighting configuration (Fig. 10-c) with a corresponding rendering (Fig. 10-f). Finally, the resultant BRDF can be applied on any geometry with novel lighting conditions (Fig. 10-g). Although the result is inaccurate compared with a ground-truth BRDF captured using a gonioreflectometer, we believe that our system can be useful for rapid material modeling in fields like visual effects, where the approximation accuracy is not the top priority.

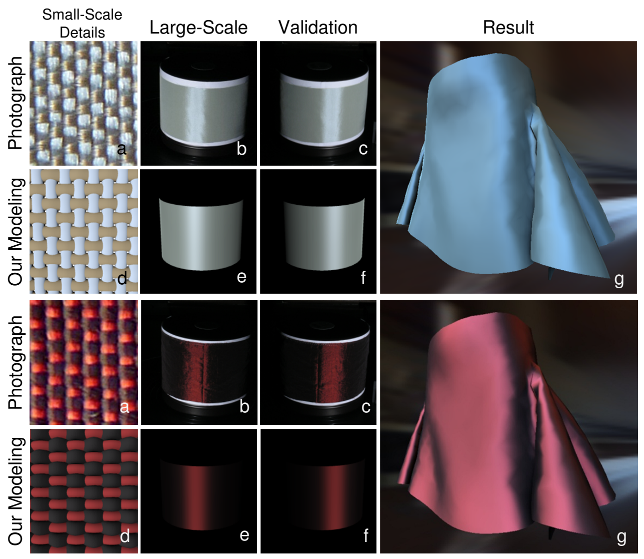


Figure 10: *Modeling of real-world materials using bi-scale constraints. First, we take only two photographs of a fabric sample at different scales (a & b). We then model the small-scale details (d) according to the close-up-view photograph (a). Next, the small-scale geometry and materials are adjusted to match our large-scale rendering (e) with the corresponding photograph (b). The final rendering (g) is done using a large-scale BRDF computed from the edited small-scale details under novel lighting. Photographs (c) and rendering results (f) using a different lighting condition are also shown for validation purpose.*

6 Results and Discussions

We conducted experiments on a workstation with an Intel 2.4GHz quad-core processor and 4GB memory. All images in our system (Fig. 1-a) are rendered with a resolution of 384×384, using directional lights importance sampled from an environment map.

Precomputation. We handle both analytical BRDF models (Lambertian, Blinn-Phong, Cook-Torrance and Ward models) and measured isotropic BRDFs from [Matusik et al. 2003]. For each analytical model, we precompute multiple versions of BRDFs with different model parameters. The optimal sampling rate of M is computed as described in Sec. 3.7: the error threshold is set to 1% of the total power for the normal domain, and 2% for the bidirectional domain. This is because in our experiments, we found that using relatively lower error threshold, for computing the sampling rate of the normal domain, will produce better quality results than using equal thresholds for both domains, given a fixed size of M. For practical consideration, we clamp the maximum size of M to $1.7M \times 6.4K$, which means we would have insufficient sampling rate when representing highly specular, mirror-like BRDFs. This problem is typical for any data-driven representation of BRDFs, and we hope that it can be alleviated in the future by exploiting the sparsity in M of such materials. We represent M with double-precision numbers in SVD for better numerical stability. For all other stages, we use single-precision numbers. After SVD, we keep the minimum number of eigen-vectors that maintain at least 98% of the power of M. The actual rank of our approximation is listed in Tab. 3, which justifies the use of random projection in our pipeline. We have found in experiments that including the two cosine terms in M (Eq. 4) not only accelerates rendering as they are precomputed, more importantly results in a desirable lower-rank structure in M when compared to a formulation without the two terms. Please refer to Tab. 3 for details about precomputation of various materials.

Rendering. In our experiments, we sample 500∼12,000 facets for small-scale geometry with different complexity, and use 88∼132 directions for visibility sampling. We apply random projection to reduce the number of rows of N to 50, and then perform SVD. Fig. 11 shows an example of using too few eigen-vectors in approx-

Material	Row# of M	Column# of M	Size of M (MB)	Final Size (MB)	Compress. rate	Rank of Our Approx.	Precomput. Time (sec)
Lambertian	8,778	448	30.00	0.23	130:1	7	22.0
Ward(α=0.3)	161,596	3,768	4,645.49	22.81	204:1	37	1,573.5
Cook-Torrance(α=0.5)	46,056	1,512	531.29	2.99	178:1	17	262.8
Cook-Torrance(α=0.2)	1,725,153	6,442	84,788.78	460.67	184:1	70	93,409.3
pearl-paint	49,141	2,042	765.58	7.50	102:1	40	681.3
silver-paint	161,596	3,768	4,645.49	37.60	124:1	61	4,276.5
silver-metallic-paint2	283,128	6,442	13,915.33	101.52	137:1	94	10,172.2
gold-metallic-paint	453,628	6,442	22,295.16	160.93	139:1	93	15,282.9
blue-metallic-paint	453,628	6,442	22,295.16	164.39	136:1	95	20,835.7
special-walnut-224	935,028	4,250	30,318.21	178.34	170:1	50	38,016.3
white-diffuse	49,141	626	234.70	1.87	126:1	10	1,138.9

Table 3: *Various statistics from precomputation of different materials. Note that only a subset of all materials we processed are shown here due to limited space.*

imating N. We observe that most large differences occur at grazing view angles. In our pipeline, the number of eigen-vectors is set to maintain at least 99.5% of the power of N, for a high-quality approximation. According to Eq. 8, we decouple the handling of N and M, and use appropriate sampling rate to represent each one separately. When combining N and M to compute $\overline{f}_r$, we interpolate N in the bidirectional domain to match that of M, which is usually higher.

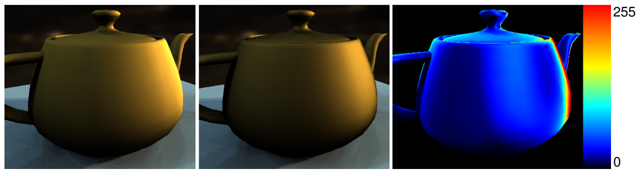


Figure 11: *The effect of SVD approximation for N. Left: rendering with 23 eigen-vectors (error = 0.15%). Center: rendering with 1 eigen-vector (error = 7.44%). Right: color-coded 3× pixel-wise difference. The error is the sum of squared differences divided by the total sum of squares over all elements of N.*

The performance of our rendering pipeline is divided into two parts. The first one is the time to compute a new $\overline{f}_r$ after an edit is performed. In our experiments, it takes 0.38∼2.43 seconds, depending on the complexity of both the small-scale geometry and materials. The second part is the time to render $\overline{f}_r$ on a large-scale object. We achieve 10∼20 fps for a scene with 32 directional lights.

We validate the result of our rendering pipeline with a ground-truth effective $\overline{f}_r$ computed as follows: we sample the ground-truth $\overline{f}_r$ with the same resolution for representing the approximated $\overline{f}_r$ in our pipeline; for each $(\boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k})$, we render the small-scale geometry towards $\boldsymbol{\omega}_{o,k}$ with a directional light from $\boldsymbol{\omega}_{i,k}$, and perform shading using small-scale materials; the average pixel value is then computed and saved as the ground-truth $\overline{f}_r(\boldsymbol{\omega}_{i,k}, \boldsymbol{\omega}_{o,k})$. To accelerate the computation, we use GPU to rasterize the geometry into pixels and perform shadow mapping for visibility determination. The rendering results comparing both versions of $\overline{f}_r$ are shown in the accompany video. Our rendering is close to the ground-truth and all major visual features are captured. We also compute numerical quality of our approximation using Peak Signal-to-Noise Ratio (PSNR), listed in Tab. 4: we measure the difference in terms of cosine-weighted BRDF, similar to [Lafortune et al. 1997]. Comparing with the ground-truth $\overline{f}_r$ computation time, we essentially reach an acceleration rate of over three orders of magnitudes in all scenes (Tab. 4).

We also compare our method to Monte-Carlo path tracing, which includes small-scale interreflections. Two of our scenes with most significant differences are shown in Fig. 12. Although path tracing results look slightly brighter due to the indirect illumination, our system captures all major visual effects, and more importantly provides fast feedback.

Scene	Comput. Time (sec) Ground-Truth	Comput. Time (sec) Ours	Accel. Rate	PSNR (db)
plate	10410.3	2.43	4284:1	17.36
teapot	7668.7	1.30	5899:1	19.27
pillow	2242.2	1.63	1375:1	20.01
cup	2992.1	0.75	3989:1	18.86
skirt	6430.8	2.14	3005:1	22.66

Table 4: *Comparisons between a ground-truth approach for computing $\overline{f}_r$ and our approximation.*

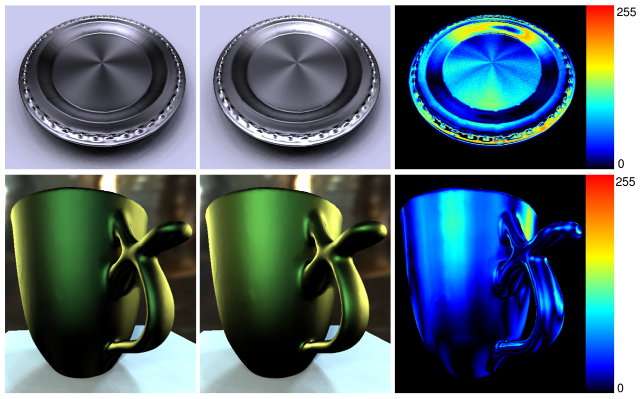


Figure 12: *Comparisons with path tracing results that include small-scale interreflections. Left: our rendering. Center: path tracing results. Right: color-coded 3× pixel-wise difference.*

Editing. We have already shown various editing results in Sec. 4. All height-field images has a resolution of 128 × 128. For the editing user interface (Fig. 1-a), the main elements are two viewports: one displaying a large-scale object rendered using $\overline{f}_r$, and the other showing a visualization of the small-scale structure. We use a Lambertian BRDF while rendering the small-scale details for a better visualization.

Limitations. Similar to previous work on reflectance filtering [Bruneton and Neyret 2011], our rendering pipeline handles direct illumination only, without considering transmission/scattering, which might be important for the appearance of certain types of small-scale details. The other limitation is the lack of support for anisotropic small-scale materials, though we can generate anisotropic large-scale appearance through the use of anisotropic small-scale geometry (e.g. Fig. 5). It would be interesting to extend BVNDF to Bidirectional Visible Local-Frame Distribution Function in order to tackle this issue.

7 Conclusion and Future Works

We present the first physically-based interactive bi-scale editing system, which introduces a novel form of material design, by manipulation of both small-scale geometry and materials. The core of our system is a high-performance reflectance filtering algorithm, which exploits the low-rank structures in the matrix formulation of our rendering problem. In addition to bi-scale material design, we show how to apply our system to physical realization of appearance, as well as rapid modeling for real-world materials using very sparse measurements.

For future work, we would like to lift the limitations mentioned in the previous section. We are also interested in material editing of more than two scales, as well as generating large-scale spatially-varying BRDFs. In addition, we would like to experiment with [Johnson et al. 2011] to capture high-precision small-scale geometry, and then use the result as input to our system.

Acknowledgements

We would like to thank Jason Lawrence for the inspiring initial discussion, Huan Wang for discussions on random projection, Min H. Kim for help on capturing data, Li-Yi Wei for help on typesetting, and Solomon Boulos and Yue Dong for 3D models. This material is based upon work supported by the National Science Foundation under Grant No. IIS-1064412.

References

ASHIKMIN, M., PREMOŽE, S., AND SHIRLEY, P. 2000. A microfacet-based BRDF generator. In *Proc. of SIGGRAPH 2000*, 65–74.

BEN-ARTZI, A., OVERBECK, R., AND RAMAMOORTHI, R. 2006. Real-time BRDF editing in complex lighting. In *Proc. of SIGGRAPH 2006*, 945–954.

BRABEC, S., ANNEN, T., AND PETER SEIDEL, H. 2002. Shadow mapping for hemispherical and omnidirectional light sources. In *Proc. of Comp. Graph. Inter.*, 397–408.

BRUNETON, E., AND NEYRET, F. 2011. A survey of non-linear pre-filtering methods for efficient and accurate surface shading. *To appear in IEEE Trans. Vis. Comput. Graph.*.

COOK, R. L., AND TORRANCE, K. E. 1982. A reflectance model for computer graphics. *ACM Trans. Graph. 1*, 1 (January), 7–24.

DONG, Y., WANG, J., PELLACINI, F., TONG, X., AND GUO, B. 2010. Fabricating spatially-varying subsurface scattering. *ACM Trans. Graph. 29*, 4 (July), 62:1–62:10.

ERSHOV, S., DURIKOVIC, R., KOLCHIN, K., AND MYSZKOWSKI, K. 2004. Reverse engineering approach to appearance-based design of metallic and pearlescent paints. *Vis. Comput. 20* (November), 586–600.

GONDEK, J. S., MEYER, G. W., AND NEWMAN, J. G. 1994. Wavelength dependent reflectance functions. In *Proc. of SIGGRAPH 94*, 213–220.

GREEN, P., KAUTZ, J., MATUSIK, W., AND DURAND, F. 2006. View-dependent precomputed light transport using nonlinear gaussian function approximations. In *Proc. of I3D 2006*, 7–14.

HAN, C., SUN, B., RAMAMOORTHI, R., AND GRINSPUN, E. 2007. Frequency domain normal map filtering. *ACM Trans. Graph. 26*, 3 (July), 28:1–28:11.

HAŠAN, M., FUCHS, M., MATUSIK, W., PFISTER, H., AND RUSINKIEWICZ, S. 2010. Physical reproduction of materials with specified subsurface scattering. *ACM Trans. Graph. 29*, 4 (July), 61:1–61:10.

HEEGER, D. J., AND BERGEN, J. R. 1995. Pyramid-based texture analysis/synthesis. In *Proc. of SIGGRAPH 95*, 229–238.

HEIDRICH, W., DAUBERT, K., KAUTZ, J., AND SEIDEL, H.-P. 2000. Illuminating micro geometry based on precomputed visibility. In *Proc. of SIGGRAPH 2000*, 455–464.

JOHNSON, M. K., COLE, F., RAJ, A., AND ADELSON, E. H. 2011. Microgeometry capture using an elastomeric sensor. *ACM Trans. Graph. 30*, 4 (August), 46:1–46:8.

KAJIYA, J. T. 1986. The rendering equation. In *Proc. of SIGGRAPH 86*, 143–150.

KERR, W. B., AND PELLACINI, F. 2010. Toward evaluating material design interface paradigms for novice users. *ACM Trans. Graph. 29*, 4 (July), 35:1–35:10.

LAFORTUNE, E. P. F., FOO, S.-C., TORRANCE, K. E., AND GREENBERG, D. P. 1997. Non-linear approximation of reflectance functions. In *Proc. of SIGGRAPH 97*, 117–126.

MATUSIK, W., PFISTER, H., BRAND, M., AND MCMILLAN, L. 2003. A data-driven reflectance model. *ACM Trans. Graph. 22*, 3 (July), 759–769.

OREN, M., AND NAYAR, S. K. 1994. Generalization of Lambert's reflectance model. In *Proc. of SIGGRAPH 94*, 239–246.

PELLACINI, F., AND LAWRENCE, J. 2007. AppWand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph. 26*, 3 (July), 54:1–54:9.

RUSINKIEWICZ, S. M. 1998. A new change of variables for efficient BRDF representation. In *Eurographics Workshop on Rendering*, 11–22.

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

TAN, P., LIN, S., QUAN, L., GUO, B., AND SHUM, H.-Y. 2005. Multiresolution reflectance filtering. In *Rendering Techniques*, 111–116.

VEMPALA, S. 2004. *The Random Projection Method*. AMS.

WESTIN, S. H., ARVO, J. R., AND TORRANCE, K. E. 1992. Predicting reflectance functions from complex surfaces. In *Proc. of SIGGRAPH 92*, 255–264.

WEYRICH, T., PEERS, P., MATUSIK, W., AND RUSINKIEWICZ, S. 2009. Fabricating microgeometry for custom surface reflectance. *ACM Trans. Graph. 28*, 3 (July), 32:1–32:6.

WU, H., DORSEY, J., AND RUSHMEIER, H. 2009. Characteristic point maps. *Computer Graphics Forum 28*, 4, 1227–1236.

ZHAO, S., JAKOB, W., MARSCHNER, S., AND BALA, K. 2011. Building volumetric appearance models of fabric using micro CT imaging. *ACM Trans. Graph. 30*, 4 (August), 44:1–44:10.