Pacific Graphics 2009
S. Lee, D. Lischinski, and Y. Yu
(Guest Editors)

Volume 28 (2009), Number 7

Edit Propagation on Bidirectional Texture Functions

Kun Xu[1†] Jiaping Wang[2] Xin Tong[2] Shi-Min Hu[1] Baining Guo[2]

[1] Tsinghua National Laboratory for Information Science and Technology
and Department of Computer Science and Technology, Tsinghua University
[2] Microsoft Research Asia

Abstract

We propose an efficient method for editing bidirectional texture functions (BTFs) based on edit propagation scheme. In our approach, users specify sparse edits on a certain slice of BTF. An edit propagation scheme is then applied to propagate edits to the whole BTF data. The consistency of the BTF data is maintained by propagating similar edits to points with similar underlying geometry/reflectance. For this purpose, we propose to use view independent features including normals and reflectance features reconstructed from each view to guide the propagation process. We also propose an adaptive sampling scheme for speeding up the propagation process. Since our method needn't any accurate geometry and reflectance information, it allows users to edit complex BTFs with interactive feedback.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.7]: Color, shading, shadowing, and texture —

1. Introduction

Faithfully modeling and rendering surface appearance is important for many graphics applications. For surfaces with geometric details and material variations, Dana et. al. [DvGNK99] proposed six dimensional Bidirectional Texture Functions (BTFs) that directly capture their appearances with respect to 2D positions, 2D viewing directions and 2D lighting directions. Nearly all surface appearance effects, such as self-occlusion, self-shadowing and reflectance, can be well modeled by BTF. Despite recent advances in BTF capturing, synthesis and rendering, editing BTF data is still a challenging task. Since points appear in different views may correspond to the same point on the meso-structure surface, to maintain appearance consistency in BTF editing, edits on one point should be automatically propagated to those points in other views which are essentially the same underlying meso-structure point. Without explicit geometry information, it is a non-trivial task to do such propagation.

Inspired by recent edit propagation methods on spatially varying materials [LFUS06, PL07, AP08], we propose a method for editing BTF data based on edit propagation scheme. In our approach, users specify sparse edits on a slice of BTF data under a certain view. These edits are then automatically propagated to other views. The consistency of the BTF data is maintained by propagating similar edits to BTF data with similar underlying geometry/reflectance. The similarity between different points is defined by *view independent features* including normal and reflectance features reconstructed from each view. Thus, points that correspond to the same meso-structure surface point but appear in different views would have the same view-independent features and thus receive the same edits. Since our method does not rely on an explicit geometry, BTFs with complex geometric structures, even non-height-field geometry, could be edited by our approach.

As in AppProp [AP08], we solve the propagation problem with a linear system defined by a dense affinity matrix of millions of elements. We also apply the column sampling technique to approximate affinity matrix for efficiency. Instead of randomly sampling columns, we propose an adaptive column sampling method which is demonstrated to produce a better approximation.

Following prior works [LFUS06, PL07, AP08], we propagate edit parameters rather than the final appearance val-

† This work was done when Kun Xu was an visiting student at Microsoft Research Asia.

ues, which makes the propagation process independent of the edit operator being applied. We have tested various BTF edit operators proposed by Kautz et al. [KBD07], including hue change, tone adjustment, angular blur, local frame change and shadow removal. To preserve the rich effects of BTF, we directly edit the raw BTF data rather than editing an approximated representation of BTF data. Furthermore, our framework is extensible. 3D position could be added as an additional view independent feature if the geometry of BTF is available, and new edit operators can also be easily integrated in.

In the context of recent works on edit propagation [AP08, PL07] and BTF editing [KBD07, MSK07], we see this work make the following contributions:

- We propose a novel method for editing BTFs based on view independent features and edit propagation scheme. In our approach, users only need to specify sparse edits and those edits would be automatically propagated to the whole data based on edit propagation scheme, and appearance consistency is automatically maintained during editing.
- Our method enables editing of BTFs with complex geometric structures, e.g. non-height-field geometry, which could not be achieved by previous BTF editing methods.
- We propose an adaptive column sampling method for approximating dense affinity matrix which is demonstrated to be more efficient than randomly sampling.

2. Related Work

Interactive Material Editing: With the advance of appearance acquisition techniques, editing captured spatially varying materials receives more and more attention in recent years [WTL*06, GTR*06, LBAD*06, PL07, AP08]. In particular, [WTL*06, GTR*06] demonstrated editing time varying effects of spatially varying BRDFs interactively. Lawrence et al. [LBAD*06] factorized spatially varying BRDFs into products of low dimensional functions in a hierarchical way, each of which can be easily edited by user. Recently, optimization based methods have been proposed for edit propagation. In these methods, users specify rough or sparse edits on some regions, then an optimization problem is set up and solved to propagate those edits to the whole dataset. These methods include image colorization [LLW04], tone adjustment [LFUS06] and material editing [PL07, AP08]. All of these propagation schemes work on an explicit domain (2D image plane) that every point on the domain is essentially different from each other. Differently, we propagate on the domain of the underlying meso-structure surface that is implicitly determined by reference plane location and viewing directions of BTFs.

BTF Editing: A lot of works have been done for BTF acquisition [DvGNK99, SSK03, KMBK03, HP03, ND06] ,compression [SSK03, KMBK03, VT04, WWS*05], rendering [SSK03, SVLD03, MMK04, WTS*05] and synthesis [TZL*02, KMBK03, HH05]. A comprehensive survey of these works could be found in [MMS*05]. Recently, Kautz et al. [KBD07] proposed a set of edit operators for BTFs and further developed an interactive BTF editing system. They directly manipulated on the raw BTF data and utilized an out-of-core scheme to achieve interactive performance. To maintain appearance consistency, the height field reconstructed from the BTF data is applied to warp the BTF data under different views. Unfortunately, it is difficult to apply this approach to BTF captured from non-height-field geometry. Based on texture synthesis techniques, Müller et al. [MSK07] proposed a procedural method for editing the spatial pattern of BTFs. Different from their method, we focus on editing material property of BTFs. Besides, instead of assuming a height-field meso-structure geometry, our method propose to use view independent features to preserve appearance consistency. Since view independent features can be easily and robustly recovered from BTFs, our method can be conveniently applied for editing BTFs with complex geometry.

3. Edit Propagation on BTF

The 6D $BTF(\mathbf{x},\mathbf{o},\mathbf{i})$ is an image-based representation that is defined on reference plane location $\mathbf{x}$, viewing direction $\mathbf{o}$ and lighting direction $\mathbf{i}$. Editing the surface reflectance of the BTF is essentially changing the material properties of the points on the underlying meso-structure surface. However, for raw BTF data whose underlying geometry is unknown, we cannot directly use the underlying meso-structure surface as editing domain. Instead, we define our editing domain on *spatial-view domain* $(\mathbf{x},\mathbf{o})$ which combines both the reference plane location and viewing direction. The underlying meso-structure surface can be represented by spatial-view domain implicitly and recovery of the meso-structure geometry is thus avoided. The BTF data could be treated as 2D light reflectance field $b(\mathbf{i})$ defined in 4D spatial-view domain:

$$(\mathbf{x},\mathbf{o}) \rightarrow b_{\mathbf{x},\mathbf{o}}(\mathbf{i}) = BTF(\mathbf{x},\mathbf{o},\mathbf{i}). \quad (1)$$

In our framework, user gives some edits on sparse samples in the spatial-view domain. Typically, the edits are directly specified on the top view. Then, the user given edits are propagated to the whole spatial-view domain. However, since points appear in different views may correspond to the same underlying point on the meso-structure surface, their appearances should keep consistent during propagating . An naive approach is to define similarity based on the L^2 distance between light reflectance fields $b(\mathbf{i})$ and to use this similarity to guide propagation. However, this approach fails because light reflectance fields $b(\mathbf{i})$ from the same meso-structure surface point varies with viewing directions due to specular reflection. Thus the similarity of light reflectance fields doesn't indicate the similarity of material property correctly. The appearance consistency in the BTF data may be

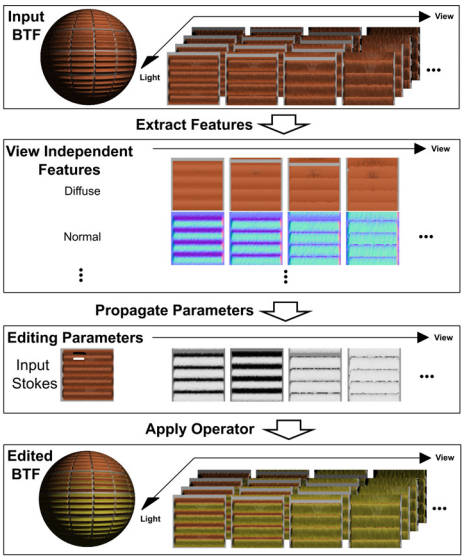

Figure 1: *The pipeline of our method. First, we extract view independent features from the input BTF data; Next, we propagate user given edit parameters in the spatial-view domain under the guidance of view independent features; Finally, we apply the desired edit operator according to propagated edit parameters.*

broken after editing. In our framework, we define similarity on *view independent features* extracted from the light reflectance fields. The view independent features, such as diffuse color, indicate material property and do not change with viewing direction. In this way, points that correspond to the same meso-structure surface point but appear in different views would have the same view independent feature and thus receive the same edits.

The basic pipeline of our work is illustrated in Figure 1. Given the input BTF data, view independent features are first extracted (Section 3.1) as a pre-process. Then, user could specify desired edits with our sketch-based user interface at a certain slice of BTF data, e.g. top view. After that, our system automatically propagate user edits to the whole BTF data based on the similarity of view independent features. The details of propagation algorithm are described in Section 3.2. Following prior edit propagation works [LFUS06, PL07, AP08], we propagate edit parameters rather than the final appearance values to make the propagation process independent of the certain edit operator being applied.

3.1. View Independent Features

In our approach, the view independent features are used to define similarity between points in the spatial-view domain. To maintain appearance consistency between views, they should not change with viewing direction. For this purpose, we choose normals and diffuse colors as view independent features. The view independent features are extracted as a pre-process step. Normal and diffuse colors are recovered from the light reflectance fields at each point in the spatial-view domain using the classical photometric stereo method [RTG97]. In particular, we exclude the shadowed values and the specular values and use the remaining values to recover normal and diffuse. Values are determined as shadowed/specular if they are too dark/bright, e.g. smaller/larger than a predefined threshold. Note that the classical photometric stereo method is robust for BTFs with lambertain surfaces, but it cannot work well for BTFs with highly specular reflectance, highly subsurface scattering or strong occlusion, since it breaks the lambertain reflectance assumption. In addition, if the underlying 3D geometry could be recovered, e.g. as height-field, we would include the 3D position as an additional view independent feature. In this case, we use the shape from shadow techniques [DD98] to recover a height-field geometry. Note that the view independent features are only used to guide the propagation process, the final edits are directly applied to the raw BTF data rather than an approximated representation defined by view independent features. Therefore, we do not need a perfectly accurate reconstruction of view independent features.

For denotation simplicity, in the remaining parts of the paper, we use subscript i, j to refer to points on the 4D spatial-view domain$(\mathbf{x}, \mathbf{o})$. The similarity z_{ij} between two points is defined as:

$$z_{ij} = \exp\left(-\frac{\|\mathbf{n}_i - \mathbf{n}_j\|^2}{\sigma_n^2}\right) \exp\left(-\frac{\|\mathbf{d}_i - \mathbf{d}_j\|^2}{\sigma_d^2}\right) \qquad (2)$$

It is the product of two Gaussian kernels of the distances between normal $\mathbf{n}_i$ and diffuse color $\mathbf{d}_i$, controlled by deviation parameters σ_n and σ_d, respectively. How to set the values of σ_n and σ_d depends on user intention. The smaller $\sigma_n(\sigma_d)$ is, the more precise propagation would depend on normal(diffuse color). In our experiments, for high dynamic range BTFs, σ_d is usually set around 0.2 to 0.4; σ_n is usually set around 0.2 to 0.4 for normalized normals. Besides, if the underlying geometry is available, we could add 3D position as an additional view-independent feature to be accounted in similarity computation between points. In this case, z_{ij} could be rewritten as:

$$z_{ij} = \exp\left(-\frac{\|\mathbf{n}_i - \mathbf{n}_j\|^2}{\sigma_n^2}\right) \exp\left(-\frac{\|\mathbf{d}_i - \mathbf{d}_j\|^2}{\sigma_d^2}\right) \exp\left(-\frac{\|\mathbf{p}_i - \mathbf{p}_j\|^2}{\sigma_p^2}\right) \qquad (3)$$

where $\mathbf{p}_i$ is the position of point i, σ_p is the corresponding deviation parameter. It could be set from 0.05 to 5.0 in our experiments.

3.2. Edit Propagation

Following the AppProp edit framework on spatially varying materials [AP08], given user specified edit parameters g, to solve for the propagated edit parameters e, we formulize the propagation process as an optimization problem minimizing the following energy function:

$$\sum_i \sum_j w_j z_{ij} (e_i - g_j)^2 + \lambda \sum_i \sum_j z_{ij} (e_i - e_j)^2 \quad (4)$$

where the sums are calculated over all points on the 4D spatial-view domain. The energy function consists of two terms. λ controls the relative contributions of the two terms, and is set to $\sum_j w_j / n$ to make the contributions of the two terms almost same, and n is the total number of points on the 4D spatial-view domain. The first term accounts for the constraint of user input, where g denotes user specified edit parameters and w is edit strength which is one for user edited points and zero for non-edited points. The second term accounts for the propagation scheme that ensures similar edits are applied to points with similar view independent features. The similarity z_{ij} between point i and j is defined in Equation 2.

Our formulation is different from AppProp [AP08] in two aspects. First, we propagate edits in the 4D spatial-view domain instead of a 2D image domain. Secondly, we use view-independent features instead of appearance and image pixel position to define similarity between points.

Algorithm: We utilize the method in AppProp [AP08] to minimize the energy function in Equation 4. To make our paper self-contained, we briefly review AppProp's solution. Please refer to their original paper for detailed derivations. Since the energy function in Equation 4 is quadratic, optimizing it is equivalent to solving a linear system:

$$(D - Z)e = \frac{1}{2\lambda} ZWg \Rightarrow e = \frac{1}{2\lambda} (D - Z)^{-1} ZWg \quad (5)$$

where e,g are respectively the vectors of edit parameters e_i and user specified parameters g_i, and Z is the affinity matrix of z_{ij}, D is a diagonal matrix with $D_{ii} = d_i = \sum_j (1 + w_j/(2\lambda)) z_{ij}$, W is a diagonal matrix whose element is the edit strength $W_{ii} = w_i$.

The dense affinity $n \times n$ matrix Z is further approximated by a low rank matrix approximation. By selecting m columns ($m \ll n$) of matrix Z, denote U as the $n \times m$ matrix composed of the m selected columns, and A as the $m \times m$ affinity matrix of the m selected points, we have:

$$Z \approx U A^{-1} U^T \quad (6)$$

With this approximation, the diagonal elements d of diagonal matrix D could be approximated by:

$$d \approx \left(\frac{1}{2\lambda} U A^{-1} U^T W + U A^{-1} U^T \right) 1_n \quad (7)$$

where 1_n is a length-n vector of ones. Finally, edit parameters e in Equation 5 could be approximated by applying

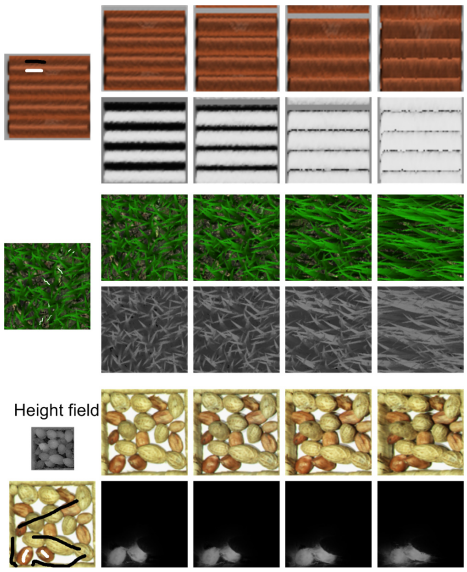

Figure 2: *Some results of propagated edit parameters. The left column indicates user specified edit parameters. The 2-th to 5-th columns show BTF slices and propagated edit parameters under different views. The first row edits on a synthetic brick BTF with parameters $\sigma_n = \sigma_d = 0.4$; the second row edits on a synthetic lawn BTF with parameters $\sigma_n = 0.2, \sigma_d = 0.4$. the third row edits on a captured peanut BTF with an additional position feature extracted from reconstructed height-field, and the used parameters are $\sigma_n = 1.5, \sigma_d = 0.25, \sigma_p = 0.1$.*

Woodbury Formula:

$$e \approx \frac{1}{2\lambda} (D^{-1} - D^{-1} U (-A + U^T D^{-1} U)^{-1} U^T D^{-1}) \cdot (U A^{-1} U^T) W g \quad (8)$$

The bottleneck lies on the matrix-matrix calculation of $U^T D^{-1} U$ whose complexity is $O(m^2 n)$.

Adaptive Column Sampling: In AppProp [AP08], they approximate matrix Z by stratified randomly sampling columns. Intuitively, with a given column number m, a more sophisticated column sampling scheme such as singular value decomposition would approximate Z better, but would require the knowledge of the full matrix Z and need much more computation. Noticing that matrix Z is an affinity matrix, which means that the i-th column and the j-th column would be similar if z_{ij} is close to 1, we propose a more efficient adaptive column sampling method:

1. Randomly select a column from Z, suppose it is the i-th column. Remove all the j-th columns from Z if the affinity z_{ij} is larger than a given threshold;
2. If the number of selected columns is already m or there is no columns left, go to Step 3; otherwise go to Step 1;
3. End.

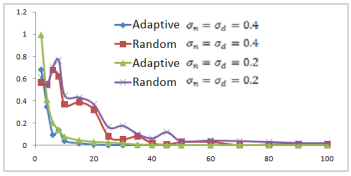

Figure 3: *Column sampling scheme comparison. It plots the relative RMS error of propagated edit parameters on the brick BTF using different number of columns and different parameters.*

In our implementation, we set the threshold as 0.9 and find it works well. Since the time complexity of adaptive column sampling method is $O(mn)$, the additional cost to utilize this sampling scheme could be ignored. We have compared the proposed adaptive column sampling with stratified random sampling using different number of columns and different parameters σ_n and σ_d in Figure 3. We find that the adaptive column sampling only needs about one-fourth number of columns compared to stratified random sampling with similar approximation error. Since the calculation complexity is $O(m^2 n)$, this would lead to a speed-up of 16 times.

4. Results and Implementation

4.1. Edit Operators

After solving the propagated edit parameters, we allow users to adjust the desired edit operator being applied to the BTF data, whose editing strength is controlled by the propagated edit parameters. We have tested various edit operators proposed by Kautz et al. [KBD07], including hue change, shadow removal, local frame change, angular blur and tone adjustment. The BTFs used in our experiment come from Bonn BTF database [SSK03] and Wang et al. [WTS*05].

Hue Change: At each spatial-view point, we first transform the light reflectance field $b_i(\mathbf{i})$ from RGB to HSV color space, then we linearly blend the original hue channel with a target hue channel using edit parameter e_i as a weight while keeping other two channels unchanged. We have applied this operator to the synthetic lawn BTF, the synthetic brick BTF and the measured pulli BTF. As shown in Figure 4(b), the appearance of the grass is changed from green to withered like. As shown in Figure 5(b), one side of the brick is changed to yellow-green. As shown in Figure 9(c), a strip of the pulli BTF is changed to green color. For the pulli BTF, we reconstruct a height-field geometry and use position as an additional feature.

Shadow Removal: In this operator, first, points are determined as shadow areas if the color value is less than a given threshold and determined shadow areas are filled using the color value computed by reconstructed reflectance and normal feature at that point. Secondly, we adjust the average brightness and saturation of the shadow area to make it match nearby regions. Edit parameter e is used to control the blending weight between original shadowed color and filled color. We have applied this operator to the synthetic block BTF. As shown in Figure 6(b), shadow is removed after editing.

Local Frame Change: At each spatial view point, we linearly blend the target normal with the original normal weighted by edit parameter e_i to obtain a rotated local frame. Next, we search in the 2D spatial slice of the rotated view direction, and return the pixel position j with the best view-independent feature match. Then, we modify the light reflectance field $b_i(\mathbf{i})$ by looking up original value in $b_j(\mathbf{i})$ using rotated light directions. We have applied this operator to the synthetic dot BTF, as shown in Figure 8(c).

Angular Blur: At each spatial view point, we apply a spherical Gaussian kernel G in light direction space to the light reflectance data $b_i(\mathbf{i})$:

$$b_i^{new}(\mathbf{i}) = (b_i \bigotimes G)(\mathbf{i}) \tag{9}$$

where the radius of Gaussian kernel G is controlled by edit parameter e. We have applied this operator to the synthetic dot BTF and the pulli BTF. As shown in Figure 8(b) and Figure 9(b), the edited area looks less specular.

Tone Adjustment: At each spatial-view point, we adjust the tonal value of the light reflectance data $b_i(\mathbf{i})$ by linearly interpolating the original tonal value with a user given target tonal value using editing parameter e_i as a weight. We have applied this operator to the measured peanut BTF data. As shown in Figure 10, after editing, some peanuts look from fresh to cooked. Notice the difference between Figure 10(b) and Figure 10(c). In Figure 10(b), with the help of additional position feature, we only edited 2 selected peanuts. In Figure 10(c), without position feature, we achieved global editing effects by changing all the peanuts.

Comparison With Kautz et al. [KBD07]: We make a comparison between our method and the method of Kautz et al. [KBD07] on editing the lawn BTF data in Figure 7. The first row shows our result, and the second row is Kautz's result. From left to right are slices of BTF from views of increasing polar angles. Our method keeps consistency in all views, as shown in Figures 7(a)-(d). However, Kautz's method fails to preserve consistency when view polar angle is large, as shown in Figures 7(g)-(h). This is not surprising because the lawn BTF contains a highly complex non-height-field geometry structure, which makes Kautz's method fail to determine correct view correspondences by a reconstructed top-view height field.

Failure Case: Figure 11 gives an example of failure case. The left image gives the top view of the BTF and input strokes. In this example, user gives a white stroke in a vertical strip and a black strop in a horizontal strip, whose intention is to select the vertical strips and then apply edits to them. However, since this BTF is almost flat and the diffuse color is similar among the whole data, the propagation pro-

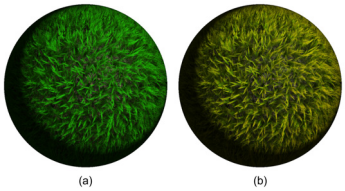

Figure 4: *Editing of the synthetic lawn BTF. (a) The original BTF; (b) Hue component changed.*

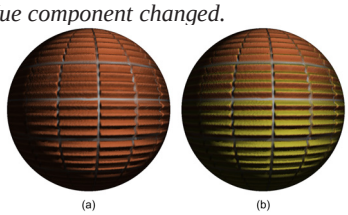

Figure 5: *Editing of the the synthetic brick BTF. (a) The original BTF; (b) Hue component changed.*

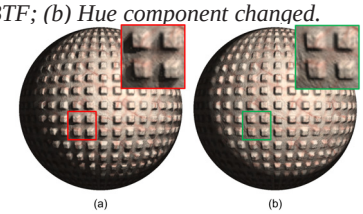

Figure 6: *Editing of the synthetic block BTF. (a) The original BTF; (b) Shadow removed.*

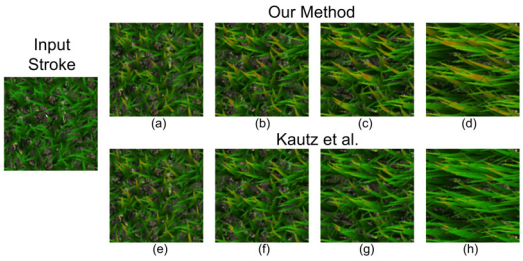

Figure 7: *Comparison between our method and Kautz's method.*

cess fails to separate the horizontal and vertical strips (As shown in the middle and the right images).

4.2. Implementation

Interactive Editing Preview: To support a fast iterative workflow, it is important to provide the user with fast visual feedback in editing. In order to provide fast visual feedback, we map the BTF to a sphere and render it under directional lighting. Instead of propagating edit parameters to the whole 4D spatial-view domain, we only propagate edits to those spatial-view points which are used in rendering. Thus, the size of the propagation domain and the cost to evaluate Equation 8 are largely reduced. This increased efficiency allows user an interactive rendering preview of edited BTF.

Performance: Our system is implemented on a consumer level PC with Intel Core2Duo 2.33G CPU and 4G RAM. The performance is reported in Table 1. The memory cost mainly lies on storing the $n \times m$ matrix U. In our experiment, the largest data is the pulli data with resolution $n = 81 * 256^2$,

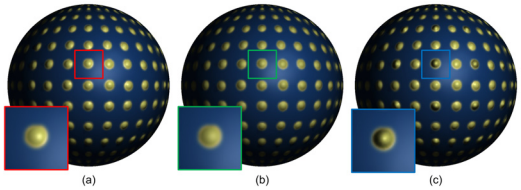

Figure 8: *Editing of the synthetic dot BTF. (a) The original BTF; (b) Less specular by angular blur; (c) Local frame changed.*

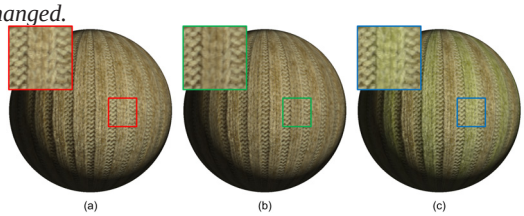

Figure 9: *Editing of the measured cloth-pulli BTF. (a) The original BTF; (b) Less specular by angular blur (with additional position feature); (c) Hue component changed (with additional position feature).*

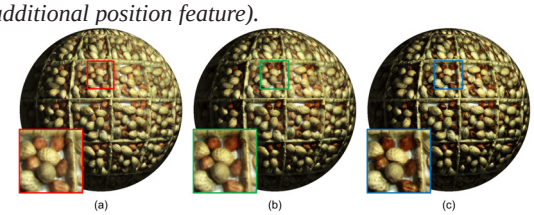

Figure 10: *Editing of the measured peanut-box BTF. (a) The original BTF; (b) Tone adjustment (with additional position feature); (c) Tone adjustment.*

Figure 11: *An example of failure.*

BTF	dot	lawn	peanut	pulli
Res.	$60 \times 60 \times 64^2$	$60 \times 60 \times 128^2$	$96 \times 96 \times 128^2$	$81 \times 81 \times 256^2$
Col.	50	100	100	100
Operator	Ang.Blur	Color	Tone	Color
Memory	47M	375M	600M	2G
Prev.T.	0.1s	0.5s	0.8s	2.0s
Prop.T.	0.4s	9.2s	22s	60s
Edit.T.	30s	6.1s	16s	42s

Table 1: *Performance. Each row lists: the resolution of the BTF data; the number of columns selected; the edit operator applied; memory cost for edit propagation; execution time for different stages of the algorithm including edit preview, edit propagation and the actual applied edit operation.*

and we choose $m = 100$ columns. Storing matrix U (32-bit floating-point precision) would cost about 2G space and could be fit in main memory. For even larger BTF data, since the calculations involving U in Equation 7 and 8 only contain matrix-matrix and matrix-vector multiplications, it is straight-forward to implement an out-of-core scheme by splitting U into sub-matrices.

5. Conclusion

We proposed an efficient method for editing material properties of bidirectional texture functions (BTFs) based on edit propagation scheme. In our method, users specify sparse edits on a certain slice of BTF, and those edits would be automatically propagated to the whole BTF data, while keeping appearance consistency between different views. Since our method does not rely on explicit geometry, users are allowed to edit BTF with complex geometric structures, e.g. non-height-field geometry. Besides, our method is extensible and allows users to easily add new edit operators and new view independent features to guide propagation. We also proposed an adaptive column sampling method which is demonstrated to produce a better approximation for dense affinity matrix than randomly sampling and is expected to be useful in image edit propagation or other high-dimensional data editing.

Currently, Our method limits to edit uncompressed BTFs. Since most compressed representations are linear approximation of the original high dimensional BTF datas, it is unclear how to map propagation process and the nonlinear edit operators to the compressed representation. In the future, we would like to explore techniques to directly edit on a compressed representation. Besides, how to extend our method to edit geometric meso-structures of BTFs is another interesting topic.

Acknowledgement: We would like to thank the anonymous reviewers for their valuable comments and insightful suggestions. This work was supported by the National Basic Research Project of China (Project Number 2006CB303106), the Natural Science Foundation of China (Project Number U0735001) and the National High Technology Research and Development Program of China (Project Number 2009AA01Z329). Kun Xu is also supported by the Microsoft Fellowship.

References

[AP08] AN X., PELLACINI F.: Appprop: all-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3 (2008), 1–9.

[DD98] DAUM M., DUDEK G.: On 3-d surface reconstruction using shape from shadows. In *Proceedings of CVPR* (1998), pp. 461–468.

[DvGNK99] DANA K. J., VAN GINNEKEN B., NAYAR S. K., KOENDERINK J. J.: Reflectance and texture of real-world surfaces. *ACM Trans. Graph. 18*, 1 (1999), 1–34.

[GTR*06] GU J., TU C.-I., RAMAMOORTHI R., BELHUMEUR P., MATUSIK W., NAYAR S.: Time-varying surface appearance: acquisition, modeling and rendering. *ACM Trans. Graph. 25*, 3 (2006), 762–771.

[HH05] HAINDL M., HATKA M.: Btf roller. In *Proceedings of the 4th International Workshop on Texture Analysis and Synthesis* (2005), pp. 89–94.

[HP03] HAN J. Y., PERLIN K.: Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Trans. Graph. 22*, 3 (2003), 741–748.

[KBD07] KAUTZ J., BOULOS S., DURAND F.: Interactive editing and modeling of bidirectional texture functions. *ACM Trans. Graph. 26*, 3 (2007), 53.

[KMBK03] KOUDELKA M., MAGDA S., BELHUMEUR P., KRIEGMAN D.: Acquisition, compression and synthesis of bidirectional texture functions. In *Proceedings of the 3rd International Workshop on Texture Analysis and Synthesis* (2003), pp. 59–64.

[LBAD*06] LAWRENCE J., BEN-ARTZI A., DECORO C., MATUSIK W., PFISTER H., RAMAMOORTHI R., RUSINKIEWICZ S.: Inverse shade trees for non-parametric material representation and editing. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers* (New York, NY, USA, 2006), ACM, pp. 735–745.

[LFUS06] LISCHINSKI D., FARBMAN Z., UYTTENDAELE M., SZELISKI R.: Interactive local adjustment of tonal values. *ACM Trans. Graph. 25*, 3 (2006), 646–653.

[LLW04] LEVIN A., LISCHINSKI D., WEISS Y.: Colorization using optimization. *ACM Trans. Graph. 23*, 3 (2004), 689–694.

[MMK04] MESETH J., MÜLLER G., KLEIN R.: Reflectance field based real-time, high-quality rendering of bidirectional texture functions. In *Computers & Graphics 28, 1* (2004), pp. 105–112.

[MMS*05] MULLER G., MESETH J., SATTLER M., SARLETTE R., KLEIN R.: Acquisition, synthesis, and rendering of bidirectional texture functions. *Computer Graphics Forum 24*, 1 (2005), 83–109.

[MSK07] MÜLLER G., SARLETTE R., KLEIN R.: Procedural editing of bidirectional texture functions. In *Proceedings of Eurographics Symposium on Rendering* (2007), The Eurographics Association, pp. 219–230.

[ND06] NGAN A., DURAND F.: Statistical acquisition of texture appearance. In *Proceedings of the 17th Eurographics Symposium on Rendering* (2006), pp. 31–40.

[PL07] PELLACINI F., LAWRENCE J.: Appwand: editing measured materials using appearance-driven optimization. *ACM Trans. Graph. 26*, 3 (2007), 54.

[RTG97] RUSHMEIER H. E., TAUBIN G., GUÉZIEC A.: Appying shape from lighting variation to bump map capture. In *Proceedings of the Eurographics Workshop on Rendering Techniques '97* (1997), Springer-Verlag, pp. 35–44.

[SSK03] SATTLER M., SARLETTE R., KLEIN R.: Efficient and realistic visualization of cloth. In *Proceedings of Eurographics Symposium on Rendering* (2003), pp. 167–177.

[SVLD03] SUYKENS F., VOM K. B., LAGAE A., DUTRÉ P.: Interactive rendering with bidirectional texture functions. In *Computer Graphics Forum 22, 3* (Sept. 2003), pp. 463–472.

[TZL*02] TONG X., ZHANG J., LIU L., WANG X., GUO B., SHUM H.-Y.: Synthesis of bidirectional texture functions on arbitrary surfaces. In *SIGGRAPH* (2002), pp. 665–672.

[VT04] VASILESCU M. A. O., TERZOPOULOS D.: Tensortextures: multilinear image-based rendering. *ACM Trans. Graph. 23*, 3 (2004), 336–342.

[WTL*06] WANG J., TONG X., LIN S., PAN M., WANG C., BAO H., GUO B., SHUM H.-Y.: Appearance manifolds for modeling time-variant appearance of materials. *ACM Trans. Graph. 25*, 3 (2006), 754–761.

[WTS*05] WANG J., TONG X., SNYDER J., CHEN Y., GUO B., SHUM H.-Y.: Capturing and rendering geometry details for btf-mapped surfaces. *The Visual Computer 21*, 8-10 (2005), 559–568.

[WWS*05] WANG H., WU Q., SHI L., YU Y., AHUJA N.: Out-of-core tensor approximation of multi-dimensional matrices of visual data. *ACM Trans. Graph. 24*, 3 (2005), 527–535.