A Real-Time FPGA-Based Architecture for a Reinhard-Like Tone Mapping Operator

Firas Hassan and Joan Carletta
The University of Akron
Outline of Presentation

• Background and goals
• Existing methods for local tone mapping
• Real-time variation on the Reinhard operator
• Experiments and results
• Future work
Outline of Presentation

- Background and goals
Luminance and dynamic range

- Luminance correspond to pixel intensity
- Different devices are sensitive to different ranges of luminance:
  - Human visual system: 14 log units or 48 bits
  - Imaging sensors: 9.5 log units or 32 bits
  - Conventional displays: 2.5 log units or 8 bits
- Mismatch in dynamic ranges makes it:
  - Hard to capture scenes as human perceive them
  - Even harder to display these scenes!
Tone mapping operators

• are used to bridge the mismatch between HDR images and display devices
• compress the dynamic range of HDR images to displayable range
• reproduce as much as possible of the visual sensation of the scene
Global tone mapping operators

- are independent of local spatial context
- perform same operation on each pixel
- do not work well when illumination varies locally
Local tone mapping operators

- vary adaptively with the local characteristics of the image
- produce higher quality tone mapped images than global TMOs
- can require complex computation
- can suffer from halo artifacts
Goals of research

- Develop algorithms for local tone mapping of gray scale HDR images such that:
  - they can be shown with clear detail on standard displays
  - processing is “real-time” (60 frames/second for standard LCD monitors)
  - processing can be easily embedded (using field programmable gate arrays)
- The system is the result of a careful trade-off of both image processing and hardware performance aspects
Outline of Presentation

• Existing methods for local tone mapping
Basic structure of local TMOs

L: luminance
I: illumination, related to lighting conditions
R: reflectance, related to object in scene
Retinex method (Jobson et al.)

- uses Gaussian surrounds centered on a pixel to estimate local illumination
  - single-scale: use one fixed-size surround (get halo artifacts)
  - multi-scale: use mean of three differently sized surrounds
- normalizes each pixel by its local illumination
- published 2004 implementation is not real-time:
  - single-scale Retinex
  - 256×256 grayscale image
  - 20 frames/sec on a digital signal processor
Reinhard method

- uses the best illumination estimate around the pixel from a Gaussian pyramid of the image
- eliminates halo artifacts better than Retinex
- published 2005 implementation is not real-time:
  - uses four-scale Gaussian pyramid
  - $1024 \times 768$ color image
  - 14 frames per second on graphics card
  - bottleneck was memory bandwidth
Reinhard method

Reinhard’s selection of best window for illumination estimate

- too small a surround gives poor estimate (contrast loss results)
- too large a surround may encompass light source (halos result)
- start with smallest surround
- consider next largest surround; if its average is not much different than the smaller surround’s, use it instead
- result: use biggest surround that doesn’t contain a big change in illumination
Other local TMOs

All are able to get rid of halo artifacts but too complex to be real-time!

- Iterative methods
  - Low curvature image simplifier
  - Gradient domain HDR compression
- Nonlinear filters
  - Bilateral filters
  - Trilateral filters
- Image appearance models
  - Pattanaik
  - iCAM
Outline of Presentation

• Real-time variation on the Reinhard operator
Block diagram of our method
Approximating a Gaussian surround

rising geometric series  falling geometric series
Implementing the window with accumulators

rising geometric series:

\[ a_1[i] = \frac{a_1[i-1]}{2} + 64P[i-7] - \frac{1}{2} P[i-14] \]

falling geometric series:

\[ a_2[i] = 2a_2[i-1] + P[i] - 128P[i-7] \]

total window

\[ P_{ave}[i-7] = \frac{1}{2} \left( \frac{a_1[i]}{128} + \frac{a_2[i]}{128} \right) \]

\( P[i] \): incoming pixel on right hand side of window
Four-scale approximate Gaussian pyramid

\[ w_{i,j} = 2^{i-8} \times 2^{j-8} \]

\[ w_{i,j} = 2^{i-7} \times 2^{j-7} \]

\[ w_{i,j} = 2^{[i/2]}-7 \times 2^{[j/3]}-7 \]

\[ w_{i,j} = 2^{[i/4]}-7 \times 2^{[j/4]}-7 \]
Hardware for the 56x56 pixel window

Horizontal Computation Block (HCB)  Vertical Computation Block (VCB)

Complete Hardware
Memory organization

- Delay block and stack
- Delay block

Diagram showing the flow of data through various scales and rows.
Log average
Normalize the pixel

Fast hardware for the reciprocal

• avoid division – it’s expensive and slow!
• borrow from the Newton-Raphson algorithm, iteratively finds roots of a function
• root of $f(x) = \frac{1}{X} - b = 0$ is reciprocal of $b$
• algorithm says here; this means

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

$$X_{n+1} = X_n(2 - bX_n)$$

• look up initial guess based on 8 bits of mantissa of $b$; one iteration then gives 17 bits of reciprocal
Outline of Presentation

• Experiments and Results
Simulation results
Simulation results
Simulation results
### Hardware synthesis results

<table>
<thead>
<tr>
<th>Input image</th>
<th>1024×768 pixels, 28 bits per pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>Altera Stratix II EP2S90F1020C3 FPGA</td>
</tr>
<tr>
<td>Total bits of memory</td>
<td>2,952,960 / 4,520,448</td>
</tr>
<tr>
<td>Total logic cells</td>
<td>17,553 / 72,768</td>
</tr>
<tr>
<td>Max operating frequency</td>
<td>83.83 MHz</td>
</tr>
</tbody>
</table>

- Compatible with a frame rate of 60 frames/sec (for not quite a full-sized LCD screen)
- Truly a real-time implementation
PSNR Study

- Our gold standard was a floating-point version of Reinhard operator.
- Using constant weights to construct the Gaussian pyramid we get PSNR which are on average 3dB lower.

<table>
<thead>
<tr>
<th></th>
<th>Const weight</th>
<th>Our method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memorial</td>
<td>30.5</td>
<td>34.9</td>
</tr>
<tr>
<td>Rosette</td>
<td>25.1</td>
<td>28.5</td>
</tr>
<tr>
<td>groveC</td>
<td>29.4</td>
<td>33.5</td>
</tr>
<tr>
<td>groveD</td>
<td>29.8</td>
<td>33.6</td>
</tr>
<tr>
<td>vinesunset</td>
<td>41.4</td>
<td>42.6</td>
</tr>
</tbody>
</table>
Outline of Presentation

• Future work
Towards a nine-level embedded real-time Reinhard operator

• Using more scales allows for better contrast, but geometric series based on powers-of-two are no longer enough

• To use more general bases, must consider:
  • the relation between the base and the size of the window
  • the accuracy of calculation, which relates to the size of the accumulator used to calculate the rising and falling geometric series
Towards tone mapping of color HDR images

- color should be an easy extension
- extract luminance from RGB triplet
  \[ L = 0.27 \times R + 0.67 \times G + 0.06 \times B \]
- tone-map the luminance
- use the mapped luminance to transform the RGB
- preliminary simulations are promising
Thank You

Questions?