On Improving JPEG Entropy Coding by means of Sub-Stream Extraction

Youngjin KIM†(a), Hyun Joon SHIN†(b), Nonmembers, Jung-Ju CHOI††(c), Member, and Youngcheul WEE††(d), Nonmember

SUMMARY We introduce an entropy coding method to enhance the compression efficiency of JPEG. Because run-length coding and early-termination work more effectively for longer zero sequences, we extract ones and negative ones from the coefficients and reduce the magnitude of all coefficients by one. The extracted coefficients are encoded with a designated entropy coding method. The proposed method can transmit images in two parts progressively, where the first contains JPEG-compatible image with a small amount of degradation and the second is used to add fine details. Our method improves the compression ratio by more than 5% without sacrificing the efficiency of JPEG.

key words: JPEG, entropy coding, sub-stream extraction

1. Introduction

In this letter, we propose an entropy coding method to improve the JPEG standard. JPEG is one of the most widely used image compression methods thanks to its high compression ratio for image quality. Although there are many alternatives that provide better compression ratios, including JPEG 2000 and the i-frame coding method of H.264/AVC [1], [2], JPEG is the most popular standard because it provides a very efficient way to encode and decode images. The requirement of computational efficiency makes improving the compression ratio of the JPEG standard challenging.

Many recent studies have suggested methods to improve the JPEG compression ratio [3]–[5] in a variety of ways, including DC prediction direction changes and inter-block coefficient predictions. Many of them focus on improving the entropy coding phase to enhance the compression ratio while maintaining the image characteristics [6]–[10]. A noteworthy approach was introduced by Lakhani [7], in which the JPEG compression ratio was improved by pairing the zero-run with the preceding non-zero coefficient, achieving the highest compression ratio. However, these methods typically could not preserve the computational efficiency of the JPEG standard, as they required extra computations such as the generation of a per-image Huffman table. Later, an arithmetic coding method was adopted to improve the compression effectiveness. It was added to the JPEG standard [11], but it has not been adopted in many real-world applications due to the complexity of its implementation and computational routines.

Our primary goal is to modify the entropy coding of JPEG to improve the compression ratio while preserving the computational efficiency of the JPEG encoding and decoding schemes. It is well known that JPEG compresses images more effectively when there are more consecutive zeros or when the magnitudes of the AC coefficients are relatively small. Based on these observations, we extract sequences of small values (zero, one, and negative one) from the AC coefficients of a JPEG block. The extracted sequences are then replaced with zero-runs. The sequences of the small values are coded separately with a designated method. This scheme allows us to transmit the image in a two-pass manner: the large AC coefficients are transmitted first and formed into an image with only a small amount of degradation, after which the small coefficient sequences are transmitted to improve the image quality. Moreover, the former part can be decoded using standard JPEG decoders without any modification.

In Sect. 2, the details of the proposed approach are described. Section 3 exhibits the experimental result of the proposed method compared to an earlier method in terms of the compression and computational efficiency. Finally, we provide a summary of this work and directions for future research in Sect. 4.

2. Sub-Stream Extraction

In the JPEG standard, a block of $8 \times 8$ pixels of an image is transformed by DCT (discrete cosine transform). The resulting coefficients are quantized and reordered [12], [13]. The later parts can be viewed as processes that decrease the magnitude of the values to encode while increasing the number of consecutive zeros such that the subsequent run-level coding and Huffman coding routines work more effectively [14], [15]. Therefore, by further reducing the magnitude of the coefficients and increasing the length of zero-runs, we can improve the compression ratio of JPEG. To do this, our basic idea is to extract small values, which are ones and negative ones in this letter, from the sequence of AC coefficients and to encode them separately.

Figure 1 illustrates our extraction process. In the AC

Manuscript revised July 2, 2012.
Nonmember

†The authors are with the Department of Information and Computer Engineering, Ajou University, Suwon 443–749, Rep. of Korea.
††The authors are with the Department of Digital Media, Ajou University, Suwon 443–749, Rep. of Korea.
a) E-mail: vzriny@ajou.ac.kr
b) E-mail: joony@ajou.ac.kr
c) E-mail: jungju@ajou.ac.kr
d) E-mail: ycwhee@ajou.ac.kr
DOI: 10.1587/transinf.E95.D.2737
coefficients of a block, we first find the ones and the negative ones. These ones and negative ones are taken out from the original AC coefficients and formed into two separate sequences: an internal sequence and a tailing sequence. To specify the location and the length of the extracted sequences in the remaining coefficients, the positions of extracted values in the original sequence are marked with zeros. As in the JPEG standard, the tailing zeros are deleted. Here, we also extract zeros found between the ones and negative ones such that we can put the ones and negative ones in their proper positions in the original sequence when decoding.

As a result, we have the sequence of the remaining coefficients, which is to be encoded by the standard JPEG encoding method, as well as the two (internal and tailing) one-zero sequences, each of which is to be compressed using the designated method. The first part can be decoded independently from the others with the standard JPEG encoder while degrading the quality only slightly. The major reason behind the choice of the Huffman-based JPEG standard coding method for the remaining sequence instead of advanced entropy coding methods such as Golomb-Rice coding is that the Huffman-based method is very efficient for both encoding and decoding. Hence, this part can maintain compatibility with JPEG.

After this extraction process, the remaining sequence has two positive characteristics: it has longer zero-runs and does not have ones or negative ones. Given that JPEG adopts a run-level encoding method, in which the consecutive zeros and the level of the following non-zero value are encoded with a single code, longer zero-runs can help to improve compression ratio drastically.

On the other hand, the absolute values of the remaining coefficients are either larger than one or equal to zero. This allows us to improve the compression ratio further by optimizing the Huffman table used in AC coefficient encoding. In JPEG, one and negative ones appear most often and are thus assigned with the shortest code values. Because the remaining sequence does not have those values, the codes for the other coefficients can be shortened by the Huffman table optimized for those sequences.

We can achieve a similar effect without time-consuming per-image Huffman table optimization. By reducing the magnitudes of the remaining AC coefficients by one and using the standard JPEG Huffman table, we can assign shorter codes to the AC coefficients in many cases. In our experiment, we found that AC magnitude reduction improved the compression ratio by 1%.

As mentioned earlier, the extracted sequences are classified into two groups, internal sequences and tailing sequences, because they have different characteristics. The former sequences are from the middle of the blocks and their values appear rather randomly in these sequences. On the other hand, the later sequences, extracted from the ends of the blocks, have relatively long sequences of consecutive zeros compared to the others. We concatenate the sequences in each group from all blocks in the image to form two long streams.

2.1 Internal Stream Compression

The internal stream contains zeros, ones, and negative ones. In most images, zeros occur more often than the other values. Therefore, we explicitly assign codewords to the values: 0 for zero, 2 for one, and 3 for negative one (see Fig. 2). Note that these codewords are identical to what one would obtain using the optimal Huffman tree. However, this explicit code-assignment helps to accelerate the encoding and decoding of the data, as an explicit Huffman table is not required.

2.2 Tailing Stream Compression

A tailing stream, on the other hand, usually has a longer sequence of consecutive zeros. Based on this observation, we can reduce the size of the code by applying a run-length encoding method to these streams. Run-length encoding assigns one codeword to a consecutive sequence of zeros. Inspired by the original JPEG encoding method, we assign a codeword to each pair of the length of consecutive zeros (run) and the sign of the following value (sign), as shown in Fig. 3.

The codeword for each run-sign pair is determined by building an optimal Huffman tree. The corresponding Huffman table can be either stored in a file or fixed for every image. In an extensive experiment, we found that the size of the Huffman table is usually larger than the gain we could achieve by optimizing the table for an image. Therefore, we use a fixed table hard-coded into the encoder and the de-
coder. To build the Huffman tree covering a wide range of images, we collect streams from a variety of standard images and measured the frequency of each run-sign pair.

In contrast to internal sequences with which length can be determined by the length of the zeros in the remaining part of the large coefficients, the length of a tailing sequence cannot be determined from other data; thus, an EOB (end of block) mark is required. Instead of storing the EOB, which can degrade the compression ratio, the count of non-zero coefficients in the tailing stream of each block is stored separately using Huffman encoding with a fixed code table. At the decoding time, the length information is deflated one by one and is used to decode the tailing stream of each block.

3. Experimental Results

We verified the performance of the proposed method using five standard images: Lenna, Baboon, Airplane, Sailboat, and Peppers. All images were initially in gray-scale. Table 1 shows the compression ratios of the standard JPEG method, the optimal Huffman coding of DCT blocks (OHCDB) approach in [7], and our method with the magnitude reduction process. Here, JPEG implementation from the Independent JPEG Group (IJG) was used as the standard JPEG encoder. We obtained DC and AC coefficients from IJG using 75 as the quality factor, and then, compressed them using IJG, OHCDB, and the proposed method. The values provided in the table are the sizes of the compressed files in bits and the ratio of improvement from the standard JPEG method. As shown in the table, the compression ratio of the proposed method is better than that of the JPEG standard by more than 5%, and only slightly worse than that by the OHCDB method.

Table 2 shows the time required to compress and decompress the Lenna image using IJG, OHCDB, and the proposed method in a variety of environments, including Microsoft Windows (compiled using Microsoft Visual Studio), Linux virtual machine (compiled using GNU C compiler), and a mobile device equipped with a Cortex-A9 CPU running at 1.2 GHz with 1 GB memory (cross-compiled using GNU C compiler). The image was compressed and decompressed 10 times with each method in each environment. The performances were measured in milliseconds. OHCDB exhibited a noticeable amount of performance degradation. In addition, the performance loss of OHCDB was more apparent on the mobile device. We believe the degradation arose due to the optimal Huffman tree construction. On the other hand, the performance of the proposed method was found to be comparable to that of the JPEG standard.

As stated earlier, the proposed method also supports two-pass transmission in which large coefficients are transmitted first to be decoded using a standard JPEG decoder, after which the zero-one streams are transmitted to give fine details. Table 3 shows the relative ratio of the former and latter parts and the PSNR after each part is transmitted. Note that the PSNR values after the second passes transmitted (the last column) are identical to those of the JPEG standard. As shown in the table, an image degraded by 10% can be constructed with only about 67% of the file after the first transmission. Moreover, this part can be understood by any standard JPEG decoder. Figure 4 shows an image decoded from the first part of the data. Although fine details are missing in the partial image, the images all have the global features of the original versions and can be used as intermediate images in a multi-pass transmission process. The fine details encoded in the extracted stream are merged into the image and are used to produce the final image.

4. Conclusion

In this letter, we proposed a novel approach to enhance
the compression efficiency of the JPEG standard. Based on the characteristics of JPEG entropy coding, we extract small values from the AC coefficients of JPEG blocks and replace them with zeros to improve the compression ratio while maintaining compatibility. The extracted data are encoded with the designated method based on Huffman coding and run-length coding. This separation process allows us to transmit an image via a two-pass approach. Because the proposed method relies on a simple Huffman coding method with global tables, it also preserves the computational efficiency of the JPEG standard while improving the overall compression ratio by more than 5%.

The major drawback of the proposed approach is that we have to apply IDCT (inverse DCT) for each pass. The coefficients of the first pass must be transformed to be built into an image with a small amount of degradation. After decoding the second pass data and adding them to the first pass, all coefficients must be transformed back to form the final image. These steps are, however, computationally costly. Progressive JPEG is not widely used in real-world applications for this reason. In the future, it may be possible to develop a DCT-based image compression method that supports multi-pass transmission without an inverse transform procedure at each pass.

Acknowledgements

This work was partially supported by the Defense Acquisition Program Administration and Agency for Defense Development under the contract number UD060048AD.

References


