Application of Uniform Quantization and Huffman Coding to the Iterated Block Matching Fractals for Image Compression

Rolando J González & Paul G Román
Advisor: Dr. Hamed Parsiani

Laboratory for Applied Remote Sensing and Image Processing
Electrical and Computer Engineering Department
University of Puerto Rico, Mayagüez Campus
Mayagüez, Puerto Rico 00681-5000
rolandog@larsip.uprm.edu / roman_paul@hotmail.com

ABSTRACT

The concept of Fractals dates back to Mandelbrot [1]. Improvements were achieved by other researchers such as Barnsley and Jaquin [2],[3]. In this work, the Iterated Block Matching Fractals (IBMF) [4] software is being coded using Uniform Quantization (UQ) and Huffman Coding (HC). Theoretical description of each coding method is presented using examples. The IBMF output parameters are coded using the cascaded UQ/HC method. The compression achieved for the Lena image is 8.5 at the Peak Signal to Noise Ratio (PSNR) of 33.7. The objectivity of this result was preserved by not including the Lena image parameters in determination of the UQ and HC tables. However, for tutorial purposes an image of a baby, which was included in the Huffman table creation, was used as the input image to be compressed by the IBMF. The result was as expected a higher compression of 14 at the PSNR of 37.8. Therefore, the limits of compression of a given image are very much related to being totally a part of the pool at one extreme or being totally out at the other extreme.

1. INTRODUCTION

Uniform Quantization and Huffman Coding theory and their implementation to the IBMF software is explained in details and compression tests are presented with numerical results. The purpose of UQ is to code the parameters of the IBMF output, assuming equally sized bins for the parameters to be coded. On the other hand, the purpose of HC is to achieve a higher compression ratio by compressing the UQ output by assigning shorter bit patterns to the most frequent symbols. The cascading of UQ and HC will permit a higher compression ratio. However, the cascading of a Non-UQ and a HC will not lead to any higher compression, because the symbols generated by the Non-UQ have equally likely probabilities of occurrence, and Huffman will not cause any compression to take place.

For testing purposes, a pool of images was used to construct Huffman tables. The last part of this report contains pictures and conclusions with results from the software.

2. UNIFORM QUANTIZATION

Quantization is the conversion of a continuous signal into a discrete signal. The value of each signal sample is represented by a value selected from a finite set of possible values. The difference between the unquantized input and the quantized output is called quantization error. Uniform

Quantization can be implemented to any type and number of inputs, but it is more efficient if it is a multiple of 2^N. To build a uniform quantizer, subtract the minimum value from the maximum value, and then divide it by the compression ratio desired. For example, in a 16 (from 0 to 15) discrete variable input with a desired compression ratio of 4, a 4 (from 0 to 3) variable output is produced which are quantization indices representing the input values.

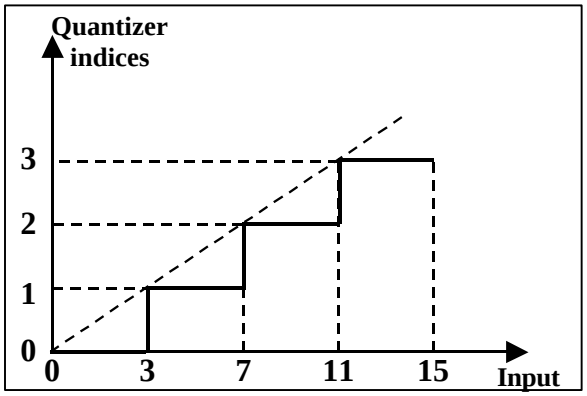

Figure 1. Uniform Quantization with a compression ratio of 4.

Fig. 1 presents an example of uniform quantization with the desired compression ratio of 4.

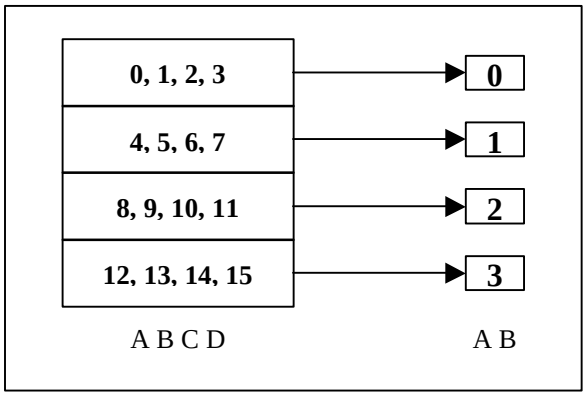

Figure 2. 16 variable input quantized to give a 4 variable output.

It can be seen how after applying quantization, the input changes from a 4 bit representation (A B C D) to a 2 bit (A B) representation giving a compression ratio of 4, Fig. 2. As discussed previously, HC achieves compression if some symbols are more likely to occur than others are.

3. HUFFMAN CODING

Huffman coding is a lossless data compression algorithm. The Huffman (or variable length coding) method is based on the fact that some symbols have a higher frequency of occurrence than others. These symbols are encoded in fewer bits than the fixed length coding producing in average a higher compression. The idea behind HC is to use shorter bit patterns for more frequent symbols. To achieve that goal, Huffman algorithm needs the probability of occurrence of each symbol. These symbols (stored in nodes), Fig. 3, are then sorted in ascending order of their probability value.

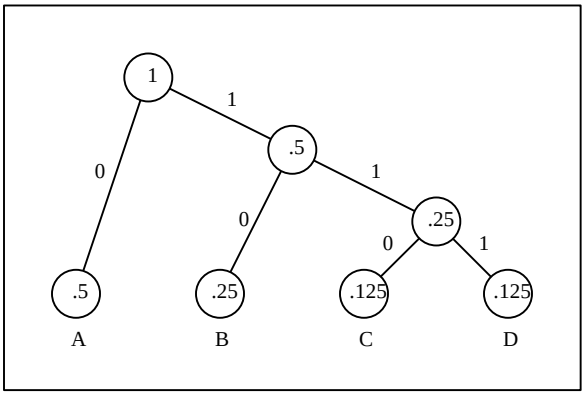

Figure 3. Huffman tree representation for a four symbols data.

The algorithm selects two nodes (children) with the smaller probabilities and constructs a new node (parent) with a probability value equals to the addition of the probabilities of its children. This process ends when the node with probability equal to 1 is constructed. The result of this algorithm is a binary tree data structure with the root being the node with probability equal to 1, and the last nodes (leaves) represent the original symbols. This binary tree is then traversed

to reach its leaves when a certain symbol is needed. Moving to a right child insert a 1 in the bit pattern of the symbol, while moving to a left child insert a 0. The result is a unique bit pattern for each symbol that can be uniquely decoded. This is because no code word can be a prefix of any larger-length code word.

Table 1. Code before Huffman Coding

Character	Code	Frequency	Total Bits
A	00	16	32
B	01	8	16
C	10	4	8
D	11	4	8
		Total	64

Table 2. Code after Huffman Coding

Character	Code	Frequency	Total Bits
A	0	16	16
B	10	8	16
C	110	4	12
D	111	4	12
		Total	56

4. APPLICATION OF UNIFORM QUANTIZATION AND HUFFMAN CODING TO THE IBMF FOR IMAGE COMPRESSION

The IBMF blocks are divided into three types: *shades*, *mid-ranges*, and *edges*. For each of these types different parameters need to be encoded with different number of bits as in Table 3, [4].

Table 3. Number of bits per parameter

Parameter	Shades	Mid Ranges	Edges
Type	2 bits	2 bits	2 bits
Mean	6 bits	6 bits	6 bits
Alpha		4 bits	4 bits
Domain X		6 bits	6 bits
Domain Y		6 bits	6 bits
Isometry			3 bits
Total	8 bits	24 bits	27 bits

As an example, Alpha values are real numbers (4 bits) from 0.0 to 2.0 and Mean values are integers (6 bits) from 0 to 255. In order to achieve these quantization results (table 3), UQ is implemented. After these parameters are quantized HC compresses them into shorter bit patterns according to their probabilities. Later when the decoder receives a bit pattern, it compares it with the Huffman tables looking for the symbol representing this bit pattern. Decoding can be uniquely achieved for each symbol, since no Huffman Code is a prefix of another longer length code.

5. GENERATING HUFFMAN TABLES USING A POOL OF IMAGES

To implement the techniques described previously, look-up tables were created using a pool of 15 images. A varied selection of images was used like faces, landscapes, diverse objects, and LANDSAT satellite images. The idea behind this is to do a statistical analysis on a large sample of data. The pool of images was fed to the IBMF encoder and the output parameters were used as inputs to the Uniform Quantizer. The output files of the UQ for the 15 images were combined along with their respective probabilities and fed into a Huffman table generator. And, a table of Huffman code is generated for each of the parameters to be coded. Then, the outputs of these algorithms are the look-up tables that the IBMF will use later to code any image. Later, a new image to be processed is then encoded accordingly by the tables. Higher compression is achieved with images, which happens to be a part of the original pool of images chosen. And, lesser compression is produced for the images, which do not have many similarities with the pool of images. This is why if more images were used to create these tables, a

better compression at higher quality would be achieved.

6. CONCLUSION

Higher compressions at similar or better qualities are produced for IBMF with UQ/HC compression method. Lena image (512 x 512), Fig. 4, was not a part of the 15 images involved in the creation of the tables. Lena was coded using the tables, giving compression ratio of 8.5 at PSNR of 33.7. On the other hand, the Baby image (384 x 384), Fig. 5, was part of the tables, so when coded with the IBMF software a higher compression ratio of 14 at PSNR of 37.8 was achieved.

ACKNOWLEDGEMENT

Work supported by NASA grant NCC 5-252

REFERENCES

[1] B. Mandelbrot, "The Fractal Geometry of Nature". San Francisco, CA: Freeman, 1982.

[2] M. F. Barnsley and S. Demko, "Iterated function systems, and global construction of fractals," Proc. Roy. Soc. London, vol. A399, pp. 243-275, 1985.

[3] A. E. Jacquin, "Fractal Image Coding: A Review," Proceedings of the IEEE, vol. 81, No. 10, Oct. 1993.

[4] Hamed Parsiani, Andres Fuentes, "Fast Near lossless Iterated Block Matching Fractals Image Compression", Proceedings of the IASTED International Conference, Signal and Image Processing , SIP'98, Oct. 28-31, 98, Las Vegas, Nevada.

Figure 4a. Original image of Lena

Figure 4b. Reconstructed image of Lena

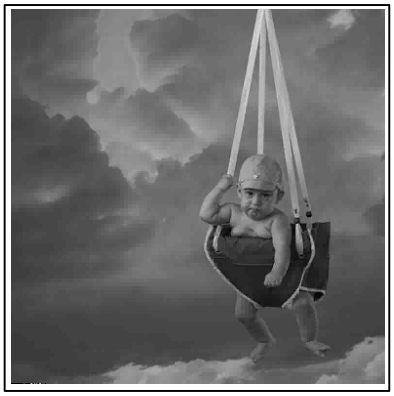

Figure 5a. Original image of baby

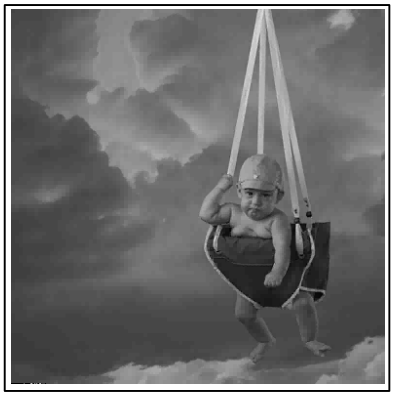

Figure 5b. Reconstructed image of baby