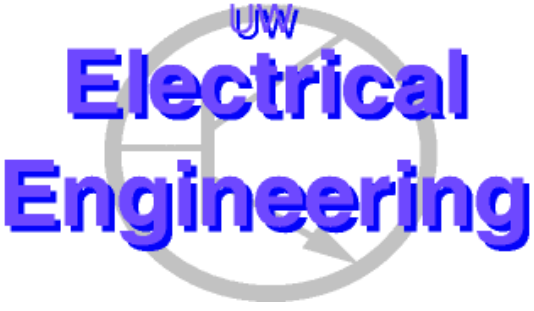
Arithmetic Compression on SPIHT Encoded Images

Todd Owen, Scott Hauck

{towen, hauck}@ee.washington.edu

Dept of EE, University of Washington
Seattle WA, 98195-2500



UWEE Technical Report
Number UWEETR-2002-0007
05/06/2002

Department of Electrical Engineering
University of Washington
Box 352500
Seattle, Washington 98195-2500
PHN: (206) 543-2150
FAX: (206) 543-3842
URL: http://www.ee.washington.edu

Arithmetic Compression on SPIHT Encoded Images

Todd Owen, Scott Hauck
{towen, hauck}@ee.washington.edu

Dept of EE, University of Washington at Seattle
Seattle WA, 98195-2500

University of Washington, Dept. of EE, UWEETR-2002-0007
May 2002

Abstract

Set Partitioning in Hierarchical Trees (SPIHT) is a wavelet based compression algorithm that offers good compression ratios, a fully progressive bit-stream, and good image quality. This paper presents the results of adding arithmetic compression to the SPIHT images in the hopes of further reducing the image size. Both the original SPIHT algorithm and a version optimized for FPGAs were used. Our tests used a model-0 adaptable arithmetic coder and show that SPIHT images can be reduced another 0-5%. Further, the compression ratio was dependent on the bits/pixel encoding level of the SPIHT image and slightly dependent on the symbol length used.

1 Introduction

Set Partitioning in Hierarchical Trees (SPIHT) is a wavelet-based image compression coder that offers a variety of good characteristics. These characteristics include:

- Good image quality with a high PSNR
- Fast coding and decoding
- A fully progressive bit-stream
- Can be used for lossless compression
- Ability to code for exact bit rate or PSNR

The main advantage of SPIHT is that it is fully progressive, meaning that we do not need the whole file to see the image. The image's PSNR will be directly related to the amount of the file received from the transmitter. This means that our image quality will only increase with the percentage of the file received. After the SPIHT transformation some regularities will exist in the file. These regularities may allow us to further compress the file. With this in mind we investigated the addition of arithmetic compression to a SPIHT encoded image. A few different code implementations are presented here and the compression results are compared.

2 Overview of Arithmetic Coding

Arithmetic coding is a well-known method for lossless data compression [1, 2]. The idea can be traced back to the work of Shannon [3], but it was first explicitly described by Elias, and described in a footnote in reference [4].

Arithmetic coding is a compression technique that can give compression levels at or very near entropy. What this means is that if we have a message composed of symbols over some finite alphabet, we can generate the exact number of bits that corresponds to a symbol (e.g. 1.6 bits/symbol). This is opposed to Huffman encoding which must output an integer number of bits per symbol (e.g. 2 bits/symbol). Arithmetic coding achieves entropy (or very near it) by grouping symbols together until an integer value of bits can be outputted for a sequence of symbols (e.g. ABC may correspond to 1011).

Arithmetic coding works by using a probability interval defined with variables L and R, which are initially set to 0 and 1 respectively. The value of L represents the smallest binary value consistent with a code representing the symbols processed so far. The value of R represents the product of the probabilities of those symbols. To encode the next symbol, which is the jth of the alphabet, both L and R must be recomputed. L and R get the following values:

$$L = L + R \times \sum_{i=1}^{j-1} p_i$$

$$R = R \times p_i$$

This preserves the relationship between L, R, and the symbols that have been previously processed. At the end of the message, any binary value between L and L + R will unambiguously specify the input message. The interval will continue to be redefined until an end-of-sequence marker is coded. An example of a coding sequence can be found below in Figure 1. Figure 1 details the coding of the sequence of symbols IUI where each symbol has a predefined probability.

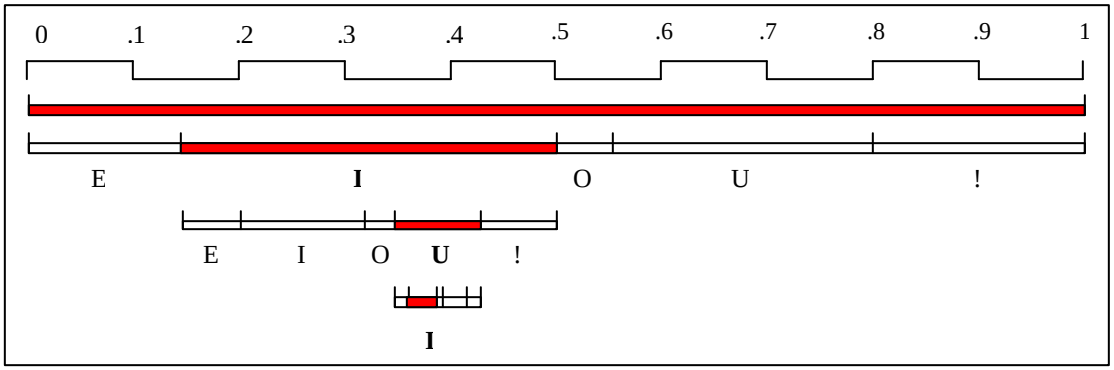


Figure 1. Arithmetic coding of the sequence IUI. The top line is a scale. Each succeeding line represents a stage of the coding process. The first stage is the initial state [0,1), which is followed by succeeding intervals. At each stage the current interval is the shaded rectangle. The value of L is the left hand side of the interval and the value of R is the length of the interval.

3 The Addition of Statistical Modeling

In theory, statistical modeling can be added to an arithmetic coder to achieve better compression levels. Our tests indicated that SPIHT images do not benefit from advanced statistical modeling, but for the purpose of understanding arithmetic coding, this topic will be briefly covered.

Statistical modeling is generally referred to as model-0, model-1, model-2, model-3, etc, where the model number indicates the level of probabilities generated. In a model-0 coder, each symbol is given a probability. In a model-1 coder, each symbol is given a probability and the probability of the next symbol based upon that symbol occurring is also calculated. In a model-2 coder, the probability of a 1st symbol is generated along with the probabilities of the next two symbols based upon that symbol occurring. The problem with high order models is that they require a large amount of memory to run and are considerably slower than a zero order model.

Our data will only show results from order-0 encoders as they gave us the best compression results.

4 Arithmetic Models Tested

For our tests we used two arithmetic coders. The first coder tested was written by Alistair Moffat, Radford M. Neal, and Ian H. Witten [5]. This character-based coder is an implementation of the CACM coder [1]. This coder uses no statistical modeling in combination with the arithmetic coder. Further research was done on the character model by changing the code so the symbol length could be varied between 2 and 16 bits in 1 bit intervals.

The second coder comes from work done by Mark Nelson [6]. Mark Nelson's code uses statistical modeling in combination with an arithmetic coder. Mark Nelson's coder is also an implementation of the CACM coder.

5 Arithmetic Coding Results

The arithmetic compressors were tested on two versions of the SPIHT algorithm. The first version was the original version of the algorithm [9] and the second was a modified version optimized for FPGAs developed in our prior work[7].

Our results were obtained from two libraries of images. The first library consisted of seven images that Tom Fry worked with for his hardware implementation of a SPIHT encoder [7]. The second library of images consisted of ten overhead views of the earth obtained from NASA satellites. The NASA images are freely available from a variety of NASA websites [8]. All images were grayscale PGM (8 bits/pixel) at 512x512 pixels. The intention was to get an idea of the performance on a specific type of data verses a random set. Overall, the compression rate was better on the random set of images. This may be due to the fact that the random images generally had more contrast to them.

Compression rates were measured using both coders and both SPIHT algorithms. The coder from Mark Nelson gave better compression than the implementation from Moffat, Neal, and Witten. For the both SPIHT algorithms, the Nelson encoder compressed the image an extra 0.5-1%. To measure the compression rate at variable symbol lengths, the code from Alistair Moffat, Radford M. Neal, and Ian H. Witten was modified so that symbol length could be varied from 2 to 16 bits. This code was modified because it was easier to modify than the code from Mark Nelson. Modifying the symbol length had little effect at improving the compression ratio. Compression results can be found in figures 2 and 3.

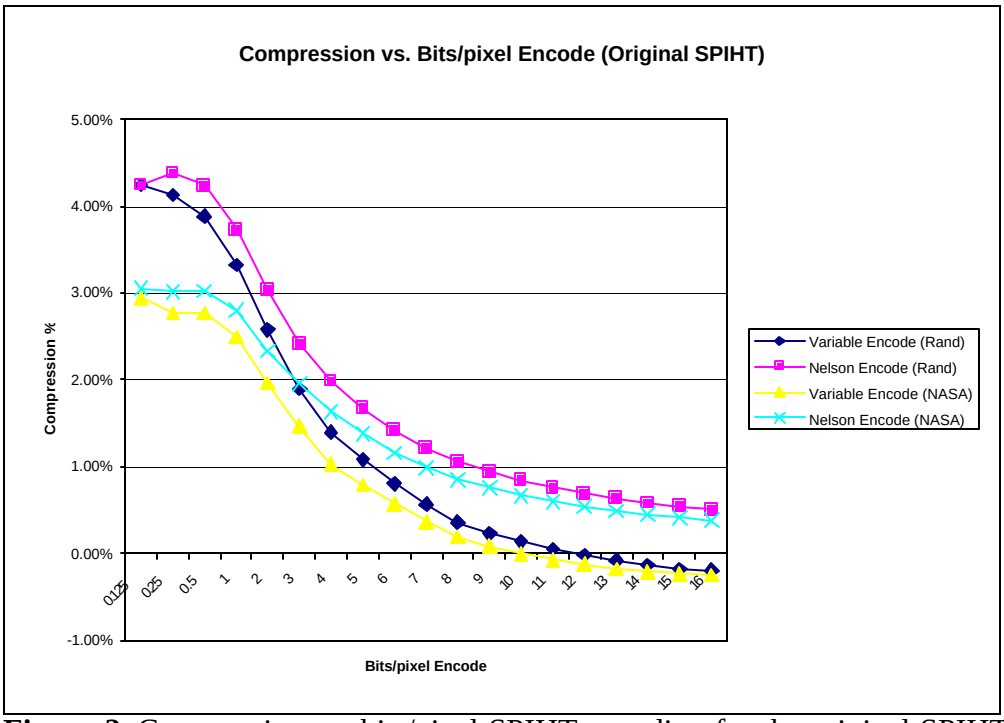


Figure 2. Compression vs. bits/pixel SPIHT encoding for the original SPIHT algorithm. Rand refers to the random set and NASA refers to the NASA image set.

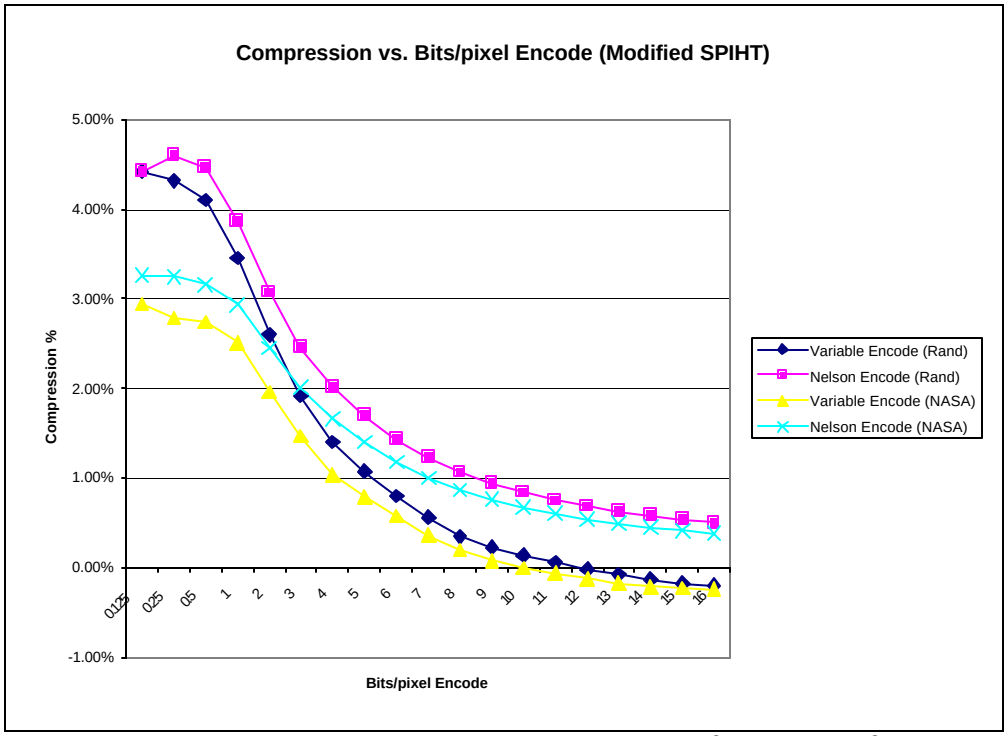


Figure 3. Compression vs. bits/pixel SPIHT encoding for the modified SPIHT algorithm. Rand refers to the random set and NASA refers to the NASA image set.

The compression rate from the arithmetic coders was not dependent on the SPIHT algorithm used, but was dependent on the bits/pixel rate of encoding. Both the standard SPIHT and modified SPIHT algorithms gave poor compression results. At best, a rate of around 4-5% was achieved at low bit rates while a rate between 0-1% was achieved at high bit rates.

When varying the symbol length, different symbol lengths were optimal at different bit rates. In general, symbol lengths with a multiple of two did best. However, a few bits/pixel encoding levels did best with a symbol length of three. Table 1 below gives the best symbol length for a given bits/pixel encoding rate.

Table 1. Best Symbol Length for a given bits/pixel SPIHT encoding rate.

Bits/pixel	0.125	0.25	0.5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Rand (Modified SPIHT)	4	4	8	8	8	8	2	2	2	2	3	2	2	2	2	2	2	3	3
NASA (Modified SPIHT)	4	4	8	8	8	8	8	2	2	2	3	2	2	2	2	2	3	3	3
Rand (Orig. SPIHT)	4	4	4	8	8	8	8	2	2	2	2	2	2	2	2	2	2	3	3
NASA (Orig. SPIHT)	4	4	8	8	8	8	8	2	2	2	3	2	2	2	2	2	3	3	3

The output from the hardware implementation gave similar compression results to our study of the original and modified SPIHT algorithm.

When statistical modeling was used in combination with the arithmetic coder, the output file was larger than the input file for both the original and modified SPIHT algorithms. This was only the case with SPIHT encoded images. When an uncompressed PGM image was compressed using both statistical modeling and arithmetic coding, a better compression rate was achieved than with just the arithmetic coder. It is still a little unclear as to why this is. This may be due to the fact that SPIHT images have little to no inter-symbol correlation.

6 Conclusion

The added compression of SPIHT encoded images using arithmetic compression was generally poor using the original and modified SPIHT algorithm. Low SPIHT bit rates got the best compression results with arithmetic compression. If low bit rates are going to be used for the transmission of images, then a hardware version of an arithmetic coder may prove useful. However, if high bit rates are going to be used, then a hardware implementation of an arithmetic coder will be of little benefit. Future work might involve looking at symbol lengths above 16 bits and how further modifications to the SPIHT algorithm effect arithmetic compression levels. Also, the results of compressing each bit-plane individually could give some insight to the differing compression rate between low and high bits/pixel encoding rates.

7 References

[1] Ian H. Witten, Radford M. Neal, and John G. Cleary. Arithmetic coding for data compression. *Communications of the ACM*, 30(6):520-540, June 1987.

[2] Glen G. Langdon, Jr. An introduction to arithmetic coding. *IBM Journal of Research and Development*, 28(2):135-149, March 1984.

[3] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379-423 and 623-656, 1948.

[4] Normand Abramson. *Information Theory and Coding*. McGraw-Hill Book Company, Inc., New York, NY, 1963.

[5] Alistair Moffat, Radford M. Neal, and Ian H. Witten. Arithmetic coding revisited. *ACM Transactions on Information Systems*, 16(3):256-294, July 1998.

[6] Mark Nelson. Arithmetic Coding + Statistical Modeling = Data Compression. *Dr. Dobb's Journal*, February, 1991. http://www.dogma.net/markn/articles/arith/part1.htm.

[7] Thomas W. Fry. Hyperspectral Image Compression on Reconfigurable Platforms. Master's Thesis, University of Washington, 2001.

[8] Visible Earth. National Aeronautic and Space Administration. http://visibleearth.nasa.gov/.

[9] A. Said, W. A. Pearlman, "A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, pp 243 – 250, June 1996.