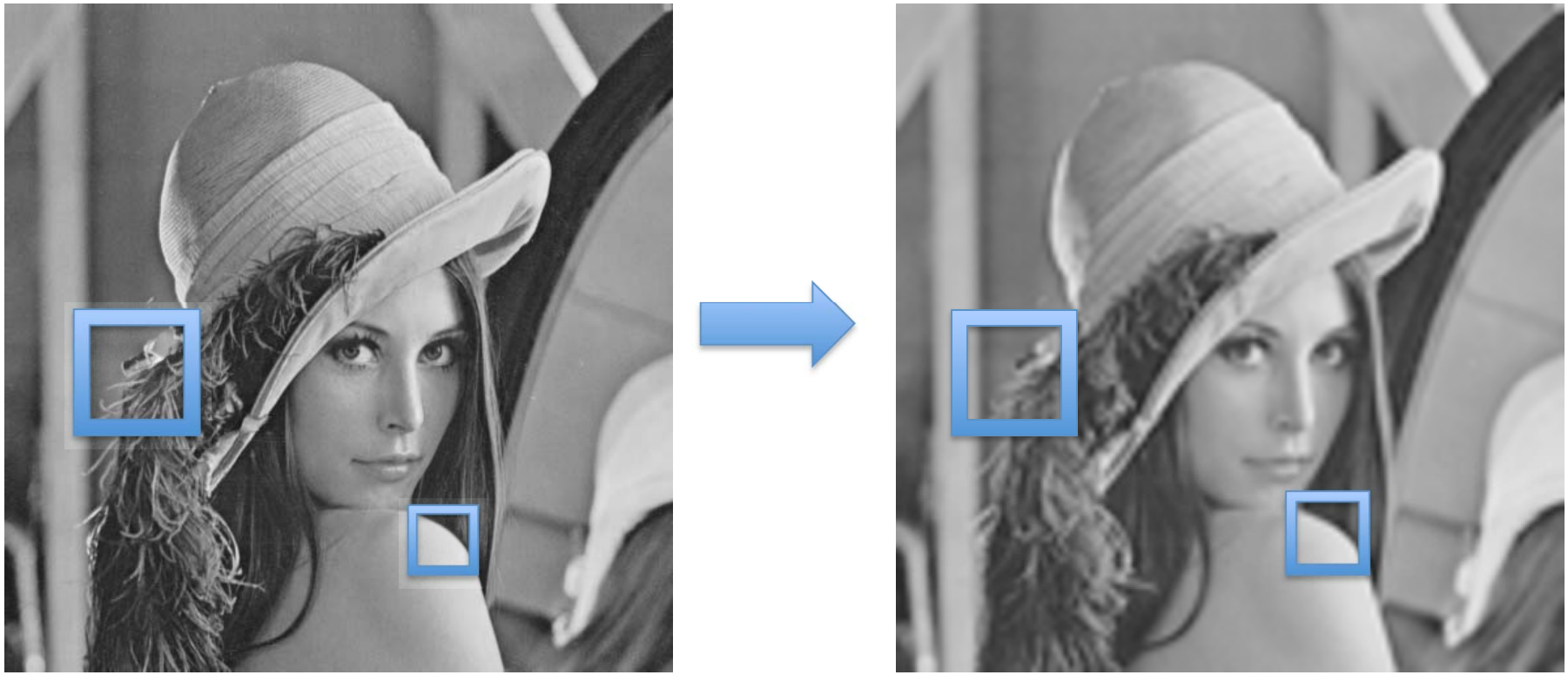
GPU-accelerated fractal imaging

Jeremy Ehrhardt

CS 81 - Spring 2009

Motivation

- Typical reconstruction filters blur edges, detail



Original

Reduced 4x, then enlarged 4x
with Mitchell filter

Motivation

- Fractal encoding
  - A **scale-independent** model of image
  - Find self-similar regions of image
    - Contractive transformations

Motivation

- Decoding fractal representation
  - Any scale
    - Larger than original image
  - Preserves appearance of high-frequency detail
  - Improved visual quality

Motivation

- Basic encoding algorithm
  - Divide image into small **range** blocks
  - Generate **domain** blocks from image somehow
  - For each range block
    - For each domain block
      - Find scale and offset for pixel luminance
      - $R = t(D) = s * D + o$
    - Pick domain with lowest mean squared error

Using the GPU

- Encoding highly parallelizable
  - Range selections don't depend on other ranges
  - Process multiple ranges at a time
  - Process multiple domains for a range at a time
  - Process multiple pixels in range, domain, or product at a time

Using the GPU

- Decoding also parallelizable
  - Short number of iterations (~10)
  - Take transform list from encoding
  - Decide on output size
  - For each pixel in output
    - Source, scale, offset don't change across iterations
    - Value depends only on image at previous iteration
    - Maps naturally to pixel shader

What I Did

- Software encoder and decoder
  - Python, using NumPy fast array extension
- GPU-assisted encoder
  - Most work done in GLSL pixel shaders

- Implementations similar
  - Easy debugging
  - All images were decoded in software

Encoding times

Encoder	Domain size	Range size	Image dimensions	Time (seconds)
Software	8	4	256 x 256	350
OpenGL	8	4	256 x 256	52
Software	4	2	256 x 256	5620
OpenGL	4	2	256 x 256	176

Domain and range sizes are pixels along edge of a square

- **CPU**: Intel Core 2 Duo @ 2.16 GHz
  - 2 GB RAM @ 0.667 GHz
- **GPU**: nVidia GeForce 7600 GT @ 0.56 GHz
  - 256 MB RAM @ 0.7 GHz

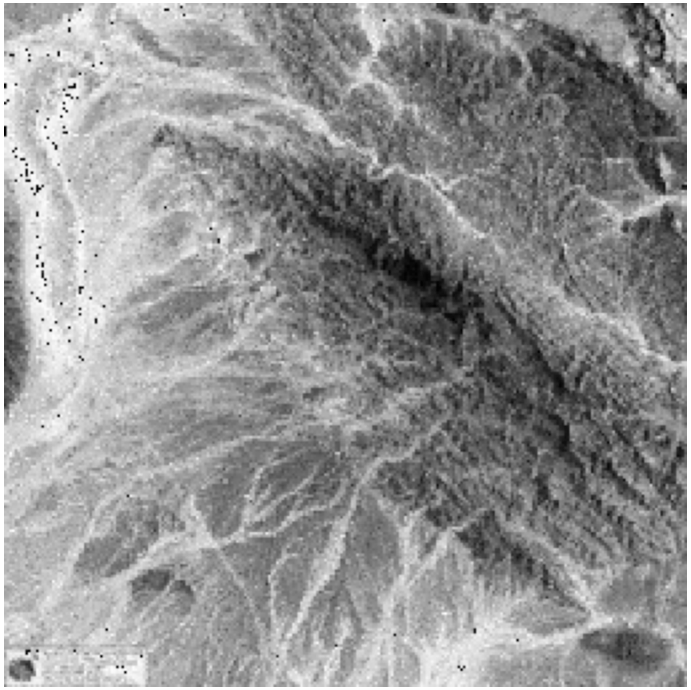
Results (no enlargement)

- Tradeoff between speckles and blocking

Fractal encoder (GPU, **4:2**)

Fractal encoder (GPU, **8:4**)

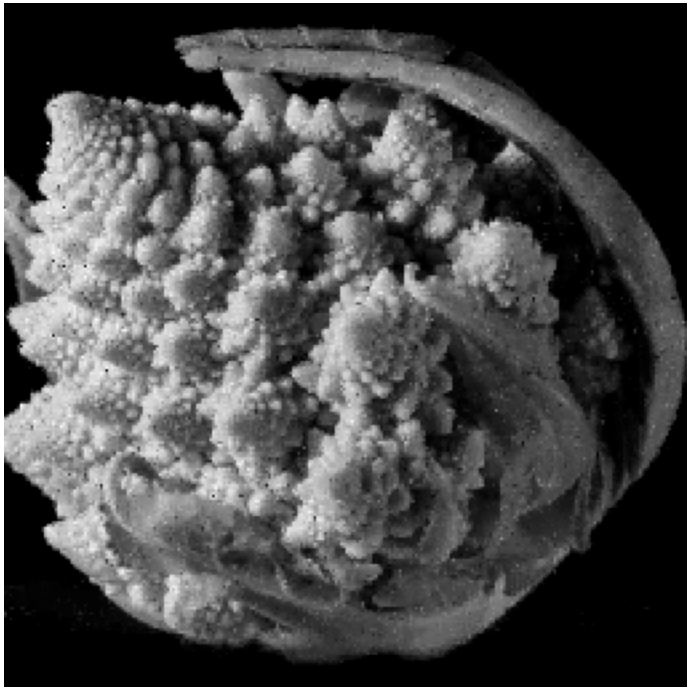
Results (no enlargement)



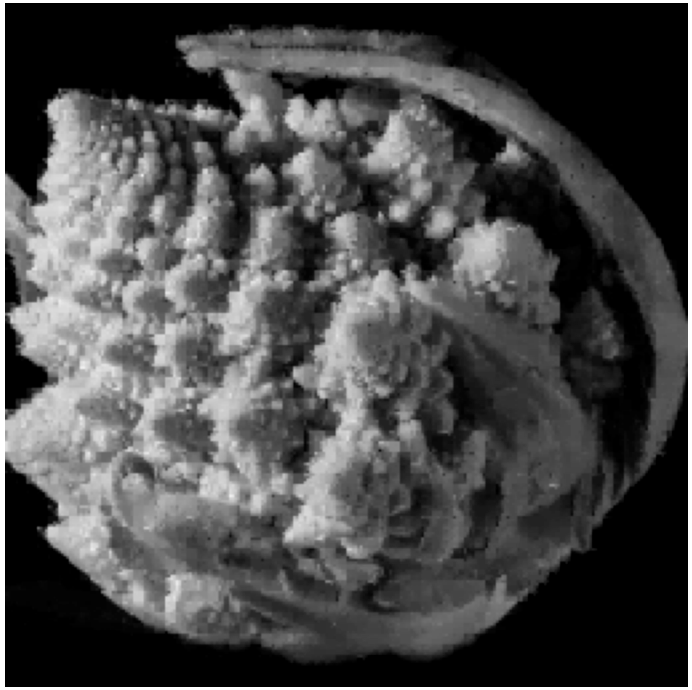
Fractal encoder (GPU, **4:2**)

Fractal encoder (GPU, **8:4**)

Results (no enlargement)



Fractal encoder (GPU, **4:2**)



Fractal encoder (GPU, **8:4**)

Results (2x enlargement)

Reduced 2x, then enlarged 2x
with fractal scaler (GPU, **4:2**)

Reduced 2x, then enlarged 2x
with fractal scaler (GPU, **8:4**)

Results (4x enlargement)

Reduced 4x, then enlarged 4x
with fractal scaler (GPU, **4:2**)

Reduced 4x, then enlarged 4x
with fractal scaler (GPU, **8:4**)

Side by Side

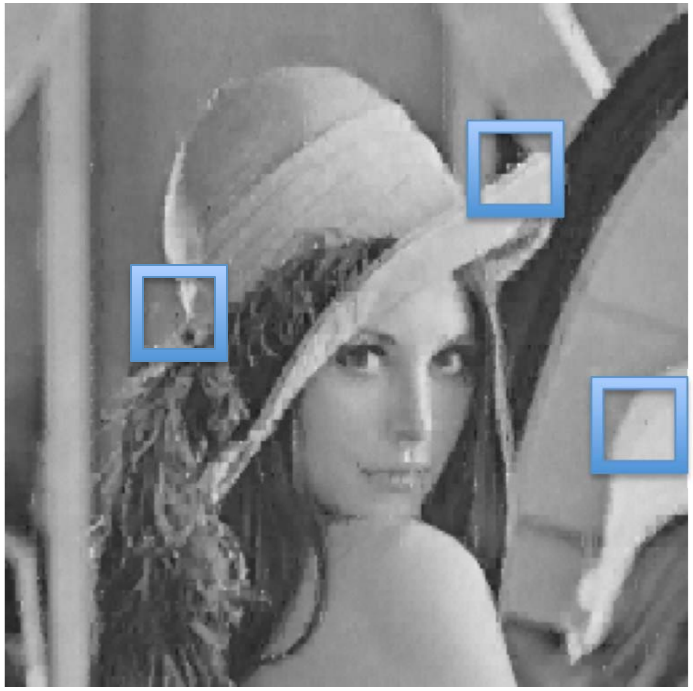
Reduced 2x, then enlarged 2x
with Mitchell filtering

Reduced 2x, then enlarged 2x
with fractal scaler (GPU, 8:4)

GPU vs CPU

- GPU has speckle noise
  - Probably due to lower accuracy

CPU, 8:4



GPU, 8:4

GPU vs CPU

- GPU has speckle noise
  - Probably due to lower accuracy

CPU, 4:2

GPU, 4:2

Accuracy issues

- Iterative nature of decoding process would tend to magnify errors
  - Testing on double-precision GPU needed

Wish list

- GPU-accelerated decoder
- CUDA or OpenCL port

Conclusions

- CPU encoder quality getting higher
  - Extremely slow
- GPU encoder is fast but has accuracy issues
  - Where are they?
  - Can they be fixed or compensated for?
  - CUDA or OpenCL port would help here