Modified JPEG algorithm with Binary Interval Transform coding with improved compression ratio.

Ilya Brailovskiy
Intel, Moscow, Russia
ilya.v.brailovskiy@intel.com
Dmitry Plotkin
Department of Computational Mathematics and Cybernetics
Moscow State University, Moscow, Russia
darksun@mail.ru

Annotation

In this paper we present new image compression algorithm which coincides with JPEG image compression in all parts but in entropy coding. By replacing VLC coding with Binary Interval Transform coding in JPEG coding pipeline we observe up to 15% improvement in compression ratio while encoding time of the new algorithm exceeds the original JPEG encoding time less than 3%.

Keywords: *Image compression, JPEG, entropy coding, Variable Length Coding, Binary Interval Transform coding.*

1. INTRODUCTION

The problem of compression and transmission of multimedia information such as images or video of various nature takes one of the key places in a plenty of application areas like web, consumer video, medicine, etc. In order to meet new requirements such as demands for low power consumptions for mobile devices as well as constant demand in increasing network bandwidth we propose new algorithm for compression of digital images. Proposed algorithm has the same quality as the JPEG [1]. In fact, it differs from JPEG only on entropy coding. At the same time it provides almost the same performance as the original JPEG and 10-15% better compression ratio. In the networking environment computational power grows drastically on both transmitter and receivers sides while the bandwidth stays approximately the same. So if we can "exchange" better compression for the price of more computations it could solve some of the problems with carrying multimedia data over network environment. If we also can manage keeping algorithms simple enough we can use them efficiently not only on desktops or servers but also on mobile clients like cell phones or PDAs. Proposed algorithm as well as developed method meets both these requirements: it allows better compression ratio with the same quality at a price of reasonable increase of computation resources.

Proposed modification of the JPEG is based on replacing Variable Length Coding (VLC) with so called Binary Interval Transformations algorithm. The method of Binary Interval Transformations [2] is a static coder. It's intended for compression of sources without memory. It is characterized by capacity of the alphabet (length of the generalized letter) and order on the "generalized" letters. It is proved, for example, that the method of Binary Interval Transformations delivers better compression ratio [2] than widely known Huffman coding [3] while providing the same level of computational complexity. Main issue in applying Binary Interval Transform to the real data coding is to find the best values for these parameters, see [4] for example. If parameters are not chosen properly the compression results could be quite unimpressive. We should note that the Binary Interval Transforms introduce some delay in coding. But on practice it is not critical at all. Unfortunately there is no theoretical basis for picking up optimal parameters because the method is quite new so we use empirical data. Later in this paper we will mostly discuss the best ways of combining JPEG and Binary Interval Transform algorithm.

We would like to discuss briefly comparison of Binary Interval transform with Arithmetic Coding. Binary Interval Transform and Arithmetic Coding are quite similar from the standpoint of introduced coding delay. However, computational complexity of Binary Interval Transform is less because it contains only "adds" and "shifts" as arithmetical operations while Arithmetic Coding contains "multiplications". Probably because of these extra computational needs Arithmetic Coding algorithm delivers the best possible compression. To some extend proposed method of Binary Interval Transform falls in the between of two algorithms: it provides better compression ratio than Huffman but little slower. It is computationally less extensive compared to Arithmetic Coding but little less efficient in compression ratio. In other words, Binary Interval Transforms deliver good trade-off from practical standpoint. It's almost as fast as Huffman Coding while providing compression rate close to Arithmetic coding. In this paper we mostly focus on JPEG modifications, some more

details on comparison of Interval Transform with Arithmetic Coding can be found in [2].

2. MODIFIED JPEG ALGORITHM

We briefly recall compression stages for image compression according to JPEG Baseline standard in coding order [5]:

- Transition from color space RGB in space YUV, sampling.
- Discrete Cosine Transform, Quantization, Zigzag-scanning.
- Huffman coding.

Proposed modification replaces Huffman coding with Binary Interval Transformations (as it has mentioned before the method has better theoretical compression ratio compared with Huffman coding).

To study this we can define Binary Interval Transformation as follows, a bit broader definition can be found in [2]. We consider all input data as binary input buffer. We split this buffer into some (fixed) number of sequential bits, for example we take each two bits, or each tree bits., etc. We will call each of these bits sequences a letter. The “letter” length is the number of bits in this letter. The coding process is following for the Binary Interval Transform coding. At first, order of letters is defined. For example, for all letters of length 2 it could be: 01, 10, 11, 00. Than we start from the first letter in the alphabet (01 in our example). We calculate intervals (and total number of the intervals) between the sequential inclusions of this letter to the input buffer. This information (number of intervals, intervals and letter itself) goes to the output stream. All inclusions of the letter in the input buffer are removed from the original stream. Modified such way input buffer is considered as input buffer. This process is repeated with other letter until there is no letters in the alphabet. Intervals are stored in binary format using Rice-Golomb coding as in [2] and in [4]. Please note, that for the last letter we are writing out only the number of intervals.

In this paper we fully follow Binary Interval Transforms optimal parameters found earlier [4] for general binary data. We did some experiments, trying to challenge the optimality of found parameters (different letter size, different order of letter) for JPEG but haven’t succeeded in that. We use Binary Interval Transforms for letters with length four bits for the best quality/performance trade-off. Alphabet order which gave us the best compression results is following: letters with 4 ones “1111”, than all with 1 one (like “1000”), next with 2 ones, with 3 one the last is “0000”.

One of the problems which should be solved as a result of this replacement is the task of defining the best way of appending Binary Interval Transformations to JPEG Baseline without VLC. Two ways of such concatenation were considered. The first way is simply submitting the image data after zigzag scan as an input of Binary Interval Transformations. However by doing this we not fully utilize valuable information about zero runs statistics variation over macroblock: more runs closer to the end of the macroblock. The other way of merging these two algorithmic components is following. After zigzag scan the image data is divided into two sequences. First part contains only nonzero elements, and the second translates initial numerical sequence into bits, putting bits "1" in case of a nonzero element of numerical sequence and bit "0" - in case of zero. For example, if we have «12, 0, 3, 4, 0, 0, 0, 0, 5, 0, 8, 0, 0, 1, 0» sequence after zigzag scan. Than two sequences will be formed:

1) first is «12, 3, 4, 5, 8, 1» – all 6 non-zero coefficients.

2) second is: «101100001010010» – 15 bits in summary.

These two sequences are further processed by different statistic coders. First part is encoded with Huffman coding, and Binary Interval Transformations are applied to the second part.

We tested both these approaches on the group of blocks in one line, which means we introduced coding delay on the level of width of the image. Comparison of these two approaches on the well known set of images Waterloo Repertoire [6] consisting of pictures of various nature, has demonstrated that using the second way of appending Binary Interval Transform to JPEG allows 10% better compression compared to the first approach.

The next step in improving compression ratio of the new method is to separate coding DC and AC coefficients in the sequence coded by VLC algorithm. For this purpose two separate tables which are generated automatically in the beginning of processing of each new image.

Tables of DC and AC coefficients are VLC table codes which, as is known, strongly depend on statistics of symbols in the stream. At first, we used the standard tables which are not adapted to the needs of our approach. Thus, the next attempt of improving the compression ratio of the new algorithm became a creation of own custom Huffman tables for DC and AC coefficients. Tables were created on the basis of statistics of frequency of occurrence of the symbols, which the algorithm collects during first path. The given statistics gathered separately for DC and AC tables on corresponding coefficients. Well known that absolute values of macroblocks elements cannot exceed 2048. (This is because of nature of Discrete Cosine Transform which is used at earlier stages). Therefore all area of values from -2048 up to 2048 can be subdivided into 12 intervals for which frequencies of occurrence were considered. Appling these customized tables we’ve built based on collected statistics significantly influenced degree of compression. It has improved in comparison with the previous static algorithm on some files up to 13 %.

We still did not fully utilize one of the main properties of the sequences received after a zigzag scanning of macroblocks - a number of zeros at the end of these sequences. To avoid this redundancy and completely re-use information about zero runs, it

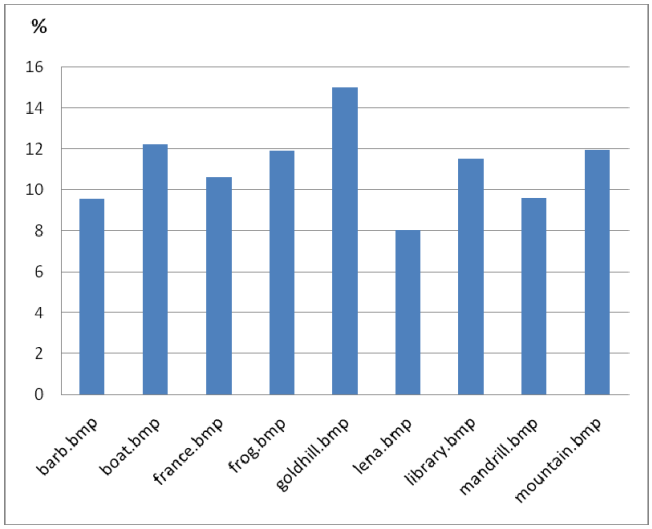
was proposed to use a special symbol which will be put instead of all sequence of zero in the end of the macroblock. Such symbol has been found and applied in Huffman's tables. As the result, the compression ratio has improved in comparison with the prior approach by 20 %.

3. SIMULATION RESULTS

Previously we were comparing our results by evaluating incremental improvements we've made using specific techniques we have considered. In this section we would like to investigate overall efficiency of the proposed JPEG modification.

For the evaluation of proposed methods we took IJG implementation of JPEG [5] and implemented new scheme with the Binary Interval Transform. As it was noticed before, we used famous Waterloo test to evaluate performance of our algorithm [6]. Our experiments proved that most of theoretically predicted results can be achieved. But this simulation also revealed several interesting effects.

Figure 1. Comparison of the new algorithm compression advantage over JPEG baseline on Waterloo test set, quantization parameter 10, in percents



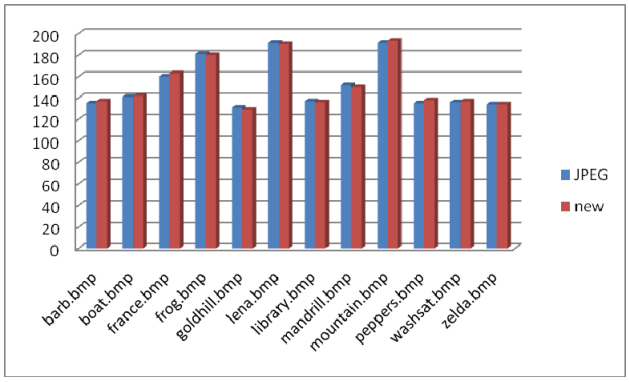
At tirst, we see on the Figure 1 (percents are calcuated by this formula: 100*(JPEG size – NEW size)/NEW size) quite consistent improvments in compression results across all types of images on the Waterloo set. However, the biggest improvments we got on artificail, not photorealistic images, with a lot of zero AC coeffecitents.

Second, quantization parameter also affects the efficiency of proposed method relative to the original scheme. It turned out that proposed algorithms gives more improvements on higher compression ratios. So we did one more thing to improve the new algorithm to be more efficient for small compression ratio (which is usually characterized by high quantization parameters, for example higher than 70) is different technique of choosing the macroblocks for determining the statistics for VLC coding. In the first scheme we took several first sequential macroblocks for the statistics. But if we take these blocks from the different parts of the image it delivers better compression results. For the best results see Table 1.

Table 1. Comparative results for the compression size for the Barb image for different quantization parameters

Quant.	10	30	50	70	90
JPEG	11558	23142	31034	41053	72971
New	11432	23012	30985	40458	72187

Figure 2. Performance simulation results (in milliseconds) for the Waterloo Set with quantization parameter 50. Left column JPEG Baseline, right column is the Modified JPEG algorithm



Third, the performance was measured. All performance results are obtained on Waterloo Repertoire Grayset2 using a system with Intel Celeron CPU 1GHz processor, 240MB RAM, running under WindowsXP. We found out that the performance of proposed algorithm is very close to the performance of the original JPEG scheme, the increase in coding time is always less than 3 msec (less than 3%).

4. CONCLUSIONS

Modified JPEG results have up to 15% better compression ratio compared to the original JPEG with the performance very close to the original JPEG. There are some potential further algorithmic improvments like selecting fixed VLC tables as well some smarter schemes for coding This work could be continued in application for video coding, where JPEG-like compression scheme are quite popular.

5. REFERENCES

1. *Digital compression and coding of continuous-tone still images*. ISO/IEC 10918-1,2,3 JPEG, 1994, 1995, 1997
2. Brailovskiy I.V. "Effective data compression with method of Generalized Interval Transforms", PhD Thesis, Moscow, IVMS RAS, 2003. (in Russian)
3. Huffman D. A "Method for the Construction of Minimum-Redundancy Codes." *Proceedings of IRE*, vol.40, N9, pp.1098-1101, September 1952.
4. Plotkin D.A. Effective data compression with Binary Interval Transformation // *Annual collection of best master thesis*. 2005, CMC MSU; pages. 81-82. (in Russian)
5. Rao K. R., Hwang J. J. "Techniques and Standards for Image, Video, and Audio Coding." Prentice Hall PTR, 1996. P. 1-27.
6. Waterloo Repertoire test suit, available at: http://links.uwaterloo.ca/.

About the author

Ilya Brailovskiy, Ph.D., is a Media Architect for Intel Corporation. His interests include a wide range of visual computing technologies. His contact email is ilya.v.brailovskiy@intel.com.

Dmitry Plotkin is a Ph.D. student at Moscow State University, Department of Computational Mathematics and Cybernetics. His contact email is darksun@mail.ru.