A SYSTEM FOR IMAGE COMPRESSION USING WAVELETS AND GENETIC PROGRAMMING

Chris Nowak
Genetic Programming

Previously used in many different applications

With Image Compression, the idea is that one can genetically evolve a program that will approximate a some part of the image. Then, the program can be stored instead of what it approximates.
My Method

Input Image

- Discrete Wavelet Packet Transform

Data Organized into Vector and split up into multiple parts

- Part 1
- Part 2
- Part n

Approximation By Genetic Programming

- Function Pointer 3
- Function Pointer 2
- Function Pointer N

Library of Functions

Compressed Image

Approximation coefficients and locations of non-zero values
A Genetic Programming System in Java

The goal was to create a system which can find programs which will approximate the Wavelet Coefficients of an image.

This was accomplished entirely through manipulation of trees. That is, there is no actual code produced by the program, but instead all programs are stored as trees.
A Genetic Programming System in Java

Tree Representation of $2.5 / \text{max}(x, 2)$
A Genetic Programming System in Java

Population Generation:
- An initial population of programs is generated to be evolved
- Different Possible Operators
- Implementation of Conditionals
A Genetic Programming System in Java

Evolution:
- Fitness Calculation
- Crossover
- Mutation
- Pruning

Crossover
Compression in the Wavelet Domain

Previous Techniques:
- JPEG 2000
- SPIHT
- Zero-Tree Compression

Why use Wavelets at all?
Compression in the Wavelet Domain

Data Organization is an important factor, since it is necessary to organize the data in such a way that it will be easy for the genetic programming system to approximate

How to accomplish this?
The Discrete Wavelet Packet Transform

Normally, in order to get higher levels of a DWT, we apply the transform to the approximation coefficients. For the Discrete Wavelet Packet Transform, the DWT is applied to the approximation, horizontal, vertical, and diagonal coefficients.
The Discrete Wavelet Packet Transform

Quantization is the key to simplifying the data once in the Wavelet Domain. By quantizing the coefficients by a certain value, we can simplify the data which we need to approximate. Larger Quantization step sizes result in more image loss, but more compression.
The Discrete Wavelet Packet Transform

After the Discrete Wavelet Packet Transform is performed, the different subbands are “stacked” on top of one another. Data is extracted in lines from the top of the stack to the bottom of the stack. By concatenating these lines, we end up with one long vector which can then be sent to the GP System.
The Discrete Wavelet Packet Transform

Problem #1: Quantization has caused there to be a large amount of zeros. This makes the data difficult for the Genetic Programming System to approximate.

Solution: Do not send the zero values to the Genetic Programming system. Instead, send the non-zero values, and store where all the zero values are located. Since there is such a large quantity of zeros, they can be encoded efficiently using Run Length Encoding. This way, we benefit from long runs of zeros, and quantization is far more effective.
The Discrete Wavelet Packet Transform

Problem #2: The approximation coefficients are very different when compared to the rest of the wavelet data. Often they have values much higher than the rest of the coefficients, which makes them difficult for the Genetic Programming System to approximate.

Solution: Do not send the approximation coefficients to be approximated. Instead, store them in the final image file. This results in less error in the compressed image, since any error in the approximation coefficients is much more noticeable than error in other coefficients.
Final Method of Compression

The Wavelet Packet Transform is applied, and the non-zero values which occur are sent to the Genetic Programming System to be evaluated. In order to control complexity, only 10 points were sent in at a time. So, any one function only was ever used to approximate 10 points. Greater compression ratios could be achieved by increasing the number of points the GP system approximates, but the amount of error increased along with this.

After the genetic programming system created the functions, a set of pointers to these functions was returned, and the programs were stored in a library.
Results

**TABLE I**

<table>
<thead>
<tr>
<th>Quantization Step Size</th>
<th>Compression Ratio (no code)</th>
<th>Root Mean Square Error</th>
<th>Compression Ratio before Genetic Programming</th>
<th>Compression Ratio (with code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>8.5:1</td>
<td>4.6</td>
<td>5:1</td>
<td>No compression</td>
</tr>
<tr>
<td>50</td>
<td>14.2:1</td>
<td>6.9</td>
<td>9.8:1</td>
<td>1.7:1</td>
</tr>
<tr>
<td>100</td>
<td>22.1:1</td>
<td>8.8</td>
<td>16.2:1</td>
<td>6.5:1</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th>Quantization Step Size</th>
<th>Compression Ratio (no code)</th>
<th>Root Mean Square Error</th>
<th>Compression Ratio before Genetic Programming</th>
<th>Compression Ratio (with code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>6.5:1</td>
<td>3.9</td>
<td>4:1</td>
<td>No Compression</td>
</tr>
<tr>
<td>50</td>
<td>12.4:1</td>
<td>7.7</td>
<td>8.1:1</td>
<td>1.2:1</td>
</tr>
<tr>
<td>100</td>
<td>18.0:1</td>
<td>10.16:1</td>
<td>13.0:1</td>
<td>2.7:1</td>
</tr>
</tbody>
</table>
Results

Original Image

Quantization: 20
Compression: 6.5:1

Quantization: 20
Compression: 12.4:1

Quantization: 20
Compression: 18:1
Results

Original Image

Quantization: 20
Compression: 8.5:1

Quantization: 20
Compression: 12.4:1

Quantization: 20
Compression: 18:1
Code Reuse

In order for this to be a viable approach for image compression, code which is developed must be reusable, so a library of functions can be developed to approximate all images.

A test was run to approximate the F16 image with the functions that had been evolved from approximating a Lena image. A quantization step size of 100 was used. Of the 670 functions that were needed to evolve the data for the F16 image, 188 of them were able to be reused from the Lena data.