representation of the variation in appearance of the surface under changes in viewpoint and illumination, including complex details due to surface mesostructure, such as self-occlusion, interreflection, and self-shadowing. Unlike the BTF synthesis method of Tong *et al.* [2002] and similar algorithms, our technique is purely image based—it avoids the nontrivial problem of estimating 3D geometric mesostructure and/or macrostructure from the image data.

Unlike all previous methods, TensorTextures is a *nonlinear* BTF model. Mathematically, TensorTextures stems from a multilinear (i.e., tensor) algebra approach to the analysis of image ensembles [Vasilescu and Terzopoulos 2002]. Here, we apply the multilinear theory in the context of computer graphics. TensorTextures may be regarded as the first instantiation of a novel, multilinear framework for image-based rendering. A major technical advantage of our framework is that the underlying tensor formulation can disentangle and explicitly represent each of the multiple factors inherent to image formation. This stands in contrast to principal components analysis (PCA), a linear (i.e., matrix) model typically computed using the singular value decomposition (SVD), which has so far been the standard BTF representation/compression method [Sattler et al. 2003] and is, in fact, subsumed by our multilinear framework. A major limitation of PCA is that it captures the overall variation in the image ensemble without explicitly distinguishing what proportion is attributable to each of the relevant factors—illumination change, viewpoint change, etc. Our method prescribes a more sophisticated tensor decomposition that further analyzes this overall variation into individually encoded constituent factors using a novel set of basis functions.

Unfortunately, there does not exist a tensor SVD that offers all the nice mathematical properties of the matrix SVD, and there are several ways to decompose tensors [Kolda 2001]. Furukawa *et al.* [2002] propose a compression method that expresses sampled BTF data as a linear combination of lower-rank tensors, but this is inadequate as a possible generalization of PCA. Although the authors report improved compression rates over PCA, their method suffers from the same fundamental drawback: It does not permit separate dimensionality reduction (compression) to be guided independently in viewing, illumination, and spatial variation, as our method does.

Harnessing the power of multilinear algebra, the algebra of higher-order tensors, our approach contributes a novel, explicitly multimodal model with which to tackle the BTF modeling/rendering problem. Our model is computed through a tensor decomposition known as the N-mode SVD, a natural extension to tensors of the conventional matrix SVD.

2 Tensor Concepts

A *tensor* is a higher order generalization of a vector (1st-order tensor) and a matrix (2nd-order tensor). Tensors are multilinear mappings over a set of vector spaces. The *order* of tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is N.[1] An element of $\mathcal{A}$ is denoted as $a_{i_1 \ldots i_n \ldots i_N}$, where $1 \le i_n \le I_n$. In tensor terminology, column vectors are referred to as mode-1 vectors and row vectors as mode-2 vectors. The mode-n vectors are the column vectors of matrix $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 I_2 \ldots I_{n-1} I_{n+1} \ldots I_N)}$ that results from *flattening* the tensor $\mathcal{A}$, as illustrated in Fig. 2.

The *mode-n product* of a tensor $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_n \times \ldots \times I_N}$ and a matrix $\mathbf{M} \in \mathbb{R}^{J_n \times I_n}$ is denoted by $\mathcal{A} \times_n \mathbf{M}$. Its result is

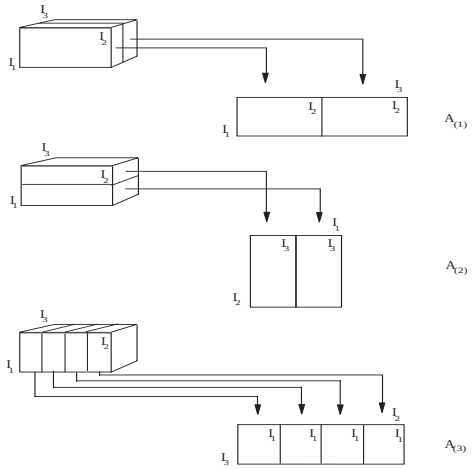

Figure 2: Flattening a (3rd-order) tensor. The tensor can be flattened in 3 ways to obtain matrices comprising its mode-1, mode-2, and mode-3 vectors (from [Vasilescu and Terzopoulos 2002]).

a tensor $\mathcal{B} \in \mathbb{R}^{I_1 \times \ldots \times I_{n-1} \times J_n \times I_{n+1} \times \ldots \times I_N}$ whose entries are $b_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} m_{j_n i_n}$.[2] The mode-$n$ product can be expressed in terms of flattened matrices as $\mathbf{B}_{(n)} = \mathbf{M}\mathbf{A}_{(n)}$.

A matrix $\mathbf{D} \in \mathbb{R}^{I_1 \times I_2}$ is a two-mode mathematical object with two associated vector spaces, a row space and a column space. The SVD orthogonalizes these two spaces and decomposes the matrix as $\mathbf{D} = \mathbf{U}_1 \mathbf{S} \mathbf{U}_2^T$, the product of an orthogonal column-space represented by the left matrix $\mathbf{U}_1 \in \mathbb{R}^{I_1 \times J_1}$, a diagonal singular value matrix $\mathbf{S} \in \mathbb{R}^{J_1 \times J_2}$, and an orthogonal row space represented by the right matrix $\mathbf{U}_2 \in \mathbb{R}^{I_2 \times J_2}$. This matrix product can be rewritten in terms of mode-n products as $\mathbf{D} = \mathbf{S} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2$.

By extension, an order $N > 2$ tensor $\mathcal{D}$ is an N-dimensional array with N associated vector spaces. The *N-mode SVD* is a generalization of the SVD that orthogonalizes these N spaces and decomposes the tensor as the mode-n product of the N orthogonal spaces,

$$\mathcal{D} = \mathcal{Z} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \ldots \times_n \mathbf{U}_n \ldots \times_N \mathbf{U}_N, \qquad (1)$$

as illustrated in Fig. 3 for the case $N = 3$. Tensor $\mathcal{Z}$, known as the *core tensor*, is analogous to the diagonal singular value matrix in conventional matrix SVD. It is important to realize, however, that the core tensor does not have a diagonal structure; rather, $\mathcal{Z}$ is in general a full tensor. The core tensor governs the interaction between the *mode matrices* $\mathbf{U}_n$, for $n = 1, \ldots, N$. Mode matrix $\mathbf{U}_n$ contains the orthonormal vectors spanning the column space of the matrix $\mathbf{D}_{(n)}$ that results from the mode-n flattening of $\mathcal{D}$, as was illustrated in Fig. 2.

Our **N-mode SVD algorithm** for decomposing $\mathcal{D}$ according to equation (1) is as follows:

1. For $n = 1, \ldots, N$, compute matrix $\mathbf{U}_n$ in (1) by computing the SVD of the flattened matrix $\mathbf{D}_{(n)}$ and setting $\mathbf{U}_n$ to be the left matrix of the SVD.[3]

[1] We denote scalars by lower case letters ($a, b, \ldots$), vectors by bold lower case letters ($\mathbf{a}, \mathbf{b} \ldots$), matrices by bold upper-case letters ($\mathbf{A}, \mathbf{B} \ldots$), and higher-order tensors by calligraphic upper-case letters ($\mathcal{A}, \mathcal{B} \ldots$).

[2] The mode-n product of a tensor and a matrix is a special case of the inner product in multilinear algebra and tensor analysis. Note that for tensors and matrices of the appropriate sizes, $\mathcal{A} \times_m \mathbf{U} \times_n \mathbf{V} = \mathcal{A} \times_n \mathbf{V} \times_m \mathbf{U}$ and $(\mathcal{A} \times_p \mathbf{U}) \times_n \mathbf{V} = \mathcal{A} \times_p (\mathbf{V}\mathbf{U})$.

[3] For a non-square, $m \times n$ matrix $\mathbf{A}$, the matrix $\mathbf{U}$ in the SVD $\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ can be computed more efficiently, depending on which

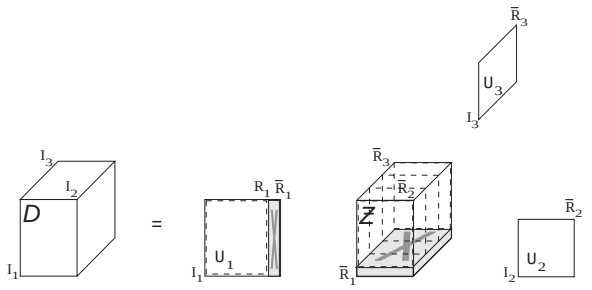

Figure 3: Three-mode tensor decomposition and dimensionality reduction through truncation. The data tensor D can be decomposed into the product of a core tensor Z and N mode matrices $U_1 \ldots U_N$; for the $N = 3$ case illustrated here, $D = Z \times_1 U_1 \times_2 U_2 \times_3 U_3$. Deletion of the last mode-1 eigenvector of U_1 incurs an approximation error equal to $\sigma_{R_1}^2$, which equals the Frobenius norm of the (grey) subtensor of $Z_{i_1 = R_1}$.

2. If it is needed, solve for the core tensor as follows:

$$Z = D \times_1 U_1^T \times_2 U_2^T \ldots \times_n U_n^T \ldots \times_N U_N^T. \quad (2)$$

Dimensionality reduction is useful for data compression in image-based rendering algorithms. Optimal dimensionality reduction in matrix PCA results from the truncation of eigenvectors associated with the smallest singular values in the SVD. Multilinear analysis admits an analogous dimensionality reduction scheme, but it offers much greater control, enabling a tailored truncation of each mode in accordance with the importance of the mode to the rendering task.

A truncation of the mode matrices of the data tensor D results in an approximation $\hat{D}$ with reduced ranks $R_1 \leq R_1, R_2 \leq R_2, \ldots, R_N \leq R_N$, where $R_n = \text{rank}_n(D) = \text{rank}(D_{(n)}) = \text{rank}(U_n)$ is the n-rank of D for $1 \leq n \leq N$. The error of this approximation is

$$\|D - \hat{D}\|^2 = \sum_{i_1 = R_1 + 1}^{R_1} \cdots \sum_{i_N = R_N + 1}^{R_N} Z_{i_1 i_2 \ldots i_N}^2 \quad (3)$$

$$\leq \sum_{i_1 = R_1 + 1}^{R_1} \sigma_{i_1}^2 + \cdots + \sum_{i_N = R_N + 1}^{R_N} \sigma_{i_N}^2; \quad (4)$$

that is, it is bounded by the sum of squared singular values associated with the discarded singular vectors, where the singular value associated with the m^{th} singular vector in mode matrix U_n is equal to the Frobenius norm $\|Z_{i_n = m}\|$ of subtensor $Z_{i_n = m}$ of the core tensor Z (Fig. 3). The truncated mode matrix is denoted $\hat{U}_n$.

Computing the optimal dimensionality reduction is unfortunately not straightforward in multilinear analysis. Truncation of the mode matrices that result from the N-mode SVD algorithm yields a reasonably good reduced-dimensionality approximation $\hat{D}$, but it is generally not optimal. See [Lathauwer et al. 2000] (also [Kroonenberg and de Leeuw 1980] and [Tucker 1966]) for an iterative, alternating least squares (ALS) algorithm that improves the mode matrices $\hat{U}_n$ and hence $\hat{D}$, although it does not guarantee a globally optimal result.

dimension of $\mathbf{A}$ is smaller, by decomposing either the $m \times m$ matrix $\mathbf{A}\mathbf{A}^T = \mathbf{U}\mathbf{S}^2\mathbf{U}^T$ and then computing $\mathbf{V}^T = \mathbf{S}^+\mathbf{U}^T\mathbf{A}$ or by decomposing the $n \times n$ matrix $\mathbf{A}^T\mathbf{A} = \mathbf{V}\mathbf{S}^2\mathbf{V}^T$ and then computing $\mathbf{U} = \mathbf{A}\mathbf{V}\mathbf{S}^+$, where the $^+$ superscript denotes the pseudoinverse.

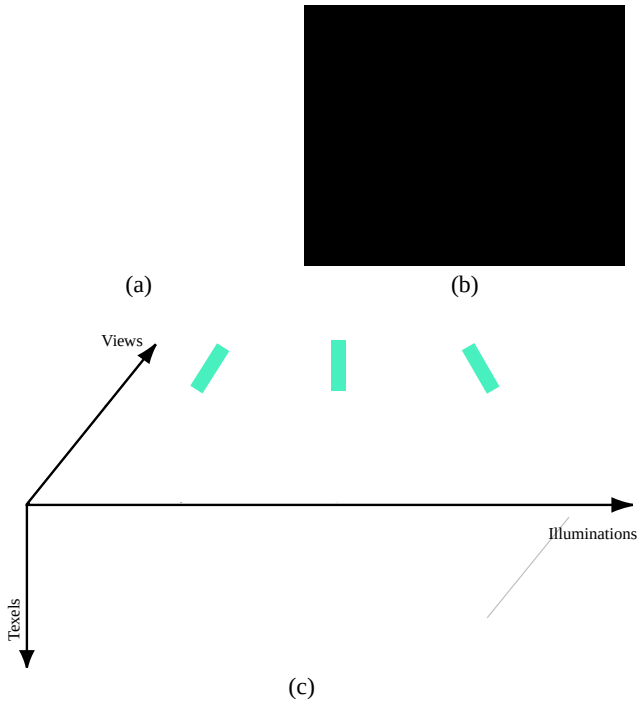

Figure 4: Image acquisition and representation. Images are acquired from several different view directions over the viewing hemisphere (a) and, for each viewpoint, under several different illumination conditions over the illumination hemisphere (b). The ensemble of acquired images is organized in a third-order tensor (c) with view, illumination, and texel modes. Although the contents of the texel mode are vectors of RGB texel values, for clarity they are displayed as 2D images in this and subsequent figures.

3 Multilinear Analysis

Given an ensemble of images of a textured surface, we define an image data tensor $D \in \mathbb{R}^{T \times I \times V}$, where V and I are, respectively, the number of different viewing conditions and illumination conditions associated with the image acquisition process, and T is the number of texels in each texture image. As a concrete example, which we will use for illustrative purposes, consider the synthetic scene of scattered coins shown in Fig. 4. A total of 777 sample RGB images of the scene are acquired from $V = 37$ different view directions over the viewing hemisphere (Fig. 4(a)), each of which is illuminated by a light source oriented in $I = 21$ different directions over the illumination hemisphere (Fig. 4(b)). The size of each image is $T = 240 \times 320 \times 3 = 230400$. Images acquired from an oblique viewing angle are rectified with respect to the frontal image acquired from the top of the viewing hemisphere.

We organize the rectified images as a 3rd-order tensor $D \in \mathbb{R}^{230400 \times 21 \times 37}$, a portion of which is shown in Fig. 4(c). In principle, we can apply the N-mode SVD algorithm from Section 2 to decompose this tensor as follows:

$$D = Z \times_{\text{texel}} U_{\text{texel}} \times_{\text{illum}} U_{\text{illum}} \times_{\text{view}} U_{\text{view}}, \quad (5)$$

into the product of three orthonormal mode matrices and a core tensor Z that governs the interaction between the different modes. The mode matrices encode the second-order statistics of each of the factors. The column vectors of the 37×37 mode matrix U_{view} span the view space. The rows of U_{view} encode an illumination and texel invariant representation for each of the different views. The column

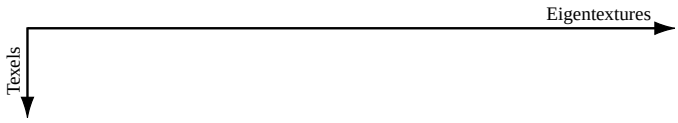

Figure 5: U_{texel} contains the PCA eigenvectors, which are the principal axes of variation across all images.

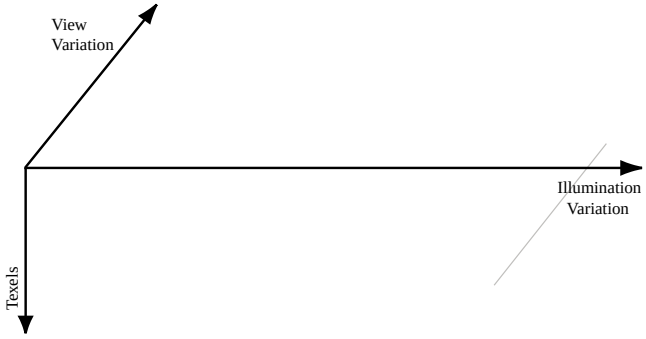

Figure 6: A partial visualization of the 37×21 TensorTextures bases of the coins image ensemble.

vectors of the 21×21 mode matrix U_{illum} span the illumination space. The rows of U_{illum} encode a view and texel invariant representations for each of the different illuminations.[4] Fig. 5 shows the column vectors of the 230400×777 mode matrix U_{texel}, which span the texel space and are, in fact, the PCA eigenvectors (i.e., "eigenimages" or "eigentextures"), since they were computed by performing an SVD on the matrix $D_{(texel)}$ obtained by mode-3 flattening the data tensor D. Hence, our multilinear analysis subsumes PCA, as we show formally in [Vasilescu and Terzopoulos 2002].

TensorTextures models how the appearance of a textured surface varies with view and illumination. The TensorTextures representation (Fig. 6) is the product

$$T = Z \times_2 U_{texel} \quad (6)$$

$$= D \times_2 U_{illum}^T \times_3 U_{view}^T, \quad (7)$$

where the second equation is preferable in practice, since it prescribes computation of the relatively small matrices U_{view} and U_{illum} rather than the generally large matrix U_{texel} that PCA would compute. Thus, TensorTextures transform eigentextures into a tensorial representation of the variation and co-variation of modes (view and illumination). It characterizes how viewing parameters and illumination parameters interact and multiplicatively modulate the appearance of a surface under variation in view direction (θ_v, φ_v), illumination direction (θ_i, φ_i), and position (x, y) over the surface.

TensorTextures is a more compact representation than PCA. In our example, PCA would decompose the image ensemble into 777 basis vectors (eigentextures), each of dimension 230400, and represent each image by a coefficient vector of length 777, which specifies what proportion of each basis vector to accumulate in order to obtain that image. By contrast, TensorTextures decomposes the image ensemble into 37×21 basis vectors of the same dimension, and represents each image by two coefficient vectors, one of length 37 to encode the view and the other of length 21 to encode the illumination. Thus, each image is represented by $37 + 21 = 58$ coefficients. Fig. 7 shows how these coefficients multiplicatively modulate the TensorTextures basis vectors in order to approximate (or render) an image.

[4] The first coordinates of the row vectors of $\mathbf{U}_{view}$ ($\mathbf{U}_{illum}$) encode the directions on the viewing (illumination) hemisphere associated with the acquired images. This information is not provided explicitly; it is learned by the decomposition from the image ensemble.

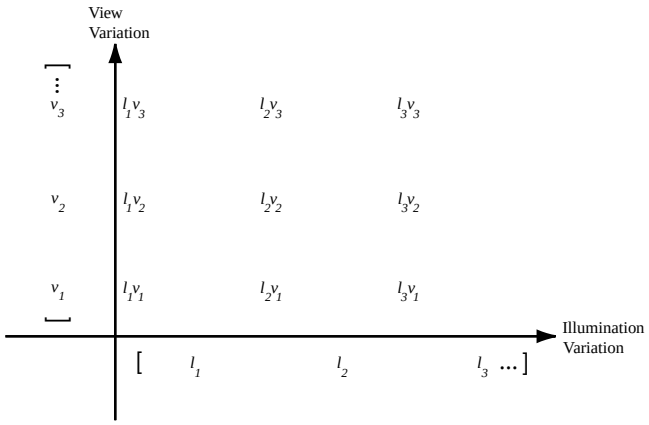

Figure 7: The lower left image is rendered by multiplicatively modulating each of the TensorTextures basis vectors with the coefficients in the view coefficient vector v and the illumination coefficient vector l.

More importantly, our multilinear analysis enables a *strategic* dimensionality reduction, which is a mode-specific version of the conventional linear dimensionality reduction of PCA. In particular, we truncate the mode matrices U_{view} and U_{illum} to obtain U_{view} and U_{illum}, and apply the aforementioned iterative ALS algorithm [Lathauwer et al. 2000] until convergence in order to improve these truncated mode matrices. Whereas dimensionality reduction in PCA results in unpredictable image degradation, multilinear models yield image degradation that can be controlled independently in viewing and illumination.

Fig. 8 compares TensorTexture image compression against PCA compression. Note in Fig. 8(c) that the 95.2% reduction of the illumination dimensionality suppresses illumination effects such as shadows and highlights, but that it does not substantially degrade the clarity of the texture, since the rank of the view mode matrix has not been reduced. However, a comparable compression using PCA results in the blurred texture of Fig. 8(b). Although the RMS error of the TensorTexture compression relative to the original image (Fig. 8(a)) is larger than the PCA compression, its *perceptual error* [Teo et al. 1994] is smaller, yielding a substantially better image quality than comparable PCA compressions. Fig. 8(d) shows the degradation of the TensorTexture if we drastically compress in the view mode. Applying PCA compression in Fig. 8(e), we retain the 111 (out of 777) most dominant eigentextures. Applying TensorTextures, we compress the dimensionality of the illumination mode from 21 to 3 ($R_{illum} = 3$) in Fig. 8(f). Since $R_{view} = 37$, we retain 37×3 TensorTexture basis vectors, equaling the number of retained PCA basis vectors. The total number of coefficients representing the compressed images is $37 + 3$. Fig. 8(g)-(i) illustrate the same scenario with 31×4 TensorTexture basis vectors.

4 Multilinear Rendering

Our TensorTextures basis (eq. (7) and Fig. 6) leads to a straightforward rendering algorithm, which is illustrated in Fig. 7. To render an image d, we compute

$$\mathbf{d} = T \times_2 \mathbf{l}^T \times_3 \mathbf{v}^T, \quad (8)$$

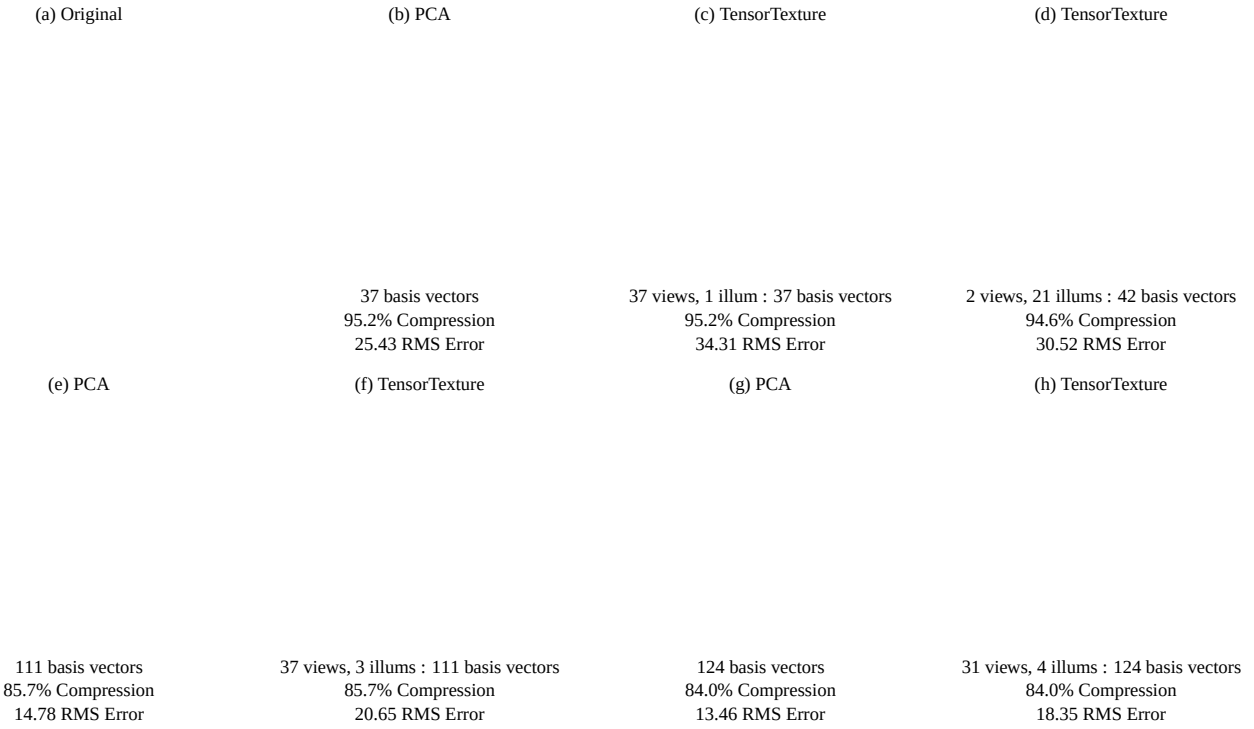

Figure 8: The perceptual error incurred by compressing the illumination representation of the TensorTextures model is smaller than that of indiscriminate PCA compression in a subspace of comparable dimension. (a) Original image. (b)–(h) PCA and TensorTexture compressions of image (a) using various numbers of basis vectors. The label above each image indicates the type of compression, while the annotations below indicate the basis set, the compression rate, and the root mean squared (RMS) error relative to the original image (a). For example, the PCA compression (e) retains 111 of the 777 most dominant eigentexture basis vectors, while the TensorTexture image compression (f) retains 111 TensorTextures bases associated with $\mathbf{U}_{\text{view}} \in \mathbb{R}^{37 \times 37}$ and $\mathbf{U}_{\text{illum}} \in \mathbb{R}^{21 \times 3}$, which reduces the illumination representation from 21 dimensions to 3 ($R_{\text{illum}} = 3$). The RMS error of the PCA-compressed images are lower, as expected, yet comparable TensorTexture compressions have the better perceptual quality.

where $\mathbf{v}$ and $\mathbf{l}$ are, respectively, the view and illumination representation vectors associated with the desired view and illumination directions. These will in general be *novel* directions, in the sense that they will differ from the *observed* directions associated with sample images in the ensemble. Given a novel view (illumination) direction, we first find the three nearest observed view (illumination) directions which form a triangle on the view (illumination) hemisphere that contains this novel direction. We then compute the novel view (illumination) representation vector $\mathbf{v}$ ($\mathbf{l}$) as a convex combination, using homogeneous barycentric coordinates, of the view (illumination) representation vectors associated with the three observed view (illumination) directions. Note that this algorithm is appropriate for a planar surface, since every texel of the rendered texture shares the same view/illumination representation. The algorithm (8) was applied in the *Treasure Chest* animation to render the coins TensorTexture on a planar surface in the chest under continuously varying view and illumination directions (Fig. 1).

When rendering a TensorTexture $\mathbf{d}$ on a curved surface, the view $\mathbf{v}_j$ and illumination $\mathbf{l}_j$ representation vectors associated with each texel j of $\mathbf{d}$ are computed with respect to the given view and illumination directions as well as the direction of the surface normal at the center of texel j. The RGB value d_j for texel j is then computed as follows:

$$d_j = T_j \times_2 \mathbf{l}_j^T \times_3 \mathbf{v}_j^T, \tag{9}$$

where T_j is a subtensor of the TensorTexture which governs the interaction between view and illumination for texel j (Fig. 9).

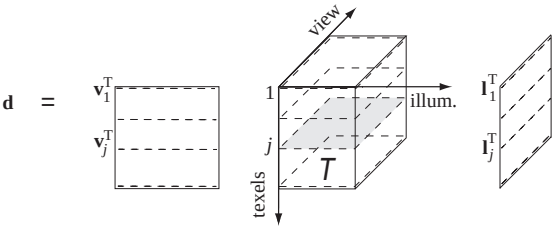

Figure 9: TensorTexture rendering when every texel j has a different associated view $\mathbf{v}_j$ and illumination $\mathbf{l}_j$ direction.

5 Additional Results

We have applied the TensorTextures algorithm to two synthetic image ensembles: The coins ensemble, which has served to illustrate our TensorTextures algorithm, and a corn image ensemble whose TensorTextures representation is illustrated in Fig. 10. Fig. 11 demonstrates the application of the algorithm of eq. (9) to render the corn TensorTexture onto a perfect cylinder that forms the head of a scarecrow, lit from two different directions. As the cylinder is rotated, the TensorTexture shows the desired 3D effects, including self-occlusion and self-shadowing between the corn kernels. Fig. 12 shows the closing shot of an animated short called *Scarecrows Quarterly* in which we have employed the TensorTex-

Figure 10: TensorTexture bases for the corn texture.

Figure 11: Renderings, with different light source directions, of the corn TensorTexture mapped onto a cylinder that forms the scarecrow head.

tured scarecrow head.

Both of our synthetic image datasets were acquired by rendering 3D graphics models of surfaces featuring considerable mesostructure. As was the case of the coins, the images of the corn surface were also acquired by rendering the surface from 37 different view and 21 different illumination directions. Both the coins and the corn TensorTexture models retain $37 \times 11 = 407$ TensorTexture basis vectors by reducing the illumination mode from 21 to 11, while retaining all of the basis vectors of the view mode in order to maintain the sharpness of the rendered images.

It takes considerable time to render each of the original sample images because of the nontrivial scene geometries and rendering methods employed. In particular, Alias|wavefront *Maya* consumed around 180 seconds on average to render the coins images and around 20 seconds on average to render the corn images on a 2GHz P4 with 1GB of RAM. After our TensorTextures model has been computed offline, the online rendering of the TensorTextures is significantly more efficient. For the coins example, the rendering of the TensorTextured surfaces for arbitrary viewpoints and illuminations, implemented in not especially well-optimized Matlab code, took on average 1.6 seconds per image on the same workstation. Furthermore, because it is image-based, the TensorTextures online rendering speed is independent of the scene complexity.

We have also applied our TensorTextures algorithm to images

Figure 12: Still from the *Scarecrows@Quarterly* animation.

of natural textured surfaces from the University of Bonn BTF database. The materials and methods for the image acquisition are detailed in [Sattler et al. 2003]. Fig. 13 shows Impalla (a stone) and Corduroy TensorTextures mapped onto spheres using the rendering algorithm of eq. (9). The TensorTextures bases were computed from ensembles of RGB sample images, each of size $256 \times 256 \times 3$, acquired under 81 view and 81 illumination directions. The image data were organized as $81 \times 81 \times 196608$ tensors D. The view and illumination mode matrices were computed in accordance with step 1 of the N-mode SVD algorithm, and their dimensionality was reduced from 81 to 61 and from 81 to 27, respectively, yielding the reduced mode matrices $U_{\text{view}} \in \mathbb{R}^{81 \times 61}$ and $U_{\text{illum}} \in \mathbb{R}^{81 \times 27}$. The 61×27 TensorTextures bases vectors were computed according to (7). As a final demonstration, we have created a *Flintstone Phonograph* animation which maps the Impalla TensorTexture on the planar turntable surface (Fig. 14).

6 Conclusion

We have introduced a multilinear approach to the image-based rendering of textured surfaces. Our TensorTextures algorithm provides a parsimonious, explicitly multifactor approximation to the bidirectional texture function (BTF). It is computed through a tensor decomposition known as the N-mode SVD, which is a natural extension to tensors of the conventional matrix singular value decomposition (SVD). We have demonstrated the algorithm in example applications to synthetic and natural texture image ensembles.

We believe that our approach can handle data sets that result from the variation of additional factors, such as BTF scale (i.e., zooming into or away from a textured surface), high dynamic range (HDR) BTF acquisition at multiple exposures, or temporally varying BTFs, such as aging skin or leaves changing colors in the fall. In future work, we would also like to incorporate into TensorTextures a variant of view-dependent displacement maps [Wang et al. 2003].

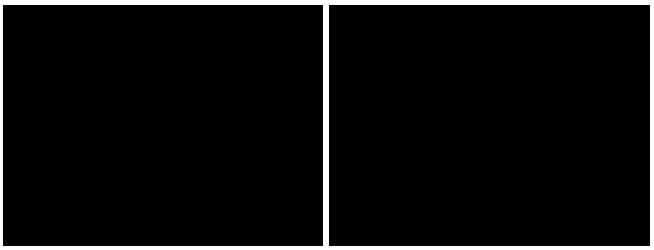

Figure 13: Tapalla (left) and Corduroy (right) TensorTextures rendered on spheres. A third order image data tensor associated with 81 viewpoint directions, 81 illumination directions, and 196608 texels (256 rows × 256 cols. × 3 channels) was employed to the TensorTextures bases $T = D \times U^{T}_{\text{illum}} \times_3 U^{T}_{\text{view}}$. Image compression was obtained by retaining 61 × 27 TensorTextures basis associated with $U_{\text{view}} \in \mathbb{R}^{81\times 61}$ and $U_{\text{illum}} \in \mathbb{R}^{81\times 27}$, which reduces the viewpoint representation from 81 dimensions to 61 ($R_{\text{view}} = 61$) and the illumination representation from 81 dimensions to 27 ($R_{\text{illum}} = 27$).

Figure 14: Still from the *Flintstone Phonograph* animation.

Acknowledgements

We thank Jared M. Silver for his valuable assistance with 3D modeling, animation, and audio production. We also thank Svetlana Stenchikova for exploring programming issues, as well as Davi Geiger, Lexing Ying, Tamara Kolda, and Lieven de Lathauwer for helpful discussions. Greg Ward provided encouragement and comments on a draft.

References

DANA, K.J., VAN GINNEKEN, B., NAYAR, S.K., AND KOENDERINK, J.J. 1999. Reflectance and texture of real-world surfaces. *ACM Trans. Graphics 18*, 1, 1–34.

FURUKAWA, R., KAWASAKI, H., IKEUCHI, K., AND SAKAUCHI, M. 2002. Appearance based object modelling using texture database: Acquisition, compression and rendering. In *8th Eurographics Workshop on Virtual Environments*, 257–266.

GORTLER, S.J., GRZESZCZUK, R., SZELISKI, R., AND COHEN, M.F. 1996. The lumigraph. *Computer Graphics (Proc. SIGGRAPH) 30*, 43–54.

HAN, J.Y. AND PERLIN, K. 2003. Measuring bidirectional texture reflectance with a kaleidoscope. *ACM Trans. Graphics (Proc. SIGGRAPH) 22*, 741–748.

KOENDERINK, J.J., AND VAN DOORN, A.J. 1996. Illuminance texture due to surface mesostructure. *Journal of the Optical Society of America 13*, 3, 452–463.

KOLDA, T.G. 2001. Orthogonal tensor decompositions. *SIAM J. Matrix Analysis and Applications 23*, 1, 243–255.

KOUDELKA, M.L., MAGDA, S., BELHUMEUR, P.N., AND KRIEGMAN, D.J. 2003. Acquisition, compression, and synthesis of bidirectional texture functions. In *Proc. 3rd Int. Workshop on Texture Analysis and Synthesis* (Oct), Nice, France, 59–64.

KROONENBERG, P., AND DE LEEUW, J. 1980. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika 45*, 69–97.

LATHAUWER, L. DE, MOOR, B. DE, AND VANDEWALLE, J. 2000. On the best rank-1 and rank-($R_1, R_2, \ldots, R_n$) approximation of higher-order tensors. *SIAM J. Matrix Analysis and Applications 21*, 4, 1324–1342.

LEVOY, M., AND HANRAHAN, R. 1996. Light field rendering. *Computer Graphics (Proc. SIGGRAPH) 30*, 31–42.

LIU, X., YU, Y., AND SHUM, H.-Y. 2001. Synthesizing bidirectional texture functions for real-world surfaces. *ACM Trans. Graphics (Proc. SIGGRAPH) 20*, 97–106.

MALZBENDER, T., GELB, D., AND WOLTERS, H. 2001. Polynomial texture maps. *Computer Graphics (SIGGRAPH 01)* (Aug), 519–528.

MATUSIK, W., PFISTER, H., BRAND, M., AND MCMILLAN, L. 2003. A data driven reflectance model. *Computer Graphics (SIGGRAPH 03) 22*, 3 (July), 759–769.

MESETH, J., MULLER, G., SATTLER, M., AND KLEIN, R. 2003. BTF rendering for virtual environments. In *Proc. Virtual Concepts 2003* (Nov), 356–363.

SATTLER, M., SARLETTE, R., AND KLEIN, R. 2003. Efficient and realistic visualization of cloth. In *Proc. Eurographics Symposium on Rendering*, 167–177.

SUYKENS, F., BERGE, K. VON, LAGAE, A., AND KLEIN, R. 2003. Interactive rendering with bidirectional texture functions. *Comp. Graphics Forum (Proc. Eurographics) 22*, 3, 463–472.

TEO, P. AND HEEGER, P. 1994. Perceptual image distortion. In *Proc. IEEE Conf. Image Processing* (Nov), 982–986.

TONG, X., ZHANG, J., LIU, L., WANG, X., GUO, B., AND SHUM, H.-Y. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Trans. Graphics (Proc. SIGGRAPH) 21*, 3, 665–672.

TUCKER, L. 1966. Some mathematical notes on three-mode factor analysis. *Psychometrika 31*, 279–311.

VASILESCU, M. A. O., AND TERZOPOULOS, D. 2002. Multilinear analysis of image ensembles: TensorFaces. In *Proc. European Conf. Computer Vision*, 447–460.

WANG, L., WANG, X., TONG, X., LIN, S., HU, S., GUO, B., AND SHUM, H.Y. 2003. View displacement mapping. *ACM Trans. Graphics (Proc. SIGGRAPH) 22*, 3, 334–339.