FREQUENCY DOMAIN HUFFMAN CODING WITH WEIGHTED PIXEL COSTS FOR IMAGE COMPRESSION AND TRANSMISSION

Zhenhuan Sun, Ziyi Xu

z98sun@uwaterloo.ca - z297xu@uwaterloo.ca
University of Waterloo
Department of Electrical and Computer Engineering
200 University Ave W, Waterloo, ON N2L 3G1

ABSTRACT

A new Huffman coding based lossy image compression method for stable image transmission is introduced in this paper. To ensure effective image transmission in a noisy environment, both high image compression ratio and robustness such as resistance to error are essential. Our proposed method begins with segregation of magnitude and phase spectra of images, followed by compressing the most informative region on both spectra with our modified Huffman encoding method. Various image transformations can be applied to acquire both spectra. The purpose of transformation is to identify and crop the region in which pixels with more significance and priority are concentrated. Huffman coding as our encoding method is modified in a way such that the cost of encoded image data is minimized to assure less error-prone and prioritized transmission of more important pixels. Simulation results of proposed method has shown that more important pixels are encoded with fewer bits while overall compression has preserved a good compression ratio and reasonable reconstruction accuracy.

Keywords: Information theory, Huffman coding, image compression, image transmission

1. INTRODUCTION

In the field of image compression, only two types of methods are concerned, lossless and lossy. Lossless image compression refers to methods which are able to reconstruct compressed images without any theoretical error, but often have unimpressive compression ratios, such as entropy coding [1]. On the contrary, lossy image compression methods reconstruct compressed images with errors but usually with very high compression ratio. Our proposed method falls in the category of lossy image compression, but a modified lossless entropy coding method based on Huffman coding is used to encode pixels with more importance and priority.

The motivation of our algorithm starts with the awareness of irrationality of assigning pixels with equivalent importance. The significance of each pixel may not be obvious in spatial domain, however various image transform methods such as Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT) can pin down the distribution of image energy in frequency domain, in which case region of pixels which is more important than others can be easily detected. In frequency domain, one can see that pixels are not equally important since most image energy is concentrated on low frequencies, the consequence or cost of losing pixels in the low frequency region is more severe than losing pixels in high frequency region. On that account, if pixels that contribute a lot to image quality are encoded with codewords with a large number of bits, there is a higher probability that these codes will be contaminated with errors during transmission, which sequentially compromises on quality of image reconstruction. Therefore, we intend to find a method which encodes pixels according to their significance, meaning more important pixels are encoded with fewer bits while less important pixels are encoded with more bits. In this manner, important pixels are prioritized during transmission and become less erroneous, but at cost of exposing less important pixels to errors. However, errors on less important pixels are tolerable since they do not contribute too much to image quality.

2. HUFFMAN CODING WITH UNEQUAL PIXEL COSTS

Huffman coding as a lossless entropy coding method has attracted our attentions for its simplicity and customization ability [2]. With the knowledge of source probability distribution, it results in optimal prefix code with minimum average codeword length [3]. However, the resulting codewords length are based on the occurrence of random variable's value, for example, symbols or pixels that are more likely to occur will be encoded with less bits compared to symbols that are less likely to occur. Despite that the structure of Huffman coding is against our intuition of encoding more important symbols with fewer bits, we have found a way to modify the Huffman coding method to yield optimal prefix codes with minimum average cost instead of length.

In order to derive formula for this variation of Huffman coding, one must first recall how to find the optimal prefix codes in terms of length under certain assumptions. Let $X \in \mathcal{X}$ be a random variable which draws values from alphabet $\mathcal{X} = \{x_1, x_2, ..., x_m\}$, the probability that X equals to any values in its alphabet is denoted as p_i and the length of each codeword for x_i is denoted as l_i, where $i = \{1, 2, ..., m\}$. Then the question of finding the optimal D-ary code with minimum average length can be formalized as a minimization problem:

$$\begin{aligned} \min_{l_1,...,l_m} \quad & \sum_{i=1}^{m} p_i l_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} D^{-l_i} \leq 1 \\ & l_i \in \mathbf{Z}^+ \end{aligned} \tag{1}$$

the optimization constraints are coming from Kraft inequality which is the necessary and sufficient condition for prefix code [3] and the length of codewords must be positive integers. This optimization problem can be easily solved if we relax the integer constraint on length to real numbers. Then optimal solution will have equality in Kraft inequality [3] since if equality does not hold, decreasing l_i will always improve optimization objective. Once inequality constraint has become equality, Lagrange multiplier can come in handy to solve the relaxed problem. The result of this optimization problem is

$$l_i^* = -\log_D p_i$$

noting that this is the solution without integer constraint, so in reality the ceiling $\lceil -\log_D p_i \rceil$ is used. To find the optimal code using Huffman coding method, the source distribution p_i is used to construct the Huffman tree.

2.1. Derivation

To modify the existing optimization problem such that we can obtain optimal prefix code with minimum average cost, we first constructed a weight function $w : \mathcal{X} \longrightarrow \mathbf{R}$ which assigns each pixel value $(e.g., x_1, x_2, ..., x_m)$ in the alphabet of X a different weight $(e.g., w_1, w_2, ..., w_m)$ according to their importance. Then weights are appended to objective function so that the minimization goal will become weighted average length which is the average cost. The resulting relaxed optimization problem can be formalized in this way:

$$\begin{aligned} \min_{l_1,...,l_m} \quad & \sum_{i=1}^{m} w_i p_i l_i \\ \text{s.t.} \quad & \sum_{i=1}^{m} 2^{-l_i} = 1 \end{aligned} \tag{2}$$

in this optimization problem we chose $D = 2$ for the purpose of finding binary prefix code. Our objective here is to find the weighted distribution on source which results in optimal codeword with minimum cost. The weighted distribution will then be applied to construct Huffman tree.

Although this optimization problem can be solved by standard Lagrange multiplier method, a different approach from information theory is considered here to show that the probability mass function (PMF) that we constructed is indeed the weighted distribution resulting in the optimal solution.

Assuming the optimal solution of this optimization problem that satisfies the Kraft equality has distribution $a_i = 2^{-l_i^*}$, which means the probability of generating a codeword with length l_i^* is a_i [3]. We construct a probability mass function (PMF) as:

$$q_i = \frac{w_i p_i}{\sum_{i=1}^{m} w_i p_i}$$

Given these two notations we can rewrite the objective function in (2) as:

$$\begin{aligned} \sum_{i=1}^{m} w_i p_i l_i &= \sum_{i=1}^{m} w_i p_i \log_2 \frac{1}{a_i} \\ &= \sum_{i=1}^{m} w_i p_i \log_2 (\frac{q_i}{a_i} * \frac{1}{q_i}) \\ &= \sum_{i=1}^{m} w_i p_i \log_2 \frac{q_i}{a_i} - \sum_{i=1}^{m} w_i p_i \log_2 q_i \end{aligned} \tag{3}$$

since q_i is a probability mass function then $\sum_{i=1}^{m} q_i = 1$, we can multiply $\sum_{i=1}^{m} q_i$ to (3) without changing anything.

$$\begin{aligned} \sum_{i=1}^{m} w_i p_i l_i &= \sum_{i=1}^{m} w_i p_i \log_2 \frac{q_i}{a_i} - \sum_{i=1}^{m} w_i p_i \log_2 q_i \\ &= \sum_{i=1}^{m} w_i p_i \sum_{i=1}^{m} q_i \log_2 \frac{q_i}{a_i} - \sum_{i=1}^{m} w_i p_i \sum_{i=1}^{m} q_i \log_2 q_i \\ &= \sum_{i=1}^{m} w_i p_i D\left(q \| a\right) + \sum_{i=1}^{m} w_i p_i H(q) \end{aligned} \tag{4}$$

where $D\left(q \| a\right)$ is Kullback–Leibler divergence between distributions q and a which measures the difference between two distributions, and $H(q)$ is the discrete information entropy of distribution q. From the definition of entropy, $H(q)$ does not depend on l_i, thus to minimize the objective function the best we can do is setting $D\left(q \| a\right)$ to 0, which means q and a are two identical distributions. Therefore, when $a_i = q_i = \frac{w_i p_i}{\sum_{i=1}^{m} w_i p_i}$, the optimal solution for this modified optimization problem is

$$l_i^* = -\log_2 \frac{w_i p_i}{\sum_{i=1}^{m} w_i p_i}$$

It concludes our proof that q_i is the best weighted distribution on source that will result in optimal prefix code with minimum average cost. Thus distribution q_i will be used in

our following algorithm to construct Huffman tree and optimal code for every pixels.

2.2. Algorithm Overview

Our algorithm begins by separating original image to its magnitude and phase components using Discrete Fourier Transform (DFT). The magnitude spectrum is used to determine the most informative region, however the phase spectrum may not be as useful as magnitude spectrum in terms of selecting important pixels, but it contains phase shift information that are sensitive to human eyes, so we need both magnitude and phase components to reconstruct images. Once the important regions are detected, crop operations are performed to these regions in both magnitude and phase spectra, and histogram analysis is conducted to determine the source distribution p_i on cropped images. Weight function is then constructed in a manner that user-defined important pixels will receive higher weights than others, and weighted source distributions q_i is computed using weight function for magnitude spectrum. Previously discussed Huffman coding method is applied to the cropped image to obtain binary sequence of codewords. The codewords are transmitted and corresponding Huffman decoding procedures are applied to retrieve the original magnitude and phase spectra. Finally, original image is reconstructed by applying Inverse Discrete Fourier Transform (IDFT) to complex plane $|X|e^{j\angle X}$ with magnitudes $|X|$ and phases $\angle X$, where X denotes original image.

2.3. Choice of Weight Function

Weight function as a user-defined function in our algorithm has its subtleties. The definition of importance among pixels varies from application to application, different applications may define a same pixel with completely different importance. Here, we list a few weight functions for some common pixel importance definition.

In contrast enhanced magnitude spectrum, when pixels with bright visual effect or intensity value close to 255 are considered important, the following functions can be used:

- Square weight function: Assigning all pixels with intensity values greater than a given threshold a fixed weight and a weight 0 for every other pixels.
- Gaussian weight function: 1D Gaussian distribution with a mean 255 and a user-defined variance is considered to assign weights in ascending order.

If frequency of occurrence is considered as definition of pixel importance, the following function can be used.

- Distributive weight function: Assigning higher weight to pixel values that are more likely to occur according to image source probability distribution.

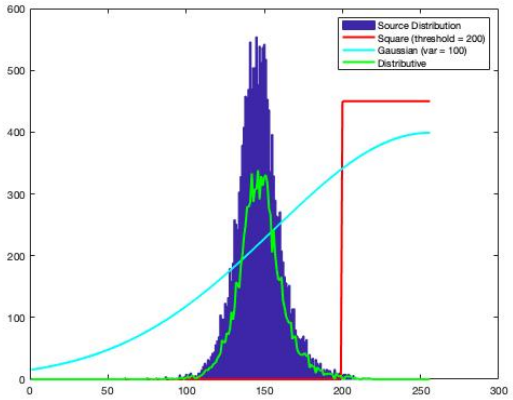

Fig. 1. Different Weight Functions Comparison for Same Source Distribution

3. SIMULATION RESULTS

The objectives of this simulation are to find out if our modified Huffman coding algorithm can indeed encode pixels according to their importance and evaluate the performance of proposed algorithm.

Table 1. Average Codeword Length for Important Pixels Only(All Pixels*) Using Distributive Weight Function

Importance	Modified Huffman	Huffman
Probability $\geq 1\%$	5.6000(53.1406)	5.5714(47.3828)
Probability $\geq 1.5\%$	5.2414(53.1406)	5.3448(47.3828)
Probability $\geq 2\%$	4.8636(53.1406)	5.1364(47.3828)
Probability $\geq 3\%$	4.3333(53.1406)	5(47.3828)

Table 2. Average Codeword Length for Important Pixels Only(All Pixels*) Using Gaussian Weight Function

Importance	Modified Huffman	Huffman
Intensity ≥ 200	31.5439(47.3672)	31.7544(47.3828)
Intensity ≥ 210	32.9574(47.3672)	33.0851(47.3828)
Intensity ≥ 220	32.9459(47.3672)	32.9730(47.3828)
Intensity ≥ 230	27.9259(47.3672)	27.9630(47.3828)

Table 3. Average Codeword Length for Important Pixels Only(All Pixels*) Using Square Weight Function

Importance	Modified Huffman	Huffman
Intensity ≥ 200	22.9298(121.7031)	31.7544(47.3828)
Intensity ≥ 210	24.1915(121.7031)	33.0851(47.3828)
Intensity ≥ 220	24(121.7031)	32.9730(47.3828)
Intensity ≥ 230	19(121.7031)	27.9630(47.3828)

*: Average Codeword Length for All Pixels.

It can be seen from table 1, 2, and 3, modified Huffman coding method has in fact reduced the number of bits required to encode pixels that are defined to be important. For example, in table 3, if pixels with intensity value greater than 230

are considered important, then our modified Huffman coding method can encode those pixels with 19 bits while standard Huffman coding needs 27.9630 bits. Note that the average codeword length in table 1 is much smaller than the length in table 2 and 3, which is expected since even with weight-adjusted distribution, pixels with probability of occurrences greater than 3% are still more likely to occur than pixels with intensity value greater than 200, thus pixels in table 1 require fewer bits to encode by the nature of Huffman coding. Regardless of the definitions of important pixels, our modified Huffman coding algorithm has managed to encode important pixels with fewer bits than the standard Huffman coding. Different weight functions reduced the average codeword length for important pixels by different amount, for example, square weight function has the most significant bits reduction on important pixels, however the average codeword length for all pixels is increased a lot from 47.3828 to 121.7031 in order to compensate for the bits reduction. On the flip side, Gaussian weight function not only reduces the codeword length (albeit insignificantly) for important pixels, but it has also managed to reduce the codeword length for all pixels by a very small amount from 47.3828 to 47.3672. It is also worth noting that, although standard Huffman coding method has already encoded pixels with high probabilities of occurrence with fewer bits, our modified Huffman coding with distributive weight function can encode those pixels with even fewer bits, but again at cost of increasing average codeword length for all pixels. We can conclude that modified Huffman coding algorithm indeed achieves the effect of encoding important pixels with fewer bits, but sometimes average codeword length for all pixels are increased to compensate for that effect.

The result of compression using proposed algorithm has shown in figure 2. We have tested the performance of using different weight functions on Lenna image's magnitude spectrum, and we have found that Gaussian weight function results in the best compression ratio and overall quality. We used a uniform weight function on the phase spectrum since all pixel values seem to be uniformly distributed, so it makes sense to assign equal weights to them.

Table 4. Quality Metrics for Reconstructed Image

MSE	SNR	Compression Ratio
73.7925	48.1172	9.2177:1

Table 4 and figure 2 have shown that reconstructed image have great compression ratio while persevering reasonable image quality.

4. NOVELTY

Our proposed algorithm was based on the intuition that pixels in images should not be treated equally. We think it is natural to take extra care to pixels with more significance. Given a user-defined pixel importance, more important pixels are encoded with fewer bits, whereas less important pixels are encoded with more bits. There are several benefits of doing this.

(a) Original Image (b) Reconstructed Image

Fig. 2. Original and Reconstructed Image Comparison

The first benefit is to give important pixels more resistance to error during transmission. For a noisy discrete memory-less channel, errors during transmission are inevitable, however codewords with more bits are more likely to contaminate with errors during transmission since they need to spend more time on the channel and will have more opportunities to deal with errors because of their length. In opposition, codewords with less bits are less likely to catch an error and even if they do, their resistance to error is much higher than codewords with more bits, because they usually have narrower value range. For example, a 2-bit binary code has value range between 0 and 3, so even if there is an error during transmission, the codebook can recover transmitted codeword by finding the closest value of received erroneous codeword, which is impossible for codewords with very long length. In this way important pixels are more protected during transmission, and less important pixels are more exposed to errors, which is acceptable as less important pixels usually do not contribute too much to image quality. In addition to error resistance, compression ratio can be improved significantly if there is a way to extract important pixels only from images. Although in our proposed method a simple crop operation is used to extract important region of pixels, there are still many insignificant pixels in the cropped region which are encoded with many bits. If we can extract important pixels only, for example, the cross area with bright pixels, then we might be able to further improve the compression ratio and image quality.

5. CONCLUSION

In this paper we have introduced a new image compression method which encodes pixels according to their significance to ensure less error-prone transmission of important pixels. Simulation results have shown that proposed method indeed encodes important pixels with fewer bits than before, however the bits reduction on important pixels by different weight functions has in various degree influence on average codeword length for all pixels. Despite the fact that proposed method may come with side effect, nevertheless, it still has space for improvement and great potential.

6. REFERENCES

[1] David JC MacKay, David JC Mac Kay, et al., *Information theory, inference and learning algorithms*, Cambridge university press, 2003.

[2] David A Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[3] MTCAJ Thomas and A Thomas Joy, *Elements of information theory*, Wiley-Interscience, 2006.