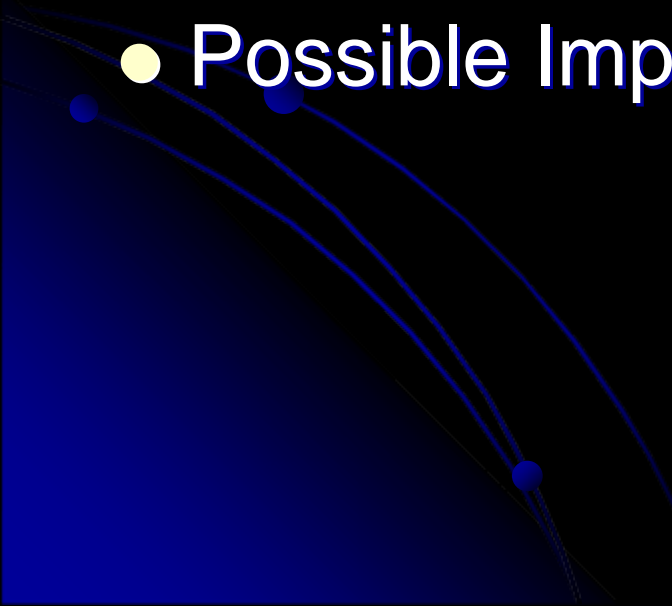


# Fractal Image Compression

By Cabel Sholdt and Paul Zeman



# Overview

- Why Fractal Image Compression
  - Mathematical Background
  - How does it work?
  - Examples
  - Possible Improvements
- 

# Why Fractal Image Compression

- Different type of compression scheme worth exploring
- Takes advantage of similarities within an image
- Advanced detail interpolation
- High theoretical compression rates
- Fast decompression times

# Mathematical Background

- Started with Michael Barnsley, and refined by A. Jacquin
- Try and find a set of transforms that map an image onto itself.
- The key is the Collage Theorem
  - States that if the error difference between the target image and the transformation of that image is less than a certain value the transforms are an equivalent representation of the image.

# How does it work?-Encoding

- Take a starting image and divide it into small, non-overlapping, square blocks, typically called “parent blocks”.
- Divide each parent block into 4 child blocks, or “child blocks.”
- Compare each child block against a subset of all possible overlapping blocks of parent block size.
  - Need to reduce the size of the parent to allow the comparison to work.
- Determine which larger block has the lowest difference, according to some measure, between it and the child block.
- Calculate a grayscale transform to match intensity levels between large block and child block precisely. Typically an affine transform is used ( $w \cdot x = a \cdot x + b$ ) to match grayscale levels.

# How does it work? – Encoding

- Upper left corner child block, very similar to upper right parent block.
- Compute affine transform.
- Store location of parent block (or transform block), affine transform components, and related child block into a file.
- Repeat for each child block.
- Lots of comparisons can calculations.
  - 256x256 original image
  - 16x16 sized parent blocks
  - $241 * 241 = 58,081$  block comparisons



# How does it work?- Decoding

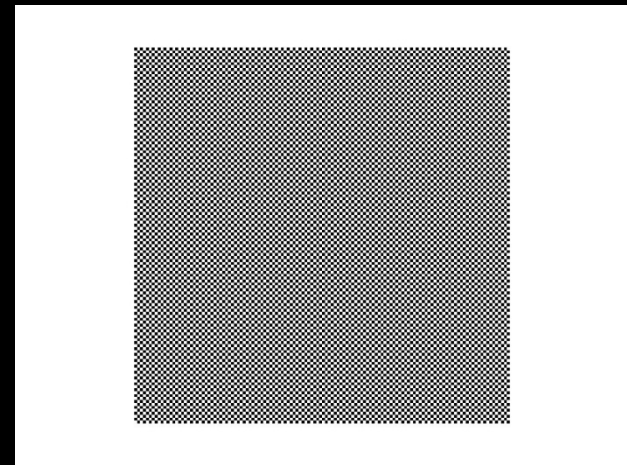
- Read in child block and transform block position, transform, and size information.
- Use any blank starting image of same size as original image
- For each child block apply stored transforms against specified transform block
- Overwrite child block pixel values with transform block pixel values
- Repeat until acceptable image quality is reached.

# Examples

- Original Image



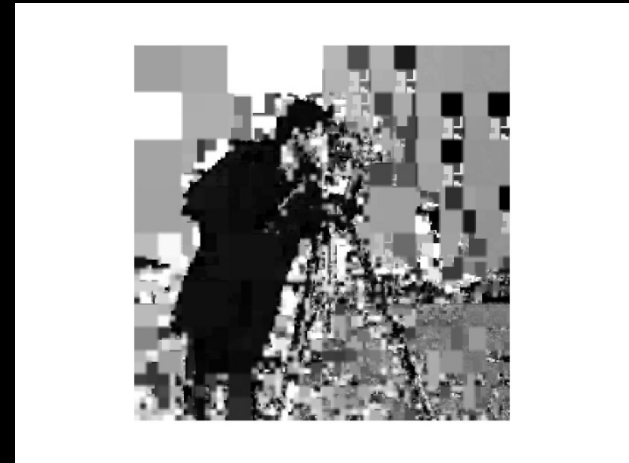
- Starting Image for Decoding





# How does it Work? – Decoding

- First Iteration



- Second Iteration



# How does it work? - Decoding

- Fifth Iteration



- Tenth Iteration



# Possible Improvements

- Greatest weakness is time for encoding
  - Possible speed ups
    - Order transform blocks into domains based off of average intensity and variance
    - Only search through blocks with similar structures
    - Do not search all possible blocks
    - Reduce number of child blocks
- Quality and Compression Improvements through
  - Quadtrees or HV Trees
  - Rotations of Transform Blocks during comparison
  - Improved grayscale transforms

Questions?

