High Quality Normal Map Compression

Jacob Munkberg  Tomas Akenine-Möller  Jacob Ström

Lund University  Lund University  Ericsson Research
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 the mesh
  - Lo-res mesh - overall shape
  - Hi-res mesh - shape + details
Normal Map Generation

- Shoot rays from the lo-res surface to the hi-res surface
- Store the normal vector from the intersection points in a texture
• Render lo res surface + normal map

Hi res - 20k triangles

Lo res - 2 triangles + normal map
We need compression!

- Render lo res surface + normal map

Hi res - 20k triangles
Lo res - 2 triangles + normal map

Increased Memory bandwidth!!!
Previous Work

- Surface normal compression
  - [Deering 1995] targeting geometry compression
  - Costly algorithm for HW ~ 12 bits per normal
- S3 Texture Compression / DXTC
  - Good for colors - not designed for normals
  - Visible artifacts (edges, subtle curvatures)
- 3Dc
  - Dedicated format for normals
  - 8 bits per texel
3Dc Overview

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

- Decompression
  - Reconstruct x & y from min/max values and texel indices.
  - Derive z from the unit length condition
  - Can be done in a pixel shader
- Supported by new [ATI] graphics cards

\[ z = \sqrt{1 - x^2 - y^2} \]
Problems with 3Dc

- Difficult scenarios
  - Slow gradients, sharp edges, directed features

3Dc
Original
**3Dc can be improved!**

- **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>xmin &lt; xmax</td>
<td>ymin &lt; ymax</td>
<td>Standard 3Dc</td>
</tr>
<tr>
<td>xmin ≥ xmax</td>
<td>ymin &lt; ymax</td>
<td>?</td>
</tr>
<tr>
<td>xmin &lt; xmax</td>
<td>ymin ≥ ymax</td>
<td>?</td>
</tr>
<tr>
<td>xmin ≥ xmax</td>
<td>ymin ≥ ymax</td>
<td>?</td>
</tr>
</tbody>
</table>
New techniques for 3Dc

- Rotation Compression
- Variable Point Distribution
- Differential Encoding
Rotation Compression

- Rotate coordinate frame for a more compact bounding box
- Storage cost: one angle per block
3Dc with Rotation

![Graph showing the relationship between PSNR (dB) and the number of angles. The PSNR increases linearly with the number of angles.]
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

3Dc

Variable Point Distribution
• Slowly varying normals are problematic:
  • Smallest interval is too wide (range/255)

  \[ x_{\text{min}} \rightarrow (x_{\text{min}}^*, \Delta x) \rightarrow x_{\text{min}}^* + \Delta x \rightarrow x_{\text{max}} \]

• The interval cannot be placed accurately enough

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

- Suppose we can reinterpret the bits!
- A suitable encoding for slow maps:
  - Use 11 bits (8.3) for $x_{\text{min}}^*$ and $y_{\text{min}}^*$ base values
  - 4 bits (2.2) for $\Delta x$ and $\Delta y$
    - $x_{\text{max}} = x_{\text{min}}^* + \Delta x$, $y_{\text{max}} = y_{\text{min}}^* + \Delta y$
  - Smallest representable interval four times smaller
  - Location of an interval border encoded with three additional fractional bits
Mapping Function

- We want to reinterpret the bits of $x_{\text{start}}$ & $x_{\text{stop}}$ while preserving $x_{\text{start}} \geq x_{\text{stop}}$
- Numbering scheme:
  - Remap triangle $x_{\text{start}} \geq x_{\text{stop}}$ to upper rectangle
- Example
  - (2x3 bit values) 32 numbers in upper rectangle: 5 bits can be extracted
  - (2x8 bit) 15 bits extracted: 11 (base) + 4 ($\Delta$)

<table>
<thead>
<tr>
<th>$x_{\text{start}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7</td>
</tr>
<tr>
<td>0   1   2   3   4   5   6   7</td>
</tr>
<tr>
<td>1   9  10  11  12  13  14  15</td>
</tr>
<tr>
<td>2   8  17  18  19  20  21  22</td>
</tr>
<tr>
<td>3   7  16  19  23  24  25  30</td>
</tr>
<tr>
<td>4   6  14  18  22  24  26  31</td>
</tr>
<tr>
<td>5   5  13  17  21  25  29  31</td>
</tr>
<tr>
<td>6   4  12  16  20  24  28  31</td>
</tr>
<tr>
<td>7   3  11  15  19  23  27  31</td>
</tr>
</tbody>
</table>

Table 4: The average PSNR for the normal maps presented

(c) The Eurographics Association 2009
Combined Scheme

- Three rotations, variable point distribution and differential encoding
- Select modes by comparing $x_{\text{start}}, x_{\text{stop}}, y_{\text{start}} \& y_{\text{stop}}$

<table>
<thead>
<tr>
<th>mode</th>
<th>X</th>
<th>Y</th>
<th>bits</th>
<th>vpd</th>
</tr>
</thead>
<tbody>
<tr>
<td>I: rot 0°</td>
<td>$x_{\text{start}} &lt; x_{\text{stop}}$</td>
<td>$y_{\text{start}} &lt; y_{\text{stop}}$</td>
<td>8+8</td>
<td>yes</td>
</tr>
<tr>
<td>II: rot 30°</td>
<td>$x_{\text{start}} \geq x_{\text{stop}}$</td>
<td>$y_{\text{start}} &lt; y_{\text{stop}}$</td>
<td>8+8</td>
<td>yes</td>
</tr>
<tr>
<td>III: rot 60°</td>
<td>$x_{\text{start}} &lt; x_{\text{stop}}$</td>
<td>$y_{\text{start}} \geq y_{\text{stop}}$</td>
<td>8+8</td>
<td>yes</td>
</tr>
<tr>
<td>IV: diff</td>
<td>$x_{\text{start}} \geq x_{\text{stop}}$</td>
<td>$y_{\text{start}} \geq y_{\text{stop}}$</td>
<td>8.3+2.2</td>
<td>no</td>
</tr>
</tbody>
</table>
Average improvement: \(~3 \text{ dB}\)
Slowly varying map example
Munkberg, Akenine/Möller, and Ström (High Quality Normal Map Compression

Figure 11: A typical game normal map flit3, rendered in a real-time shader development application, with a cube reflection map-

Figure 12: The normal map flit3, rendered in a high-end renderer, with HDR environment mapping, texture filtering and advanced anti-aliasing-

Left: 3Dc-Middle: unocompr-essed map-Right: our algorithm-

As can be seen in the images, 3Dc shows more natural artifacts, and some features even disappear-

Our new algorithm shows higher quality, even though some artifacts remains-

©The Eurographics Association 2006.
Results - Off-line rendering

Munkberg, Akenine/Möller, and Ström (High Quality Normal Map Compression

Figure 11: A typical game normal map flt 3, rendered in a real-time shader development application, with a cube reflection map-

Figure 12: The normal map flt 3, rendered in a high-end frame renderer, with HDR environment mapping, texture filtering and advanced anti-aliasing-

Left: 3Dc-Middle: uncompressed map-Right: our algorithm-

As can be seen in the images, 3Dc shows more unwobbling artifacts, and some features even disappear-Our new algorithm shows higher quality, even though some artifacts remains-

© The Eurographics Association 2006.
3Dc Decompression

produce \( x_i \)

\[
x_i = x_{\text{min}} + i \frac{(x_{\text{max}} - x_{\text{min}})}{7}
\]

produce \( y_j \)

\[
y_j = y_{\text{min}} + j \frac{(y_{\text{max}} - y_{\text{min}})}{7}
\]
Variable Point Distribution

produce $x_i$

$$x_i = x_{\text{min}} + i \frac{(x_{\text{max}} - x_{\text{min}})}{a}$$

$a \in \{1, 3, 5, 7, 15, 31\}$

produce $y_j$

$$y_j = y_{\text{min}} + j \frac{(y_{\text{max}} - y_{\text{min}})}{b}$$

$b \in \{1, 3, 5, 7, 15, 31\}$
Differential Encoding

compute \((x_{\text{min}}, x_{\text{max}})\) from \((x_{\text{start}}, x_{\text{stop}})\)

produce \(x_i\)
\[x_i = x_{\text{min}} + i \times (x_{\text{max}} - x_{\text{min}}) / a\]
\[a \in \{1, 3, 5, 7, 15, 31\}\]

produce \(y_j\)
\[y_j = y_{\text{min}} + j \times (y_{\text{max}} - y_{\text{min}}) / b\]
\[b \in \{1, 3, 5, 7, 15, 31\}\]
Rotation Decoding

compute \((x_{\text{min}}, x_{\text{max}})\) from \((x_{\text{start}}, x_{\text{stop}})\)

produce \(x_i\)
\[
x_i = x_{\text{min}} + i \cdot (x_{\text{max}} - x_{\text{min}})/a
\]
\[a \in \{1, 3, 5, 7, 15, 31\}\]

compute \((y_{\text{min}}, y_{\text{max}})\) from \((y_{\text{start}}, y_{\text{stop}})\)

produce \(y_j\)
\[
y_j = y_{\text{min}} + j \cdot (y_{\text{max}} - y_{\text{min}})/b
\]
\[b \in \{1, 3, 5, 7, 15, 31\}\]
Backward compatible with 3Dc and DXT5

compute \((x_{\text{min}}, x_{\text{max}})\)
from
\((x_{\text{start}}, x_{\text{stop}})\)

produce \(x_i\)
\[x_i = x_{\text{min}} + i \cdot \frac{(x_{\text{max}} - x_{\text{min}})}{a}\]
\(a \in \{1/3, 5/7, 15/31\}\)

compute \((y_{\text{min}}, y_{\text{max}})\)
from
\((y_{\text{start}}, y_{\text{stop}})\)

produce \(y_j\)
\[y_j = y_{\text{min}} + j \cdot \frac{(y_{\text{max}} - y_{\text{min}})}{b}\]
\(b \in \{1/3, 5/7, 15/31\}\)
Conclusions

• Higher quality than 3Dc
  • Still at 8 bits per texels
  • More flexibility with new modes

• Simple HW extensions

• Backwards compatible
  • 3Dc is a subset of our approach
  • DXT5 can be decoded with same HW

• API support?
Thank You!

- Swedish Foundation for Strategic Research (Mobile Graphics Grant)
- NVIDIA Fellowship
- ATI for making all the details of 3Dc openly available
- Pixologic
- Illuminate Labs
- Questions?
### Average PSNR over all maps

<table>
<thead>
<tr>
<th>mode</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3Dc</td>
<td>36.4</td>
</tr>
<tr>
<td>3Dc + Point Distr.</td>
<td>37.5</td>
</tr>
<tr>
<td>3Dc + Point Distr. + Rot</td>
<td>38.8</td>
</tr>
<tr>
<td>3Dc + Point Distr. + Rot + Diff</td>
<td>39.4</td>
</tr>
</tbody>
</table>
Mapping Examples (5 bit)

- Upper half of rect:
  \( x_{\text{start}} = 6, \ x_{\text{stop}} = 2 \)
  - \( v = (x_{\text{stop}} << 3) \text{ OR } x_{\text{start}} \)
  - \( v = 22 \)

- Lower half of rect:
  \( x_{\text{start}} = 7, \ x_{\text{stop}} = 5 \)
  - \( v = (\text{NOT}(x_{\text{stop}}) << 3) \text{ OR } \text{NOT}(x_{\text{start}}) \)
  - \( v = 16 \) (NOT = bitwise inversion)

- Encoding: \( x_{\text{start}} \) lower part of \( v \), \( x_{\text{stop}} \) upper part. Invert both if \( x_{\text{stop}} > x_{\text{start}} \)
## Variable Point Distribution

<table>
<thead>
<tr>
<th>aspect ratio ( a = \frac{y_{\text{max}} - y_{\text{min}}}{x_{\text{max}} - x_{\text{min}}} )</th>
<th>distribution ( (d_x \times d_y) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a &lt; 1/8 )</td>
<td>32 ( \times ) 2</td>
</tr>
<tr>
<td>( 1/8 \leq a &lt; 1/2 )</td>
<td>16 ( \times ) 4</td>
</tr>
<tr>
<td>( 1/2 \leq a \leq 2 )</td>
<td>8 ( \times ) 8</td>
</tr>
<tr>
<td>( 2 &lt; a \leq 8 )</td>
<td>4 ( \times ) 16</td>
</tr>
<tr>
<td>( a &gt; 8 )</td>
<td>2 ( \times ) 32</td>
</tr>
</tbody>
</table>