LOSSY-TO-LOSSLESS COMPRESSION OF IMAGES BASED ON BINARY TREE DECOMPOSITION

Armando J. Pinho and António J. R. Neves

Signal Processing Lab, DET / IEETA
University of Aveiro, 3810-193 Aveiro, Portugal
ap@det.ua.pt — an@ieeta.pt

ABSTRACT

A new lossy-to-lossless quality-progressive method for image coding is presented. This method is based on binary tree decomposition and context-based arithmetic coding. Experimental results obtained with the eighteen 8-bit ISO images show that the proposed method attains an average lossless compression gain of 7.9% with respect to JPEG2000. Compared to JPEG-LS, the compression gain is 3.1%, but this compression standard does not allow lossy-to-lossless decoding.

Index Terms— Image coding, data compression.

1. INTRODUCTION

Usually, image compression techniques are classified according to their capability for recovering the original data. Lossy methods waive away that capability in exchange for increased compression. On the other hand, lossless techniques are stick to the fundamental principle of exact recovery of the original data, even if that implies modest compression rates. Lossy methods are typically used in consumer products, such as photographic cameras. Lossless methods are generally required in applications where cost, legal issues or value play a decisive role, such as, for example, in remote and medical imaging or in image archiving.

The application is, therefore, an important aspect when choosing between a lossy or a lossless representation. However, there might be situations where different end users may require different levels of reproduction quality, for example, if they have reproduction equipments (displays, printers) with different capabilities. Users may have different downloading capacities, and may want to trade some loss of quality for a reduced downloading time. Finally, different users may want to pay different amounts for a given image and, therefore, may want to trade image quality for money. In conclusion, the image provider may be asked to deliver copies of the same image at an eventually large number of different qualities, including lossless.

This work was supported in part by the Fundação para a Ciência e a Tecnologia (FCT).

Basically, there are two possible approaches to address this problem. One, is based on brute force, requiring from the provider the daunting task of maintaining or having to generate on request all possible versions of the image. The other, relies on embedded bit-stream technology, where a single bit-stream may be able to generate, potentially, as many different images as the number of bits it possesses. This latter approach is offered by the most recent standard for image coding, JPEG2000 [1, 2].

In this paper, we present a lossy-to-lossless quality-progressive method for image coding. The proposed technique allows the reconstruction of a full resolution image from an arbitrary fraction of the bit-stream, resulting in an image with a reduced number of intensity levels. This method is based on binary tree decomposition and context-based arithmetic coding. It was inspired by the work done by Chen *et al.* regarding the compression of color-quantized images [3]. During encoding, the tree is traversed from the root to the leaf nodes, in a specific order, with the aim of minimizing the reconstruction error. For each node, the encoder sends the representative intensity of each of its two children nodes and the location of the pixels having an intensity change. The encoding of these pixel locations is performed by means of a context-based arithmetic encoder with variable size contexts.

2. THE CODING METHOD

2.1. Constructing the tree

We denote by $\mathcal{I} = \{I_1, I_2, \ldots, I_M\}$ the set of image intensities and by $p(I)$ the fraction of pixels having intensity I. Each node, n, of the binary tree represents a certain subset, $\mathcal{S}^n$, of the intensities of the image, i.e., $\mathcal{S}^n \subset \mathcal{I}$. Each node possesses a representative intensity, I^n, which is the weighted average of the intensities associated to the node, i.e.,

$$I^n = \sum_{I \in \mathcal{S}^n} I p(I).$$

Whenever $|\mathcal{S}^n| > 1$, this set is partitioned into two other subsets, $\mathcal{S}_l^n$ and $\mathcal{S}_l^n$, having, as close as possible, identical sizes. The intensities that are assigned to each node are such that $I_i < I_j, \forall I_i \in \mathcal{S}_l^n, \forall I_j \in \mathcal{S}_r^n$.

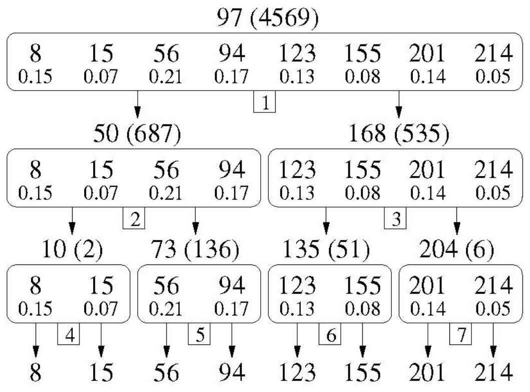

Fig. 1. Small example of binary tree which is used to illustrate how the coding algorithm works.

Figure 1 shows an example of a decomposition tree where $\mathcal{I} = \{8, 15, 56, 94, 123, 155, 201, 214\}$. The numbers below the intensity values indicate the fraction number of pixels having that intensity. The numbers over the boxes (which represent the nodes of the tree) indicate the representative intensities and the corresponding reconstruction squared errors (the latter written inside parenthesis). Finally, the numbers inside the small squares are used to identify the nodes.

2.2. Traversing the tree

Although, in our explanation, we are separating the construction of the tree from the part of traversing the tree and encoding the associated information, in fact these two operations can be done simultaneously. The first node to be encoded is the root node, for which the value of I^1 is transmitted ($I^1 = 97$ in the example of Fig. 1). Then, until the whole tree is traversed, the following procedure is iterated:

1. From the nodes available for processing, i.e., the children nodes of all nodes for which the corresponding I^n has already been transmitted, the node associated with the largest reconstruction error is chosen. Let us call it node n.

2. Encode the identification of this node, i.e., the value of n (the numbering of nodes is such that it can be reproduced by the decoder without additional side information) and the representative intensities of its two children, I_l^n and I_r^n, corresponding to subsets $\mathcal{S}_l^n$ and $\mathcal{S}_r^n$ ($\mathcal{S}^n = \mathcal{S}_l^n \cup \mathcal{S}_r^n$).

3. Encode the location of the pixels with intensity belonging to $\mathcal{S}_l^n$. Since the location of the pixels with intensity belonging to $\mathcal{S}^n$ is known by the decoder, then it is only needed to encode, for each of those pixels, if its intensity belongs now to $\mathcal{S}_l^n$ or to $\mathcal{S}_r^n$.

Therefore, according to the example shown in Fig. 1, the nodes would be traversed in the following order: 1, 2, 3, 5, 6, 7, 4.

2.3. Encoding pixel locations

The location of the pixels that have the intensity changed due to a node splitting is encoded using context-based arithmetic coding. Contexts are constructed based on the template shown in Fig. 2, where the context pixels are numbered according to their distance to the encoding pixel. The context is constructed using a sequence of bits,

$$b_1 b_2 \ldots b_k \qquad (1)$$

where

$$b_i = \begin{cases} 0, & \text{if} \quad |I(i) - I_l^n| \le |I(i) - I_r^n| \\ 1, & \text{otherwise} \end{cases}$$

and where $I(i)$ denotes the intensity of the pixel corresponding to position i of the context template in the current reconstructed image.

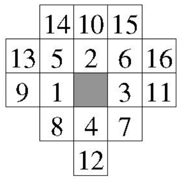

Fig. 2. Context template used in this work. Note that the use of non-causal pixels is possible, because the image is encoded/decoded using multiple passes (in fact, as many as the number of different intensities).

2.4. Context adaptation

In his work on color-quantized image coding, Chen *et al.* noticed that the results could be improved if the size of the context template, i.e., the value of k in (1), is progressively reduced during encoding/decoding [3]. A further improvement to this idea was presented in [4], still for the case of color-quantized images. According to the model proposed in [4], the size of the context is chosen by means of a relation that depends on the number of intensities already processed, n, and the number of pixels of the image, N:

$$k(n) = \lceil \alpha(N) - \log_2(n+1) \rceil ,$$

where

$$\alpha(N) = m \log_2 N + b,$$

i.e.,

$$k(n) = \left\lceil \log_2 \frac{N^m 2^b}{n+1} \right\rceil .$$

The function $\alpha(N)$ was adjusted using training data from a number of images, resulting in $m = 0.671$ and $b = -0.859$ (for further details, see [4]).

3. EXPERIMENTAL RESULTS

For testing the performance of the compression method described in this paper, we used the eighteen 8-bit ISO images (also known as the JPEG continuous-tone test image set). This set of images has been used during the development of the JPEG-LS image coding standard, and cover a wide range of image types, including natural, synthetic and compound images, and image formats (RGB, CMYK, CIELab, YUV and grayscale).

Table 1 presents the compression results, in bits per pixel (bpp), obtained with this image set. Besides presenting the results produced by the proposed method, which are denoted by "BTC" in the table, we also include, for comparison, results obtained with JPEG-LS (the current standard for the lossless compression of continuous-tone images [5, 6]), with lossless JPEG2000, and using exhaustive search for finding the best context size ("BTC-BEST").

JPEG2000 lossless compression was obtained using version 5.1 of the JJ2000 codec with default parameters for lossless compression[1]. JPEG-LS coding was obtained using version 2.2 of the SPMG JPEG-LS codec with default parameters[2].

4. DISCUSSION AND CONCLUSION

The results presented in Table 1 show an average gain of 7.9% of the proposed method over lossless JPEG2000. The gain in relation to JPEG-LS is smaller, 3.1%, but this technique does not produce an embedded bit-stream and, therefore, fails to provide lossy-to-lossless capability.

It is interesting to note that the additional compression gain that can be obtained if the best context is sought ("BTC-BEST") is only marginal (about 1 percentual point). This result gives an indication that the context adaptation model is quite accurate, despite the fact that had been calibrated with color-quantized images [4].

In Fig. 3, we give an example of the operational rate-distortion curves that are typically obtained with the proposed method, both for the L_∞ (maximum absolute) and L_2 (root mean squared) errors. For comparison, similar curves are presented for JPEG2000. As can be seen, for higher rates, JPEG2000 usually losses to BTC. However, at lower rates, JPEG2000 is generally superior to BTC. In our opinion, this happens because BTC provides a full resolution image right from the beginning, which is clearly a current drawback and disadvantage in comparison to JPEG2000.

In conclusion, we have presented a new image coding method that provides an embedded bit-stream, allowing lossy-to-lossless compression. This method is not based on predictive nor on transform coding. Instead, it relies on the direct encoding of the intensity image, by means of a suitable binary tree decomposition. The experimental results attained show its competitiveness in relation to the current standard that is able to offer similar lossy-to-lossless capability, i.e., JPEG2000.

[1]http://jj2000.epfl.ch.

[2]The original web-site of this codec, http://spmg.ece.ubc.ca, is currently unavailable. However, it can be obtained from ftp://www.ieeta.pt/~ap/codecs/jpeg_ls_v2.2.tar.gz.

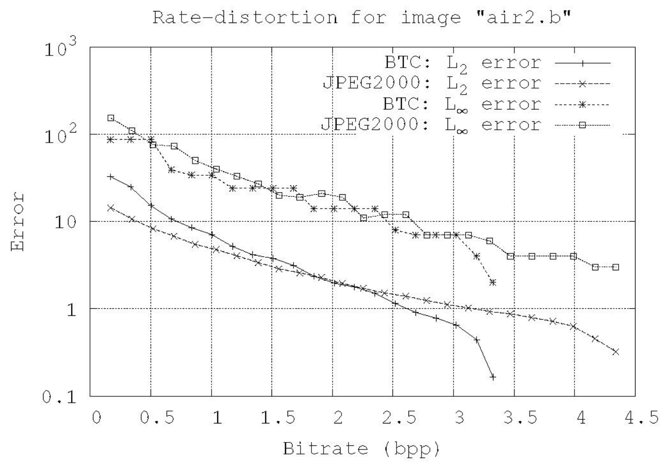

Fig. 3. Operational rate-distortion curves for the proposed method and for JPEG2000 for image "air2.b".

5. REFERENCES

[1] ISO/IEC, *Information technology - JPEG 2000 image coding system*, ISO/IEC International Standard 15444–1, ITU-T Recommendation T.800, 2000.

[2] D. S. Taubman and M. W. Marcellin, *JPEG 2000: image compression fundamentals, standards and practice*. Kluwer Academic Publishers, 2002.

[3] X. Chen, S. Kwong, and J.-F. Feng, "A new compression scheme for color-quantized images," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 12, no. 10, pp. 904–908, Oct. 2002.

[4] A. J. Pinho and A. J. R. Neves, "A context adaptation model for the compression of images with a reduced number of colors," in *Proc. of the IEEE Int. Conf. on Image Processing, ICIP-2005*, vol. 2, Genova, Italy, Sept. 2005, pp. 738–741.

[5] ISO/IEC, *Information technology - Lossless and near-lossless compression of continuous-tone still images*, ISO/IEC 14495–1 and ITU Recommendation T.87, 1999.

[6] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. on Image Processing*, vol. 9, no. 8, pp. 1309–1324, Aug. 2000.

Table 1. Lossless compression results, in bits per pixel, on the 8-bit images of the ISO set, for JPEG-LS, JPEG2000, and the proposed method, BTC. The BTC-BEST column shows the compression results when the best context size is used.

Image	Size	JPEG-LS	JPEG2000	BTC	BTC-BEST
air2.b	720×1024	4.179	4.519	3.358	3.334
air2.g	720×1024	4.135	4.474	3.209	3.184
air2.r	720×1024	4.014	4.288	3.581	3.547
bike3.b	781×919	4.223	4.807	3.858	3.817
bike3.g	781×919	4.246	4.839	3.757	3.722
bike3.r	781×919	4.678	5.248	4.234	4.195
bike.black	2048×2560	1.592	2.022	1.445	1.419
bike.cyan	2048×2560	4.041	4.223	4.083	4.029
bike.magent	2048×2560	4.567	4.701	4.612	4.558
bike.yellow	2048×2560	4.306	4.476	4.351	4.299
cafe.black	2048×2560	3.891	4.452	3.731	3.696
cafe.cyan	2048×2560	4.948	5.184	5.001	4.947
cafe.magent	2048×2560	5.230	5.441	5.254	5.208
cafe.yellow	2048×2560	5.265	5.477	5.325	5.273
cats.b	3072×2048	2.635	2.594	2.449	2.429
cats.g	3072×2048	2.594	2.562	2.475	2.452
cats.r	3072×2048	2.609	2.573	2.413	2.393
chart.a	1752×2375	1.189	1.296	1.216	1.153
chart.b	1752×2375	1.088	1.193	1.105	1.039
chart.l	1752×2375	1.680	2.032	1.592	1.527
chart_s.b	1688×2347	2.613	3.001	2.052	1.989
chart_s.g	1688×2347	2.784	3.133	2.275	2.194
chart_s.r	1688×2347	2.913	3.167	2.367	2.257
cmpnd1.b	512×768	1.308	2.220	1.312	1.282
cmpnd1.g	512×768	1.257	2.182	1.251	1.224
cmpnd1.r	512×768	1.245	2.164	1.246	1.219
cmpnd2.b	1024×1400	1.339	2.175	1.367	1.334
cmpnd2.g	1024×1400	1.281	2.109	1.308	1.278
cmpnd2.r	1024×1400	1.356	2.158	1.399	1.365
faxballs.a	1024×512	0.777	0.897	0.592	0.540
faxballs.b	1024×512	1.106	1.237	1.342	1.162
faxballs.l	1024×512	0.812	0.925	0.667	0.601
finger	512×512	5.662	5.693	5.913	5.865
gold.u	360×576	3.117	3.243	3.138	3.113
gold.v	360×576	3.591	3.693	3.594	3.559
gold.y	720×576	4.477	4.639	4.596	4.549
graphic.a	2644×3046	2.609	2.455	2.622	2.606
graphic.b	2644×3046	2.501	2.471	2.552	2.522
graphic.l	2644×3046	1.963	2.030	1.950	1.936
hotel.u	360×576	3.121	3.298	3.179	3.146
hotel.v	360×576	3.325	3.516	3.340	3.304
hotel.y	720×576	4.381	4.624	4.491	4.430
tools.black	1524×1200	3.355	4.126	2.933	2.908
tools.cyan	1524×1200	5.552	5.827	5.069	5.024
tools.magent	1524×1200	5.666	5.902	5.248	5.203
tools.yellow	1524×1200	5.752	6.014	5.273	5.221
us	512×448	2.629	3.081	2.308	2.269
water.b	3072×2048	1.860	1.863	1.739	1.727
water.g	3072×2048	1.784	1.769	1.713	1.698
water.r	3072×2048	1.794	1.789	1.679	1.665
woman.black	2048×2560	3.605	3.773	3.561	3.533
woman.cyan	2048×2560	4.223	4.301	4.310	4.287
woman.magent	2048×2560	4.565	4.600	4.641	4.613
woman.yellow	2048×2560	4.412	4.462	4.459	4.432
Average	—	**3.106**	**3.268**	**3.011**	**2.973**