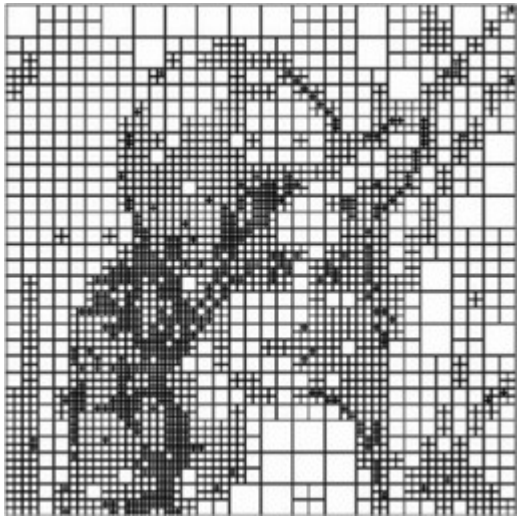
F.I.C. ⇔ Fractal Image Compression

An open source library written in Java, implementing the concepts of fractal image compression, along with a simple implementation — a proof of concept application.


`input image → partition sheme → output image`

» Brought to you, by

- Kanakarakis *c00kiemon5ter* Ivan — p3060190
- Ntanasis Periklis — p3070130
- Sarbinowski Pavlos — p3080179

Project Description

The project's goal, is to provide a reusable, free and open source library for fractal compression of images. Along with the library, comes a small application, indicating the library's usage and serving as an example and proof of concept of the ideas behind fractal compression.

» What is it ?

Fractal compression is a lossy compression method for digital images, based on fractals. The method is best suited for textures and natural images, relying on the fact that parts of an image often resemble other parts of the same image.

» Source

The code repository is hosted on GitHub.

The source for the library can be found under `/src/lib/` while the app lives under `/src/app/`.

For the documentation refer to the javadoc.

Usage Scenarios

Usually, images are expensive, especially when the quality is greater than needed. Compression is, in almost every case, necessary. Databases holding thousands or millions of images, or network data transfers, can benefit from different compression schemes, in terms of less storage space needed and less data to transfer over the wire, less bandwidth consumption and thus more efficient and faster browsing and less cost and charges by web hosting services.

Lossy compression schemes, such as fractal compression, can be used for images, that need not great detail and quality, like landscapes, but not medical images. Low bandwidth and few resource systems can benefit by lossy compression schemes, as they process less data, whereas big volume data would result to delays and greater chance or increased error rates or even deformation of data.

» **Fractal compression features**

- *Asymmetric Compression*: Encoding is extremely computationally expensive because of the search used to find the self-similarities. Decoding, however is quite fast.
- *Resolution Indepedence*: An inherent feature of fractal compression is that images become resolution independent after being converted to fractal code. A fractal compressed image can be decompressed to any size, without loss in its quality.
- *Interpolation*: The resolution independence of a fractal-encoded image can be used to increase the display resolution of an image. This process is also known as "fractal interpolation". Fractal interpolation maintains geometric detail very well compared to traditional interpolation methods like bilinear interpolation and bicubic interpolation.

Related Projects

There have been attempts of using fractal compression in the business and open source community in the past.

Historically Michael Barnsley led development of fractal compression in 1987 and was granted several patents on the technology. This made it difficult for developers to experiment and grow the interest on the technology, and thus not many project exist today based on the fractal compression scheme. Michael Barnsley and Alan Sloan formed Iterated Systems Inc. which was granted over 20 additional patents related to fractal compression.

The most widely known and successful implementation of fractal compression is by onOne Software company, which developed a fractal compression scheme (under the Iterated Systems Inc. license) as a Photoshop plugin. Another successful real world usage of fractal compression is by Microsoft in its Encarta multimedia encyclopedia, also under license.

In the Open Source world, Ullrich Hafner developed a library called Fiasco. It is now used by the Netpbm library and toolkit. Femtosoft provided another example implementation.

Numerous research papers have been published during the past few years discussing possible solutions to improve fractal algorithms and encoding hardware.

Responsibilities

- *Kanakarakis c00kiemon5ter Ivan*:

  System Architect, library and application design and component logic separation. Supervision and implementation of components and communication between them.

- *Ntanasis Periklis*:

  Implementation and design of various components, mostly focused on image block comparison.

- *Sarbinowski Pavlos*:

  Implementation and design of various components, mostly focused on affine transformations.

Every member contributed in the research conducted for the project, the preparation of the presentations and reports.

Achievements

This project is one of the few available freely and open source, dealing with the concept of fractal compression for images.

» Currently implemented

- different kinds of image segmentation
- different kinds of comparators supporting many metrics (AE, MAE, MSE, RMSE)
- many transformations to choose from, such as: flip, flop, shear, rotate, scale and special optimized transforms.
- a really compact fast and efficient fractal model
- a proof of concept application, as a usage example of the library.

» Todo

- Quadtree based and HV partitioning.
- Normalization of images can yield better results for the presented image quality.
- Histogram based image grouping to minimize image comparisons, thus faster compression.
- Multithreading support for even faster compression.

Application Workflow

The applications' workflow, is quiet simple. The application expects configuration arguments such as the tile size, the initial scale factors, enabling of debug information and others, to set its internal state and a command along with a file as input. There are two commands: compress and decompress. In case of compress, the expected file should be an image. In case of decompress, the expected file should be a compressed file as resulted by the compression step.

Compression:

Once the application initializes its state, it reads the image and starts the compressor. The compressor splits the image into tiles (segmentation step) and proceeds to compare each tile with every other part of the image after transforming the image with a set of selected transforms. The blocks that match most, are kept linked along with the transformation that has been applied. The end result is a list of tiles which are linked to a list of transforms, each of whom is linked with a list of points in the original image, that, applied to the tile that's linked to them, the result represents that part of the original image.

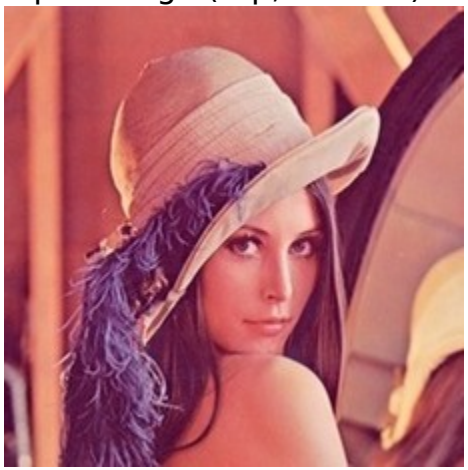
Decompression:

Decompression works the same way. The application reads a compressed file, as generated by the compression step, and forms back the original image.

Typical timings for compression is about 3min and less than a second for decompression.

Results

Input image (top, colored) and results (bottom) for tiles of `8px`, `4px` and `2px` respectively



Conclusions

Fractal Image Compression is a much promising and still young technology that can fit well in many areas of the multimedia systems' world. Despite the great compression time, the benefits of compression and resolution independence are much broader (for example, mapping services like google maps)

The Future

The project isn't finished. There are many things to experiment and play with.

New partition schemes (Quadtrees, HV partitioning) and optimizations (Histogram grouping, multithreading) that can yield better results faster, are yet to be discovered and implemented.

The base is here and everyone can join and expand the limits.

References

Wikipedia

- Image Compression
- Fractal Compression
- Iterated Function System
- Image Processing
- Binary Space Partitioning

Papers

- Fractal Image Compression @ google.schoolar
- Fractal Image Compression: Theory and Application book @ Amazon
- Fractal Imaging book @ Google Books
- University of Waterloo publications