Overview

- Introduction to normal mapping
- Previous work
- Tight Frame Compression
- Evaluation
Overview

- Introduction to normal mapping
- Previous work
- Tight Frame Compression
- Evaluation
Normal Maps

- Add geometric detail with texture maps
- Store value of the local normal vector
- Realistic, detailed appearance at low cost
Normal Map Generation

- Create two versions of geometry
  - Lo-res
    - overall shape
  - Hi-res
    - shape + details
Normal Map Generation

- Shoot rays
  - from the lo-res surface
- to the hi-res surface
Normal Map Generation

- Store the normal vector from the intersection points in an RGB texture
Normal Map Generation

- Render lo res surface + normal map

![20k triangles](image1.png) ![2 triangles + normal map](image2.png)
Motivation

- We need compression!

20k triangles

2 triangles + normal map

Increased texture bandwidth!!!
Overview

• Introduction to normal mapping

• Previous work

• Tight Frame Compression

• Evaluation
Previous Work

- Surface normal compression [Deering 95]
- S3 Texture Compression/DXTC [Iourcha99]
- 3Dc [ATI05]
  - Dedicated format for normal maps
- e3Dc [Munkberg 06]
  - Enhanced 3Dc with rotations and diff-coding
- Adaptive bit rate [Wong06,Yang06]
- Vector Quantization [Yamakasi et. al 06]
Design choices

- Fixed rate encoding at 8 bpp
  - Fast random access
  - Simple decompressor
  - 4x4 texel blocks
- Use advantages from e3Dc
  - Rotation encoding
  - Differential encoding
  - Variable point distribution
- Exploit coherence between channels
3Dc Overview

- Divide the input file in 4x4 blocks of texels
3Dc - Projection

- Project the normals on the xy plane and find min/max values of the bounding box.
Map each texel to a quantized \((x, y)\) value

- Eight levels in \(x\) & \(y\); \((3,3)\) bits to select \((x_i, y_i)\)
3Dc - Compressed Block

- Compressed form
  - 4x8 bits for $x_{\text{min}}$, $x_{\text{max}}$, $y_{\text{min}}$, $y_{\text{max}}$
  - 6x16 bits for per texel index
  - Total: 128 bits per block : 8 bits per texel

<table>
<thead>
<tr>
<th></th>
<th>6b</th>
<th>6b</th>
<th>6b</th>
<th>6b</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6b</td>
<td>6b</td>
<td>6b</td>
<td>6b</td>
</tr>
<tr>
<td></td>
<td>6b</td>
<td>6b</td>
<td>6b</td>
<td>6b</td>
</tr>
<tr>
<td>xmin</td>
<td>8b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xmax</td>
<td>8b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ymin</td>
<td>8b</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ymax</td>
<td>8b</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Problems with 3Dc

- Difficult scenarios
  - Slow gradients, sharp edges, directed features
3Dc can be improved - e3Dc

- Observation (used in DXT1)
  - Swap min & max values → same reconstruction levels
  - One bit unused per channel!
  - Use these to signal new modes!

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_{min} &lt; X_{max}</td>
<td>Y_{min} &lt; Y_{max}</td>
<td>Standard 3Dc</td>
</tr>
<tr>
<td>X_{min} ≥ X_{max}</td>
<td>Y_{min} &lt; Y_{max}</td>
<td>Rotation 30</td>
</tr>
<tr>
<td>X_{min} &lt; X_{max}</td>
<td>Y_{min} ≥ Y_{max}</td>
<td>Rotation 60</td>
</tr>
<tr>
<td>X_{min} ≥ X_{max}</td>
<td>Y_{min} ≥ Y_{max}</td>
<td>Differential mode</td>
</tr>
</tbody>
</table>
Rotation Compression

- Rotate coordinate frame for a more compact bounding box
- e3Dc uses three angles: 0, 30 and 60 degrees
Variable Point Distribution

- 3Dc: points in a 8x8 grid
- Our approach: use aspect ratio of bbox
  - BBox twice as wide -> 16x4 instead of 8x8
  - Automatic selection -> No extra cost
Variable Point Distribution in e3Dc
Differential Encoding

- Slowly varying normals are problematic:
  - Smallest interval is too wide (range/255)
  - The interval cannot be placed accurately enough
Differential Encoding

- Slowly varying normals are problematic:
  - Smallest interval is too wide \((\text{range/255})\)
  - The interval cannot be placed accurately enough

Reinterpret the bits differentially!
\((x_{\text{min}}, x_{\text{max}}) \rightarrow (x_{\text{min}}^*, \Delta x)\)
Overview

- Introduction to normal mapping
- Previous work
- Tight Frame Compression
- Evaluation
Tight Frame Compression

- Example: directed lines
- coherence between channels!
Tight Frame Compression

- OBB is tighter than AABB
- Store two point \( p \) & \( q \)
  - Enough to define local coordinate frame \( e_1, e_2 \)
  - arbitrary rotation
- Store box aspect ratio
  - aspect ratio = height/width
- Variable Point Distribution still works!
Target: 128 bits per block - 8 bpp

Indices with 6 x 16 bits as before

32 bits left for encoding OBB

- p & q are encoded using 7+7 bits each
- Four bits for the aspect ratio, a
- sixteen levels as $a_i = 1/32 + h_i/16$, $i = 0...15$

Ex:

```
a_1

\[ \text{a}_7 \]

\[ \text{a}_{15} \]
```
Additional Mode

- Unused bit combinations?
- \( p_x \geq q_x \) & \( p_y \geq q_y \)
- Use to flag a mode!
Tight Frame Differential Mode

- Trigger when $p_x \geq q_x$ and $p_y \geq q_y$
- Encode all normals in a small square
  - Limit the square side length
- Compact representation but high resolution!
  - **Lower left corner** 2x11 bits
  - **Side** of the square: 8 bits
  - 8x8 grid over the square
  - $6 \times 16 + 2 \times 11 + 8 = 126$ bits
Tight Frame Differential Mode

- **Differential mode resolution**
  - **corner** 2x11 bits, **side**: 8 bits,
  - Limit max square side to 1/4
  - min square size = 1/(4*2^8) = 1/1024

- **Standard mode resolution**
  - p and q with (2x7) bits each
  - max size = 1
  - min square size = 1/128
Overview

• Introduction to normal mapping
• Previous work
• Tight Frame Compression
• Evaluation
Evaluation

- PSNR
- Max error
  - A few bad normals “cracks” a smooth surface
- Angular deviation [Abate 05]
  - Angle between compressed and original normal
  - Motivation: Even a small error in the specular reflection is visible
  - Presented as a histogram
Test Images
### PSNR

<table>
<thead>
<tr>
<th></th>
<th>3Dc</th>
<th>e3Dc</th>
<th>TF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30.87</td>
<td>32.74</td>
<td>33.50</td>
</tr>
</tbody>
</table>
Max angular deviation

![Graph showing max angular deviation for different models and algorithms]
Error distribution

![Graph](image)
False Color

3Dc  e3Dc  Tight Frame
Rendered Quality

SSIM: structural similarity image metric

Original 100%
3Dc 93.3%
e3Dc 95.7%
Tight Frame 96.3%
Pros & Cons

• Increased hardware cost
  • Twice the number of “multiply and divide” units compared to e3Dc, but still lightweight

• Unlike e3Dc, there is no backward-compatibility with 3Dc
  • The format cannot be used for encoding two 1D signals
  • Not depending on 3Dc patent

• More robust results!
Conclusions

• Higher quality than 3Dc
  • Still at 8 bits per texels
  • More flexibility with OBB, VPD and diff-mode

• Rather modest HW extensions

• API support?
  • Potential candidate for OpenGL ES?
Thank you!

- http://graphics.cs.lth.se