**TensorTextures:**
Multilinear Image-Based Rendering

Computer Graphics

**Motivation**

- **Goal:** Generation of photorealistic virtual environments
- **Classical Computer Graphics: Model-based Rendering**
  - From object models to images
  - Model specifies geometry of a scene and surface properties
  - Images are generated by projecting 3D model onto an image plane and computing surface shading
- **Photorealism requires complex models**
  - Difficult
  - Time consuming

**Image-Based Rendering**


- World is modeled by a collection of images (and possibly some coarse geometry)
- These images are used to synthesize novel images representing the scene from arbitrary viewpoints and illuminations
- Advantages:
  - Rendering is decoupled from the scene complexity
  - Photorealism is improved

**Our Contribution**

- We introduce a tensor framework for image-based rendering (IBR)
  - Specifically, rendering of 3D textured surfaces
- Surface appearance is determined by the complex interaction of multiple factors:
  - Scene geometry
  - Illumination
  - Imaging

**Bidirectional Texture Function**

- **BTF:** Captures the appearance of extended textured surfaces with
  - Spatially varying reflectance
  - Surface mesostructure (3D texture)
  - Subsurface scattering
  - Etc.
- Generalization of **BRDF**, which accounts only for surface microstructure at a point

**BTF Texture Mapping**

[Dana et al. 1999]

<table>
<thead>
<tr>
<th>Concrete</th>
<th>Pebbles</th>
<th>Plaster</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="Concrete.png" alt="Concrete" /></td>
<td><img src="Pebbles.png" alt="Pebbles" /></td>
<td><img src="Plaster.png" alt="Plaster" /></td>
</tr>
<tr>
<td>Standard Texture Mapping</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BTF Texture Mapping</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
BTF

- Reflectance as a function of position on surface, view direction, and illumination direction

\[ f_{\text{BTF}}(x, y, \theta_v, \phi, \theta_i, \phi_i) \]

- The BTF captures shading and mesostructural self-shadowing, self-occlusion, interreflection

TensorTexture Mapping

- Standard Texture Mapping
- TensorTexture Mapping

TensorTextures: Learns BTFs from ensembles of sample images
Nonlinear generative BTF model

Background

- BTF introduced by Dana et al. [1999]
- BTF acquisition devices
  - [Debevec et al. 2000]
  - [Dana 2001]
  - [Furukawa et al. 2002]
  - [Han & Perlin 2003] (BTF Kaleidoscope)
- BTF based rendering methods
  - Polynomial texture maps
    - [Malzbender et al. 2001]
  - Synthesis of BTFs for curved surfaces
    - [Liu et al. 2001]
    - [Tong et al. 2002]

TensorTextures Overview

1. Mathematical foundations: Eigentextures
   - Linear Analysis / Principal Components Analysis
     - fixed viewpoint, changing illumination
     - changing viewpoint and illumination
2. TensorTextures
   - Nonlinear (multilinear) Analysis / Tensor decomposition
3. Experiments and results
"Eigentextures" – PCA (Matrix Algebra)

Simple Data Acquisition:
Fixed Viewpoint, Varying Illumination

Sample images are points in "pixel space"

The 1-Mode Case
(fixed viewpoint, varying illumination)

Principal Components Analysis (PCA) - Eigentextures

Image Representation using PCA

• Eigentextures – captures variation across illuminations

• Note: This is a linear representation

Its PCA Representation

An arbitrary image

d = Uc
Sampling Multiple Viewpoints and Illuminations

- This poses a 3-mode BTF estimation problem
  - Viewpoint, illumination, and pixel modes

Rectifying Homography

- Image unwarping \( \mathbf{p}' = \mathbf{Hp} \)
  - \( \mathbf{H} \) can be computed given at least 4 fiducials \( \mathbf{p}' \) & \( \mathbf{p} \)

Applying PCA

- Eigentextures – variation across views and illuminations

PCA Reconstruction

- Original
- 111 basis vectors
- 33 basis vectors

TensorTextures (Tensor Algebra)
This leads to a multilinear BTF learning method.

**Background on Tensor Decomposition**

- **Factor Analysis**: Psychometrics, Econometrics, Chemometrics, ...
- **SVD**: [Beltrami, 1873] (Giornale di Matematiche 11) “Sulle funzioni bilineari”
  - [Eckart and Young, 1936] (Psychometrika) “The approximation of one matrix by another of lower rank”
  - [Kroonenberg and De Leeuw, 1980] – 3-mode ALS
  - [Franco, 1992] – tensor algebra
  - [Denis & Dhome, 1989]
  - [De Lathauwer, 1997]

- A matrix $D \in \mathbb{R}^{n \times n}$ has a column and row space
- SVD orthogonalizes these spaces and decomposes $D$
  $$D = U_1SU_2^T$$
  ( $U_1$ contains the "eigentextures")
- Rewrite in terms of mode-$n$ products

**Matrix Decomposition - SVD**

$D = S \times_1 U_1 \times_2 U_2$
Tensor Decomposition

- \( D \) is a \( n \)-dimensional matrix, comprising \( N \)-spaces
- \( N \)-mode SVD is the natural generalization of SVD
- \( N \)-mode SVD orthogonally projects these spaces & decomposes
  \( D \) as the mode-\( n \) product of \( N \)-orthogonal spaces

\[
D = Z \times_1 U_1 \times_2 U_2 \times_3 U_3 \times_4 \cdots \times_n U_N
\]

- \( Z \): core tensor; governs interaction between mode matrices
- \( U_n \): mode-\( n \) matrix, is the column space of \( D_{(n)} \)

Tensor Texture Decomposition

\[
D = \mathbf{Z} \times_1 U_{\text{texels}} \times_2 U_{\text{illums}} \times_3 U_{\text{views}}
\]

N-Mode SVD Algorithm

1. For \( n=1, \ldots, N \), compute matrix \( U_{(n)} \) by computing the SVD of the flattened matrix \( D_{(n)} \) and setting \( U_{(n)} \) to be the left matrix of the SVD.

2. Solve for the core tensor as follows

\[
Z = D \times_1 U_{1}^T \times_2 U_{2}^T \cdots \times_N U_{N}^T
\]

Computing \( U_{\text{views}} \)

- \( D_{\text{views}} \): flatten along the view point dimension
- \( U_{\text{views}} \): orthogonalize the column space of \( D_{\text{views}} \)

Computing \( U_{\text{illums}} \)

- \( D_{\text{illums}} \): flatten along the illumination dimension
- \( U_{\text{illums}} \): orthogonalize the column space of \( D_{\text{illums}} \)
Computing $U_{\text{texels}}$

- $D_{\text{texels}}$: flatten $\mathcal{D}$ along the pixel dimension
- $U_{\text{texels}}$: orthogonal column space of $D_{\text{texels}}$ — eigenimages

N-Mode SVD Algorithm

1. For $n=1,\ldots,N$, compute matrix $U_n$ by computing the SVD of the flattened matrix $D_n$ and setting $U_n$ to be the left matrix of the SVD.

2. Solve for the core tensor as follows

$$Z = \mathcal{D} \times_1 U_1^T \times_2 U_2^T \cdots \times_N U_N^T$$

Mode-N Product

- Mode-n product is a generalization of the product of two matrices
- It is the product of a tensor with a matrix
- Mode-n product of $\mathcal{B} \in IR^{I_1 \times I_2 \times \cdots \times I_N}$ and $\mathcal{A} \in IR^{I_1 \times I_2 \times \cdots \times I_N}$, and $M \in IR^{I_1 \times I_2}$

$$\mathcal{B} = \mathcal{A} \times_n M$$

$$\langle A \times_n M \rangle_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_{n-1} i_{n+1} \cdots i_N} m_{i_n}$$

TensorTextures: $T = Z \times_3 U_{\text{pixels}}$

- Explicitly represent covariance across factors

TensorTextures: explicit variation in illuminations and viewing direction
TensorTextures vs. PCA

• Multilinear Analysis / TensorTextures:
  \[ D = Z \times U_{\text{texels}} \times U_{\text{illums}} \times U_{\text{views}} \]

• Linear Analysis:
  \[ D_{\text{levels}} = U_{\text{levels}} \times Z_{\text{levels}} \times (U_{\text{views}} \otimes U_{\text{illums}})^T \]

  data matrix  basis matrix  coefficient matrix

• TensorTextures subsumes PCA / Eigentextures

Strategic Dimensionality Reduction

TensorTextures Dimensionality Reduction

System Diagram

Computing \( v_{\text{new}} \): Homogeneous Barycentric Blend

\[ v_{\text{new}} = v_i \Delta t_i + v_j \Delta t_j + v_k \Delta t_k \]

\[ \Delta t_i = \frac{[(v_i - v_{\text{new}}) \cdot (v_j - v_{\text{new}})]}{|v_i - v_j|^2} \]
Synthesis Algorithm / Texture Representation

\[ \mathbf{d} = \mathbf{T} \times_2 \mathbf{v}_{\text{new}} \times_3 \mathbf{v}_{\text{new}} \]

Rendered Texture for a Planar Surface

Rendered Textures for Cylinder

Rendering on Arbitrary Geometry

Bonn natural BTF datasets

TensorTextures renderings

Video

TensorTextures