Image processing on GPUs

Rahul Narain

COMP790-058: GPGP

March 7, 2007

Image processing

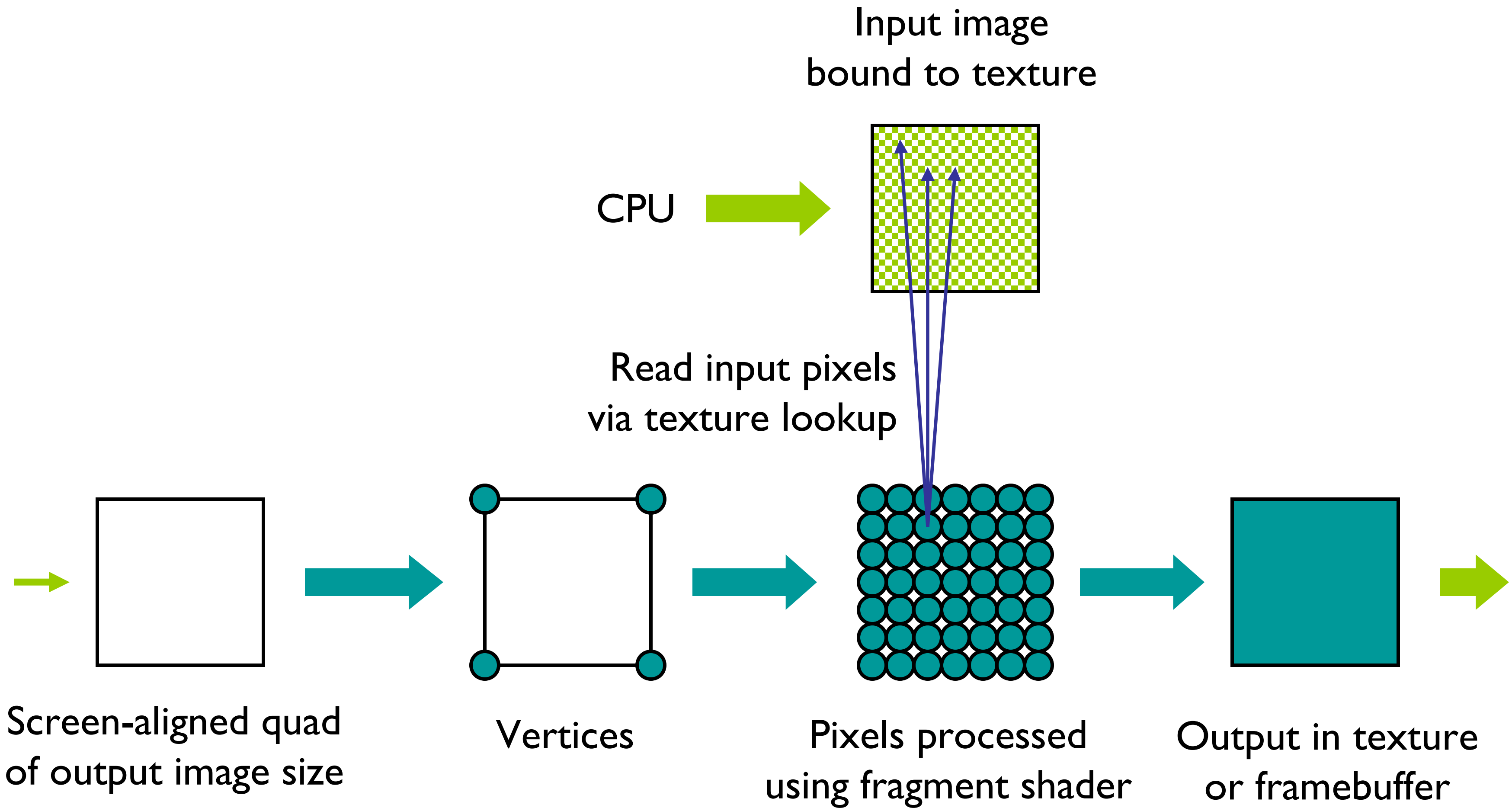
- Image = 2D array of color values (1D or 3D)
- Most image processing algorithms are inherently parallel
  - Do "the same thing" for every pixel
- Memory intensive with coherent lookups

Image processing

Image processing maps well to GPUs

2D image	2D texture
Per-pixel operations	Fragment program
Memory intensive	Fast texture lookup
Accuracy is not critical	Good!

Image processing on GPUs

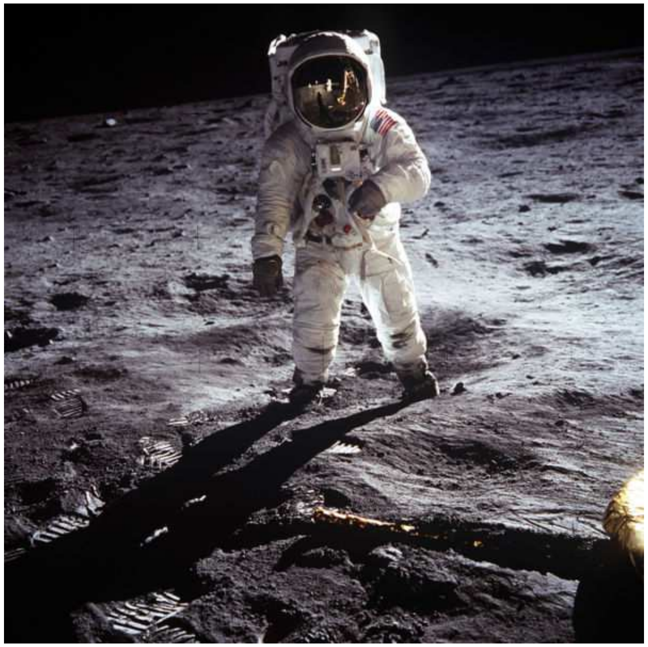


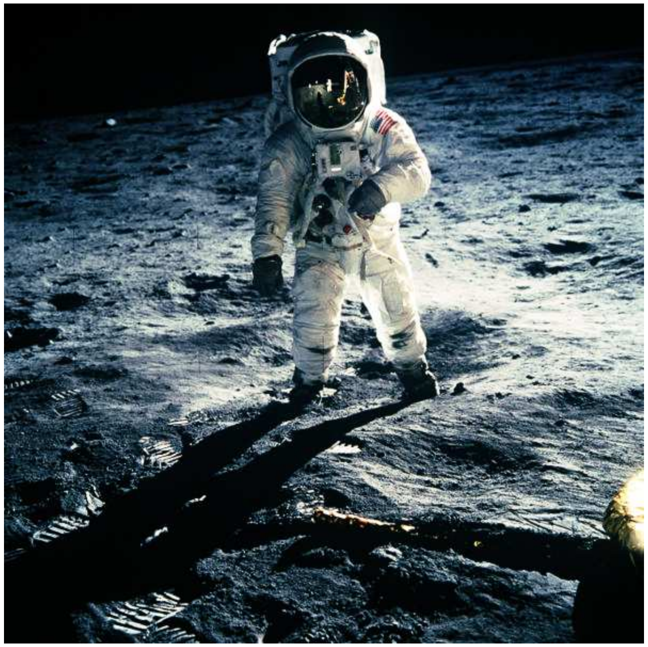
Topics

- Color correction
- Convolution
- Wavelet transforms
- Anisotropic diffusion and depth of field
- HDR and tone mapping

Color correction

- Brightness/contrast, hue/saturation, gamma, thresholding, Levels and Curves, …





Color correction

- Process each pixel independently

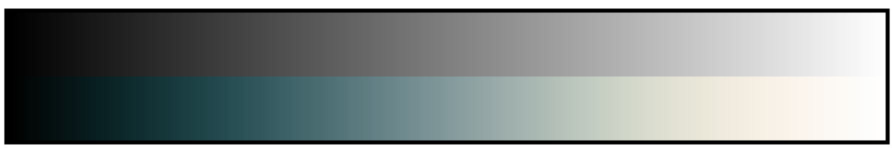
$$t : \mathbb{R}^3 \longrightarrow \mathbb{R}^3$$

- Usually process each channel independently

$$t_R, t_G, t_B : \mathbb{R} \longrightarrow \mathbb{R}$$

- Pass three lookup tables as a 1D RGB texture

$$g_R[x,y] = t_R[f_R[x,y]]$$



Convolution

$$g[x,y] = \sum f[x+i,y+j]\, h[i,j]$$

- Pass kernel h and sampling coordinates $[i,j]$ as uniform data arrays
- Requires N or N^2 texture lookups per pixel
  - Used to be a problem on old graphics cards
  - `EXT_convolution` is only supported by SGI

Convolution

Convolution with limited texture lookups:

1. Clear output buffer
2. For each pass:
    1. In vertex program, generate k texture coordinates corresponding to adjacent pixels
    2. In fragment program, compute partial sum of k terms and add to output buffer

Requires N/k passes

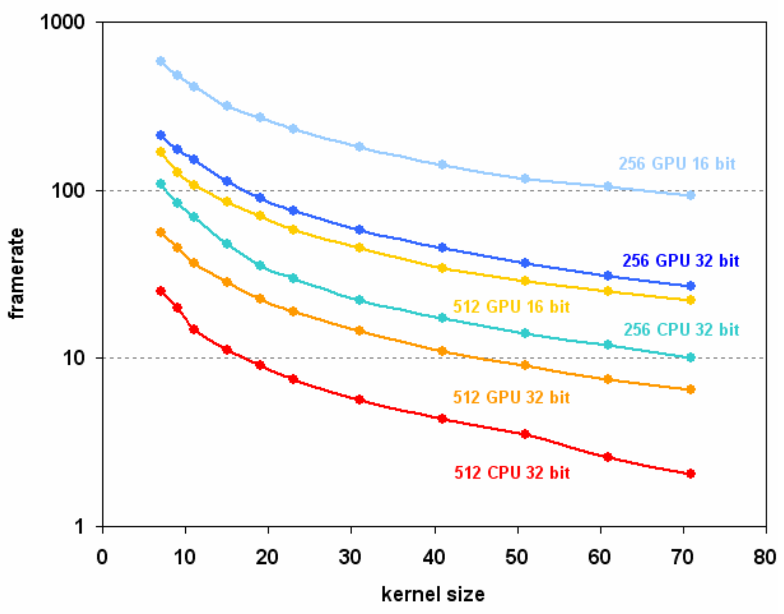
Convolution

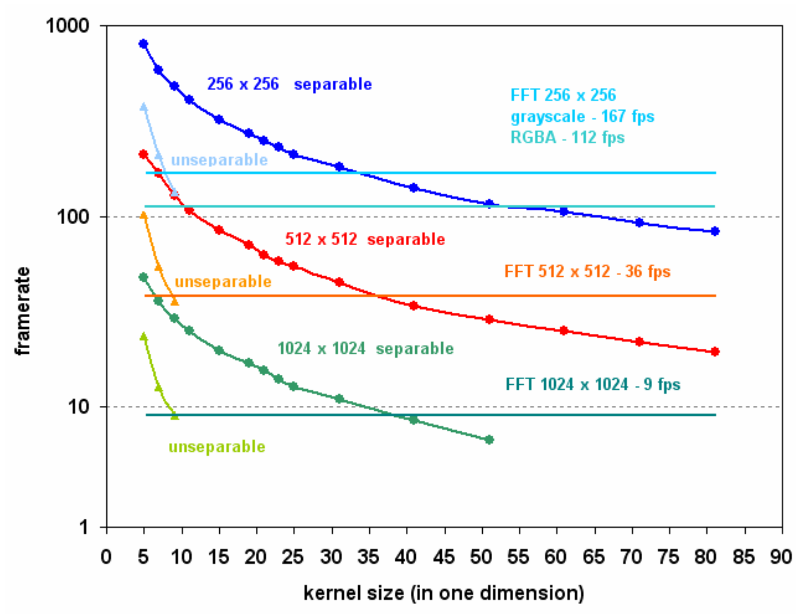
- Now only limited by fragment program instruction length
- All texture lookups access nearby pixels

  Very fast due to cache coherence

Convolution

- Fialka and Čadík: NVIDIA GeForce 6600
- GPU outperforms CPU in all cases





Convolution

- 3D convolution for volume data
- Current GPUs don't allow high-precision 3D textures

    Load slices into several 2D textures instead

- Multiple passes to loop over slices
- Only 16 textures can be bound at a time

    Use multi-pass algorithm if kernel is wider in z

Non-linear filtering

- Median filter

$$g[x,y] = \text{median}\,\{\, f[x+i,y+j] \,\}$$

- Can be done naïvely for smallish filter sizes

  Known fast algorithms are not parallelizable

- Even then, naïve GPU is faster than fast CPU
- Viola et al: 1.17× speedup on 5×5×5 volume filter using NVIDIA GeForce FX 5800

Non-linear filtering

- Bilateral filter

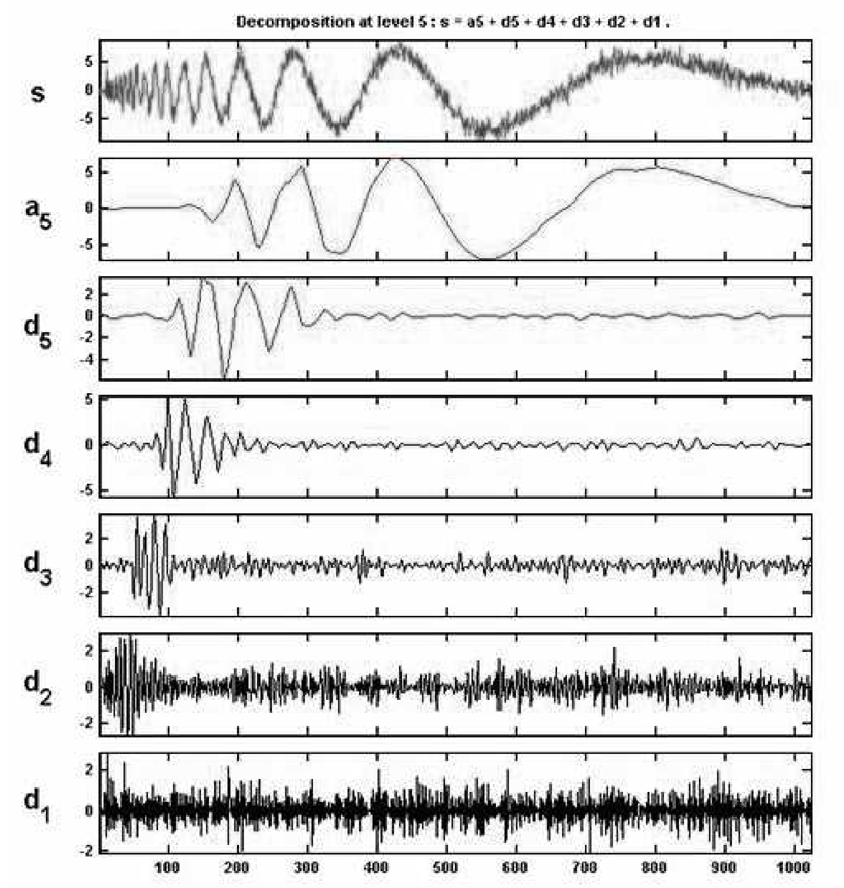
$$g[x] = k^{-1} \sum f[x'] \, h_s[x'-x] \, h_r[f[x']-f[x]]$$

$$k = \sum h_s[x'-x] \, h_r[f[x']-f[x]]$$

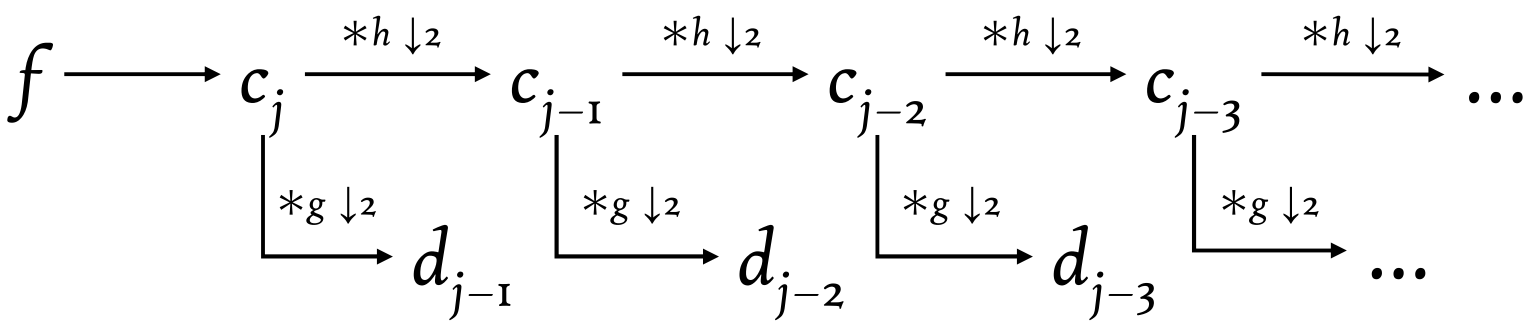
- Naïve approach: 1.52× speedup [Viola et al]
- Paris and Durand's fast approximation [2006] should be parallelizable on GPU

Wavelet transforms

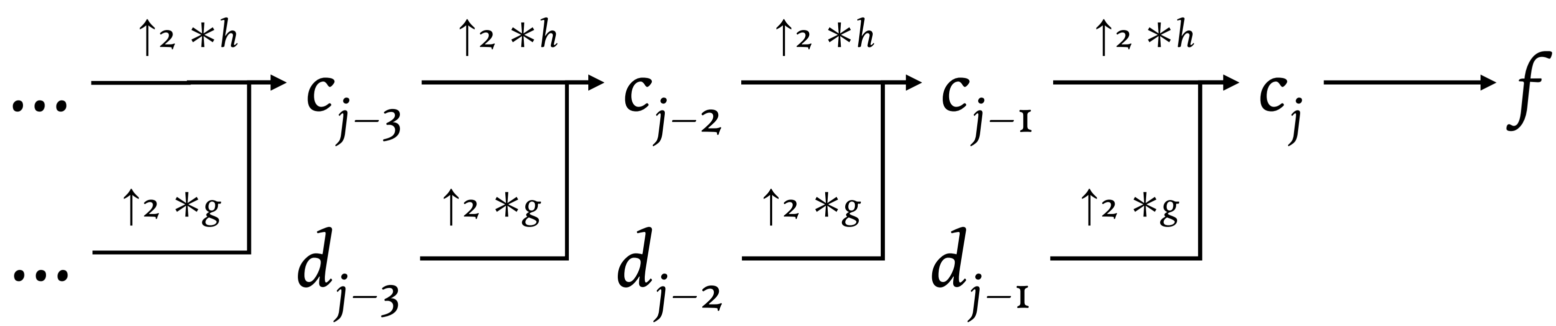
- Multi-resolution decomposition of a signal
- Basis functions are localized in both position and frequency



Wavelet transforms



Decomposition



Reconstruction

Wavelet transforms

- All wavelet coefficients stored in a texture
  - Two for ping-pong
- Each pass reads/writes a subset of the texture
- Convolutions are separable

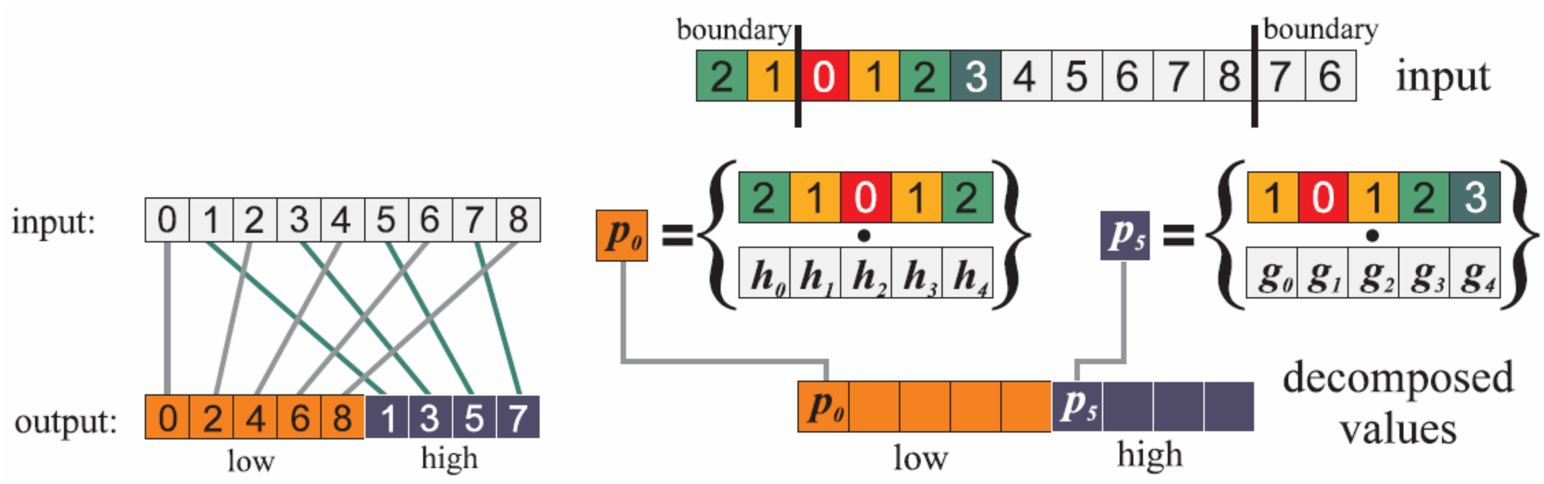
Wavelet transforms

- Forward DWT:

$$c_{j-1}[n] = \sum h[k]\, c_j[2n-k], \quad d_{j-1}[n] = \sum g[k]\, c_j[2n+1-k]$$

$$\mathbf{z}_{j-1} = [\mathbf{c}_{j-1}\ \mathbf{d}_{j-1}]$$

- Boundary extension using indirection texture



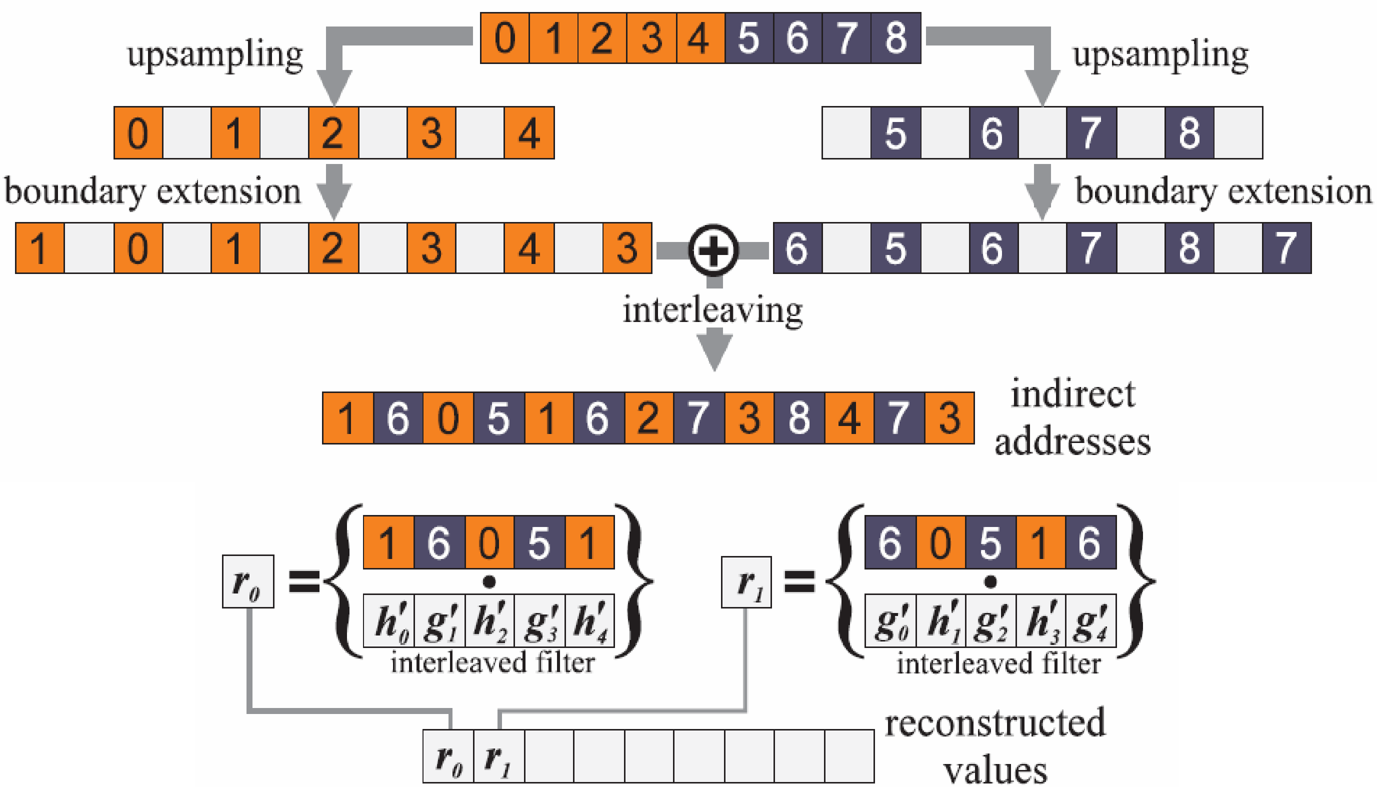
Wavelet transforms

- Inverse DWT:

  $c_j[n] = \sum h[k]\, c'_{j-1}[(n-k)/2] + \sum g[k]\, d'_{j-1}[(n-k)/2]$

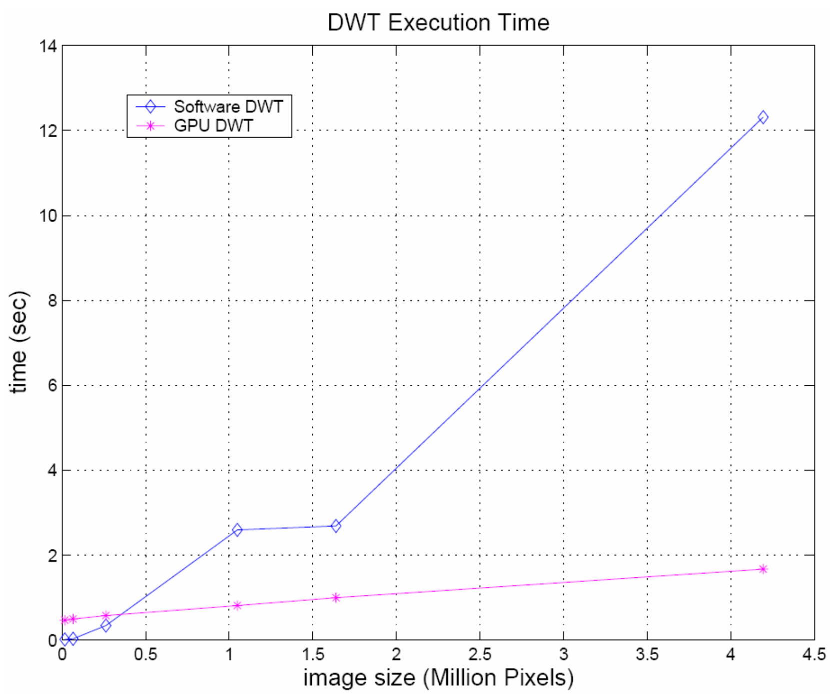
- Two cases depending on whether n is even
- Avoid conditionals using precomputed indirection texture

Wavelet transforms



Wavelet transforms

- Wong et al: NVIDIA GeForce 7800 GTX
- Performance gain over CPU for large images



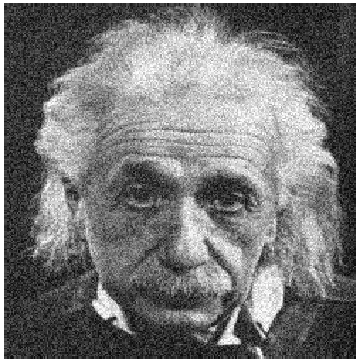
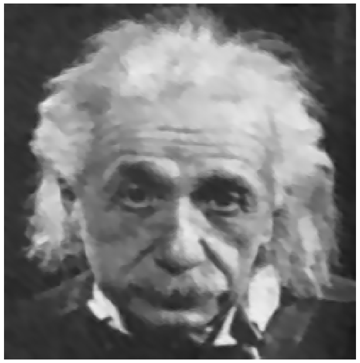
Diffusion

- Diffuse intensities over image at varying rates
- Anisotropic diffusion

  low diffusion at edges

- Depth of field

  radius of confusion

Diffusion

$$u' = \nabla \cdot (g \, \nabla u)$$

- Discretize differential equation over pixel grid
    - Finite differences in space
    - Implicit 1st-order Euler in time
- Solve linear system of equations per iteration

$$\mathrm{A}^k(\mathbf{u}^k)\, \mathbf{u}^{k+1} = \mathbf{r}^k(\mathbf{u}^k)$$

Diffusion

- A is sparse, banded with known structure
- Don't want to represent whole matrix in memory
- Structure of A allows simplification

Diffusion

Rumpf and Strzodka [2001]:

- Use Jacobi or conjugate gradient iterations

  e.g. $\mathbf{x}^{i+1} = F(\mathbf{x}^i) = \mathrm{D}^{-1}(\mathbf{r} - (\mathrm{A} - \mathrm{D})\mathbf{x}^i)$
- Corresponds directly to image blending
- Can be implemented directly in OpenGL!
- NVIDIA GeForce 3: 8ms per iteration on 256×256 image

Diffusion

1. Upload original image $\mathbf{u}^0$ to texture
2. For each timestep k:
   1. Initialize r.h.s. $\mathbf{r}^k$ (usually equals $\mathbf{u}^k$)
   2. (If necessary) calculate image of diffusion coefficients $\mathbf{g}^k$ using lookup table
   3. Initialize $\mathbf{x}^0 = \mathbf{r}^k$
   4. For each iteration i:

      Calculate $\mathbf{x}^{i+1} = F(\mathbf{x}^i)$ using image blending
   5. Store the solution $\mathbf{u}^{k+1} = \mathbf{x}^{i+1}$

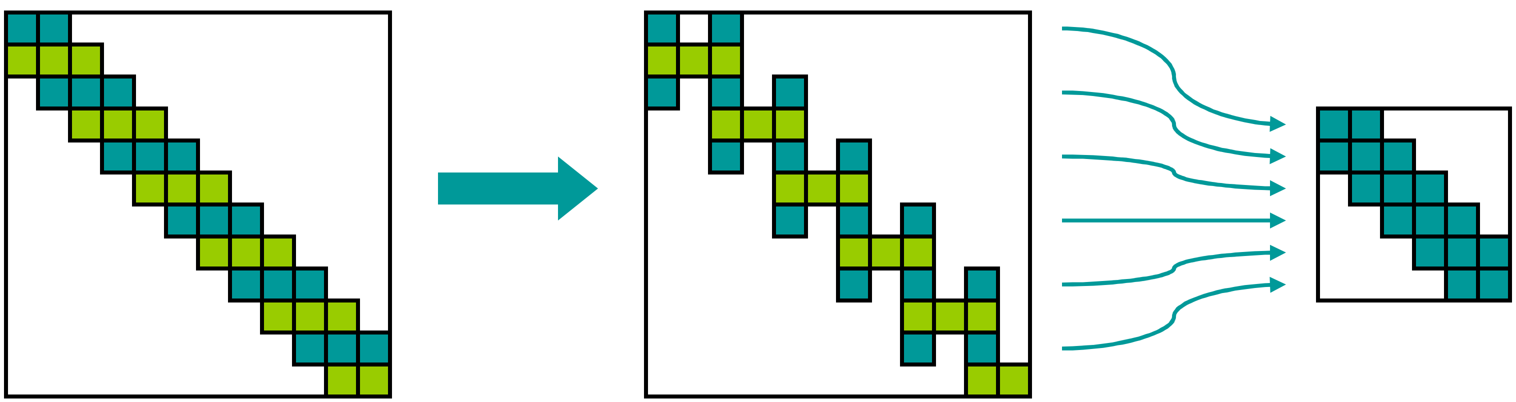
Diffusion

Kass et al [2005]:

- Approximate by two 1D diffusions instead
- n linear systems for n rows, tridiagonal A's
- Represent A's using 3 channels of each row of 2D texture
- Solve in parallel using *cyclic reduction*
- NVIDIA GeForce 7800: 0.15s for 1024×1024

Diffusion

1. Gaussian elimination on odd rows in parallel
2. Copy smaller system of even rows to new texture; solve recursively
3. Propagate solution to odd rows

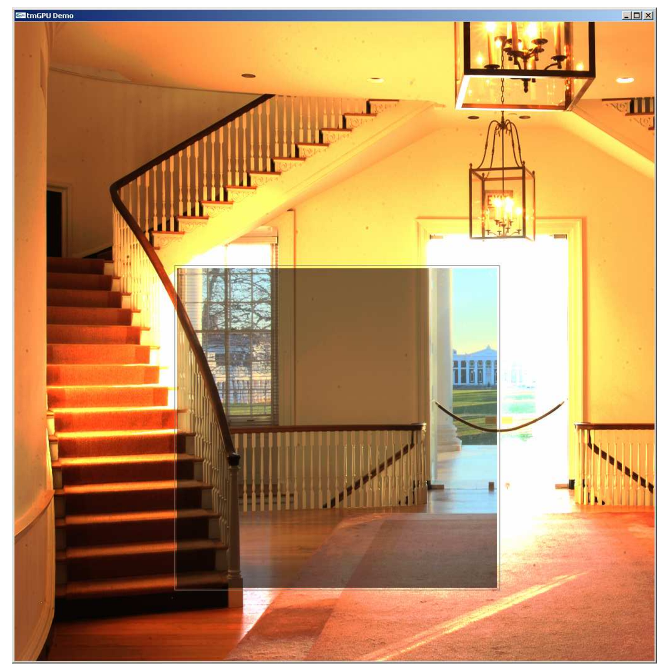


HDR

- OpenEXR: `half` datatype = 16-bit floating point
- Identical to native `half` datatype on GPUs
- Floating-point textures allow HDR

Tone mapping

- Displaying HDR images on LDR devices
- Reduce the dynamic range of an HDR image while “looking the same”
- Several techniques
- Reinhard et al.’s method has been implemented in real-time on the GPU



Tone mapping

- Compute log average luminance
- Rescale pixel luminances by average
- Find local average luminance of each pixel
  - Convolve with Gaussian filters of various widths
  - Compare to find best scale for each pixel
- Apply transfer function based on per-pixel local average luminance

Tone mapping

First pass

- Compute log average luminance

    Sum over entire image by repeated reduction

Several passes

- Convolve rescaled image with Gaussian filters of various widths and compare

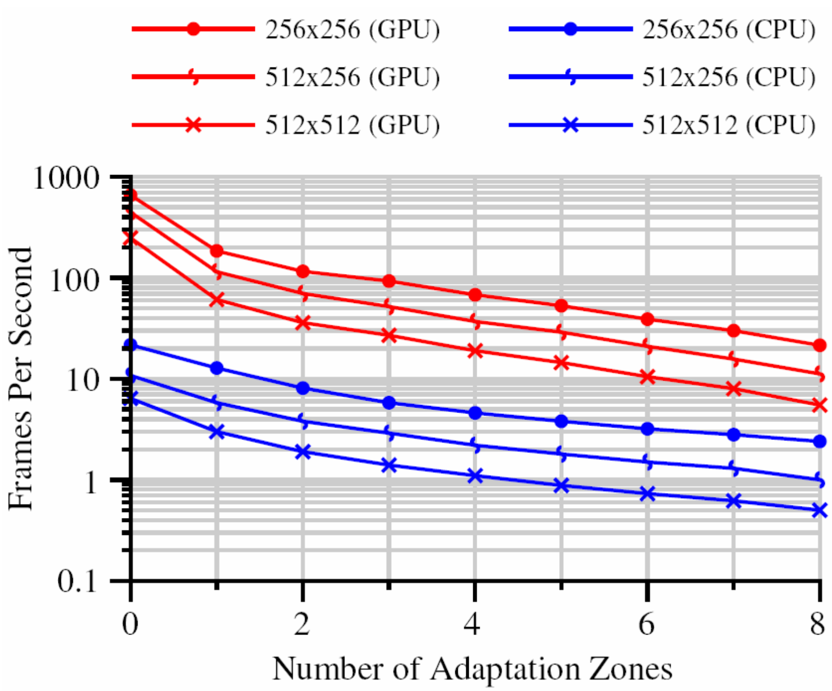
    Accumulate results for “best” scale in texture

Final pass

- Apply transfer function

Tone mapping

- Goodnight et al: ATI Radeon 9800
- GPU is faster that CPU in all cases



Conclusion

- GPUs significantly accelerate image processing
  - Pixel-level parallelism
  - High memory bandwidth
- Previously slow operations now run at interactive rates on GPU

References

- *GPU Gems: Programming Techniques, Tips, and Tricks for Real-Time Graphics*. Edited by Randima Fernando. NVIDIA Corporation, 2004.
- Ondřej Fialka and Martin Čadík. "FFT and Convolution Performance in Image Filtering on GPU." Proceedings of the Conference on Information Visualization, 2006.
- Ivan Viola, Armin Kanitsar, Eduard Gröller. "Hardware-Based Nonlinear Filtering and Segmentation using High-Level Shading Languages." Proceedings of IEEE Visualization 2003.
- Sylvain Paris and Frédo Durand. "A Fast Approximation of the Bilateral Filter using a Signal Processing Approach." European Conference on Computer Vision, 2006.
- Matthias Hopf and Thomas Ertl. "Hardware Accelerated Wavelet Transformations." Proc. EG/IEEE TCVG Symposium on Visualization, 2000.

References

- Tien-Tsin Wong, Chi-Sing Leung, Pheng-Ann Heng, Jianqing Wang. "Discrete Wavelet Transform on Consumer-Level Graphics Hardware." IEEE Transactions on Multimedia, 2005.
- Michael Kass, Aaron Lefohn, John Owens. "Interactive Depth of Field." Pixar Technical Memo #06-01, 2006.
- Martin Rumpf and Robert Strzodka. "Nonlinear Diffusion in Graphics Hardware." Proceedings of EG/IEEE TCVG Symposium on Visualization, 2001.
- Nolan Goodnight, Rui Wang, Cliff Woolley, Greg Humphreys. "Interactive Time-Dependent Tone Mapping Using Programmable Graphics Hardware." Eurographics Symposium on Rendering, 2003.
- Erik Reinhard, Michael Stark, Peter Shirley, Jim Ferwerda. "Photographic Tone Reproduction for Digital Images." ACM Transactions on Graphics, 2002.