

# USING GENE EXPRESSION PROGRAMMING TO ACHIEVE LOSSLESS IMAGE COMPRESSION

*John D. Zechlin*

Utah State University

## 1. Introduction

As the quality of image capture devices continues to grow and the demand for efficiency and speed of communications systems continues to increase, image compression is becoming more and more vital to the success and usability of those devices and systems. One solution to image compression is using Gene Expression Programming to evolve a model for a given image and using it later to decompress the image. The basic idea behind the lossless compression of an image is as follows:

- Step 1: Perform any necessary preprocessing in order to get a data set that will easily evolve in the GEP environment.
- Step 2: Evolve the data set using GEP.
- Step 3: Depending on the method being implemented, use the evolved model to either evolve a more accurate model or construct a data array that represents the residual between the evolved model and the original image.
- Step 4: Package the resulting data using Huffman and run-length encoding in order to improve compression.

## 2. Preprocessing

**(a) FiveValues:** The FiveValues function is the most basic and standard method of preprocessing that was used. Given an image, the FiveValues function traverses the image starting at the second row and the second column. The intent is to use at least the first column and the first row as seed values when reconstructing (or decompressing) the image. In the five value model, the last column is also

saved. Except for these seed values, every pixel is represented by a row in the new data array. A row of five values is saved for each pixel that is traversed. For a given pixel with the coordinates  $(i,j)$ , the five values that are saved are the pixel values at the following coordinates:  $(i,j-1)$ ,  $(i-1,j-1)$ ,  $(i-1,j)$ ,  $(i-1,j+1)$ , and  $(i,j)$ . The last value is the actual pixel value and becomes the column designated as the target value in the evolution.

One variation of the above method is simply leaving out the last input value,  $(i-1,j+1)$ . One advantage to using four values instead of five is the need for only the first row and first column as seed values. Another advantage is that the resulting data set becomes less complex and therefore easier to evolve. The subsequent loss that occurs has to do with the existence of duplicates, which will be covered in section 3.

Due to some of the problems that arise when using the FourValues or the FiveValues functions, another alternative is using a six-value file. This method is identical to the FiveValues, except that it adds one more column. The additional column is simply an index value that provides a unique number for each row in the data, starting at 1 and ending at the quantity of pixels in the data set. While using six values solves the problem of duplication, it increases the complexity of the evolution.

**(b) Time Series:** The Time Series function is similar to the FiveValues function in that it is a simple traversal of the image. In this approach, the entire image is traversed in a zig-zag pattern starting at the first pixel. The finished data array is a file that is one column wide and contains enough rows to represent each pixel in the image. Because of the zig-zag pattern,

every pixel is preceded and followed by adjacent pixels in the resulting array. Rather than traversing in a straight-line fashion, a diagonal zig-zag pattern is used to ensure that the previous values are still relatively close (and thus, correlated) to the target value.

**(c) Discrete Wavelet Transform:** The DWT is one step that can be taken in the preprocessing phase. Because the DWT yields four smaller images from the source image, the idea is that evolving those four parts separately will be less complex than evolving the source image.

(d) Discrete Cosine Transform: The DCT is another approach that is meant to make the evolution phase less complex.

### 3. Challenges and Solutions

**(a) Duplicate lines of input with different output:** One recurring challenge that frequently presents itself is the existence of duplicate lines of input with different target values. For example, the series 3 5 8 9 could be duplicated on several rows of the FiveValue data set with the fifth value (the target value) being different, such as the following: 3 5 8 9 12 and 3 5 8 9 6. Unfortunately, duplicate rows such as this will almost always occur in the file. The disadvantage of this occurrence is that it limits the accuracy of any evolved model. There is simply no correct answer for a given set of four inputs when several different target values are found throughout the data set.

#### *i. Removing the duplicates*

One attempt at eliminating the problem of duplicate rows is simply eliminating the duplicate rows altogether. The drawback to this approach is that the resulting data set is not completely representative of the image (or, the portion of the image), which may alter any evolution that is performed.

#### *ii. Adding index value to file*

This approach can be implemented during the construction of the data set by adding one additional column as the image is traversed. The value saved represents the number of rows

of data that have been saved. In other words, for any data set  $x$  rows in length, there will be corresponding indices from 1 to  $x$  in the file. Adding an index to the file effectively eliminates all instances of duplicate rows. Because there is an index, the duplicate rows will vary by the index if nothing else. Although this approach appears to solve the problem, the results do not show significant improvements.

**(b) Too many Zeros:** The existence of a large number of zeros (or any number, for that matter) presents another challenge when attempting to evolve a model. During most standard evolutions (when a four or five value data set is used), numerous zeros are rarely a problem. It will, however, occur when attempting to evolve residual files and during the implementation of the DCT method, which is further discussed in section 4. The existence of zeros in the data set presents a similar problem as duplicate rows and in some cases, causes the same problem. For example, several single residual values might all be surrounded by zeros, which would make it impossible to evolve an accurate model. While numerous zeros don't always cause an exact duplicate row, they do cause something that is close to a duplicate row, which can be just as hard to predict. An example of this would be something like:

```
0 0 0 4 2
0 0 2 0 2
0 0 3 0 2
```

Although the three rows above are not duplicates, the existence of three zeros in each row significantly complicates the evolution. For this reason, any evolution of a data set that contains a large number of zeros will likely yield unfavorable results.

**(c) Variation in image:** In some cases, an image may look largely different in some areas than it does in other areas. This variation can result in a reduction in accuracy of the evolution that is performed. A solution to this problem is the segmentation of the image, followed by subsequent compression using any of the methods listed below. The challenge

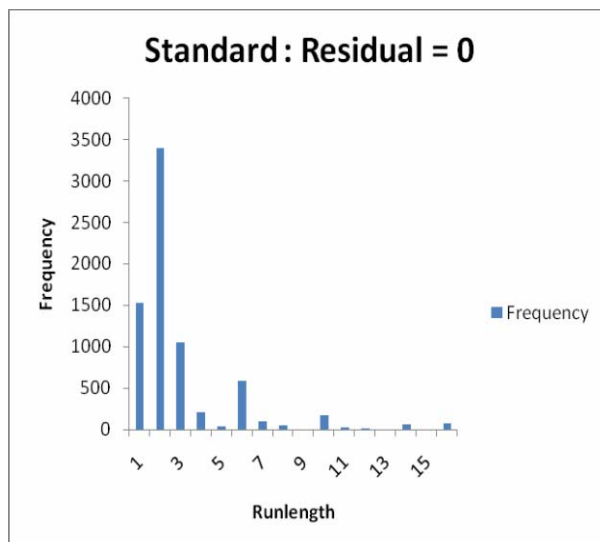
would be to effectively segment the image in a way that would lead to better compression. Since each image is different, each picture would require different segmentation schemes in order to produce desirable results.

#### 4. Detailed Methods

**(a) Standard Approach:** This method is the basic starting point for all other methods of image compression using gene expression programming. Starting with either the four or five value files, an evolution is performed to a point in which further evolutions would not yield any significant improvements. Once the evolution is performed, the model is compared to the original image and a residual file is created from the difference between each pixel value in the model and in the original image. The model, residual array, and either two or three rows or columns from the original image are saved as a lossless representation of the image. If the four value function is used, only the first row and first column of the original are needed. The last column in the image is added if the five value function is used. Because the residual array is the most costly component of this compression scheme, Huffman and Run-length encoding can be applied to it in order to further compress the image.

*Table 1: Results from standard method expressed in average run length of residual values.*

Residual Value	Ave run length	Occurrences (% of total pixels)
-4	1.08	2.60%
-3	1.05	3.81%
-2	1.08	5.72%
-1	1.42	6.10%
0	3.018	34.33%
1	1.42	6.61%
2	1.05	6.11%
3	1.05	4.22%
4	1.07	2.56%
<-4 & >4	n/a	27.94%



*Figure 1: Standard method histogram of run lengths of residual = 0.*

Variations to this method include adding an index value to eliminate duplicates, using a zig-zag traversal pattern to compliment the addition of the index value, and using a standard segmentation scheme such as evolving only half the image. Although these variations appear to solve some of the inherent problems with the above compression scheme, they either do not yield considerable improvements or in the case of segmentation, the cost of performing several evolutions is not worth the improvement that is produced.

Table 2: Results from DWT method expressed in average run length of residual values.

Residual Value	Ave run length	Occurrences (% of total pixels)
-4	1.07	2.59%
-3	1.06	2.30%
-2	1.05	3.41%
-1	1.16	5.18%
0	1.24	9.54%
1	2.2	16.64%
2	1.08	3.87%
3	1.04	3.00%
4	1.04	2.14%
<-4 & >4	n/a	51.33%

Table 3: Results from DCT method expressed in average run length of residual values.

Residual Value	Ave run length	Occurrences (% of total pixels)
-4	1	0.02%
-3	1.05	0.06%
-2	1.17	0.16%
-1	1.2	1.05%
0	15.1786	21.40%
1	1.2	1.09%
2	1.07	0.15%
3	1.03	0.06%
4	1	0.03%
<-4 & >4	n/a	75.96%

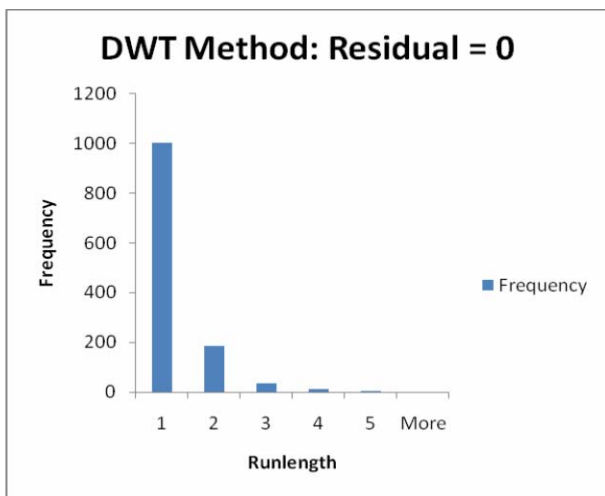


Figure 2: DWT method histogram of run lengths of residual = 0.

**(b) Wavelet Approach:** The Discrete Wavelet Transform produces four images that are derived from the original image. The concept of using this for image compression is based on the fact that one of the images produced by the DWT (the filtered image) will have most of the image data contained in it and the other three will have minimal data. If a successful evolution is performed on an image that is one fourth the size of the original image, then the other  $\frac{3}{4}$  of the image can be dealt with in other ways (run-length, Huffman, etc.). Unfortunately, the evolution is less successful with this method due to less correlation and more variation between neighboring pixels.

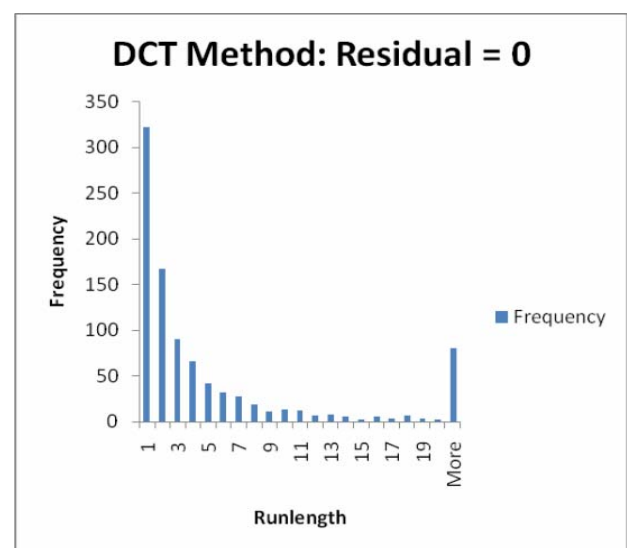


Figure 3: DCT method histogram of run lengths of residual = 0.

**(c) Discrete Cosine Transform:** In this approach, the Discrete Cosine Transform of the image is used to evolve a time series model.

**(d) The Seed Approach:** Once an evolution is completed on a given data set, a new set of data is produced from the evolved model. This approach seeks to exploit the new set of data in order to evolve a more accurate solution. The first step is to evolve a model using the standard Six Value file, which yields a standard model that gets values that are merely close to the target values. Another evolution can then be performed on the original six values and the model values from the original evolution. This can be applied to the data set as many times as is needed in order to get a sufficient model. Because additional models only add one function to the overall size of the compressed image, it is more practical to use  $x$  seed functions than it would be to segment the image into the  $x$  parts, for example. The challenge with this method remains the same as many of the others: once a model is produced, there are always some values that are not easily predicted. In this case, those values seem to remain large, even when multiple seed functions are applied. Fortunately, the large values do tend to shrink as more seed models are applied, even though those values do stay relatively large.

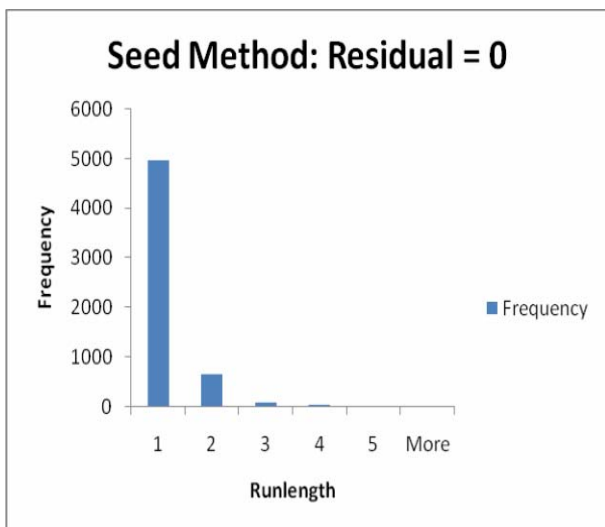
*Table 4: Results from seed method expressed in average run length of residual values.*

Residual Value	Ave run length	Occurrences (% of total pixels)
-4	1.07	4.66%
-3	1.1	6.52%
-2	1.13	8.54%
-1	1.16	10.16%
0	1.16	10.21%
1	1.15	9.05%
2	1.1	6.99%
3	1.08	5.10%
4	1.05	3.64%
<-4 & >4	n/a	35.14%

One variation of this method that seems to yield better results (with a cost) is segmenting the data set after several seed models are applied. In order to achieve the improved results, the segmentation is determined by the residual value of each pixel. Because of the scattered manner of this segmentation, an additional array is required to determine which function will be used to calculate the final values. The result is that the additional array consumes just as much space as is saved with more accurate evolutions.

## 5. Conclusion and Recommendation for Further Work

Each of the above outlined methods has its own benefits and downfalls. Where one method fails, another succeeds. Although the methods covered here do not completely solve the problem, further work on this topic could include investigating solutions for long-running series, problem pixels, and pixel variation. One possible approach would be to perform further preprocessing in order to separate common regions in the image which may result in more accurate evolutions.



*Figure 4: Seed method histogram of run lengths of residual = 0.*