Polynomial Texture Maps

Tom Malzbender
Dan Gelb
Hans Wolters

Hewlett-Packard Laboratories
Texture Mapping
[ Catmull ‘74 ]

Pro:
• Photographic input
• Simplicity
• Hardware Support

Con:
• Unrealistic Silhouettes
• Static Lighting

Bump Mapping
[ Blinn 78 ]

Pro:
• Lighting Variations

Con:
• Per-pixel lighting computation
• Filtering is Problematic
• Procedural Synthesis (Not image-based.)
[ Rushmeier ‘97 ]
PTM Demonstration:

**Top:** Polynomial Texture Map

**Bottom:** Conventional Texture Map

Advantages:
- Image based unlike Bump Mapping
- Simpler to evaluate than Bump Mapping
- Can leverage Mip Mapping
Acquiring PTM’s Photographically

- Fixed object, fixed camera.
- Limited to Diffuse Objects.
Acquiring Lighting Models

Debevec

Goerghiades '99
Modeling Pixel Color Changes Directly

\[ L(u,v;l_u,l_v) = a_0 l_u^2 + a_1 l_v^2 + a_2 l_u l_v + a_3 l_u + a_4 l_v + a_5 \]
Modeling Pixel Color Changes Directly

\[ L(u,v;l_u,l_v) = a_0 l_u^2 + a_1 l_v^2 + a_2 l_u l_v + a_3 l_u + a_4 l_v + a_5 \]
Polynomial Texture Mapping

PTM: Store RGB per pixel and store polynomial coefficients ($a_0-a_5$) per textel:

$$L(u,v;l_u,l_v) = a_0l_u^2 + a_1l_v^2 + a_2l_ul_v + a_3l_u + a_4l_v + a_5$$

Why Polynomials?

- Compact Representation
- Consist solely of multiplies and adds.
- Cheap to evaluate on both modern CPUs and VLSI
What PTMs Capture

- Shading effects.
- Self shadowing.
- Interreflections.
- Sub-surface scattering.
**Light Direction Parametrization**

\[ L(u,v; l_u, l_v) = a_0 l_u^2 + a_1 l_v^2 + a_2 l_u l_v + a_3 l_u + a_4 l_v + a_5 \]

- \( u, v \) - texture coordinates
- \( a_0-a_5 \) - fitted coefficients stored in texture map
- \( l_u, l_v \) - projection of light direction into texture plane

*\( l_u, l_v \) can be scan-converted without normalization.*
Fitting PTMs to Image Data

• Given \( N \) light sources we compute the best fit for \((a_0-a_6)\) in the \(L_2\) norm using S.V.D.

• SVD computed once for a given lighting arrangement.

\[
\begin{bmatrix}
  l^2_{u0} & l^2_{v0} & l_{u0}l_{v0} & l_{u0} & l_{v0} & 1 \\
  l^2_{u1} & l^2_{v1} & l_{u1}l_{v1} & l_{u1} & l_{v1} & 1 \\
  \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
  l^2_{uN-1} & l^2_{vN-1} & l_{uN-1}l_{vN-1} & l_{uN-1} & l_{vN-1} & 1 \\
\end{bmatrix}
\begin{bmatrix}
a_0 \\
a_1 \\
\vdots \\
a_5 \\
\end{bmatrix}
=
\begin{bmatrix}
L_0 \\
L_1 \\
\vdots \\
L_{N-1} \\
\end{bmatrix}
\]
PTM Fitting Errors

- Smoothing is not spatial, it occurs in light space.
- High spatial frequencies are well preserved.
- Hard shadows become softer.
- Point lights become area lights.
PTM Formats

<table>
<thead>
<tr>
<th>Format</th>
<th>Per Pixel Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRGB</td>
<td></td>
</tr>
<tr>
<td>RGB</td>
<td>+ R,G,B</td>
</tr>
<tr>
<td>ENC</td>
<td>Index to L.U.T. storing Polynomial Coefficients</td>
</tr>
</tbody>
</table>
Scale and Bias

\[ a_i = \lambda_i (a_i' - \Omega_i) \]

- Allows polynomial coefficients to be stored as 8 bit values.
- Handles large dynamic range among coefficients.
- 12 Global values stored per texture map (LRGB).
Mip Mapping PTM’s vs Bump Maps

• Mip-mapping bump maps effectively smooths geometry.
  [ Schilling 97 ]

• PTMs are linear in polynomial coefficients so mip-mapping PTMs is accurate.

\[ L(u,v;l_u,l_v) = a_0 l_u^2 + a_1 l_v^2 + a_2 l_u l_v + a_3 l_u + a_4 l_v + a_5 \]

\[ \frac{1}{n} \sum_{i,j \in \Omega} L_{r,g,b}(a_{0-5}(u_i,v_j)) = L_{r,g,b}\left(\frac{1}{n} \sum_{i,j \in \Omega} a_{0-5}(u_i,v_j)\right) \]
Evaluation – MMX / SSD Implementation

Parallel computation

- Fixed point arithmetic
- Pack 4 PTM coefficients in 64 bit integer MMX register
- Parallel multiply/adds

Yields 6.5M pixels/sec on a 1 Ghz CPU (software only).
Evaluation – Programmable Hardware

Vertex Processing
- Store precomputed tangent and binormal per vertex
- Vertex code projects light vector onto tangent and binormal

Pixel Processing
- $l_u, l_v$ passed from vertex stage
- PTM coefficients stored in 2 textures
- Calculate using dot products / multiplies / adds
- Single pass on current hardware
Bump Maps as PTM’s

- If PTM rendering methods are implemented, they can be used for rendering bump maps.
- Provides specular and diffuse effects.

\[
I = I_a k_a + I_d k_d (N \cdot L) + I_s k_s (N \cdot H)^n
\]

- Precompute \( N \cdot V \) PTM L.U.T.
- Convert Normals to PTM using L.U.T.
- Render PTM
  - Diffuse - evaluate \( N \cdot L \)
  - Specular - evaluate \( (N \cdot H)^n \)
Complex Shading Effects
Combine with hardware lighting
- Use with existing Phong lighting
- PTM models more complex reflectance effects
  Examples:
  - Anisotropic Fresnel
  - Off-specular
2D Applications

- Enhancement of Cuneiform Tablets w/ Zuckerman USC
- PTMs for Short Image Sequences
- PTMs for Depth of Focus Effects.
Surface Normal Extraction

Yields maximum surface brightness
Surface Normal Extraction

For a diffuse object, coordinates of \((l_u, l_v)\) that maximize luminance yield local surface normals.

Setting \(\frac{\partial L}{\partial u} = \frac{\partial L}{\partial v} = 0\) yields:

\[
\begin{align*}
 l_u &= \frac{a_2 a_4 - 2a_1 a_3}{4a_0 a_1 - a_2^2} \\
 l_v &= \frac{a_2 a_3 - 2a_0 a_4}{4a_0 a_1 - a_2^2}
\end{align*}
\]

Providing a surface normal per textel:

\[
\vec{N} = (l_{u0}, l_{v0}, \sqrt{1 - l_{u0}^2 - l_{v0}^2})
\]
Specular Enhancement

Cuneiform tablet courtesy of Dr. Bruce Zuckerman at U.S.C.
Diffuse Gain - a reflection transformation that:

• Keeps the surface normal fixed.
• Increases the curvature (second derivative) of the reflectance function by $g$.

\[
\begin{align*}
    a_0' &= ga_0 \\
    a_1' &= ga_1 \\
    a_2' &= ga_2 \\
    a_3' &= (1 - g)(2a_0l_{u0} + a_2l_{v0}) + a_3 \\
    a_4' &= (1 - g)(2a_1l_{v0} + a_2l_{u0}) + a_4 \\
    a_5' &= (1 - g)(a_0l_{u0}^2 + a_1l_{v0}^2 + a_2l_{u0}l_{v0}) + \\
        (a_3 - a_3')l_{u0} + (a_4 - a_4')l_{v0} + a_5
\end{align*}
\]
Light Direction Extrapolation

• Input images are collected across a hemisphere of light directions, i.e. \(-1 \leq l_u, l_v \leq 1\)

• PTM’s can be evaluated outside of the hemisphere, 
  \((l_u, l_v < -1 \text{ or } l_u, l_v > 1)\)
PTMs as Parametric Images

For each \((u,v)\) we have:

\[
a_0 l_u^2 + a_1 l_v^2 + a_2 l_u l_v + a_3 l_u + a_4 l_v + a_5
\]
Depth of Focus
Palletization

- Light Space Lookup table – contains polynomials.
- LUT constructed by K-means clustering.
- RGB values can be stored in L.U.T. or image space.

Each textel stores:
R, G, B, Index

8 bit index

PTM

L.U.T.
Palletization

- Light Space Lookup table – contains polynomials.
- LUT constructed by K-means clustering.
- RGB values can be stored in L.U.T. or image space.

Each textel stores:
Index

PTM

| a₀, a₁, a₂, a₃, a₄, a₅, a₆, R, G, B |
| a₀, a₁, a₂, a₃, a₄, a₅, a₆, R, G, B |
| a₀, a₁, a₂, a₃, a₄, a₅, a₆, R, G, B |
| ... |
| ... |
| a₀, a₁, a₂, a₃, a₄, a₅, a₆, R, G, B |

8 bit values only

L.U.T.
Compression

- Allows better rate/distortion tradeoff than palletization.
- Similar to compression of multispectral images.
- Removes correlations within and between byte planes.
- Sacrifices pixel independence.
- Visible artifacts don’t appear until ~10 bits/pixel.

**Perceptually lossless results:**

<table>
<thead>
<tr>
<th>Original Size</th>
<th>Lossless</th>
<th>Loss = 1 grey level</th>
<th>Loss = 2 grey levels</th>
<th>Loss = 4 grey levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>72 - 144 bits</td>
<td>27.4 bits</td>
<td>17.1 bits</td>
<td>13.5 bits</td>
<td>10.1 bits</td>
</tr>
</tbody>
</table>
Future Work, Conclusions + Web Tools

- PTM’s are fast, compact, effective representations.
- PTM’s encoding opacity channels?
- Full BRDF’s can be modelled using PTMs by trading off spatial variation with viewing angle.
- Applications in Medicine, Forensics, Paleontology

Tools available at [hpl.hp.com/ptm](http://hpl.hp.com/ptm)

- sample PTMs
- PTM viewer
- Polynomial Fitter
- PTM format document
The End

hpl.hp.com/ptm