CS-1992-9

# Lossless Compression of Region Edge Maps

Stephen R. Tate

Department of Computer Science
Duke University
Durham, North Carolina 27706

# Lossless Compression of Region Edge Maps

Stephen R. Tate*

Department of Computer Science
Duke University
Durham, NC 27706

**Abstract**

In this note, we describe a lossless compression technique that has been applied to bitmaps defining regions of images. The results show considerable improvement over previous methods. The basic technique is to use context-based statistical modeling fed into an arithmetic coder.

## 1 Introduction

A method for lossy compression that has recently been examined by Novik and Tilton [2] can be explained as follows: The image is divided into regions of relatively uniform intensity, and a description of region boundaries as well as the average intensity of each region is stored. The quality of the compressed image is related to how the phrase "relatively uniform intensity" in the preceding sentence is defined. The method chosen by Novik and Tilton for identifying regions is an incremental procedure in which each pixel starts off in its own region, and then neighboring regions are combined if the difference in their intensity levels is below a certain threshold. Such a method is ideally suited for grid-connected SIMD machines, such as the MassPar used by Novik and Tilton.

The compressed image produced by this scheme consists of two files: the edge map file and the region intensity file. Lossless compression of either of these files improves the effective

*This work was performed while the author was visiting NASA/CESDIS, and was partially supported by NASA subcontract 550-63 of prime contract NAS5-30428.

compression ratio of the original lossy method. In this note, we are concerned with lossless compression of the edge map file — the file defining the borders of the uniform intensity regions. The results we obtain show that standard techniques of lossless compression can be adapted to the special properties of the edge maps to obtain improved compression ratios.

## 2 Compressing the Edge Maps

Regions in an image can be entirely described by two bits per pixel location. In effect, these two bits answer the questions "Is the pixel to the south in the same region as me?" and "Is the pixel to the east in the same region as me?". Clearly, these two bits of information completely describe the image regions.

The raw format for the edge map file uses a byte for each pixel location, with only the least significant two bits used for the information described above. Clearly, there is a lot of waste in this representation, and 4:1 compression can be achieved by simply packing the bits so that room is not wasted. Interestingly enough, packing the bits first and then running the result through the standard compressors produced larger files than compressing the wasteful raw format (probably due to the larger symbol alphabet of the packed representation).

The compression method we have designed for the edge map files is a simple adaptive context modeler fed into an arithmetic coder. The basics of context modeling and arithmetic coding can be found in the book by Bell, Cleary, and Witten [1]. For a pixel location's context, we use the pixel locations immediately above and to the left of the current location. The context is shown pictorially in Figure 1, where "X" marks the current pixel location, and the other pixels denote previously coded values that can be used as context. Since there are four locations used for the context, and each value is two bits, the context is uniquely specified with 8 bits, for 256 different context values. It is an important property of edge maps that we are allowed to have such meaningful contexts with a relatively small number of different possible contexts (so that the gathered statistics are reliable). The modeler keeps track of how many times each pixel value has occurred in each context, and these counts are

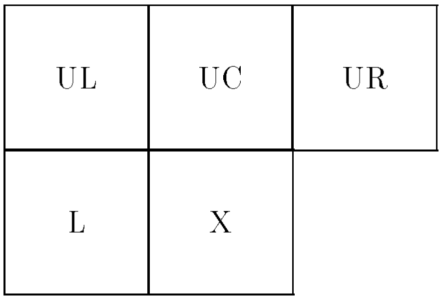


Figure 1: Context locations for encoding pixel X.

used as the prediction probabilities for the arithmetic coder.

A problem common to all adaptive coders is the "zero-frequency problem". To be able to encode all possible input files, we must allocate a non-zero portion of the prediction probabilty to each symbol, even those with zero counts. There are many ways of handling this problem, the simplest being to initialize all symbol counts to one, thereby avoiding zero counts. In this scheme, a symbol $z$ that occurs in context $c$ is has an estimated probability of

$$\mathsf{Prob}(z|c) = \frac{\mathsf{count}(z|c) + 1}{\mathsf{count}(c) + |A|},$$

where $\mathsf{count}(z|c)$ is the total number of times that character $z$ has occurred in context $c$, $\mathsf{count}(c)$ is the total number of times that context $c$ has occurred, and $|A|$ is the size of the symbol alphabet.

For small data sets, if might be useful to value the accumulated counts more than the original estimation used to overcome the zero-frequency problem, so we tried the formula

$$\mathsf{Prob}(z|c) = \frac{\lambda\mathsf{count}(z|c) + 1}{\lambda\mathsf{count}(c) + |A|},$$

where $\lambda$ is a weight factor that we give to the counts. Through experiments, values of 4 or 5 for $\lambda$ gave the best compression ratios, so we selected a value of $\lambda = 4$ for the final compression tests.

| | e00 | e01 | e02 | e03 | e04 |
|---|---|---|---|---|---|
| 2D Context | 10384 | 30185 | 29220 | 18964 | 9519 |
| LZC | 13053 | 38327 | 45408 | 37245 | 24571 |
| WNC | 11355 | 35219 | 43671 | 37699 | 25990 |
| ADAP | 11359 | 35112 | 41927 | 35091 | 23378 |
| LZRW3A | 30818 | 69828 | 81299 | 67376 | 46001 |
| Lossless JPEG | 20852 | 54909 | 59479 | 48278 | 31054 |

Table 1: Compressed file size for 5 sample edge map files.

For a coder, we simply used the publically available fast arithmetic coder written by Radford Neal, and obtained via anonymous ftp from Internet host `fsa.cpsc.ucalgary.ca`.

## 3 Experimental Results

In this section we present the results for our two dimensional context modeler. Five sample edge map files were used, all of which were produced from the same original image (a 368 by 468 Landsat Thematic Mapper image of the Ridgely area), but with various quality thresholds. The highest quality image file is labeled e00 and the quality decreases in the progression of files that ends at e04. According to Tilton, the quality represented by e04 is typical of the reconstructed image quality desired in his study.

For comparison, we compressed all the files with the various compression methods available in the "crush" compression package, which includes LZC (the standard UNIX compress utility), WNC (the Witten, Neal, and Cleary arithmetic coder), ADAP (an arithmetic coder with an adaptive model), and LZRW3A (a fast variant of Lempel-Ziv compression due to Ross Williams). In addition, we used the lossless JPEG compression method in an attempt to draw out some of the two dimensional dependencies in the data. For all test files, our new compression program attained higher compression ratios than any of the previous methods. The compressed file sizes are shown in Table 1. Notice that the savings on the file e04 is particularly impressive, and that this is the file described as typical by Tilton.

| | e00 | e01 | e02 | e03 | e04 |
|---|---|---|---|---|---|
| 2D Context | 3.25s | 3.51s | 3.65s | 3.61s | 3.55s |
| LZC | 0.56s | 0.93s | 0.98s | 0.86s | 0.71s |
| WNC | 4.13s | 5.08s | 5.46s | 5.21s | 4.75s |
| ADAP | 4.66s | 5.68s | 5.88s | 5.61s | 5.21s |
| LZRW3A | 0.46s | 0.71s | 0.75s | 0.63s | 0.56s |
| Lossless JPEG | 3.08s | 6.40s | 6.91s | 5.68s | 4.01s |

Table 2: Encoding times for the 5 sample edge map files.

| | e00 | e01 | e02 | e03 | e04 |
|---|---|---|---|---|---|
| 2D Context | 4.45s | 4.70s | 4.63s | 4.70s | 4.66s |
| LZC | 0.31s | 0.46s | 0.46s | 0.46s | 0.40s |
| WNC | 5.65s | 6.66s | 7.13s | 6.83s | 6.26s |
| ADAP | 6.25s | 7.25s | 7.56s | 7.23s | 6.71s |
| LZRW3A | 0.18s | 0.28s | 0.30s | 0.25s | 0.26s |
| Lossless JPEG | 2.48s | 4.90s | 5.23s | 4.36s | 3.15s |

Table 3: Decode times for the 5 sample edge map files.

The speed of encoding and decoding for the 2 dimensional modeler is comparable to the previous methods based on arithmetic coding. While the speed cannot compete with the faster Lempel-Ziv family of compressors, it is clearly within reasonable limits. For exact encoding and decoding speeds, see Tables 2 and 3.

# 4 Conclusion

The ideas behind our two dimensional modeler are not entirely new, but the results here clearly show the importance of chosing the correct method for the data to be compressed. Since the edge map data doesn't conform to typical grey-scale image characteristics, image compression techniques such as the lossless JPEG algorithm do not perform well, and one-dimensional coders fail to take advantage of the two-dimensional dependencies in the data.

By combining two-dimensional contexts with adaptive text-compression techniques, we have achieved compression ratios greater than any of the previously available methods.

## References


[1] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice-Hall, Englewood Cliffs, New Jersey, 1990.

[2] D. A. Novik and J. C. Tilton. "Adjustable Lossless Image Compression Based on a Natural Splitting of an Image into Drawing, Shading, and Fine-Grained Components", Space and Earth Science Data Compression Workshop, 1992.