Lossles compression of bilevel images

Cédric Hannotier
channoti@ulb.ac.be

Mathieu Petitjean
mpetitje@ulb.ac.be

***Abstract*—Compression babyyy**

Keywords*—*compression, jpp

I. Introduction

This report presents the implementation of a lossless coder for bilevel image compression. Several techniques are developed and their performances are compared.

II. Techniques description

The different coding schemes that have been implemented rely on several transforms and entropy coders that are introduced here.

A. Run-Length Encoding (RLE)

The RLE transforms a bilevel image into a sequence of runs. A run is the number of successive occurrences of the same value in the file. For example, the sequence $[0, 0, 0, 1, 1, 1, 1]$ would be coded $[3, 4]$ by a coder assuming the first received symbol is a 0.

B. Move to Front transform (MTF)

The purpose of the MTF is to reduce the entropy of the image [1]. It is also based on the concept of runs but replaces each symbol of a run by 0 except the first one. The first one is replaced by the index of the symbol in the alphabet of the source. For example, for a sequence $[1, 1, 1, 0, 0, 0, 0]$, the alphabet is $[0, 1]$. The index of 1 in the alphabet is 1, so that the sequence becomes $[1, 0, 0, 0, 0, 0, 0]$. The 1 is moved to the front of the alphabet (hence the name of the MTF), and the sequence of 0's now needs to be encoded. The alphabet is now $[1, 0]$, so that the index of 0 is 1. The final sequence is $[1, 0, 0, 1, 0, 0, 0]$ and the final alphabet is $[0, 1]$.

C. Two-Role Encoder (TRE)

The TRE is a variant of the RLE which is more suited when one symbol is more probable than the other [1], so that it is meant to be used after the MTF. Each non-zero element is shifted by 8 bits, while the sequence of zeros is simply encoded as the run length. To follow the previous example, the sequence $[1, 0, 0, 1, 0, 0, 0]$ would be encoded as $[256, 2, 256, 3]$. Here, the assumption that no run length will exceed 256 is done. The stream can be decoded knowing that the beginning of a run is always greater than 255, while a run cannot be greater than 255.

D. Exp-Golomb coding

The Exp-Golomb coding is a variable length prefix code meant for entropy coding. The i^{th} is mapped to the binary representation of $i+1$, preceded by M zeros with $M = \lfloor \log_2(i) \rfloor$. When the short codewords are assigned to most probable symbols, compression can be achieved [2].

E. Arithmetic coding

An arithmetic encoder converts a sequence of data symbols into a single fractional number and can approach the optimal fractional number of bits required to represent each symbol. In general, it outperforms prefix codes [2]. An integer version of the coder was implemented to cope with finite precision problems. When the interval corresponding to the symbol stream becomes to small, some bits are already extracted and the interval is made wider again.

III. Compression schemes

Various combinations of the previously detailed techniques are implemented in the codec. The encoder will try for all possible techniques and write to file the most advantageous method. They are listed here:

- **RLE-Gb**: the image is RLE-encoded then compressed with Exp-Golomb. A LUT also needs to be transmitted to allow the mapping of shortest codewords to most probable symbols. Two variants are possible, which are the scanning of the image in horizontal or vertical order.
- **RLE-Ath**: same process as RLE-Gb but the Golomb code is replaced by the arithmetic encoder. The symbols and their probabilities also need to be transmitted in a LUT.
- **M2F-Ath**: the image is transformed using the M2F and then is compressed by the arithmetic encoder. The dictionnary of the M2F and the probabilities for the arithmetic coder need to be added to the file.
- **Benzid**: the method is the one proposed in [1]. The image is vertically shrinked to reach 8bpp, is transformed using the M2F, TRE and compressed using the arithmetic coder.

In the compressed file, a 3-bits flag identifies the chosen method. For some of them, additional size information is also needed to correctly parse the file while decoding.

IV. Performance analysis

V. Conclusion

References

[1] R. Benzid. "Lossless compression of binary image using move-to-front transform and two-role encoder". In: *Electronics Letters* 47.2 (2011), pp. 104–105. ISSN: 0013-5194. DOI: 10.1049/el.2010.2362.

[2] Iain E. Richardson. *The H.264 Advanced Video Compression Standard*. 2nd. Wiley Publishing, 2010. ISBN: 0470516925, 9780470516928.