TEXTURE COMPRESSION USING SMOOTH PROFILE FUNCTIONS

J. RASMUSSON\textsuperscript{1,2}, J. STRÖM\textsuperscript{1}, P. WENNERSTEN\textsuperscript{1}, M. DOGGETT\textsuperscript{2} AND T. AKENINE-MÖLLER\textsuperscript{2}

\textsuperscript{1} ERCSSON RESEARCH
\textsuperscript{2} LUND UNIVERSITY
Problems:

- Memory can get full
- Bus can get full (performance bottleneck)
TEXTURE COMPRESSION HELPS

Benefits:

• More textures fit in memory
• Less traffic on bus = higher performance
• Less traffic on bus = lower power consumption
POWER SAVINGS

› Especially good for mobile devices
... unless you want to carry an extra battery.
NICER RENDERING WORKS AGAINST US

Low Bandwidth

Computer Graphics infancy

Texture Compression

High Bandwidth

Higher resolution textures

New Rendering Techniques
RADIOSITY NORMAL MAPPING

› Improves lighting realism in 3D games.
  - Catches light interactions between images: light bouncing, soft shadows, color bleed
› First used by Valve in Half-Life 2
› Other examples include Mirrors Edge from DICE, EA.
› Can be thought of as “directional light maps”.

image courtesy of DICE EA
RADIOSITY NORMAL MAPPING

› Sample the light in three directions in every pixel.

\[
\begin{pmatrix}
\sqrt{\frac{2}{3}}, & 0, & \frac{1}{\sqrt{3}}
\end{pmatrix}
\]

images courtesy of Valve
RADIOSITY NORMAL MAPPING

› Sample the light in three directions in every pixel.

\[
\left\{-\frac{1}{\sqrt{6}}, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}}\right\}, \quad \left\{\frac{2}{\sqrt{3}}, 0, \frac{1}{\sqrt{3}}\right\}
\]

images courtesy of Valve
RADIOSITY NORMAL MAPPING

› Sample the light in three directions in every pixel.
› We will refer to these components as “radiosity light maps”.

\[
\begin{align*}
\left\{ -\frac{1}{\sqrt{6}}, -\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}} \right\} \\
\left\{ -\frac{1}{\sqrt{6}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{3}} \right\} \\
\left\{ \frac{2}{\sqrt{3}}, 0, \frac{1}{\sqrt{3}} \right\}
\end{align*}
\]
A normal map is used to decide how much of every radiosity light map should be used.

Together with the albedo this gives the final image.

The normal map and the albedo can often be repeated/reused. The radiosity lightmaps are unique to each surface.

A lot of data! Mirror’s Edge has 3GB of compressed data.

Images courtesy of Valve
OLD CODECS NOT IDEAL

› Radiosity light maps contain a lot of smooth transitions.
› Traditional texture codecs not good at smooth transitions.
› DXT1 can only handle four colors per block

Radiosity Light Map from Mirror’s Edge, Image courtesy of DICE, EA
OLD CODECS NOT IDEAL (CONT.)

› ETC2 has a planar mode that can handle linear gradients, but not faster changes.

original

ETC2

linear model works well

linear model breaks down
OLD CODECS NOT IDEAL (CONT.)

› ETC2 has a planar mode that can handle linear gradients, but not faster changes.
› Higher quality codecs exist (e.g. Microsoft’s BC7) but comes at a penalty of doubling the bit rate.

original

ETC2

linear model works well

linear model breaks down
TRAIN OF THOUGHTS

› Problem:
  – How to compress radiosity lightmaps well?

› Inspiration:
  – Smooth blocks contain very little information. Should compress well!
  – ETC2 planar mode was built on this notion.

› Observation:
  – Most smooth blocks vary only in one direction.

A one-dimensional problem!

a lot of variation along this direction
no variation along this direction
GRAY SCALE EXAMPLE

A lot of variation along this direction

No variation along this direction

Position along red axis

Position along green axis

Gray scale
GRAY SCALE EXAMPLE

a lot of variation along this direction

no variation along this direction

gray scale

position along red axis

modeled as constant

position along green axis
We approximate this with a simple non-linear function that we call “profile function”.

\[ f(x) = \begin{cases} 
1 & \text{if } x \geq 1 \\
3x^2 - 2x^3 & \text{if } 0 < x < 1 \\
0 & \text{if } x \leq 0 
\end{cases} \]
PROFILE FUNCTIONS

gray scale

extrusion direction

profile function
GRAY SCALE EXAMPLE

Establish line

Ax + By + C = 0

f(d + 0.5)

d
GRAY SCALE EXAMPLE

calculate signed distance $d$ to this line

Establish line $Ax + By + C = 0$

$$f(d + 0.5)$$
GRAY SCALE EXAMPLE

calculate signed distance $d$ to this line

Establish line $Ax+By+C=0$

dark value

$f(d + 0.5)$

-0.25

+0.15
GRAY SCALE EXAMPLE

calculate signed distance $d$ to this line

Establish line $Ax+By+C=0$

$f(d + 0.5)$

bright value

dark value

-0.25

+0.15
GRAY SCALE EXAMPLE

› With just the parameters A, B and C in $Ax+By+C=0$, it is possible to describe a rather large set of blocks:
GRAY SCALE EXAMPLE

With just the parameters A, B and C in $Ax+By+C=0$, it is possible to describe a rather large set of blocks:
GRAY SCALE EXAMPLE

› With just the parameters A, B and C in $Ax+By+C=0$, it is possible to describe a rather large set of blocks:
GRAY SCALE EXAMPLE

› With just the parameters A, B and C in $Ax + By + C = 0$, it is possible to describe a rather large set of blocks:
With just the parameters $A$, $B$ and $C$ in $Ax+By+C=0$, it is possible to describe a rather large set of blocks:

Since $A$ and $B$ only represents a rotation, one can instead store $\theta$, and use $A=\cos(\theta)$, $B=\sin(\theta)$. 
Gray Scale Example

By introducing a division by $w$ before using the function, the width of the function can be varied.

gray scale = $f((d + 0.5)/w)$

small $w = \text{small width}$

medium width

large $w = \text{large width}$
GRAY SCALE EXAMPLE

› With the extra parameter \textit{width}, more blocks are possible:

- sharp edges
- smooth transitions
EXTENSION TO COLOR

› Instead of using the output from the function \( f() \) directly as a gray scale value, it is used for interpolation between two colors:

\[
i = f\left(\frac{d+0.5}{w}\right)
\]

› \( \text{color} = (1-i)\times\text{color}_A + i\times\text{color}_B \)
EXTENSION TO COLOR

› Thus by storing colorA and colorB, in addition to $\theta$, C and width, many blocks can be represented.
› Unique color in every pixel possible!
THE OTHER DIRECTION

 › Typically, the block is not completely constant in the other direction -> block artifacts.

 › To mitigate these artifacts, we linearly compensate the intensity in the other direction.

first direction – most of the variability nonlinear model
THE OTHER DIRECTION

› Typically, the block is not completely constant in the other direction -> block artifacts.

› To mitigate these artifacts, we linearly compensate the intensity in the other direction.

second direction – still some variability

linear model

first direction – most of the variability

nonlinear model
THE OTHER DIRECTION

- Typically, the block is not completely constant in the other direction -> block artifacts.
- To mitigate these artifacts, we linearly compensate the intensity in the other direction.
- We calculate the signed difference $d_2$ from the dominant direction, and add $d_2 \gamma$ to each color component.
The Other Direction

- Typically, the block is not completely constant in the other direction -> block artifacts.
- To mitigate these artifacts, we linearly compensate the intensity in the other direction.
- We calculate the signed difference $d_2$ from the dominant direction, and add $d_2 \gamma$ to each color component.

\[ \text{add } -0.5\gamma(1,1,1) \text{ to color in this pixel} \]
THE OTHER DIRECTION

- Typically, the block is not completely constant in the other direction -> block artifacts.
- To mitigate these artifacts, we linearly compensate the intensity in the other direction.
- We calculate the signed difference $d_2$ from the dominant direction, and add $d_2 \gamma$ to each color component.

![Diagram showing linear compensation](image)

add $0.7 \gamma (1,1,1)$ to color in this pixel
Sometimes, the profile function $3x^2 - 2x^3$ is not ideal. We therefore have two other profile functions.

- asymmetric single: different widths on each side
- asymmetric double: middle color, different widths
FURTHER REFINEMENT

› Some blocks have significant variability in both directions
› The fourth function “corner” can sometimes help – two functions multiplied by each other.
FALLBACK

› Some blocks are far too irregular to be represented using smooth functions. Noisy blocks, high-frequency blocks.
› Use a subset of ETC2 as a fallback.
## BIT DISTRIBUTION

<table>
<thead>
<tr>
<th>Bit Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>colorA RGB666</td>
<td>18</td>
</tr>
<tr>
<td>colorB RGB666</td>
<td>18</td>
</tr>
<tr>
<td>width</td>
<td>5</td>
</tr>
<tr>
<td>rotation</td>
<td>7</td>
</tr>
<tr>
<td>translation</td>
<td>7</td>
</tr>
<tr>
<td>second direction tilt ($\gamma$)</td>
<td>6</td>
</tr>
<tr>
<td>function selector</td>
<td>2</td>
</tr>
<tr>
<td>ETC2 fallback</td>
<td>1</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>64</strong></td>
</tr>
</tbody>
</table>
RESULTS

› We have tested our algorithm against DXT1 and ETC2
› Same bitrate, quality measured using Peak Signal to Noise Ratio (PSNR)

› Four different data sets:
  - Radiosity Lightmaps from the game “Mirror’s Edge” by DICE/EA
  - Radiosity Lightmaps from the game “Medal of Honor” by DICE/EA
  - 64 “regular textures”, both game textures and photos
  - 24 “Kodak images”, photos only
RESULTS MIRROR’S EDGE

› Proposed scheme 41.2 dB
  - ETC2: 40.5 dB (-0.7 dB)
  - DXT1: 38.4 dB (-2.8 dB)

› Greatest improvement on large resolution mipmaps
MEDAL OF HONOR TEST SET

- Proposed 37.06 dB
  - ETC2 37.01 dB (-0.05 dB)
  - DXT1 34.15 dB (-2.91 dB)
- Test set contained many small lightmaps of size 16x16 pixels and less
- Excluding these increased the advantage
  - ETC2 -0.53 dB
  - DXT1 -3.05 dB

Image courtesy of DICE/EA
REGULAR TEXTURES

› Works for radiosity lightmaps – what about regular textures?
› Works for these too!
› 64 images (photos and game textures)
› Proposed scheme 34.38 dB
  – ETC2 34.04 dB (-0.34 dB)
  – DXT1 33.13 dB (-1.25 dB)
KODAK IMAGE DATABASE

› Contains only photos
› Publicly available
› Proposed scheme 37.67 dB
  – ETC2 37.44 dB (-0.23 dB)
  – DXT1 36.02 dB (-1.65 dB)
SOME EXAMPLES

› Radiosity lightmap
SOME EXAMPLES

› Radiosity light map

original

proposed

DXT1

ETC2
SOME EXAMPLES

› Natural image example
HARDWARE COMPLEXITY

› We have made a rough estimate of the complexity of the decoder for the proposed system
› Roughly 5 times the size of DXT1 hardware
› Roughly 4 times the size of ETC2 hardware
SUMMARY

› We have created a texture compression system targeted for slowly varying textures
› Main idea is to describe block in only one direction using profile functions
› Works well on radiosity lightmaps – up to 0.7 dB better than ETC2
› Bonus: Also works on regular textures – up to 0.34 dB better than ETC2
ACKNOWLEDGEMENTS

› Thanks to Henrik Halén at EA/DICE for providing us with light map textures from recent games.

› Thanks to Jason Mitchell at Valve for images.