Inverse Bi-scale Material Design

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

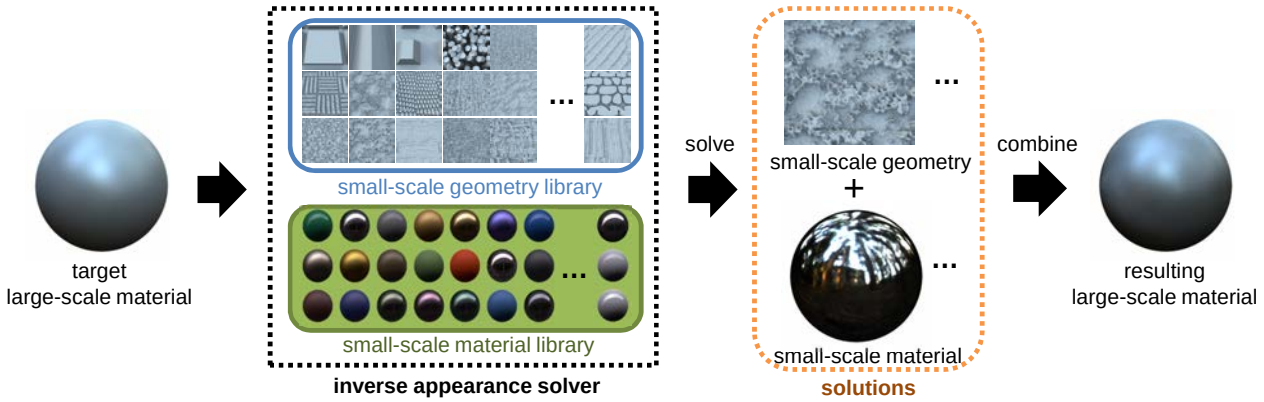

Figure 1: *Given a target large-scale material, our inverse appearance solver efficiently searches through precomputed small-scale material and geometry libraries. Then it returns a few small-scale materials and geometries as results, which approximate the target large-scale appearance when combined.*

Abstract

One major shortcoming of existing bi-scale material design systems is the lack of support for inverse design: there is no way to directly edit the large-scale appearance and then rapidly solve for the small-scale details that approximate that look. Prior work is either too slow to provide quick feedback, or limited in the types of small-scale details that can be handled. We present a novel computational framework for inverse bi-scale material design. The key idea is to convert the challenging inverse appearance computation into efficient search in two precomputed large libraries: one including a wide range of measured and analytical materials, and the other procedurally generated and height-map-based geometries. We demonstrate a variety of editing operations, including finding visually equivalent details that produce similar large-scale appearance, which can be useful in applications such as physical fabrication of materials.

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

Keywords: inverse appearance computation, bi-scale, microfacet

Links: DL PDF WEB VIDEO

1 Introduction

Deriving large-scale materials from small-scale details is a powerful appearance modeling technique. From a physical point of view, the large-scale look is determined by averaging the appearance of small-scale details [Bruneton and Neyret 2012]. Previous work has succeeded in modeling a wide range of real-world materials, from velvet [Westin et al. 1992], brushed metal [Ashikmin et al. 2000], to woven fabrics [Zhao et al. 2011].

Recently, researchers further pushed the idea by introducing interactive bi-scale material design [Wu et al. 2011; Iwasaki et al. 2012]: the user can manipulate both the small-scale material and geometry; the corresponding change on the large-scale appearance is then rapidly computed. However, one major limitation in existing work is that, the editing operations can only be applied to the small-scale details. There is no way to directly design the large-scale appearance, and then automatically find small-scale details to approximate it. Tedious trial-and-error on the small-scale is needed to achieve even simple edits on the large-scale appearance.

To address the above problem, we would like to build an inverse bi-scale material design system, which turns out to be highly challenging: first, there is a large number of degrees of freedom in the small-scale materials and geometries, making it difficult to efficiently solve for the optimal result in this huge space; second, due to the visibility factor, the effect of small-scale geometry over the large-scale appearance is both non-linear and non-local; moreover, inverse bi-scale material computation is an ill-posed problem, as there might be multiple small-scale materials and geometries, which correspond to similar appearance on the large scale [Han et al. 2007]. Previous papers in this field (e.g., [Weyrich et al. 2009]) solve sophisticated non-linear optimization problems, a process that is far from interactive. In addition, existing approaches (e.g., [Ershov et al. 2004]) handle very limited types of materials and geometry. It is non-trivial to extend these frameworks to more general cases.

This paper presents a novel appearance computational framework for inverse bi-scale material design. Inspired by previous work

based on large databases [Hays and Efros 2007], we precompute a material library and a geometry library, which include a wide range of small-scale materials and geometries. We also propose a search algorithm to find desired small-scale details in these two large libraries, unlike existing approaches that directly optimize for unknown small-scale materials and geometries (e.g., [Weyrich et al. 2009]). Our algorithm typically computes results in a few seconds, while previous work requires a couple of hours. The key to the efficiency is that we only evaluate a few representatives, which are obtained by exploiting the intrinsic relationships among all materials / geometries in the libraries and then apply CX matrix decomposition [Drineas et al. 2008].

Our design system can also find *visually equivalent details*, which are different small-scale details that produce similar appearance on the large scale. Potentially, this could be very useful for applications such as appearance fabrication, where multiple practical constraints need to be considered. Moreover, one may use our system for building exterior design, to edit the looks of a building that change with view distances.

In summary, this paper makes the following contributions:

- We present the first near-interactive inverse bi-scale material design system, the core of which is a novel inverse appearance computational framework. We convert the solving of a sophisticated optimization problem as in previous work, into a search in large precomputed libraries, which is both faster and more general.
- Our work completes bi-scale material design, by adding the key functionality missing in previous forward design systems [Wu et al. 2011; Iwasaki et al. 2012], which frees the designers from manual trial-and-error.
- We build a large library of small-scale geometry, consisting of over 164,000 pieces of procedurally-generated and height-map-based geometry. The wide variety in our library makes it a valuable database for future research in computer graphics and computer vision.

Terminology. We follow the terminology of *large/small-scale* from previous work [Wu et al. 2011; Iwasaki et al. 2012], instead of using *micro/milli-scale*. 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

Modeling Large-scale Appearance from Small-scale Details. Microfacet BRDFs (e.g., [Cook and Torrance 1982; Oren and Nayar 1994; Ashikmin et al. 2000]) are modeled by statistical distributions of the orientations and shadowing of perfectly specular or Lambertian facets, assuming no correlations between the two factors. Wang et al. [2011] propose a dual-scale statistical model for stationary, isotropic indoor surface appearance. More complex appearance can be modeled by explicitly constructing the small-scale structures, and then performing an expensive simulation of light interactions [Westin et al. 1992; Gondek et al. 1994]. Zhao et al. [2011] extends the idea to volumetric appearance models, using results from micro CT imaging. Previous work in this field focuses on the analysis, rather than the synthesis, of small-scale details.

Appearance Fabrication. Weyrich et al. [2009] use a milling machine to fabricate a height-field. Its effective large-scale appearance closely matches a custom reflectance distribution function. The idea is further pushed to realize custom subsurface scattering properties [Hašan et al. 2010; Dong et al. 2010]. In addition, prototyping printers are developed [Matusik et al. 2009; Malzbender et al. 2012] to produce reflectance distribution functions on a sheet of paper, or a special structure with spherical dimples. The goal of [Ershov et al. 2004] is similar to ours. They propose a reverse engineering method to figure out pigment composition that approximates a designated paint appearance. Concurrent to our work, Lan et al. [2013] fabricate large-scale spatially-varying anisotropic BRDF, by printing an optimized height-field coated with glossy inks. Unfortunately, existing work is not directly applicable to our problem, either because the speed of algorithms is not suitable for interactive design, or because the methods cannot be easily generalized to handle the wide range of materials and geometry that we process.

Interactive Appearance Design. Significant research effort has been devoted to the development of interactive single-scale appearance design (e.g., [Pellacini and Lawrence 2007; Lawrence et al. 2006; Ben-Artzi et al. 2006]), which supports rapid visual feedback by exploiting factors like fixed lighting and/or geometry. Bi-scale material design systems [Wu et al. 2011; Iwasaki et al. 2012] quickly compute the large-scale appearance from small-scale details. However, they do not support editing in the reverse direction. Manual trial-and-error is required to find small-scale details that achieve an equivalent large-scale appearance change. Moreover, it is not clear to the designer if a better solution exists.

Inverse Rendering. One major goal for inverse rendering algorithms (e.g., [Ramamoorthi and Hanrahan 2001; Marschner et al. 1999]) is to estimate reflectance properties from photometric measurements, typically with known geometry and illumination. When both the lighting and BRDFs are unknown, additional assumptions are needed to resolve the ambiguity. Recently, Romeiro and Zickler [2010] tackle this problem, by assuming a statistical distribution of natural illumination. While our problem can be viewed as generalized inverse rendering, and we have similar small-scale material-geometry ambiguity, it is unclear how to extend previous work to inverse bi-scale appearance computation, where the known function is 4D (large-scale BRDF) and the two unknown factors are 4D (small-scale BRDF) and 6D (small-scale geometry), respectively.

3 Preliminaries

The effective large-scale appearance, $\overline{f}_r$, is computed as the integral of the product of a rotated isotropic BRDF f and a Bidirectional Visible Normal Distribution Function (BVNDF) γ, defined on the small scale [Wu et al. 2011; Iwasaki et al. 2012]:

$$\overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \int_{S^2} f(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\gamma(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)d\boldsymbol{n}, \qquad (1)$$

where

$$f(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = f_r(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o)(\boldsymbol{n} \cdot \boldsymbol{\omega}_i), \qquad (2)$$

$$\gamma(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{(\boldsymbol{n} \cdot \boldsymbol{\omega}_o)}{a_v(\boldsymbol{\omega}_o)} \int_A V(\boldsymbol{\omega}_i)V(\boldsymbol{\omega}_o)\delta(\boldsymbol{n} - \boldsymbol{n'})dA. \qquad (3)$$

Here f_r is a small-scale BRDF, $\boldsymbol{\omega}_i$ is the lighting direction, and $\boldsymbol{\omega}_o$ is the view direction. S^2 is the unit sphere, V is the visibility function, A is a surface patch, and $a_v(\boldsymbol{\omega}_o)$ is the visible projected area of A along $\boldsymbol{\omega}_o$ (see Fig. 3 in [Wu et al. 2011] for an illustration). $\boldsymbol{n'}$ is the normal associated with dA. Note that compared to the original definition of f and γ in [Wu et al. 2011; Iwasaki et al. 2012], we slightly moved a few terms around in order to apply the same idea to solve for materials and for geometry (Sec. 4). Throughout this paper, we assume that BRDFs (e.g., f_r) are spectrally-averaged single-channel functions. Details about how to handle spectral information are described in Sec. 4.2.

3.1 BVNDF and BRDF Representations

We propose high-precision representations that are inspired by all-frequency rendering under environment map illumination [Green et al. 2006], where the BRDF is fitted with lighting- and view-dependent Gaussian lobes, and the environment map is prefiltered for different lobes. First, we represent a BVNDF as a linear combinations of von Mises-Fisher (vMF) distributions [Han et al. 2007] at different view and lighting directions. For a particular directional pair, γ is represented as:

$$\gamma(\boldsymbol{n}) = \sum_j \alpha_j \rho(\boldsymbol{n}; \kappa_j, \boldsymbol{\mu}_j). \tag{4}$$

Here κ_j is the inverse width, $\boldsymbol{\mu}_j$ is the central direction and α_j is the corresponding weight. $\rho(\cdot)$ is a Gaussian-like probability distribution over the unit sphere:

$$\rho(\boldsymbol{n}; \kappa, \boldsymbol{\mu}) = \frac{\kappa}{4\pi \sinh(\kappa)} e^{\kappa(\boldsymbol{n}\cdot\boldsymbol{\mu})}. \tag{5}$$

Using vMFs allows us to compactly represent all-frequency normal distributions, including δ-like spikes. For each lobe, only α, κ and $\boldsymbol{\mu}$ are stored. Please refer to Fig. 5 for an example.

Next, we represent a BRDF as prefiltered responses with respect to vMF lobes of various sizes and orientations. To be specific, we substitute Eq. 4 into Eq. 1, which yields:

$$\overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \sum_j \alpha_j F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \kappa_j, \boldsymbol{\mu}_j), \tag{6}$$

where

$$F(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o, \kappa, \boldsymbol{\mu}) = \int_{S^2} f(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \rho(\boldsymbol{n}; \kappa, \boldsymbol{\mu}) d\boldsymbol{n}. \tag{7}$$

Here F is our BRDF representation, which contains all the information about the small-scale BRDF, in order to compute $\overline{f}_r$. In practice, we store F as a mipmap: each of its elements is the precomputed product between f and vMFs of different $\boldsymbol{\mu}$ and κ. Essentially, computing $\overline{f}_r$ becomes multiple look-ups in the precomputed mipmap. One issue here is that F is 7D, which is difficult for efficient storage. In the appendix, we show how to reduce the dimensionality of F to 4, using a half-angle parameterization.

Note that our framework is independent of the underlying BVNDF / BRDF representations. We choose the current ones, after balancing precision, compactness and efficiency. Other representations (e.g., [Iwasaki et al. 2012]) might also be employed under different circumstances.

4 Inverse Bi-scale Appearance Computation

Given a target large-scale BRDF $\hat{f}_r$, our goal is to solve for the small-scale geometry γ and material f_r, which best approximate $\hat{f}_r$ when viewed on the large-scale:

$$\underset{\gamma, f_r}{\operatorname{argmin}}\, d(\hat{f}_r, \overline{f}_r), \tag{8}$$

where

$$d(\hat{f}_r, \overline{f}_r) = \int_\Omega \int_\Omega ||\hat{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) - \overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)||^2 d\boldsymbol{\omega}_i d\boldsymbol{\omega}_o. \tag{9}$$

To find the optimal $\overline{f}_r$ in Eq. 8, we need to solve for two unknown functions according to Eq. 1, which is highly challenging. Therefore, we first introduce two simple inverse solvers, which only solve for geometries (Sec. 4.1) or materials (Sec. 4.2), while fixing the other factor. These two solvers are then combined to build a general inverse appearance solver in Sec. 4.3.

4.1 Geometry Solver

Directly solving for every single detail of an unknown geometry is difficult, even for a simpler case [Weyrich et al. 2009] than ours, where shadowing and masking are not considered. It is remarkably challenging to develop an efficient solver, due to the large number of degrees of freedom in a piece of geometry, the non-linear relationship between $\overline{f}_r$ and small-scale geometry, along with the ill-posedness of the problem.

To address these issues, we precompute BVNDFs for a large geometry library G, which consists of more than 164K pieces of procedurally-generated and height-map-based geometry, and directly search for the optimal geometry that approximates the target appearance:

$$\underset{\gamma \in G}{\operatorname{argmin}}\, d(\hat{f}_r, \overline{f}_r). \tag{10}$$

Inspired by [Hays and Efros 2007], the idea is that we will get useful results, by searching in a sufficiently large library, which includes a wide variety of small-scale geometry (detailed in Sec. 5).

One naïve search strategy is to enumerate all elements in the library, and select ones whose large-scale appearance differs the least from our target, according to Eq. 10. However, this is not efficient, especially when the size of the library is large. Ideally, we would like to find a few representatives, then quickly compute approximation errors for all elements in the library, by just evaluating these representatives. To do this, we employ CX matrix decomposition [Drineas et al. 2008], which is a randomized algorithm for approximating a matrix using its original column vectors, with theoretical relative-error guarantees.

Specifically, we first discretize each γ (Eq. 3) with sufficient sampling rate, the results of which are assembled into a vector. Please see Sec. 5.2 for more details. Next, we construct a matrix B, by assigning each discretized γ_j to a column vector b_j. Then, we apply CX decomposition to B as $B \approx CX$. Here C contains a few original column vectors from B. The column number of C is the minimum number that results in a user-specified approximation error. Now b_j can be expressed as a linear combination of columns from C:

$$b_j \approx \sum_k b_{c(k)} X_{kj}, \tag{11}$$

Note that $c(k)$ indicates that the k-th column of C is from column $c(k)$ of B. Fig. 2 illustrates an example.

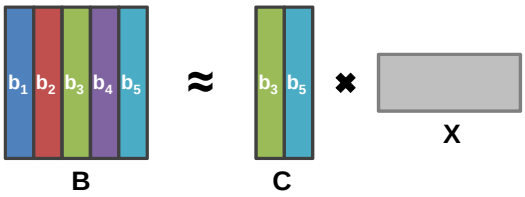

Figure 2: *CX decomposition. Matrix B is approximated as the product of C and X, where C consists of a few original columns from B.*

Observe in Eq. 1 that $\overline{f}_r$ is linear with respect to γ, if we fix the small-scale material. Applying this linearity to Eq. 11 yields:

$$\boxed{\overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \gamma_j) \approx \sum_k \overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o; \gamma_{c(k)}) X_{kj}.} \tag{12}$$

Now we only need to compute $\overline{f}_r(\cdot; \gamma_{c(k)})$ for different $c(k)$ (i.e., representatives). By multiplying with X, we obtain $\overline{f}_r(\cdot; \gamma_j)$

for all geometries in the database, which is significantly faster than an exhaustive computation of all $\overline{f}_r$s in our experiments.

Finally, we calculate the approximation error with respect to the target appearance (Eq. 9), for every piece of small-scale geometry γ_j. We select a user-specified number of candidates, which are closest to the target. Then we compute more accurate errors by directly evaluating $\overline{f}_r$ of the candidates, and sort the results according to their errors. Note that we precompute $c(\cdot)$ and X for runtime efficiency (Sec. 5.2).

4.2 Material Solver

Designing a material solver is very similar to building a geometry solver. This is because fixing the BVNDF γ and solving for the material f_r is exactly the dual problem of the geometry solving process (Eq. 1). Here we treat the F (Eq.18) for each material as a long column vector, which is then used to construct the matrix B for CX decomposition, as described in Sec. 4.1. Another modification is to change the measure of error to account for additional spectral information:

$$\underset{f_r \in M}{\operatorname{argmin}} (1-\beta) d(\hat{f}_r, \overline{f}_r) + \beta ||\lambda(\hat{f}_r) - \lambda(\overline{f}_r)||^2. \qquad (13)$$

Note that M is the material library, β is a user-specified parameter, which indicates the preference over spectral difference or spectrally-averaged reflectance difference, and λ is the color of an original RGB-channel BRDF, averaged over different lighting and view directions.

4.3 General Solver

Having developed two solvers for a single unknown factor, it is straightforward to combine them to solve the general problem, where both the small-scale material and geometry are unknown. Starting with an initial f_r or γ (randomly selected or user-specified), we fix it and alternatively solve for the optimal materials / geometries. Once a solver returns results, the best candidate is set as a constant for the other solver, and vice versa. This process is repeated until convergence or a user-specified condition is met.

5 Precomputation

5.1 Material Library Construction

For the material library, we include 100 measured BRDFs from the MERL database [Matusik et al. 2003]. Due to the narrow range of real-world materials that the MERL database covers, we enlarge our library by adding Ward, Blinn-Phong and Cook-Torrance BRDFs [Dorsey et al. 2007]. These analytical BRDFs are created by randomly sampling their corresponding parameters, and then mixing with Lambertian lobes of different magnitudes. The total number of distinct BRDFs is 4,101. For every material, we compute its prefiltered mipmap F (Eq. 18) and the average color λ. Moreover, we increase the spectral variation in the library by color-shifting each existing BRDF to 200 randomly sampled colors, resulting in $4,101 \times 201 = 824$K effective materials. Essentially, we are limited by the availability of measured BRDFs in the public domain. As techniques in appearance measurement advance, we hope that more measured materials can be added to our library to replace artificial ones.

Before computing for material representatives, we perform PCA to the original matrix B to exploit its numerical coherence, which helps speed up a subsequent CX decomposition (Sec. 4.1). As the size of M is relatively small (4K distinct BRDFs), we directly compute the co-variance matrix, and perform standard SVD-based compression using LAPACK, by keeping 99.9% of the original energy. The result is $B \approx U_B \Sigma_B V_B^T$, where U_B and V_B^T are orthonormal matrices, and Σ_B is a diagonal one. We then apply CX decomposition on the smaller-sized $U_B \Sigma_B$, instead of B. Since $U_B \Sigma_B$ approximately preserves the inner products among column vectors in B, our CX decomposition result is close to that of B.

5.2 Geometry Library Construction

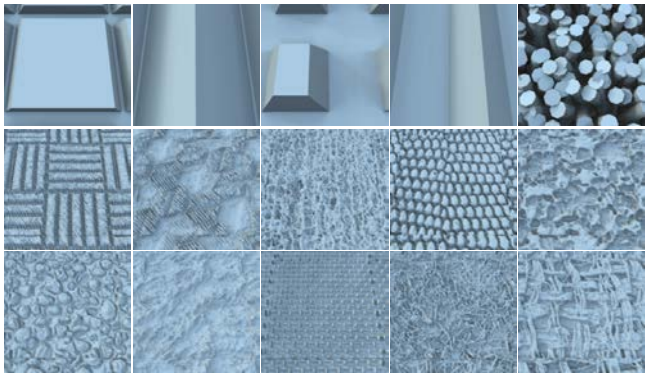

Figure 3: *Geometry samples for constructing our BVNDF library. The top row shows procedurally-generated geometries; the rest are derived from height maps.*

Our geometry library is composed of procedurally-generated and height-map-based geometries, as illustrated in Fig. 3. For procedural geometry, we use random parameters as input to predefined procedural geometry generators (e.g., grooves, bricks and rods) to directly create triangular meshes. Note that some of the results are non-height-field geometries. On the other hand, we harness a large number of existing high-quality textures, such as Brodatz [1999] and Mayang [2013] texture databases, to generate height-map-based geometries (see Fig. 4 for an illustration). Specifically, for each texture, we first convert it to a grey-scale image and down-sample to 512×512; next, we sample a few small patches of 128×128; for each patch, Fast Fourier Transform is used to remove the low-frequency components; we then apply histogram equalization; finally, triangular meshes are generated from the patches by applying different amplitudes. The total number of geometries is 164,431. For every piece of geometry, we render it on the GPU with different lighting and view directions to get its BVNDF, which is then fitted with vMFs, using the Expectation Maximization algorithm [Han et al. 2007]. Please refer to Fig. 5 for an example.

We mentioned the discretization of γ in Sec. 4.1. In practice, it would require significantly high sampling rate, in order to discretize any δ-like lobe in γ with high precision. To tackle this issue, we classify BVNDFs into two categories: specular and non-specular ones. Specular BVNDFs contain a non-negligible number of δ-like vMFs. They are not discretized, or processed with CX decomposition. During runtime, we directly evaluate $\overline{f}_r$ for all specular BVNDFs. For non-specular ones, we discretize them and assemble the results as a matrix B (Sec. 4.1). However, the size of B is 7.7TB, which is too big for efficient processing. In order to reduce its size, our first step is to restrict $\boldsymbol{\omega}_i$ to only 7 directions, quasi-randomly distributed over the upper hemisphere. We find that 7 is an adequate number for sampling $\boldsymbol{\omega}_i$, which will be demonstrated in Sec. 7 & 8. The reduced matrix is now 614.4GB. Next, similar to material library construction, we perform SVD-based compression

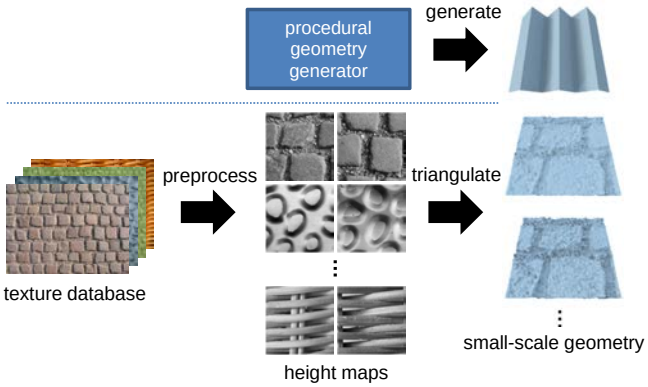

Figure 4: *Populating geometry for our library. For procedural geometry, we generate various instances by randomly sampling the corresponding parameters. For height-map-based geometry, we first transform textures into gray-scale patches, which are then converted to small-scale geometry by applying different amplitudes.*

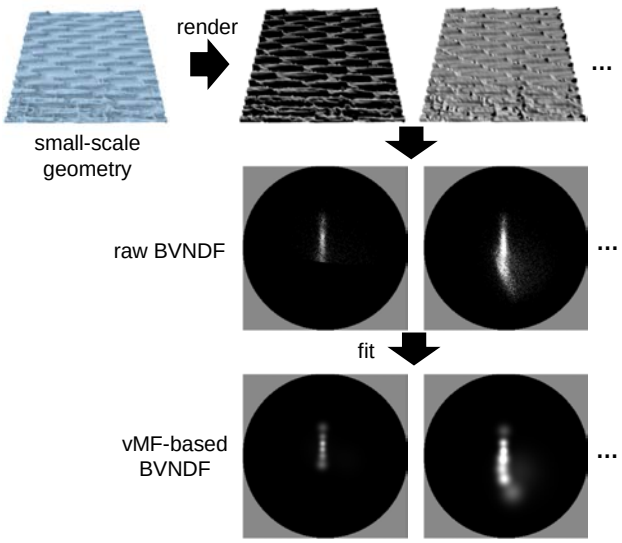

Figure 5: *Computing and fitting a BVNDF. A piece of geometry is directly rendered on GPU, to compute its visible and unshadowed normal distributions under different illumination and view directions (i.e., BVNDF). The results are then fitted using a few vMF lobes, which compactly represent the raw data. Here we visualize a normal distribution by projecting the upper hemisphere onto a 2D plane.*

on the reduced matrix, to further exploit data coherence. Specifically, we employ the state-of-the-art random projection technique with power iterations [Halko et al. 2011] to compress such a big matrix. The size of the compression result is 1.5GB, while still retaining 99.0% of the original energy. Finally, CX decomposition is applied to find BVNDF representatives, as well as the weights to express all BVNDFs.

6 Implementation Details

Sampling $\hat{f}_r$. When measuring the approximation error using Eq. 9 in practice, we need to sample the bidirectional domain and calculate the error at these sampled directional pairs. Recall that in Sec. 5.2, we restrict the lighting directions in BVNDFs to 7 different $\boldsymbol{\omega}_i$s only. Thus we sample $\boldsymbol{\omega}_i$ from these directions. In terms of $\boldsymbol{\omega}_o$, we would like to sample them such that the main features of the target appearance are captured. After experiments, we choose to sample $\boldsymbol{\omega}_o$ using a mixture of two strategies: uniform random sampling, and importance sampling based on the norm of the gradient of $\hat{f}_r$. The former strategy makes sure that every direction has a non-zero probability of being sampled, while the latter one captures the changes in the angular domain, which are characteristic features of a BRDF in our experiments.

Weighting BRDF Samples. Since we exploit linearity in our derivation (Eq. 12), we could not use a non-linear, more perceptually meaningful norm (e.g., [Matusik et al. 2003]), when measuring the error between the target BRDF and our solution. Directly using Eq. 8 generates solutions which focus on features with high numerical values, such as specular highlights. Other perceptually important features with low BRDF values, like the diffuse lobe, might not be well matched. To alleviate this problem, we add non-uniform weights to Eq. 9, so that $d(\hat{f}_r, \overline{f}_r)$ becomes:

$$\int_\Omega \int_\Omega ||w(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \left[\hat{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) - \overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)\right] ||^2 d\boldsymbol{\omega}_i d\boldsymbol{\omega}_o, \tag{14}$$

where $w(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o)$ is defined as:

$$w(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = \frac{1}{\sqrt[4]{p_{\omega_i}(\boldsymbol{\omega}_o)}} \tag{15}$$

Here $p_{\omega_i}(\boldsymbol{\omega}_o)$ is the probability of sampling $\boldsymbol{\omega}_o$ given $\boldsymbol{\omega}_i$, using the aforementioned sampling method. The idea is to put more weights on diffuse lobes compared to uniform weighting, which aligns better with our perception of materials [Matusik et al. 2003].

Handling BVNDF rotations. If a piece of geometry is rotated along the normal direction, its corresponding large-scale appearance is likely to be different, especially when the geometry contains anisotropic features, such as grooves (see bottom two rows of Fig. 7 for examples). To account for this rotation effect in our framework, one natural idea is that, for every piece of geometry, explicitly rotate it to a few angles, compute its BVNDF, and perform SVD followed by CX decomposition on the union of all rotated geometry. However, there are two major drawbacks: first, the size of the library becomes much larger, which grows linearly with the predefined number of rotations; second, the number of rotations are fixed in precomputation and cannot be changed afterwards. To address these issues, we postpone the rotation to runtime and rotate the target large-scale appearance accordingly, which is equivalent to rotating the small-scale geometry. The user can specify the number of rotations, ψ, and then rotate the target appearance by $i \times \frac{2\pi}{\psi} (i = 0, 1, ..., \psi - 1)$ before executing the inverse solver. Here we trade the runtime performance for flexibility and a smaller footprint of the precomputed library.

7 Inverse Bi-scale Material Design

We present a simple inverse bi-scale material design system, as shown in Fig. 6, based on our solvers described in the previous section. A typical workflow is as follows. The designer first loads a target large-scale BRDF, which is rendered in Fig. 6-a. Then, a slider-based editing interface (Fig. 6-c) shows up, for adjusting corresponding parameters of the target BRDF. Once a desired large-scale look is specified, the designer can run the inverse appearance solver, to compute for small-scale materials and / or geometry that best approximate the target appearance. After the results are computed, they are recommended to the designer in the order of increasing approximation error (Fig. 6-d & e). The designer can

then freely combine the candidate material and geometry to generate large-scale appearance (Fig. 6-b). Our system renders both the target and the approximated large-scale appearance side-by-side for visual comparison. Alternatively, if the designer has some idea about which small-scale material to use, she can directly load the material and then invoke the geometry solver to solve for the best possible geometry only. The material solver could be used in a similar fashion.

The remainder of this section shows a variety of editing operations using our inverse design system. Please also refer to the accompanying video for editing demonstrations.

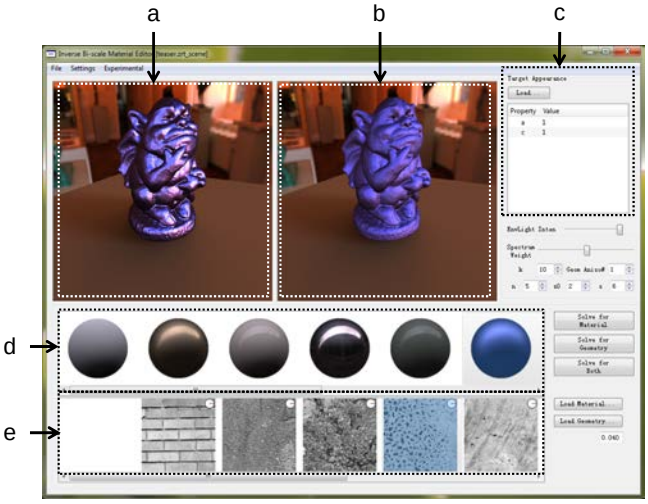

Figure 6: *User interface of our inverse design system. The user can adjust the target large-scale appearance (a) via a slider-based interface (c). Our solver then computes several small-scale material (d) and/or geometry (e) candidates, which best approximate the target appearance. The resulting large-scale appearance (b) is shown side-by-side with the target one for visual comparison.*

Angular sharpen / blur. We demonstrate angular sharpen / blur operations on the target appearance. In the first two rows of Fig. 7, the large-scale materials are isotropic Ward BRDFs ($\alpha = 0.08$ for the first row, $\alpha = 0.30$ for the second). We fix the small-scale material as a measured *grease-covered-steel* BRDF, and employ the geometry solver to solve for small-scale geometry only. The results are shown in the second column of Fig. 7. We can see that the small-scale geometry for the Ward BRDF ($\alpha = 0.30$) is much rougher than the other Ward ($\alpha = 0.08$), which essentially causes more blur over the angular domain.

Anisotropy. In order to match anisotropic target appearance, we need to use anisotropic small-scale geometry, since isotropic BRDFs are assumed on the small-scale. The bottom two rows of Fig. 7 show two examples. The target large-scale appearance is anisotropic Ward BRDFs ($\alpha_x = 0.30$, $\alpha_y = 0.08$ for the third row, and $\alpha_x = 0.08$, $\alpha_y = 0.30$ for the fourth). Following the previous example, we use *grease-covered-steel* as the small-scale material. The geometry solver successfully finds a groove-like structure, which when combined with the small-scale material, produces anisotropic reflections on the large-scale. Note that using the BVNDF rotation method described in Sec. 6, we find the same piece of geometry for the two different anisotropic Ward BRDFs, which only differ by a rotation of the local frame.

Our system can handle various forms of the target appearance. In Fig. 8, the target appearance, as shown on the left, is a measured

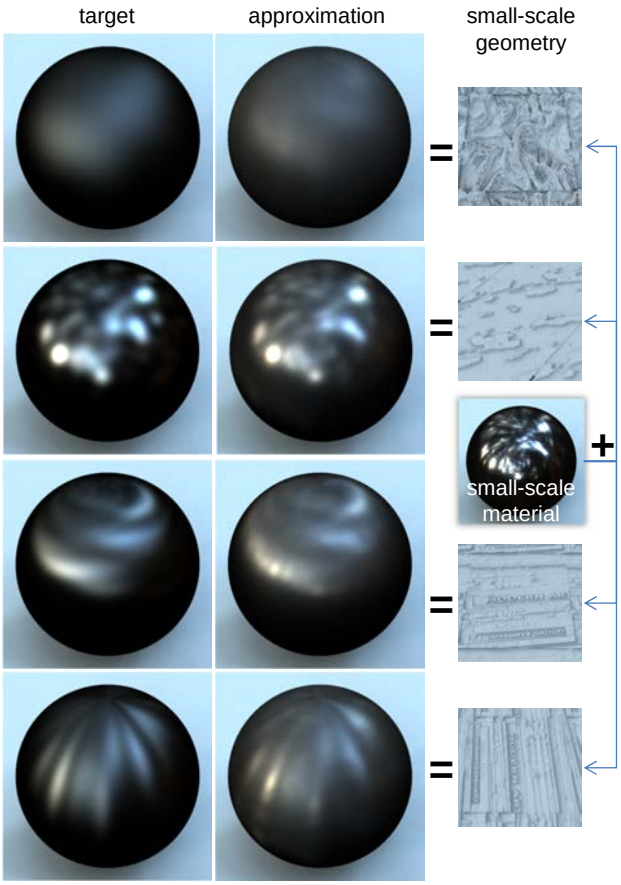

Figure 7: *Angular adjustments on the target appearance. Left: the target large-scale materials, from top to bottom, Ward BRDF (α = 0.08, α = 0.30, α_x = 0.30 & α_y = 0.08, α_x = 0.08 & α_y = 0.30). Center: our approximations. Right: the small-scale material (measured grease-covered-steel) and geometry used to generate the approximations.*

anisotropic BRDF of purple satin from [Ngan et al. 2005], fitted with the model from [Kurt et al. 2010] to fill in the missing data in the original measurements. From our general solver, we obtain interesting results: a Ward BRDF ($\alpha = 0.30$) with a Lambertian lobe, and a piece of geometry which resembles a woven pattern.

Brightness. We can also change the brightness of the target appearance, and then solve for corresponding small-scale material and geometry in our design system. In Fig. 9, the target materials are a measured *blue-metallic-paint* BRDF, whose brightness is decreased / increased. The general solver gives interesting results. For the decreased brightness case, the same *blue-metallic-paint* BRDF is picked, and a binary height-map-based geometry is used. The small-scale structure has a similar-shaped normal distribution as a plane, since its plateaus and valleys are all planar. Meanwhile, the valleys receive less light, which results in a decrease of brightness on the large-scale. For the increased brightness case, a brighter and more specular *blue-metallic-paint2* BRDF is used. It is modulated with a rough geometry to blur its specularities to match the target appearance.

Constrast. Similar to its counterpart in image processing, we define a contrast-adjusted BRDF $f_r(c, \cdot)$ as:

$$f_r(c, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) = (f_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) - \tau)c + \tau. \quad (16)$$

Figure 8: *Inverse solving a measured purple satin BRDF from [Ngan et al. 2005], shown on the left. The center image is our approximation. The small-scale details are a measured Ward BRDF (α = 0.30) with a Lambertian lobe, and a structure similar to a woven pattern.*

Here τ is the average of f_r over the bi-directional domain, and c is a parameter which represents the contrast. For $c = 1$, we get the original f_r; when $c = 0$, we have a BRDF that returns τ for all $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$. An example of contrast-adjusted appearance is shown in Fig. 10. The target appearance is a measured *light-brown-fabric* BRDF, whose contrast is decreased / increased (top-left / bottom-left image of Fig. 10). We employ the general solver to find small-scale details that approximate the corresponding looks (*polyurethane-foam* for decreased contrast, and *black-oxidized-steel* of a reddish average color for increased contrast).

Visually Equivalent Details. One interesting point about bi-scale material design is that different small-scale materials and geometry can generate similar large-scale appearance. We define such small-scale materials and geometry as Visually Equivalent Details (VEDs). VEDs are potentially important for appearance realization, in which the user needs to consider many practical issues including the cost of materials, the structural integrity and the manufacturing time for the geometry, etc. Exploring VEDs would allow the user to find the small-scale material / geometry, which is most suitable for physical fabrication, while maintaining a desired large-scale appearance.

The inverse design system presented in this paper is, to our knowledge, the first test bed that facilitates near-interactive exploration of VEDs. Fig. 11 shows an example. The target appearance is a Blinn-Phong BRDF ($n = 60$) with a Lambertian lobe. Our system helps us find a variety of VEDs, as listed in the figure. The result small-scale materials are, from left to right, Ward($\alpha = 0.16$) with a Lambertian lobe, Ward($\alpha = 0.05$) with a Lambertian lobe, *pickled-oak-260*, *gold-paint*, Cook-Torrance($m = 0.19$, $F_0 = 0.23$) and *blue-metallic-paint2*. Note that the color-shifted variants of the original BRDFs (Sec. 5.1) are selected to match the target spectral distribution.

8 Results and Discussions

We conducted experiments on a workstation with a quad-core Intel i7 3.5GHz CPU, 32GB of memory and a GeForce GTX 660 graphics card. Interactive rendering of appearance during the design process is shown in the accompanying video.

Precomputation. In terms of each BRDF in the material library, we precompute two versions of F (Eq. 18): one is the achromatic version, which is used in the inverse solvers for fast computation; the other is the chromatic version, which is needed in large-scale appearance rendering. The size is 23.0MB for one achromatic F, and 68.9MB for its chromatic counterpart. The total size of the material library is 17.9GB.

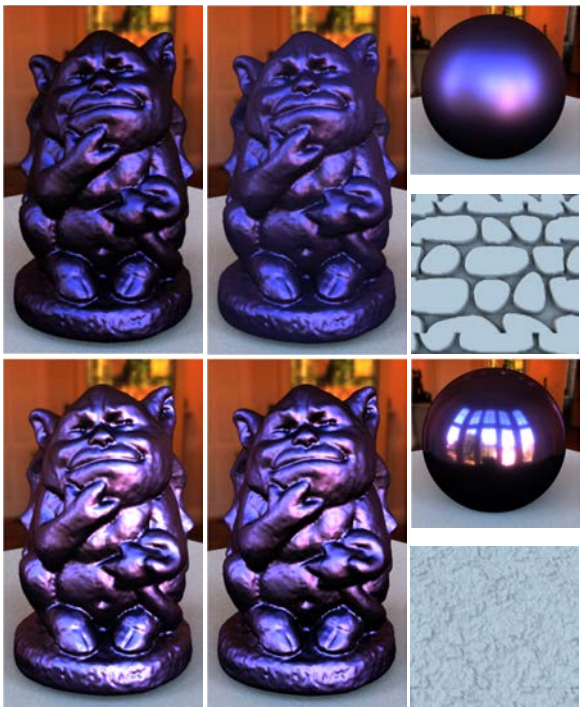

Figure 9: *Adjusting the brightness of the target appearance (measured blue-metallic-paint). Left: the target large-scale materials. Center: our approximations. Right: the small-scale materials (from top to bottom, measured blue-metallic-paint and blue-metallic-paint2) and geometry.*

For every piece of geometry in the library, we compute its BVNDF by rendering the corresponding triangular mesh on GPU using shadow mapping, for $7,744$ different lighting and view direction pairs, similar to [Wu et al. 2011]. During rendering, we also duplicate the geometry around itself as if it is tiled over a large-scale surface. Next, eight vMF lobes are used to fit the raw BVNDF data for each directional pair. This is the most computationally expensive part of the precomputation, which takes about 130 seconds for one piece of geometry. Due to the large number of geometry in our library, we distribute GPU rendering and vMF fitting on a cluster of 30 Windows-based nodes: each one has an Intel Xeon 2.13GHz CPU, 4GB RAM and dual GeForce GTX 580 graphics card. The total computation time over the cluster is 8 days and 21 hours. The size of one BVNDF is 1.2MB, and the total size of the BVNDF library is 196.6GB.

As described in Sec. 4 & 5, we perform SVD followed by CX decomposition to express materials / geometries in libraries as a linear combination of representatives. The error threshold in CX decomposition is 1.5%. The majority of time is spent on the SVD step. For the material library, it takes 2.7 minutes on the in-core SVD. For the geometry library, we first divide all geometry into 1,023 specular BVNDFs and 163,408 non-specular BVNDFs. It takes 23.0 hours to perform the random-projection-based SVD on non-specular BVNDFs. Here the bottleneck is disk-memory data transfer, due to the huge sizes of related matrices. Thus, we use a 512GB Solid State Drive to accelerate the process. In the end, we obtain 156 representative BRDFs and 597 representative BVNDFs.

Inverse Computation. At the start of our system, the representative BRDFs and BVNDFs are loaded into memory to save time for the inverse computation. In our experiments, we take 490 samples when discretizing the target appearance $\hat{f}_r$, and retrieve 10 candi-

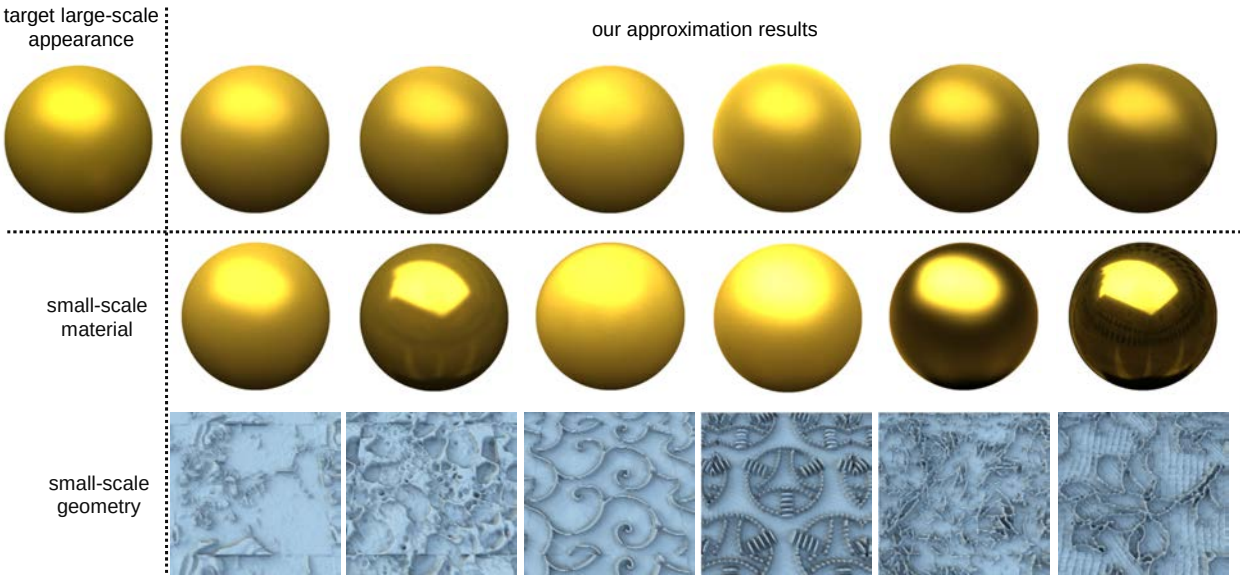

Figure 11: *Visually equivalent details. We can use our inverse solver to find a wide variety of different small-scale materials and geometries (bottom two rows), which achieve similar large-scale appearance (top row). Each approximation result in the top row is computed by combining the small-scale material and geometry shown in the same column. Here the target material is a Blinn-Phong BRDF ($n = 60$) with a Lambertian lobe. Small-scale materials are color-shifted variants of (from left to right) Ward($\alpha = 0.16$) with a Lambertian lobe, Ward($\alpha = 0.05$) with a Lambertian lobe,* pickled-oak-260*,* gold-paint*, Cook-Torrance($m = 0.19$, $F_0 = 0.23$) and* blue-metallic-paint2*.*

dates as results. Running the material solver one time takes 0.8 seconds on average. For the geometry solver, the performance is dependent on the number of BVNDF rotations. Typical timing results are 4.0 seconds for $\psi = 1$, and 18.8 seconds for $\psi = 8$. The non-linear growth in time with respect to ψ is due to the better parallel utilization when ψ is big. For the general solver, the execution time is 25.1 seconds, when 5 random initial starting points are used. We observe in experiments that the general solver usually converges after just two iterations.

User Interface. We render both the target appearance and our approximation in the inverse design system. Since the large-scale appearance varies from diffuse one to high-specular material, we importance sample the lighting directions, according to a sampled version of the BRDF. To provide quick visual feedback, the rendering is done progressively and the result is accumulated in a buffer. 32 new light rays are shot each time the rendering updates, which takes 0.2∼0.7 seconds.

Validation. To validate our inverse appearance solver, we generate large-scale appearance from small-scale BRDFs and geometry that are already in our libraries, then test if the solver can compute exactly the same results as input, or results that produce similar large-scale looks. Since the number of different combinations of materials and geometry in our libraries is huge, we randomly sample 10,000 of them as test cases. We perform three validation experiments to test the geometry solver, the material solver and the general solver (Tab. 1). The efficiency of a solver is measured by 7% relative error percentile, which is the percentage of results that have no more than 7% of relative approximation error (the measure of error divided by the measure of the target appearance, as defined in Eq. 14). We choose the 7% relative error percentile here, because it is hard to notice the difference between the target and our approximation, when the error is below this threshold, as shown in Fig. 12. When testing the general solver, we use only 5 random geometries / materials as starting points, and continue with the one that has the minimum error.

Solver	**7% Relative Error Percentile**	**Exact Recovery Rate**
Geometry	98.94%	95.79%
Material	99.57%	96.98%
General	93.31%	19.77%

Table 1: *Statistics from validation experiments.*

Overall, we achieve more than 98% success rate (i.e., 7% relative error percentile) for the geometry / material solver, and more than 93% for the general solver. Even for the exact recovery rate, we are able to obtain more than 95% for the geometry / material solver. Although the recovery rate for the general solver is about 20%, it is not necessary to use exactly the original small-scale material and geometry, for producing similar large-scale looks. Moreover, we find that the general solver is very sensitive to the initial condition, as it is essentially a local optimizer. Thus in practice, the designer could impose additional constraints on the starting points to make the solver more efficient, compared to doing random guesses.

9 Limitations and Future Work

Similar to other data-driven methods, one major limitation of our framework is that the approximation quality of results from our solvers, with respect to a given target large-scale material, is constrained by the diversity of the two precomputed libraries. This problem can be alleviated by adding more versatile materials / geometries to our libraries. Another limitation is that currently we do not support target large-scale appearance generated from spatially-varying small-scale materials (e.g., the green-yellow grooves in Fig. 9 of [Wu et al. 2011]). It will be interesting to quickly solve for multiple small-scale materials, as well as their spatial distributions over the small-scale geometry.

Due to performance concerns, our approach handles direct illumination only, following the convention in bi-scale material de-

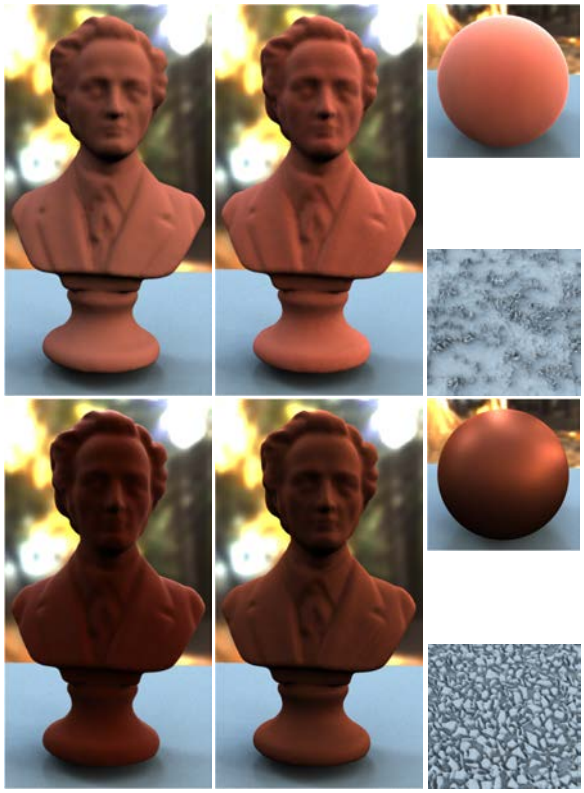

Figure 10: *Adjusting the contrast of the target appearance (measured light-brown-fabric). Left: the target large-scale materials. Center: our approximations. Right: the small-scale materials (from top to bottom, polyurethane-foam and black-oxidized-steel) and geometry.*

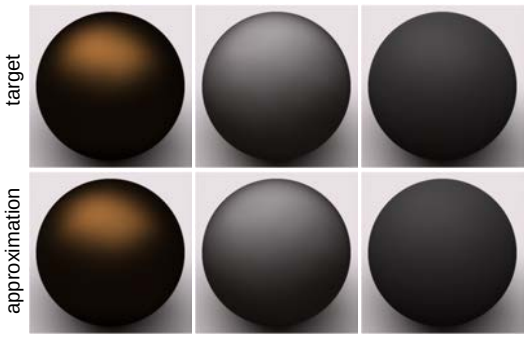

Figure 12: *Results from validation experiments. Each approximation result has the maximum relative approximation error that is less than 7% among all test cases. From left to right: results using the geometry solver, the material solver and the general solver.*

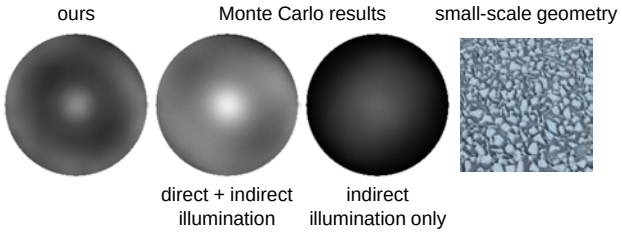

Figure 13: *Darkened large-scale appearance, due to the direct-illumination-only assumption. Here we compare three large-scale BRDF slices, obtained by fixing ω_i as the zenith. From left to right: our result, full Monte Carlo path tracing result, Monte Carlo result with indirect illumination only, and the small-scale geometry. The small-scale material is a Cook-Torrance BRDF ($m = 0.2$). Our result appears darker than the full simulation result, due to the lack of indirect illumination.*

sign [Wu et al. 2011; Iwasaki et al. 2012] and fabrication [Weyrich et al. 2009; Lan et al. 2013]. In certain cases with a material of high albedo and a geometry of significant ambient occlusion on the small-scale, this simplification may cause a darkening effect over the large-scale appearance, compared to a more expensive Monte Carlo path tracing result that includes indirect illumination. An example is shown in Fig. 13. It will be a promising future direction to take into account interreflections, while maintaining near-interactive feedback.

It will also be interesting to extend our framework to handle volumetric scattering. In addition, we would like to include specific manufacturing constraints in our system, and physically fabricate the results from our inverse solvers.

Acknowledgements

We would like to thank anonymous reviewers for help on improving the exposition, Larry Wilen for useful discussions, Dikpal Reddy for pointing to the state-of-the-art random projection technique, Xin Sun and Mingming He for preparing the height-map database, Lu Zhao for proofreading, Paul Debevec, Keith Bruns and Bernhard Vogl for environment maps used in rendering. This material is based upon work supported by the US National Science Foundation under Grant No. IIS-1064412. Hongzhi Wu was partially supported by the Fundamental Research Funds for the Central Universities (No. 2013QNA5011).

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.

BRODATZ, P. 1999. *Textures: A Photographic Album for Artists and Designers*. Dover Publications, Aug.

BRUNETON, E., AND NEYRET, F. 2012. A survey of nonlinear prefiltering methods for efficient and accurate surface shading. *IEEE Trans. Vis. Comput. Graph. 18*, 2 (Feb.), 242–260.

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.

DORSEY, J., RUSHMEIER, H., AND SILLION, F. 2007. *Digital Modeling of Material Appearance*. Morgan Kaufmann Publishers Inc.

DRINEAS, P., MAHONEY, M. W., AND MUTHUKRISHNAN, S. 2008. Relative-error CUR matrix decompositions. *SIAM Journal on Matrix Analysis and Applications 30*, 2 (Sept.), 844–881.

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.

HALKO, N., MARTINSSON, P. G., AND TROPP, J. A. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review 53*, 2 (May), 217–288.

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.

HAYS, J., AND EFROS, A. A. 2007. Scene completion using millions of photographs. *ACM Trans. Graph. 26*, 3 (July).

IWASAKI, K., DOBASHI, Y., AND NISHITA, T. 2012. Interactive bi-scale editing of highly glossy materials. *ACM Trans. Graph. 31*, 6 (Nov.), 144:1–144:7.

KURT, M., SZIRMAY-KALOS, L., AND KŘIVÁNEK, J. 2010. An anisotropic BRDF model for fitting and Monte Carlo rendering. *SIGGRAPH Comput. Graph. 44*, 1 (Feb.), 3:1–3:15.

LAN, Y., DONG, Y., PELLACINI, F., AND TONG, X. 2013. Bi-scale appearance fabrication. *ACM Trans. Graph. 32*, 4 (July), 145:1–145:12.

LAWRENCE, J., BEN-ARTZI, A., DECORO, C., MATUSIK, W., PFISTER, H., RAMAMOORTHI, R., AND RUSINKIEWICZ, S. 2006. Inverse shade trees for non-parametric material representation and editing. In *Proc. of SIGGRAPH 2006*, ACM, 735–745.

MALZBENDER, T., SAMADANI, R., SCHER, S., CRUME, A., DUNN, D., AND DAVIS, J. 2012. Printing reflectance functions. *ACM Trans. Graph. 31*, 3 (June), 20:1–20:11.

MARSCHNER, S. R., WESTIN, S. H., LAFORTUNE, E. P. F., TORRANCE, K. E., AND GREENBERG, D. P. 1999. Image-based BRDF measurement including human skin. In *Proc. of EGWR'99*, 131–144.

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

MATUSIK, W., AJDIN, B., GU, J., LAWRENCE, J., LENSCH, H. P. A., PELLACINI, F., AND RUSINKIEWICZ, S. 2009. Printing spatially-varying reflectance. *ACM Trans. on Graph. 28*, 5 (Dec.), 128:1–128:9.

MAYANG. 2013. Mayang's Free Texture Library `http://www.mayang.com/textures/`.

NGAN, A., DURAND, F., AND MATUSIK, W. 2005. Experimental analysis of BRDF models. In *Proc. of EGSR'05*, 117–126.

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. on Graph. 26*, 3 (July), 54:1–54:9.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. A signal-processing framework for inverse rendering. In *Proc. of SIGGRAPH 2001*, ACM, New York, NY, USA, 117–128.

ROMEIRO, F., AND ZICKLER, T. 2010. Blind reflectometry. In *Proc. of ECCV'10*, Springer-Verlag, Berlin, Heidelberg, ECCV'10, 45–58.

RUSINKIEWICZ, S. M. 1998. A new change of variables for efficient BRDF representation. In *Proc. of EGWR'98*, 11–22.

WANG, C.-P., SNAVELY, N., AND MARSCHNER, S. 2011. Estimating dual-scale properties of glossy surfaces from step-edge lighting. *ACM Trans. Graph. 30*, 6 (Dec.), 172:1–172:12.

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. 2011. Physically-based interactive bi-scale material design. *ACM Trans. Graph. 30*, 6 (Dec.), 145:1–145:10.

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 (Aug.), 44:1–44:10.

Appendix

We show in this section how to represent F as a 4D function. First, we perform a change of parameters for f (Eq. 2)

$$\begin{aligned} & f(\boldsymbol{n}, \boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \\ =& f(R(\boldsymbol{n}), R(\boldsymbol{\omega}_i), R(\boldsymbol{\omega}_o)) \\ =& f(R(\boldsymbol{n}), (\sin\theta_h, 0, \cos\theta_h), (-\sin\theta_h, 0, \cos\theta_h)) \\ \triangleq& f'(R(\boldsymbol{n}), \theta_h). \end{aligned} \tag{17}$$

Our parameters are similar to the ones in the half-angle parameterization [Rusinkiewicz 1998]. θ_h is half of the angle between $\boldsymbol{\omega}_i$ and $\boldsymbol{\omega}_o$, defined as $\theta_h = \frac{1}{2}\cos^{-1}(\boldsymbol{\omega}_i \cdot \boldsymbol{\omega}_o)$. R is a transformation, which rotates $\boldsymbol{\omega}_i$ to $(\sin\theta_h, 0, \cos\theta_h)$, and $\boldsymbol{\omega}_o$ to $(-\sin\theta_h, 0, \cos\theta_h)$.

Substituting Eq. 4 and 17 into Eq. 1, we have

$$\begin{aligned} & \overline{f}_r(\boldsymbol{\omega}_i, \boldsymbol{\omega}_o) \\ =& \int_{S^2} f'(R(\boldsymbol{n}), \theta_h) \sum_j \alpha_j \rho(\boldsymbol{n}; \kappa_j, \boldsymbol{\mu}_j) d\boldsymbol{n} \\ =& \sum_j \alpha_j \left(\int_{S^2} f'(R(\boldsymbol{n}), \theta_h) \rho(\boldsymbol{n}; \kappa_j, \boldsymbol{\mu}_j) d\boldsymbol{n} \right) \\ =& \sum_j \alpha_j \left(\int_{S^2} f'(R(\boldsymbol{n}), \theta_h) \rho(R(\boldsymbol{n}); \kappa_j, R(\boldsymbol{\mu}_j)) dR(\boldsymbol{n}) \right) \\ =& \sum_j \alpha_j \left(\int_{S^2} f'(\boldsymbol{n}, \theta_h) \rho(\boldsymbol{n}; \kappa_j, R(\boldsymbol{\mu}_j)) d\boldsymbol{n} \right) \\ =& \sum_j \alpha_j F(\theta_h, \kappa_j, R(\boldsymbol{\mu}_j)), \end{aligned}$$

where

$$F(\theta_h, \kappa, \boldsymbol{\mu}) = \int_{S^2} f'(\boldsymbol{n}, \theta_h) \rho(\boldsymbol{n}; \kappa, \boldsymbol{\mu}) d\boldsymbol{n}. \tag{18}$$

Now F has only 4 dimensions, which is significantly lower than its original definition in Eq. 7.