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

gh2007

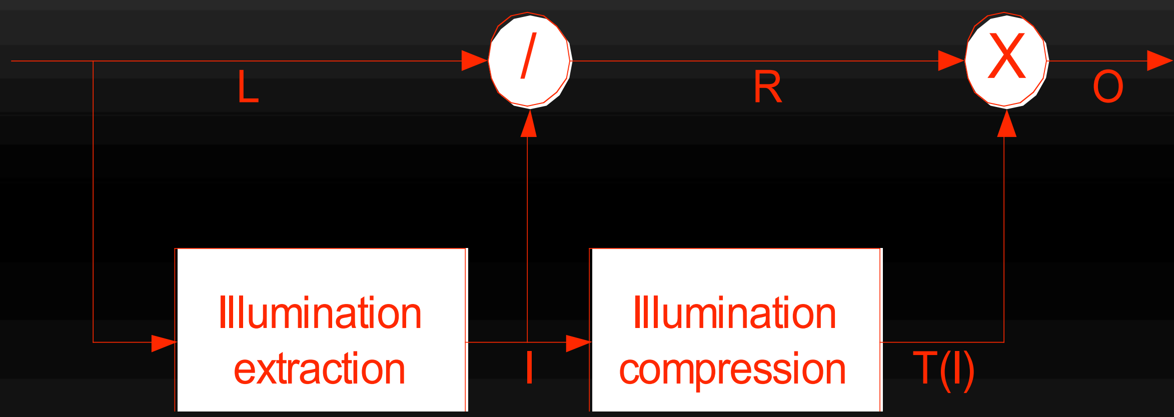
Goals of research

- 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
  - 1024×768 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

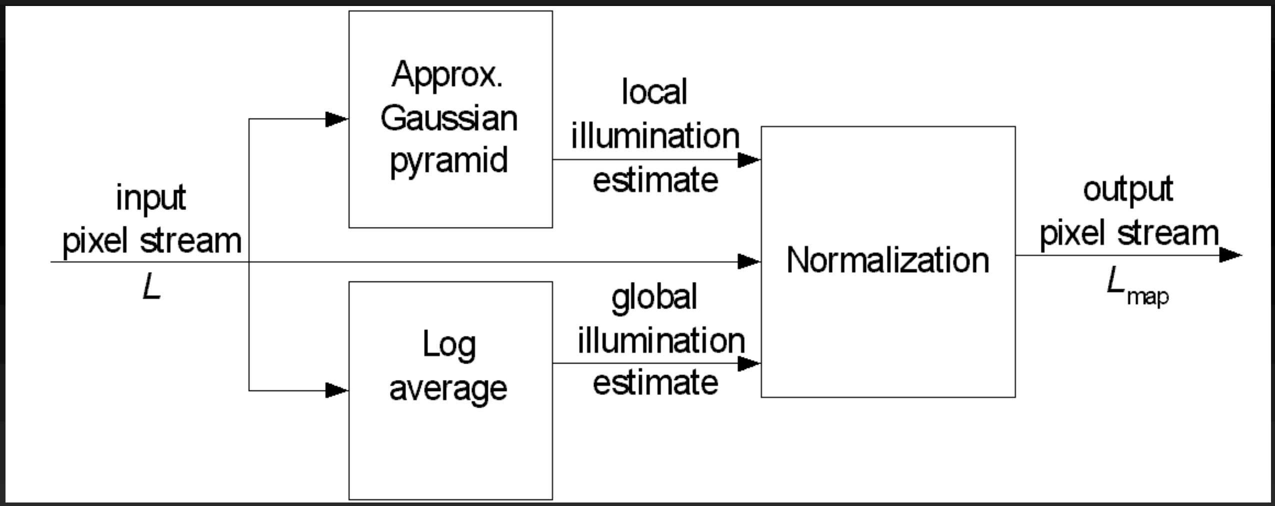
All are able to get rid of halo artifacts but too complex to be real-time!

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

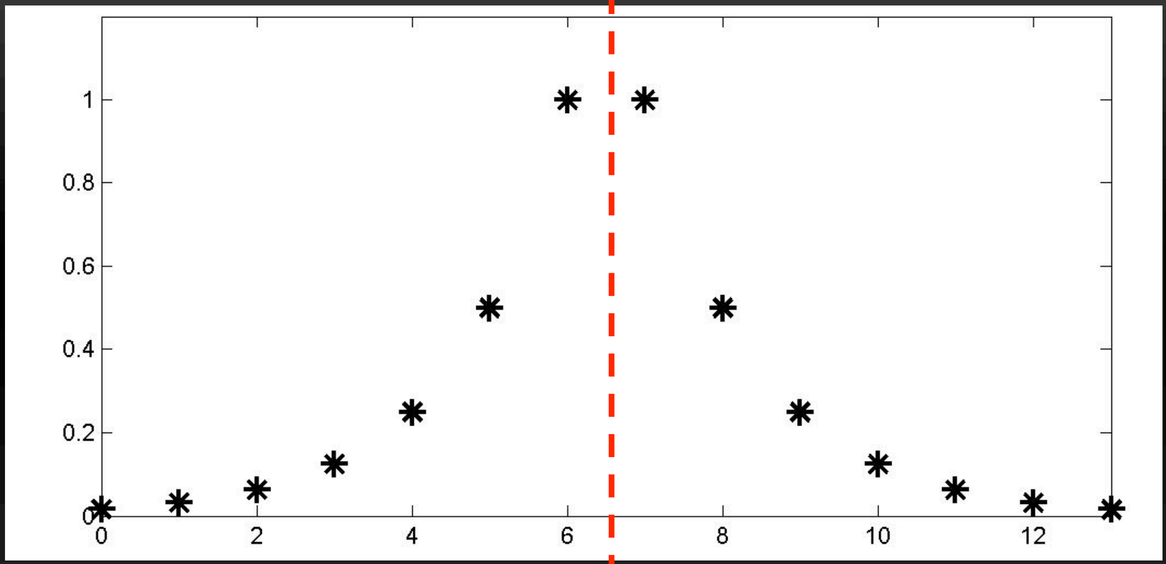
Outline of Presentation

- Real-time variation on the Reinhard operator

Block diagram of our method



Approximating a Gaussian surroud



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:

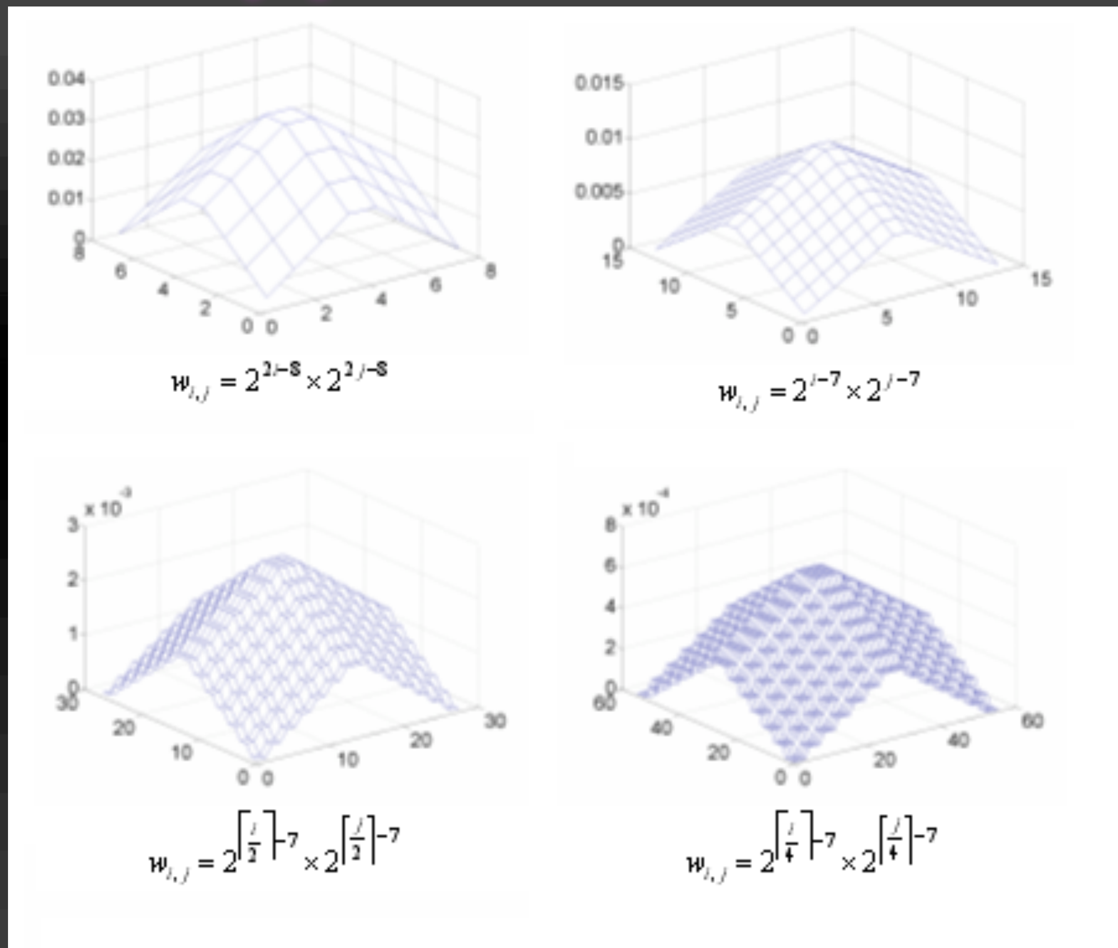
$$\mathrm{a}_2[i] = 2a_2[i-1] + P[i] - 128P[i-7]$$

total window

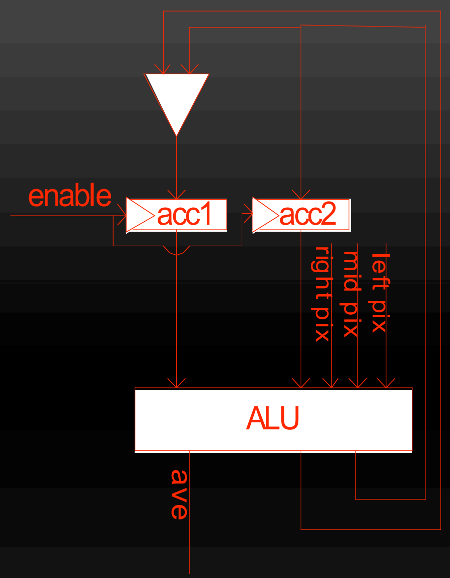
$$P_{ave}[i-7] = \frac{1}{2}\left(\frac{a_1[i]}{128} + \frac{a_2[i]}{128}\right)$$

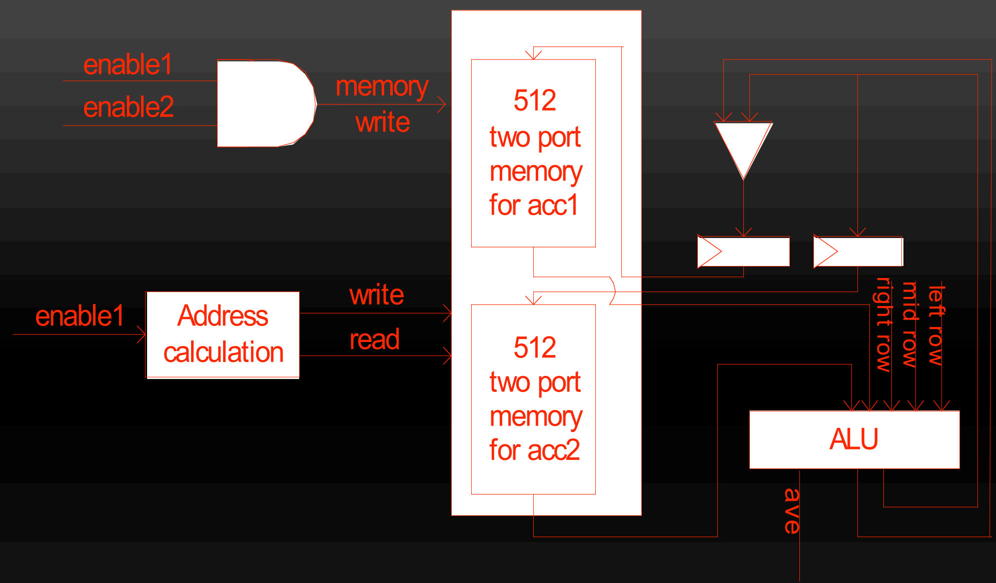
$P[i]$: incoming pixel on right hand side of window

Four-scale approximate Gaussian pyramid

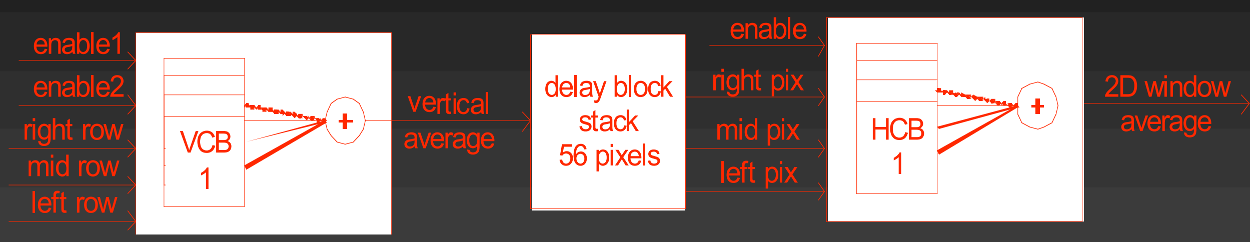


Hardware for the 56x56 pixel window



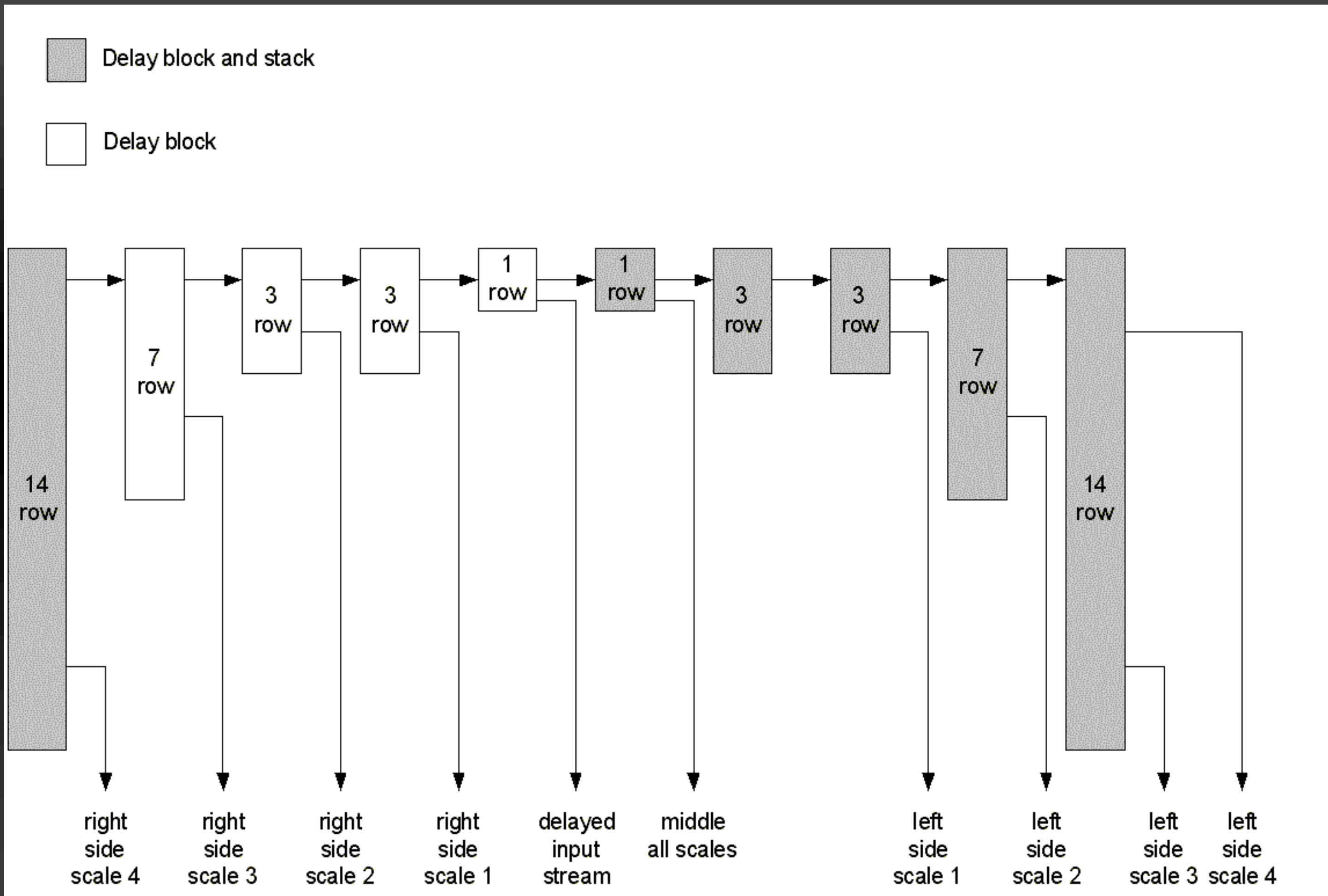


Horizontal Computation Block (HCB) Vertical Computation Block (VCB)

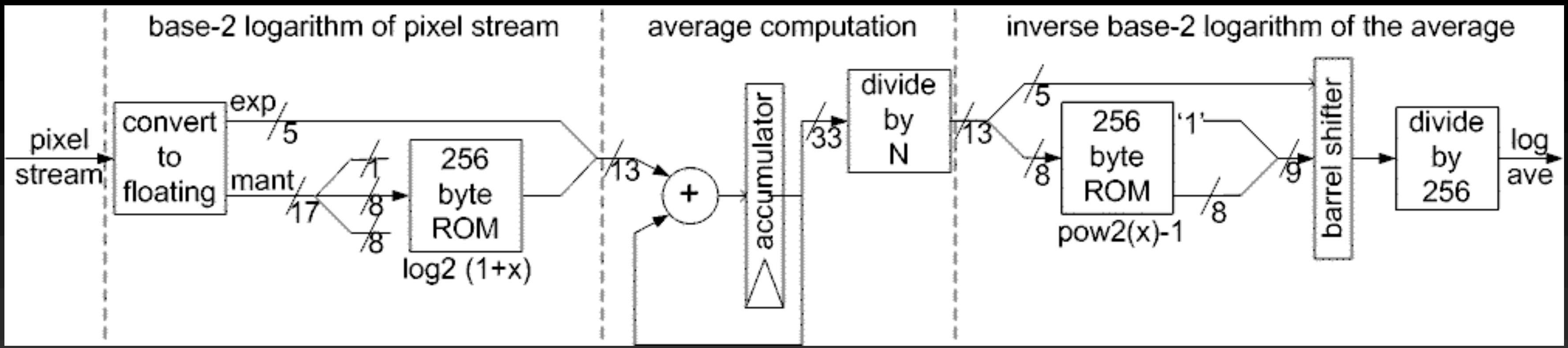


Complete Hardware

Memory organization



Log average


base-2 logarithm of pixel stream
average computation
inverse base-2 logarithm of the average
pixel stream
convert to floating
exp
5
mant
17
1
8
8
256 byte ROM
log2 (1+x)
13
+
accumulator
33
divide by N
13
5
8
256 byte ROM
pow2(x)-1
'1'
8
9
barrel shifter
divide by 256
log ave

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 $$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$ here; this means $$X_{n+1} = X_n(2 - bX_n)$$
- 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

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

- 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

	Const weight	Our method
Memorial	30.5	34.9
Rosette	25.1	28.5
groveC	29.4	33.5
groveD	29.8	33.6
vinesunset	41.4	42.6

Outline of Presentation

- Future work

Towards a nine-level embedded real-time Reinhard operator

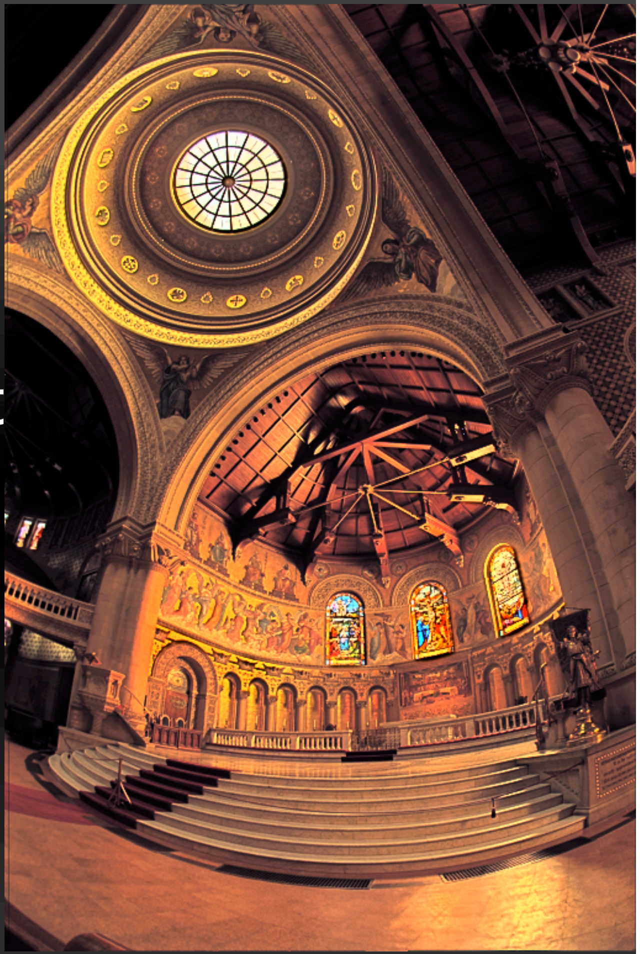
- Using more scales allows for better contrast, but geometric series based on powers-of-two are no longer enough
- To use more general bases, must consider:
  - the relation between the base and the size of the window
  - the accuracy of calculation, which relates to the size of the accumulator used 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 ?