Interactive Editing and Modeling of Bidirectional Texture Functions

Jan Kautz
University College London

Solomon Boulos
University of Utah

Frédo Durand
MIT – CSAIL

Abstract

While measured Bidirectional Texture Functions (BTF) enable impressive realism in material appearance, they offer little control, which limits their use for content creation. In this work, we interactively manipulate BTFs and create new BTFs from flat textures. We present an out-of-core approach to manage the size of BTFs and introduce new editing operations that modify the appearance of a material. These tools achieve their full potential when selectively applied to subsets of the BTF through the use of new selection operators. We further analyze the use of our editing operators for the modification of important visual characteristics such as highlights, roughness, and fuzziness. Results compare favorably to the direct alteration of micro-geometry and reflectances of synthetic reference data.

Keywords: BTFs, editing, material appearance

1 Introduction

Manipulating material appearance remains challenging for content creators. For complex materials, Dana et al. [1999] introduced the *bidirectional texture function* (BTF), a sampled 6D data structure parameterized by position (x,y) as well as light (ω_i) and view (ω_o) direction: $b(x,y;\omega_i,\omega_o)$. Essentially, BTFs are textures that vary with view and light direction and are acquired by taking photographs of a material under many view/light configurations (ideally an orthographic view and directional white light). BTFs can represent a rich class of appearances: from simple materials like plastic to complex ones, such as cloth, that have important mesostructures and exhibit complex lighting effects such as shadowing and masking. This usually makes a BTF slice unlike any analytical BRDF model. Renderings done with acquired BTFs are extremely realistic since they contain all the subtleties of real materials.

A current major limitation of BTFs is that the user is limited to the measured data and cannot easily modify the material appearance. BTF editing is vital to make BTFs a practical appearance model and to offer better return on investment from BTF acquisition by enabling different appearances from the same measurement. However, editing BTFs is challenging since the input data is high-dimensional and does not directly encode information about the material, such as geometry or reflectance.

We introduce a set of *editing operators* that enable the manipulation of view- and light-dependent BTF effects (Fig. 1). For effective editing, these operators can be restricted to work on subsets of the BTF, e.g., shadow areas, using *selections*. Operators and selections work directly on the raw BTF data without reverting to an approximate representation, but may leverage material-specific information. Furthermore, we propose an out-of-core editing architecture that enables interactive BTF manipulation despite large data sizes and computation times, called *BTFShop*. The combination of these three components allows for effective and interactive BTF editing as well as modeling from simple 2D textures.

Figure 1: *Successive edits applied to knitwear using our system.*

Our approach is inspired by photo-editing tools such as Adobe Photoshop. We focus on visually plausible, albeit not necessarily physically correct editing. We exploit inverse algorithms such as shape from shadow when possible but often fall back to much simpler heuristics to modify an aspect of a material. This focus on simple phenomenological manipulation is both the strength and the limitation of our technique. On one hand, we preserve the material's richness because we directly manipulate the BTF data. Furthermore, the parameters of our operators usually have a direct mapping to visual characteristics such as tone distribution. On the other hand, we cannot expect to precisely modify the full range of physical effects, as we do not model the physics of the underlying material. Our results show that despite such limitations, we enable a wide range of material modifications.

1.1 Related Work

BTFs were introduced by Dana et al. [1999] and a good overview is given by Müller et al. [2005]. Our work is orthogonal and complementary to the acquisition [Dana et al. 1999; Sattler et al. 2003; Koudelka et al. 2003; Neubeck et al. 2005; Ngan and Durand 2006], compression [Sattler et al. 2003; Koudelka et al. 2003; Vasilescu and Terzopoulos 2004], rendering [Sattler et al. 2003; Suykens et al. 2003; Meseth et al. 2004] and synthesis [Tong et al. 2002; Koudelka

et al. 2003; Haindl and Hatka 2005] of BTFs. Our system takes acquired BTF data as its input and allows a user to modify the data. Any BTF rendering or synthesis technique can then be used.

BTFs are traditionally encoded as raw data, but material-specific parametric representations exist [Dana and Nayar 1998; Ginneken et al. 1999; Cula and Dana 2001; Haindl et al. 2005; Magda and Kriegman 2006]. We focus on general BTFs and do not use an approximate representation of BTFs.

Little attention has been paid to BTF editing. Zhou et al. [2005] "paint" BTF patches onto surfaces with seamless blending between different BTF samples. Editing of the BTFs themselves is not supported. Dong et al. [2005] proposed self-similarity based editing for bump maps and BTFs (encoding only light variation), and Dischler et al. [1999] introduced a system where a user interactively edits the 3D meso-structure of a surface. Our approach is different; we enable the user to edit all major visual phenomena that are encoded in a BTF and are not limited to BRDFs or other subsets of BTFs.

Editing BRDF parameters is very common and most 3D content creation applications (like Maya or 3D Studio Max) have a material editor. Interactive editing of BRDFs is possible [Colbert et al. 2006], even for measured spatially-varying BRDFs [Lawrence et al. 2006]. The latter technique also avoids fitting a parametric model and instead works with acquired data. However, these material editors do not allow users to create or modify complex materials, such as wool, which cannot be represented through a BRDF.

Part of our work is inspired by studies that relate material properties with simple image characteristics such as contrast, [Pont and Koenderink 2002], pixel histogram [Dana and Nayar 1998; Ginneken et al. 1999; Leung and Malik 1997], and sharpness [Fleming et al. 2004]. Recent work has also shown that the appearance of a material in a photograph can be significantly altered with simple editing [Adelson 2001; Khan et al. 2006]. We use these observations and enable the user to change a BTF's appearance through operators and selections that act on such image characteristics and evaluate their effect with synthetic reference data.

1.2 Overview

A BTF can represent visually rich materials because it encapsulates the result of the complex interaction of lighting, normals, surface reflectance, shadowing and masking, parallax, and inter-reflections. Essentially, each spatial 2D slice of a BTF corresponds to the image of a 3D scene (the material's micro-geometry and local BRDF) rendered with the full rendering equation for a given directional lighting ω_i and a given view direction ω_o. A BTF texel with spatial coordinate p and directions ω_i and ω_o corresponds to

$$b(p;\omega_i,\omega_o) = f_r(p',\mathbf{R}_{\mathbf{n}_{p'}}(\omega_i),\mathbf{R}_{\mathbf{n}_{p'}}(\omega_o))(\mathbf{n}_{p'}\cdot\omega_i)\cdot V(p',\omega_i) + \int b(q';\omega_i,-\omega_s)d\omega_s \quad (1)$$

where $p' = \mathrm{ray}(p,\omega_o)$ is the intersection point closest to the viewer (which incorporates masking effects) in direction ω_o (which accounts for parallax), f_r is the BRDF, $\mathbf{R}$ rotates the global directions $\omega_{i/o}$ into the local coordinate system at p' with normal $\mathbf{n}_{p'}$, V is the visibility (corresponding to shadowing), and $q' = \mathrm{ray}(p',\omega_s)$ is the closest visible point from p' in direction ω_s. This accounts for direct as well as indirect illumination and corresponds to the rendering equation for a white directional light source in direction ω_i. In fact, measured BTFs also implicitly include effects such as subsurface scattering.

The equation illustrates why BTF editing is challenging and why we do not try to invert it. The individual components are not directly accessible in the data, and depend heavily on each other, e.g., the BRDF interacts with the local normal, which in turn depends on the geometry. Reconstructing these components is a very challenging inverse rendering task. Furthermore, we do not think that it is easy for a user to manipulate such micro-scene properties to obtain a desired material appearance.

Original | Blue | Dull | Diff. Painting | Angular Blur Tool

Figure 2: *A synthetic wallpaper BTF (computed using PBRT) is changed to be more bluish and less glossy by re-creating the BTF (again using PBRT) as well as using our editing tools: more bluish with the differential painting tool (hue shift by 150°; Sec. 2.1.1); reduced gloss with the angular filter (35° blur; Sec. 2.1.2). The differences between our approach and reference BTFs are minor.*

Our approach to BTF editing is influenced by previous work on material appearance and relies on simple manipulations that modify the raw data directly. As such, our editing tools are often approximate and not necessarily physically correct. Yet, we demonstrate that visually meaningful modifications that correspond to common BTF effects are possible. For validation, we compare our results with synthetic reference BTFs, which we generate by directly rendering micro-geometry within PBRT [Pharr and Humphreys 2004] (thousands of renderings each with a different view and light configuration). We aim to enable editing of all major components of a BTF: *local shading* (reflectance, roughness, shading), *geometry* (geometric structure, overall height, parallax), *shadowing and masking* (removal, modification, creation), and *global effects* (inter-reflections, translucency, fuzziness).

We introduce our BTF editing operators in Section 2. The full potential of these operators is achieved, when applied to subsets of the BTF through the use of selection operators, which we detail in Section 3. In Section 4, we present the system challenges that interactive BTF editing poses and our own solution: *BTFShop*. Finally, we show a variety of editing examples and demonstrate that this approach can also be used to interactively create new BTFs from simple texture maps (see Sec. 5).

2 BTF Editing Operators

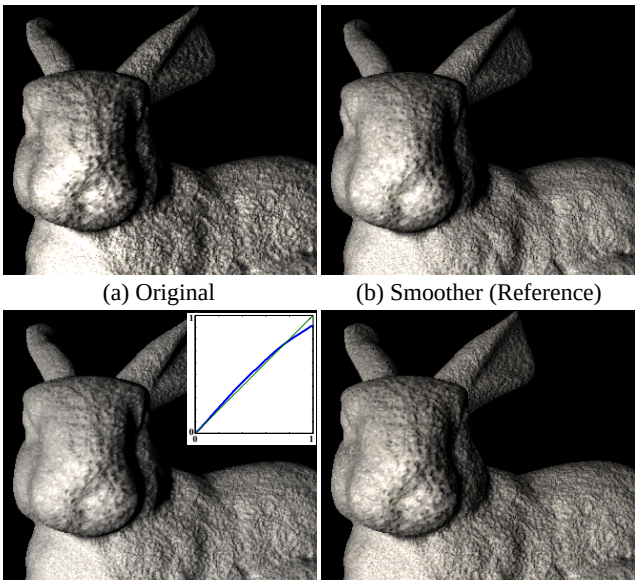
We introduce editing operators that address three major types of BTF effects: shading, shadowing, and parallax. These operators are designed to enable effective modification and creation of BTFs, yet are simple enough to allow interactive editing. We also provide simpler editing tools inspired by traditional photo editing, and show that they are surprisingly effective at modifying material properties such as roughness and fuzziness, especially when applied to subsets of the BTF.

2.1 Shading

Shading modification is challenging due to interplay with other effects, such as shadowing and inter-reflections. Care must be taken not to remove important subtleties that come from this interplay. We therefore perform modifications *relative* to existing data. We first present the extension of simple image editing tools to the 6D BTF domain and assess their effectiveness at modifying material appearance, before introducing new BTF-specific operators.

2.1.1 Tone and Color Manipulation for Editing Shading

We propose a differential painting tool, which is similar to texture painting, with the notable difference that changes performed in a certain view/light configuration (usually top view and fully lit) must be propagated to all other spatial slices of the BTF while respecting effects such as shading and shadowing. We use a differential update strategy, where the *change* from the old value to the new value (taken from the original and the modified slice) is applied to all tex-



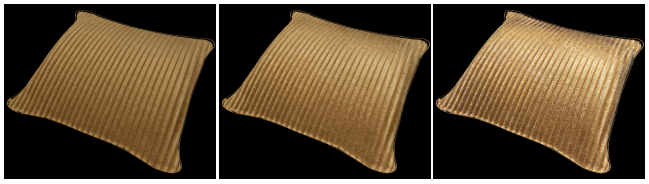
(a) Original (b) Smoother (Reference)

(c) Smoother (Curve Tool) (d) Smoother (L./V. Stretch)

Figure 3: *Roughness Change. A synthetic stone BTF computed using PBRT is applied to the Stanford bunny (a). It is changed in (b) to be less rough by re-creating the BTF with PBRT using a smoother height field. The same effect is achieved (c) with the curve tool (Sec. 2.1.1). Qualitatively, the curve brightens dark values, and darkens bright values, reducing contrast. The light/view stretch tool (Sec. 2.2) also allows for roughness changes ($s_{i/o} = 0.4$), as seen in (d). Both tools achieve results similar to the reference.*

els at the same spatial location. In particular, we transform the data into the HSV color space and propagate the difference of the hue component and the ratio of the saturation and value component to all the other texels. This ensures that shading and shadowing information is preserved; highlight pixels can be left untouched with the appropriate selection. The fourth image of Figure 2 shows that this differential albedo painting is virtually indistinguishable from a synthetic reference BTF (second image), which was obtained by modifying the reflectance of the 3D model used to generate the original BTF. Additionally, we provide a hue/saturation/lightness operator, where the user directly sets the changes that are to be propagated, instead of computing them from a modified BTF slice.

We now adapt tools that modify the distribution of intensities (*tone manipulators*) to enable material changes. In particular, we adapt the ubiquitous "curve" tool used in photo editing to finely control contrast and brightness distribution. Our curve tool changes the color distribution (per channel or simultaneously on all), by remapping each texel of the BTF according to a smooth curve (Catmull-Rom spline). We build on work by Pont and Koenderink [2002] who showed that the distribution of shadow pixels and the contrast of a texture as a function of view angle indicates the *roughness* of a material. We demonstrate that applying tone manipulation to a BTF effectively modifies roughness and hardness without knowledge of the BTF's geometry or shading. We use a virtual stone BTF example generated from synthetic micro-geometry and a texture map (Fig. 3), whose roughness is decreased by reducing the height of the synthetic height field. The same perceived change in roughness can be achieved by applying the curve tool to the original stone BTF (Fig. 3c). A major advantage of the curve tool is that it does not require the solution of inverse problems, such as shape from shading, which is prone to errors when applied to complex materials. Yet, it enables the modification of material appearance that looks like a modification of the micro-geometry. We also provide a brightness/contrast tool, which is easier to use for brightness-only changes.



Less Specular Original More Specular

Figure 4: *The specularity was changed using the angular blur (45°, applied twice) and angular sharpen filter (45°, applied twice).*

Appropriate manipulation of intensity distribution in combination with selections will also be demonstrated to yield changes in translucency, fuzziness, inter-reflections, and highlights (Sec. 3).

Discussion Differential painting assumes that there are no strong inter-reflections. If this is not the case, inter-reflections will be modified along with the local shading, instead of only the local shading. Roughness can only be modified with the curve tool when it is caused by small bumps; the geometric detail of bigger bumps is too obvious for the curve tool to work. Furthermore, the material needs to have a certain roughness to begin with, otherwise the curve tool cannot change the distribution of light vs. dark pixels.

2.1.2 Angular Blur and Sharpen for Editing Specularity

We introduce BTF-specific versions of the traditional blur/sharpen operations that yield very different effects from their spatial counterparts and alter the specularity of materials. Our *angular blur* and *angular sharpen* apply a spherical kernel H to the light-directional samples for each spatial location:

$$b_{x,y,\omega_o}(\omega_i) := (b_{x,y,\omega_o} \otimes H)(\omega_i).$$

We build on theoretical results showing that the specularity of a material corresponds to its light-angular sharpness [Ramamoorthi and Hanrahan 2001]. More specular materials exhibit higher frequencies in this domain than less specular materials. This is independent of underlying normals and enables modification of highlights using blur and sharpen operations in the angular domain without extracting BRDF and normal information (Fig. 4).

The synthetic wallpaper BTF from Figure 2 is used to compare this operator with reference data. The greenish leaves are made less glossy in the third image using PBRT. The rightmost image demonstrates the same modification using the angular blur operation (using an additional reduction in brightness with the tone manipulator as well). There are virtually no differences between the reference BTF and our modification.

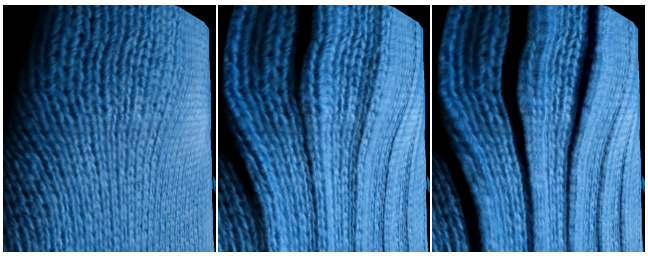
Discussion Ideally, this operator would only modify the underlying BRDF. However, visibility as well as indirect illumination is filtered as well (see Eq. 1). In practice, this makes the material appear softer, as shadow boundaries are blurred.

2.1.3 Local Frame Rotation

The underlying micro- and meso-structure of a material influences the *shading* due to variations in local surface frame. Our *Local Frame Rotation* operator rotates the local frame at each spatial location according to a user-specified height/normal map. Looking up BTF data relative to the new local frame changes the shading in a manner similar to normal mapping:

$$b_{x,y}(\omega_i, \omega_o) := b_{x,y}(\mathbf{R}_{\mathbf{n}(x,y)}(\omega_i), \mathbf{R}_{\mathbf{n}(x,y)}(\omega_o)),$$

where $\mathbf{R}_{\mathbf{n}(x,y)}$ is the rotation matrix that rotates the light or view direction into the new local coordinate frame given by the new user-supplied normal $\mathbf{n}(x,y)$. Note that the true normals or reflectances of the material are not needed since this is a relative operator. Figure 5 middle shows an example, where we have added a bump to the wool BTF using this operator.



Original | Local Frame Modification | Adding Shadows & Parallax

Figure 5: *Small bumps are added to the original wool BTF (rendered on a curved object) using the local frame tool and a user-supplied bump map (Sec. 2.1.3). Then, appropriate shadows are added (Sec. 2.2) and the underlying structure is changed to account for the bump (Sec. 2.3).*

Discussion Local frame rotation does not modify inter-reflections, shadowing, and masking accordingly. If these effects cannot be assumed minor, explicit modification by the user is required (Fig. 5).

2.2 Shadowing and Masking

Shadows are an integral part of BTFs, caused by the BTF's geometry. Desired editing operations comprise the removal, modification and creation of shadows.

Shadow Removal Shadow removal is an important, yet difficult operation. For instance, when modifying BTF geometry it is needed to fix newly unoccluded areas. Shadow areas need to be filled in a seamless manner, blending in with the surrounding area, but efficiently enough to allow for interactive editing. We provide an automatic *Shadow Filler* that leverages information from other illumination directions to fill shadow regions, without requiring reconstruction of the material's geometry.

The operator works in two stages. It first fills selected shadow regions (see Sec. 3 for the shadow selection operator) with data from corresponding top-lit BTF slices: $b_{x,y,\omega_o}(\omega_i) := b_{x,y,\omega_o}((0,0,1)^T)$. By construction, this data does not contain any shadows, which effectively removes shadows from the selected areas. However, shading differences appear due to the difference in lighting directions, see Figure 6. In a second step, the operator adjusts the average brightness and saturation of the shadowed regions to be equal to the average brightness and saturation of the unshadowed regions (separately for each spatial BTF slice), removing any shading differences; see again Figure 6. It is possible to restrict the unshadowed region further, such that the adjustment is performed with respect to similar regions and not overall averages.

Shadow Modification and Creation The manipulation and creation of shadows does not require a separate operator. Instead, a special selection operation in combination with tone manipulation or the shadow filler is used, see Section 3.

Light/View Stretch for Editing Thickness Shadowing and masking are critical effects of micro-geometry and are a major component that makes BTFs different from spatially-varying BRDFs. The precise geometric factors involved are complex but we propose simple heuristic tools that enable convincing modifications. Our *Light/View Stretch* virtually modifies the shadowing and/or masking by stretching the light- and/or view-dependent directional hemisphere of BTF values:

$$b_{x,y}(\omega_i,\omega_o) := b_{x,y}(\mathbf{S}_i(\omega_i),\mathbf{S}_o(\omega_o)),$$

with $\mathbf{S}_{i/o}(\omega) = (\omega_x,\omega_y,\omega_z + s_{i/o})^T$ and $s_{i/o}$ being the stretch parameter. This is similar to the local frame rotation, but the hemisphere of BTF values is stretched instead of tilted. This operator is used to modify the perceived "thickness" of a BTF (view-dependent stretch), which is independent of the underlying geometric structure or complexity. See adjacent figure, where we show the view-dependent BTF values for a fixed light direction and how stretch reduces the apparent depth.

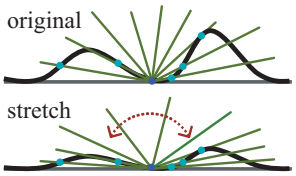


Figure 7 shows an example. A grass BTF made of little bent cylinders is created using PBRT. The thickness of the BTF is halved using the stretch tool, which compares favorably to re-creating the BTF with scaled cylinders.

When using the stretch operator, the light- and view-dependent distribution of shadow pixels changes accordingly, making a surface appear rougher or softer. We confirm this in Figure 3d, where we stretched the view and light directions, which compares favorably to the reference BTF.

Discussion The shadow removal tool only adjusts average brightness and saturation of the copied samples, which is not always sufficient to replicate the shading of the surrounding area. E.g., materials with strong color bleeding are difficult to handle due to color changes. Furthermore, the original data is assumed to be accurate, but in practice there can be noticeable lighting variations within a spatial BTF slice due to an imperfect acquisition setup. Figure 8 demonstrates this with the Lego BTF; strong lighting variations within texture slices prevent accurate adjustments. Yet, we have found the shadow removal tool to work well if its underlying assumptions are met. However, it is important to precisely select the shadow areas — shadows that are not selected cannot be removed.

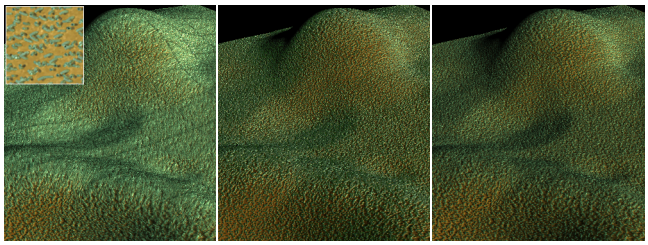
The stretch operator is effective but inherently modifies the resulting shading, as the light and/or view direction is altered when looking up BTF values. Said differently, our tool cannot transform geometry and normals consistently because we do not have such information. For typical modifications this change in shading is minor, see Fig. 7 and 3d. It is most noticeable with glossy materials, where the highlight shifts position, see Fig. 8. While the stretch operator can also be used for modifying roughness (Fig. 3d), the curve tool allows for better fine-scale manipulation of roughness.

Original | Shadows Removed | Closeup

Figure 6: *The shadows in the BTF are removed with our automated shadow filler (after shadow selection). The automatic process proceeds by filling in shadow areas from fully-lit slices, which are then adjusted in brightness and saturation (see right column).*

2.3 Geometric Modification and Parallax

The appearance of many materials depends on their meso-structure, especially for materials that are not very thin. These materials may then exhibit strong parallax, i.e., texels that share a given x,y spatial location in a BTF do not necessarily correspond to the same geometric point on the micro-geometry. This discrepancy needs to be compensated for during spatial editing. Our approach is to provide



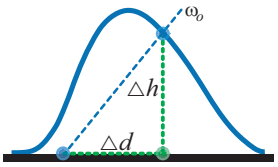
(a) Original (b) Halved Height (Reference) (c) Halved Height (BTFShop)

Figure 7: *The original grass BTF (a) is created with PBRT. The grass is scaled vertically by 1/2 and the BTF is re-created with PBRT (b). The same effect is achieved using the stretch tool ($s_o = 0.4$), see (c).*

a pair of operators that can be used for geometric modification as well as parallax compensation.

Height Field Extraction For these operators to work, we first need to extract a height field approximation of the underlying meso-structure. We exploit the fact that BTFs with significant parallax exhibit strong shadowing effects and use the method by Daum and Dudek [1998] to compute a height field from shadows. This algorithm proceeds as follows. The light-varying texture slices from the BTF (top view) need to be segmented into shadowed/unshadowed regions, which is assisted by the user using the shadow selection tool (see next section). An initial, planar surface estimate is then iteratively refined by "carving" the shadow regions of each segmented texture along the corresponding light direction into the estimate. This results in an approximation of the underlying geometry. The resulting height field is also useful for several selection operators, see next section.

Parallax Operators The *Parallax Unwarp* operator remaps the BTF data according to this height field in a fashion similar to view warping. Each stored BTF sample is offset by Δd according to the height Δh, which aligns the resulting BTF slices spatially, allowing for safe spatial editing. Note that disocclusion issues are not critical here, as BTFs commonly represent thin materials and because the data is intended to be warped back. The *Parallax Warp* manipulator maps the BTF data back according to its original height field. BTF samples are offset back by $-\Delta d$, which re-introduces parallax.



Geometric Modification The parallax operators allow a user to impose a new geometric structure by providing a different height field during the warping step, see Figure 5 right. This also introduces the correct *masking* (occlusion from viewpoint) in the modified BTF.

Parallax Compensation For very thin or fuzzy materials, parallax is not a critical factor and spatial editing can be performed based on the top view. Other materials such as the Lego BTF exhibit more significant relief and parallax must be compensated for. In Figure 9, we show an example of editing the Lego BTF with warping. As expected, parallax compensation is necessary here. Even though parallax compensation assumes heightfield-like BTFs, it works well even when the geometry is more complex. See Figure 14, where it is applied to wool.

Discussion Inspecting commonly available BTFs (see throughout the paper) shows that the underlying meso-structure of a BTF is usually thin and heightfield-like, which makes the heightfield-assumption valid. In fact, most examples shown in this paper were done without parallax compensation, as the difference is barely noticeable in most cases. However, even with parallax compensation

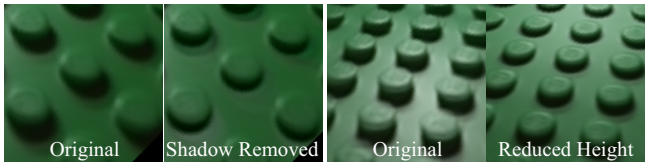


Figure 8: Left: *filling in the shadow of the Lego BTF exhibits small artifacts, due to shading variations in the BTF data.* Right: *the stretch tool effectively changes the height of the Lego ($s_o = 0.8$), but also modifies shading.*

(a) Original (b) Unwarp (c) Edit (d) Rewarp

Figure 9: *Coloring one dot of the Lego. Parallax effects, see spatial BTF slice in (a), require the BTF to be unwarped according to its height field, yielding (b). A spatial selection can now be used to color the dot consistently for all views (c). Parallax is reintroduced according to the height field producing the final result (d).*

it is difficult to manipulate steep areas, such as the sides of the little bumps on the Lego, as they cover only a small area in the parallax-corrected BTF slices, see Fig. 9b and c. Furthmore, modifying the geometry does not modify any dependent effects, such as shading or inter-reflections, and needs to be dealt with separately by the user.

2.4 Miscellaneous

Our *Copy & Paste* operator is similar to image editing. In the case of BTFs, a spatial region is copied and pasted for all view/light combination. Attention must be paid to parallax and shadowing, as described above. Figure 1 shows an example, where we have copied over parts of one BTF to create a different material.

3 Selections

The operators described in the previous section achieve their full potential when they are selectively applied to subsets of the BTF through *selections*, which are central to our approach. Similar to image editing, a selection is a per-pixel value between 0 and 1 that determines how much subsequent operations affect each sample:

$$b^{\text{new}}(\cdot) := (1 - s(\cdot))b^{\text{old}}(\cdot) + s(\cdot) f(b^{\text{old}}(\cdot)),$$

where s is the value of the selection, f is the operator, and "$\cdot$" stands for the positions and directions. The use of continuous values is important to enable feathering and smooth gradation of the effect of an operation in either space and/or angle.

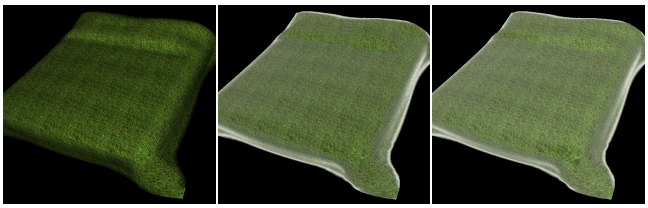
We define two types of basic selection operators: domain-driven and data-driven. Domain-driven selections choose subsets of the spatial or directional domain, while the data-driven selections choose samples based on their values. Combining and extending simple selections using Boolean and morphological operations in the form of a *selection tree* can greatly increase selection flexibility.

3.1 Selection Tree

We organize selections into selection trees in order to facilitate the combination of selections.

Boolean Combination Selections can be combined using *unions* and *intersections*, as well as be *subtracted* from each other. Furthermore, selections can be *inverted*.

Morphological Operation Morphological operations work on a single selection and modify its content. *Feathering* smoothes a selection by convolving it with a Gaussian kernel. We provide both angular and spatial feathering.



Original Fuzzy (Reference) Fuzzy (BTFShop)

Figure 10: *A bedspread BTF (computed using PBRT) is changed to include asperity scattering (fine fibers are added). We make the same change using our angular range selection (Sec. 3.2) and brightness tool (select view&light elevation angles* $90^\circ \rightarrow 0^\circ$*, increase brightness by* .3*; select view&light elevation angles* $90^\circ \rightarrow 60^\circ$*, increase brightness by* .6*). The results are very similar.*

Erosion and *Dilation* allows a user to shrink or grow a selection spatially. We extend erosion and dilation, such that it can be performed in light- or view-dependent way: a selection will be eroded or dilated only along the light- or view-direction for a given BTF slice, which is achieved using oriented structuring elements (similar to filter kernels). Light-dependent dilation of a shadow selection in combination with brightness reduction permits the enlargement of a shadow region. Similarly, light-dependent erosion and the shadow removal tool enable shrinking of a shadow region.

3.2 Domain-Driven Selections

Spatial Mask A *Spatial Mask* can be used to make detailed spatial selections. It selects the same region for all view and light directions. Combining it with the warp/unwarp operators accounts for parallax. The mask can be painted by the user, or selected based on color-similarity (using a spatial BTF slice, e.g., the top view). It is commonly used when editing patterns (Fig. 2).

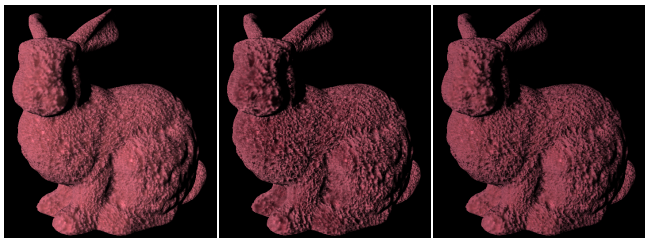
Angular Range An *Angular Range* selection enables the user to select certain light- or view-angles. This can either be a cone of directions (with a smooth falloff) or intervals along azimuth and elevation (Fig. 12 upper right).

The angular range selection is especially useful for editing *fuzzy* materials. Fuzzy materials attenuate rather than mask or shadow parts of a material, due to light scattering. An effect that is most notable at grazing angle, where it can introduce a "glow", called asperity scattering [Koenderink and Pont 2003]. It is correlated with brightness and low contrast at grazing view and light angles. Figure 10 shows a comparison with reference data, where we have changed a bedspread to be more fuzzy by adding small fibers using PBRT. Our operators allow us to add fuzziness by first selecting the intersection of light and view grazing angles (with a smooth falloff) and then increasing brightness. The results are very similar, even though our operators are not physically based.

BRDF The *BRDF* selection selects all pixels according to a user-specified BRDF f_r, which can also be relative to a normal map, i.e., $s_{x,y}(\omega_i,\omega_o) = \pi f_r(\mathbf{R}_{\mathbf{n}(x,y)}(\omega_i),\mathbf{R}_{\mathbf{n}(x,y)}(\omega_o))\cos\theta_i$. Intuitively speaking, we evaluate the user-defined BRDF at each texel and use its result as the selection value. This operation is particularly useful in creating a BTF from a single photograph. Such a BTF has initially no view- or light-dependence and this selection can be used to introduce the BRDF to the data. For example, a user would select all pixels according to the specular component of a BRDF and then increase brightness, effectively creating a specular highlight (see Section 5).

3.3 Data-Driven Selections

Thresholding The *Thresholding* selection permits the selection of all pixels with intensity values above or below a certain threshold (with smooth falloff). This is commonly used to select highlight or diffuse texels. No information about the underlying geometry or reflectance is needed. However, this operator assumes high-dynamic range BTF data; otherwise, highlights cannot be clearly separated. This operator was used for Figure 2, where we selected highlight vs. diffuse pixels before modifying them. The selection process, i.e., finding the right threshold, only takes a few seconds.



Original Solid (Reference) Solid (BTFShop)

Figure 11: *A translucent sponge material (computed using PBRT) is modified to be solid (subsurface scattering turned off). We make the same change using our editing operators (right). After selecting shadows (smooth threshold* $t = .8 \rightarrow .85$*), we decrease brightness by .2, shift the hue by* -6° *and decrease saturation by .15, which reduces brightness and undoes the color shift.*

Shadow Area The *Shadow Area* selection finds shadow areas by looking at the ratio of the frontal, fully lit spatial slice and each of the other spatial slices:

$$s_{\omega_i,\omega_o}(x,y) = \begin{cases} 0 & \text{if } b_{\omega_i,\omega_i}(x,y)/\left(S(\omega_i,\omega_o)\cdot b_{\omega_i,\omega_o}(x,y)\right) \geq t, \\ 1 & \text{if } b_{\omega_i,\omega_i}(x,y)/\left(S(\omega_i,\omega_o)\cdot b_{\omega_i,\omega_o}(x,y)\right) < t. \end{cases}$$

Each spatial slice is scaled by the factor $S(\omega_i,\omega_o)$ such that its brightness is equal to the frontal, fully-lit slice. This factor is different for each texture slice, hence the dependency on ω_i and ω_o. The threshold t is used to determine shadow/non-shadow areas. Fractional shadow selection is enabled using a smooth step function instead of a binary decision. This selection assumes that shadows darken an area more than pure shading would. In case of strong inter-reflections or subsurface scattering this assumption may be violated and accurate shadow selection becomes difficult.

This is an important selection operator and is mainly used for shadow removal and translucency manipulation. In order to perform shadow removal, the output of this selection operator serves as input to the filler, as has already been shown in Fig. 6 (with a smooth threshold $t = .7 \rightarrow .95$).

The translucency of a material is related to the lightness of the shadow areas [Fleming et al. 2004]. Lighter shadows occur in more translucent materials, as light travels inside the material into the shadow regions, whereas darker shadows appear in solid materials. Additionally, translucency influences the hardness of shadow boundaries. We demonstrate translucency modification in Figure 11. An original translucent sponge is made more solid by selecting shadow areas and decreasing brightness. It compares well to re-generating the sponge without subsurface scattering in PBRT. Minor differences are noticeable on the left side of the bunny, where strong subsurface scattering violates the shadow selection assumption and hence not all shadow regions are selected accurately.

Height Field The *Height Field* operator selects all areas that are above or below a certain height threshold. This operator can be used to change the amount of inter-reflections in a material, as the most noticeable change due to inter-reflections is the increase in brightness in concavities. We use this observation in Figure 17, where we selected concavities based on the height threshold and increased brightness to make the wool look softer. Despite visually pleasing results, this is of course only a crude approximation to reality, as no light-dependency is introduced.

Directional Height Field The *Directional Height Field* selection is used to select shadow- and masking-areas based on a given height

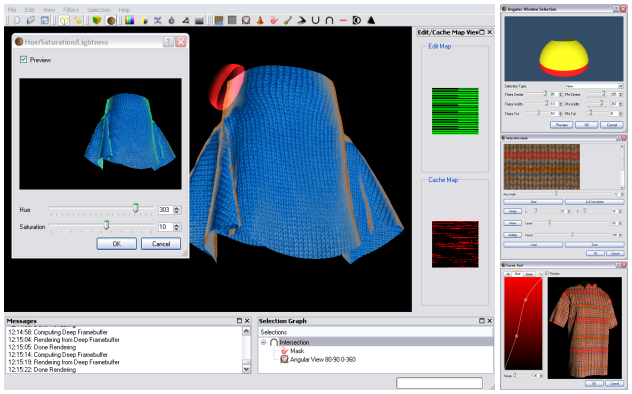


Figure 12: *Screenshot of the user interface. An object is rendered with the BTF and the selected part (grazing angle here) are shown in orange. A transparent hemispherical view-dependent slice is shown, depicting that grazing angles are selected (rim around hemisphere). The hue/saturation UI is shown as well. On the right, some more of our tools are shown: angular selection, selection mask, and curve tool.*

field, which can be derived with our height field extraction operator. To this end, rays are traced from each point on the BTF along the light- (for shadowing) or view-directions (for masking). Points become part of the selection when a self-intersection (occlusion) along the ray occurs. Adding shadows due to structural changes is enabled with this selection. Figure 5 illustrates this: a new shadow was added (due to a bump) using the heightfield-based selection and decreasing brightness in those areas (rightmost image).

3.4 Discussion

The selection tools are user-driven. Parameters need to be chosen for the specific BTF that is currently edited. For instance, the shadow selection requires the user to set a threshold parameter, as shadow intensity varies from BTF to BTF. We provide previews of the selections, such that the user can quickly adjust the parameters.

Selections are essential for editing global effects, such as interreflections, translucency, and scattering. The results depend on the appropriate choice of parameters and therefore on the user. In our experience, global effects are rather easy to modify, which is facilitated by the immediate visual feedback.

4 Architecture for Out-of-Core BTF Editing

The large size of BTFs makes memory management critical. We need an architecture that enables the manipulation of data that does not fit in main memory. On-demand lazy evaluation is mandatory because we must provide the user with rapid feedback. We first present the interaction offered to the user to make the architecture challenges concrete and then present our tile-based on-demand out-of-core architecture.

4.1 Interface and Visualization

In order to provide visual feedback about arbitrary parts of the BTF, we map it on an object and render it under user-specified lighting (Figure 12). Immediate feedback is provided after an editing operation is applied and the rendering is continually updated. A complete history is kept for undo.

Selections are critical in editing and we visualize them on the rendered models as a color mask, which is effective for spatial selections. Complex selections that involve light and view dependencies need to visualize different slices of the 6D selection. We provide a tool that displays the hemispherical light- or view-slice directly on the model for the current view or light direction and a user-selected spatial location (Figure 12 upper left).

4.2 Architecture

The 6D nature of BTFs makes memory and computation critical issues. For instance, current BTFs from the Bonn database use 1.6GB each (high-dynamic range data stored as RGBE [Ward 1992]). Anticipating a further increase in BTF resolution and due to the wish to run our editing approach on non-high-end machines, we require a scalable system design that can work with limited memory. Computation times for applying an editing operation to the whole BTF range from tens of seconds to several minutes and, in order to enable an interactive work flow, it is crucial to design an architecture that provides immediate feedback even for costly operations.

We achieve scalability and interactivity using a multi-threaded, tile-based architecture with a cache system. Internally, the BTF is split into smaller tiles and stored in a hierarchical cache (memory and disk), where only the most recently used tiles are in memory. Responsiveness of the system is ensured by processing BTF data in separate worker threads, which allows a user to continue editing. The worker threads process individual tiles, and schedule visible tiles of the BTF first. In order to provide fast visual feedback, the current rendering is continually updated with newly processed tiles, as soon as they are finished computing. A deep frame buffer stores affected BTF tiles, locations, and weights for each pixel, which makes re-rendering efficient. The system ensures that dependencies on previous editing operations are clear before succeeding operations are scheduled (e.g., the spatial blur operation on tile K is dependent on all spatially neighboring tiles). The sheer size of BTFs prevents the explicit storage of 6D selection masks like for image editing. Instead, selections are computed per tile on-the-fly, with a cache of recently used ones. This architecture ensures that our editing system can make efficient use of new multi-core systems.

We use the 32-bit RGBE format [Ward 1992] to store high-dynamic range BTF data, as it reduces data size by a factor of four compared to full floating point. We considered using further compression such as PCA to reduce tile storage costs, but then compression/decompression would increase computational load. Furthermore, performance analysis revealed that memory costs are dominated by access time, not by transfer rates.

5 Results

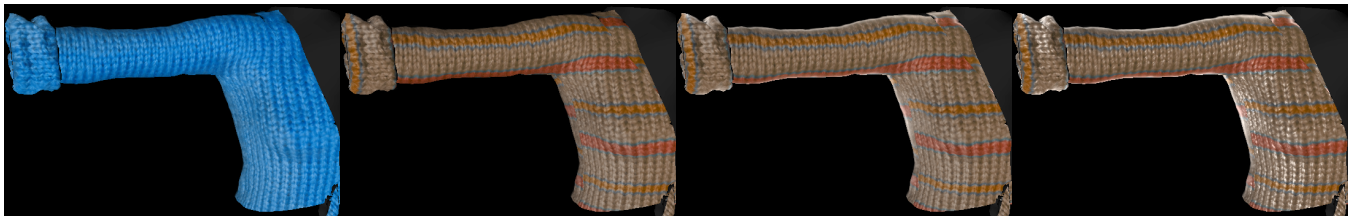
All our BTFs have 81×81 view and light directions and a spatial resolution of 256×256. Stored in high-dynamic range RGBE format, this amounts to 1.6GB of data per BTF. Through experimentation, we found that a good compromise between interactive feedback and processing time is achieved using tile sizes of $32^2 \times 3^2$ (spatial$\times$ directional), i.e., there are 46656 unique tiles.

It is important to give fast visual feedback to editing operations and selections. For a simple hue manipulation, our system provides comprehensive feedback in only 2.1 seconds (dual-core 2.6Ghz AMD Opteron) for Figure 13, meaning that every visible tile is updated. This amounts to processing 80MB of data, as there are about 5% of all tiles visible. An angular Gaussian blur (25° variance) takes about 9.7 seconds for the same configuration, which is due to the increased complexity of the operator, but is still fast enough for interactive feedback.

The time it takes to process a full BTF is heavily dependent on the chosen operation. For a simple operator it takes about half a minute to finish the full BTF; for more complex manipulations (like angular blur) processing times can go up to several minutes. However, our system always provides immediate feedback and lets the user continue to interact with our system, making the timings for finishing the full BTF less crucial.

5.1 Interactive BTF Modification

Figure 13 shows a variety of operations applied to a wool BTF. The tone and color manipulation tools, such as hue/saturation and brightness/contrast, in combination with selections prove to be very



(a) Original (b) Hue Changes (c) Fuzziness (d) More Nylon

Figure 13: *The original blue wool blouse (a) is changed to a patterned wool blouse using the hue operation and spatial selection (b). We then select grazing-angle texels and, using the brightness/contrast tool, we increase their brightness (c). Finally, we make the knitwear look more specular by increasing the brightness of highlight pixels with the curve tool (d).*

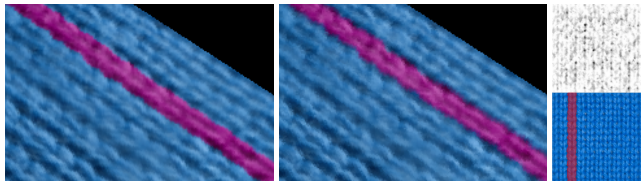


Figure 14: *A stripe of the wool BTF is colored differently. The BTF on the left was done without warping, whereas the BTF in the middle was done with the unwarp/warp operator. The height field and selection is shown on the right.*

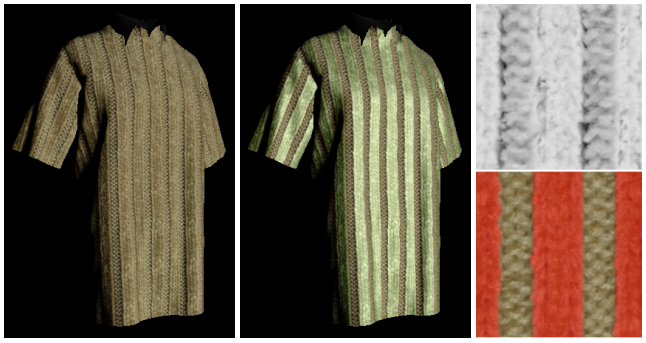


Figure 15: *Stripes of the original sweater BTF (left) are colored green and made more specular (middle). The top right shows the automatically deduced height field [Daum and Dudek 1998] and the bottom right shows the user-painted selection mask.*

effective. The wool's appearance can be changed considerably in a matter of minutes. The fast feedback provided by our system allows users to judge results quickly.

As discussed earlier, BTF slices contain parallax effects due to the underlying micro-geometry of a BTF. Figure 14 shows the effect of the parallax compensation operation on the wool BTF. Not using it incorrectly shifts the colored stripe a bit to the left. This shift is quite small for typical BTFs such as wool, and is often not needed. However, it is large enough for BTFs like the Lego (Figure 9) to be important.

Figure 15 shows an example, where we first needed to use parallax compensation to align all texture slices. The automatically deduced height field (computed in about 15 seconds) is only an approximation to the underlying micro-geometry but faithful enough to align the texture slices. We then color every other stripe of the sweater greenish and make them more specular. At the end, we re-warp the BTF again according to the height field.

The same input model was used for the teaser image (Figure 1). We use the Copy&Paste tool to add a wool stripe, which we then make more golden by increasing the brightness of highlight pixels and modifying them to be more gold colored with the curve tool. After changing the hue of the middle stripe, we add a layer of lint

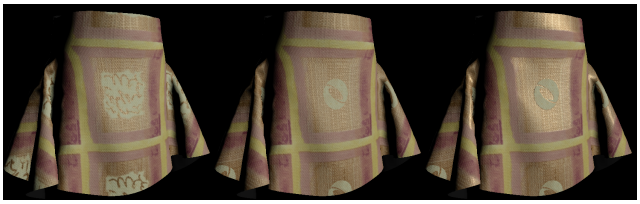


Figure 16: *The original wallpaper (left) is modified with the Copy&Paste tool to show a different logo (middle). The middle part is made more golden with the angular sharpen tool.*

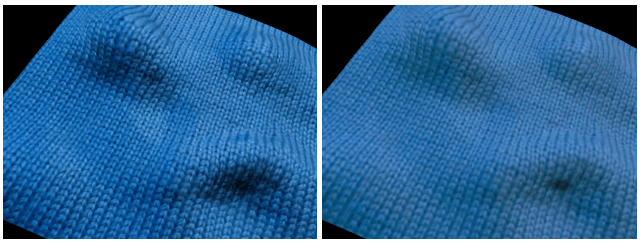


Figure 17: *The wool BTF is made "softer" by brightening concavities and blurring out the shadows.*

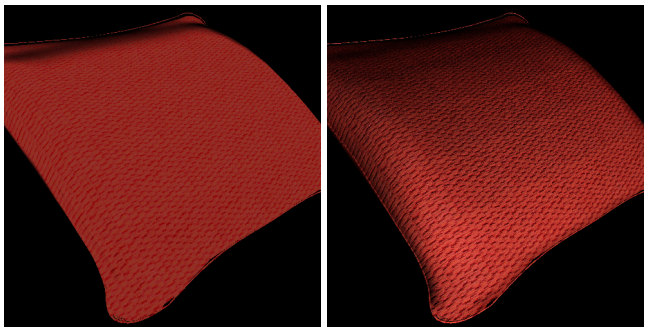
by selecting diffuse areas (BRDF selection) for grazing light angles and reducing contrast in those areas. Finally, we add a colored logo.

The Copy&Paste tool is an effective tool to change patterns in a BTF. In Figure 16, we remove the original white pattern by copying parts of the gold over it. After making a spatial selection using the SIGGRAPH logo, we replicate a small square of the original white area and repeatedly paste it onto the BTF. The spatial selection ensures that the paste operation only overwrites the inside of the logo. We then select the golden area (spatial selection) and use the angular sharpen tool and the curve tool to make it more glossy.

Finally, Figure 17 shows how the original wool BTF is made "softer" by increasing brightness in concavities, as well as blurring out shadow boundaries.

5.2 Interactive BTF Creation

The flexibility of our editing tools enables the creation of a full BTF starting from a single texture image. In Figure 18, we start with a 2D photograph of cloth, which corresponds to a flat BTF that is constant with respect to light and view direction. We first paint a crude height field based on the grey-scale version of the image. Lambertian shading is then applied using the BRDF selection and a brightness decrease (using the height field for local normal directions). Similarly, a small specular term is added using the BRDF selection (Ashikhmin-Shirley model) and an increase in brightness. Shadow areas are introduced with the directional height field tool. After their selection, they are darkened by reducing brightness. A small amount of asperity scattering is added by selecting the intersection between grazing view and light directions. Brightness is



Texture Only Cloth Material (BTFShop)

Figure 18: *An example of creating a red cloth. The original texture is edited to include BRDF, shadowing, scattering, and parallax effects.*

Original Texture Only Recreated

Figure 19: *The original BTF shown on the left is recreated from a single texture only (applied in the middle, shown at the bottom). Although there are minor differences between the original and the recreated version, it shows the effectiveness of our editing approach.*

then increased and the contrast is lowered. Finally, we perform a height field warp according to our crude height field to add "depth" to the material.

As a proof of concept, we recreated the corduroy BTF from the Bonn database (Figure 19) starting from a single texture slice (the frontal, fully lit; see bottom of figure). First, we introduce Lambertian shading and add a small glossy component (specular exponent of 5) using the BRDF selection. Shadows are created by selecting the tops of the ridges (painted with the selection mask tool) and applying the light-directional dilation, which selects shadow areas (after subtracting the original selection mask). Shadow areas are darkened using the brightness/contrast tool. We enhance the brightness at grazing angles by selecting the intersection of view and light grazing angles. The contrast at grazing views is decreased slightly. The local frame rotation tool is used to create the impression of ridges. The complete editing process, including trying different parameters for the various tools, takes about 10–15 minutes. More examples of a similar creation process can be found in Figure 20.

6 Discussion and Limitations

In our experience, the proposed approach works best for tweaking the appearance of a material as well as for the creation of a BTF from a flat texture. The user can change albedos and specularities of a particular part of the BTF, change the underlying geometry,

Figure 20: Left: *a Chinese blouse created from a single texture.* Right: *a flannel shirt from a single texture.*

remove a bump, modify shadows, and so forth. The approach is not meant to turn a wool BTF into bark, but manipulations, such as modifying the wool's fuzziness, are easily accomplished and intuitive due to interactive feedback. In fact, the operators are in our experience usually much easier to control than modifying a material in a physically-based ray-tracer (which we needed for generating comparisons for Figure 2, 3, 7, 10, and 11).

However, there are limitations to our approach. The modified BTFs may not be physically correct. For instance, the reflections of fuzz may not be fully accurate or reciprocal. Fortunately, for a wide range of contexts, physical accuracy is secondary to visual appearance. Our operators usually change one specific aspect of a BTF but not necessarily others that are connected. For instance, changing the meso-structure does not automatically change shadows accordingly. However, our operators provide the possibility to edit the major attributes of a material under the specific assumptions made by each operator.

Our approach is a first step towards intuitive editing of BTFs. It is based on a collection of operators that can modify specific parts of a BTF but it lacks "unified" tools. Ideally, a BTF editing system would enable the user to modify high-level properties of a material, such as "fuzziness" or "softness," but our current understanding of the perception of material appearance is too limited for such an approach and remains future work.

7 Conclusions

In this paper, we have presented a new approach for editing realistic BTFs. We have derived BTF-specific editing operations and have verified their expressiveness using comparisons with reference BTFs. Our proposed editing operations in combination with selections provide a flexible approach to editing BTFs. We have shown that this approach also allows for the creation of a full BTF starting from a single texture. Our editing system, BTFShop, overcomes difficult system issues that arise from the six-dimensional nature of BTF data. The system can handle large amounts of data and long computation times, yet provide interactive feedback.

Acknowledgements We would like to thank the University of Bonn [Sattler et al. 2003] and UCSD [Koudelka et al. 2003] for making their BTF data available. We are grateful to Mary Williamson and Omari Dennis for their PBRT tools. We further thank the internal MIT reviewers, the anonymous reviewers, and the referees for their very helpful comments and suggestions. This work was supported by a National Science Foundation CAREER award 0447561 "Transient Signal Processing for Realistic Imagery." Frédo Durand acknowledges a Microsoft Research New

Faculty Fellowship and a Sloan fellowship. Solomon Boulos acknowledges support from the University of Utah Brown Fellowship. Jan Kautz acknowledges support from an Emmy-Noether fellowship from the German Research Foundation.

References

ADELSON, E. H. 2001. On Seeing Stuff: the Perception of Materials by Humans and Machines. In *Human Vision and Electronic Imaging VI*, B. Rogowitz and T. Pappas, Eds., vol. 4299 of *Proceedings of SPIE*, 1–12.

COLBERT, M., PATTANAIK, S., AND KRIVANEK, J. 2006. BRDF-Shop: Creating Physically Correct Bidirectional Reflectance Distribution Functions. *IEEE Computer Graphics and Applications 26*, 1 (Jan.), 30–36.

CULA, O., AND DANA, K. 2001. Compact Representation of Bidirectional Texture Functions. In *Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, vol. 1, 1041–1047.

DANA, K., AND NAYAR, S. 1998. Histogram Model for 3D Textures. In *Conference on Computer Vision and Pattern Recognition (CVPR'98)*, 618–633.

DANA, K. J., VAN GINNEKEN, B., NAYAR, S. K., AND KOENDERINK, J. J. 1999. Reflectance and Texture of Real-World Surfaces. *ACM Transactions on Graphics 18*, 1 (Jan.), 1–34.

DAUM, M., AND DUDEK, G. 1998. On 3-D Surface Reconstruction Using Shape from Shadows. In *Conference on Computer Vision and Pattern Recognition (CVPR'98)*, 461–468.

DISCHLER, J.-M., AND GHAZANFARPOUR, D. 1999. Interactive Image-based Modeling of Macrostructured Textures. *IEEE Computer Graphics & Applications 19*, 1 (Jan.), 66–74.

DONG, J., QI, L., REN, J., AND CHANTLER, M. 2005. Self-Similarity Based Editing of 3D Surface Textures. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, 71–76.

FLEMING, R., JENSEN, H. W., AND BÜLTHOFF, H. 2004. Perceiving Translucent Materials. In *Symposium on Applied Perception in Graphics and Visualization 2004*, 127–134.

GINNEKEN, B. V., KOENDERINK, J., AND DANA, K. J. 1999. Texture Histograms as a Function of Irradiation and Viewing Direction. *International Journal of Computer Vision 31*, 2-3 (Apr.), 169–184.

HAINDL, M., AND HATKA, M. 2005. BTF Roller. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, 89–94.

HAINDL, M., GRIM, J., PUDIL, P., AND KUDO, M. 2005. A Hybrid BTF Model Based on Gaussian Mixtures. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, 95–100.

KHAN, E. A., REINHARD, E., FLEMING, R., AND BÜLTHOFF, H. 2006. Image-Based Material Editing. *ACM Transaction on Graphics (Proc. SIGGRAPH) 25*, 3 (July), 654–663.

KOENDERINK, J., AND PONT, S. 2003. The Secret of Velvety Skin. *Machine Vision and Applications 14*, 4, 260–268.

KOUDELKA, M., MAGDA, S., BELHUMEUR, P., AND KRIEGMAN, D. 2003. Acquisition, Compression and Synthesis of Bidirectional Texture Functions. In *Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis*, 59–64.

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. *ACM Transactions on Graphics (Proc. SIGGRAPH) 25*, 3 (July), 735–745.

LEUNG, T., AND MALIK, J. 1997. On Perpendicular Texture: Why do we see more flowers in the distance? In *Conference on Computer Vision and Pattern Recognition (CVPR '97)*, 807–813.

MAGDA, S., AND KRIEGMAN, D. 2006. Reconstruction of Volumetric Surface Textures for Real-Time Rendering. In *17th Eurographics Symposium on Rendering*, 19–30.

MESETH, J., MÜLLER, G., AND KLEIN, R. 2004. Reflectance Field Based Real-Time, High-Quality Rendering of Bidirectional Texture Functions. *Computers & Graphics 28*, 1 (Feb.), 105–112.

MÜLLER, G., MESETH, J., SATTLER, M., SARLETTE, R., AND KLEIN, R. 2005. Acquisition, Synthesis, and Rendering of Bidirectional Texture Functions. *Computer Graphics Forum 24*, 1 (Mar.), 83–110.

NEUBECK, A., ZALESNY, A., AND GOOL, L. V. 2005. 3D Texture Reconstruction from Extensive BTF Data. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis*, 13–18.

NGAN, A., AND DURAND, F. 2006. Statistical Acquisition of Texture Appearance. In *17th Eurographics Symposium on Rendering*, 31–40.

PHARR, M., AND HUMPHREYS, G. 2004. *Physically Based Rendering: From Theory to Implementation*. Morgan-Kaufmann.

PONT, S., AND KOENDERINK, J. 2002. Bidirectional Texture Contrast Function. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision*, 808–822.

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. A Signal-Processing Framework for Inverse Rendering. In *Proceedings of ACM SIGGRAPH 2001*, 117–128.

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2003. Efficient and Realistic Visualization of Cloth. In *14th Eurographics Symposium on Rendering*, 167–178.

SUYKENS, F., VOM, K. B., LAGAE, A., AND DUTRÉ, P. 2003. Interactive Rendering with Bidirectional Texture Functions. *Computer Graphics Forum 22*, 3 (Sept.), 463–472.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of Bidirectional Texture Functions on Arbitrary Surfaces. *ACM Transactions on Graphics (Proc. SIGGRAPH) 21*, 3 (July), 665–672.

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2004. TensorTextures: Multilinear Image-Based Rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH) 23*, 3 (Aug.), 336–342.

WARD, G. 1992. Real Pixels. In *Graphics Gems II*, J. Arvo, Ed. Academic Press.

ZHOU, K., DU, P., WANG, L., MATSUSHITA, Y., SHI, J., GUO, B., AND SHUM, H.-Y. 2005. Decorating Surfaces with Bidirectional Texture Functions. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (Sept.), 519–528.