ISSN 1060-992X, Optical Memory and Neural Networks, 2017, Vol. 26, No. 4, pp. 274–279. © Allerton Press, Inc., 2017.

Statistical Encoding Algorithm for Hierarchical Image Compression

M. V. Gashnikov

Samara National Research University, Samara, 443086 Russia

e-mail: mih-fastt@yandex.ru

Received July 18, 2017; in final form, October 11, 2017

Abstract—We analyze statistical encoding algorithms as one of the type of general methods of image compression. An approach we proposed allows us to improve effectiveness of a lossy image compression. We develop a statistical encoding algorithm that remains effective when compressing images with raw errors. It can be used as a part of any methods of image compression performing encoding of decorrelated signals with nonuniform distribution of probabilities. Coding specific data of a hierarchical compression method, we experimentally compared our algorithm with ZIP and ARJ archivers.

Keywords: image compression, statistical encoding, quantization, coefficient of compression ratio, compression ratio

DOI: 10.3103/S1060992X17040063

1. INTRODUCTION

In the present time, body of data related to images increases permanently. As an example, we can mention growth of video data obtained when shooting the Earth's surface from aircrafts [1–3]. If earlier, they have to deal only with images obtained from airplanes and satellites, now the researchers are concerned with multispectral and hyperspectral data that can contain hundreds of large format high-resolution channels. Mass application of unmanned aerial vehicles adds to the problem of storage of such image data.

Of course, there are other examples except the Earth's surface shooting. Increase in resolution and number of images takes place in different branches of knowledge, such as geoinformatics, medicine, printing industry, science of crime detection, and so on. In this situation, compression of images has no alternative, and the necessity to increase the efficiency of image compression methods is obvious.

Many image compression methods use decorrelation techniques [4]. Then data are quantized [5] and encoded (finish compression is performed) [6, 7]. For example, in the framework of the differential methods [8] of image compression they perform the decorrelation by transition to a residual signal (difference between incoming and predicted values of pixels). After that, the residual signal is quantized and encoded.

In the case of transform coding based on the discrete cosine transforms [9] (JPEG [10]) or the wavelet transforms [11] (JPEG-2000 [12]) a quantized field of the transforms (the results of transformation) has to be encoded.

When applying a hierarchical grid interpolation (HGI) for compression of images [13, 14] we interpolate pixels of an image basing on thinned versions of the same image and quantize and encode interpolation errors (post- interpolation remainders).

In all the examples discussed above, finally we need to encode a quantized signal. Of course, at a decorrelation step of any compression method it is necessary to construct the quantized signal most suitable for encoding ("final compression"). This is why as a rule the quantized signal has a specific (strongly nonuniform) distribution of probabilities. To make the best use of this non-uniformity for increasing the coefficient of compression, usually the quantized signal is statistical encoded [6, 7] allowing one to reduce data volume with the aid of an unequal-probability of the signal readings.

Thus there is a problem to choose (or to develop) an algorithm of statistical encoding for a given method of image compression. The most frequently used are the Lempel-Ziv algorithm [15], the Huffman coding [16], and the arithmetic coding [17]. Most of the other algorithms involve the ideas of the algorithms listed above.

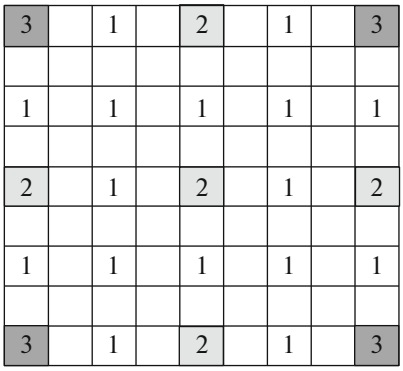

Fig. 1. Distribution of pixels of scale equations at image for number of levels $L = 4$ (free sells correspond to level number 0).

Computational complexity of the arithmetic coding is comparatively high. One other substantial defect of the arithmetic coding is that patents cover many of arithmetic coders. Computational complexity of the Lempel-Ziv algorithm is also high. Moreover, it is not sufficiently effective without an additional final compression algorithm. As such algorithm they usually use the Huffman coding. This is the reason why the most attractive is the Huffman coding whose complexity is less and the coefficient of compression is the best among all the variable-length codes.

However, due to the specific character of the distribution of probabilities of a coded signal direct application of the Huffman coding can lead to a substantial loss of the efficiency of the codding. The coded signal frequently contains a value (usually, zero value) which occurs much more often than the others (this is one of the most important aims of quantizing). The cause of the loss of the efficiency is in inability of the Huffman coding to assign a codelength less than 1 bit to any symbol (even the one that occurs very frequently). Thus, the coefficient of compression cannot exceed the width of the input signal and the degree of compression (in bits per reading) can sufficiently exceed the entropy of the compressed signal. In fact, just this is a loss of efficiency of the encoding (the algorithm is noticeable behind the theoretical limit).

In this paper, we propose a two-stream statistical coder well adapted to the distribution of probabilities of a coded signal when compressing images. Although we developed our coder for the hierarchical compression method, its main idea is suitable for a wide variety of methods of image compression based on the decorrelation and successive quantization of the compressed signal.

2. HIERARCHICAL REPRESENTATION OF IMAGES

The method of the hierarchical grid interpolation (HGI) [18, 19] for compression of images is based on the hierarchical representation of the compressed image. That means that we represent the image as a set of L scale equations. The highest scale level that is the level number $(L-1)$ is a two-dimensional array of readings of an input image that is 2^{L-1} times thinned over each coordinate. The next scale level is the level number $(L-2)$. It is the input image from which we exclude the readings of the previous scale level $(L-1)$ thinned with the step 2^{L-2}; and so on up to the level number 0 (see Fig. 1).

This representation is alike a quadtree (a pyramid image representation) [4, 5] but has the advantage since it is irredundant. In a formalized form we write of our hierarchical representation of the image $\mathbf{X}$ as a set of the scale levels $\mathbf{X}_l$:

$$\mathbf{X} = \bigcup_{l=0}^{L-1} \mathbf{X}_l, \quad \mathbf{X}_l = \begin{cases} \{x_l(m,n)\}, & \text{if} \quad l = L-1, \\ \{x_l(m,n)\} \setminus \{x_{l+1}(m,n)\}, & \text{if} \quad 0 \le l \le L-2, \end{cases} \tag{1}$$

where $\{x_l(m,n)\}$ is an array of image pixels thinned with the step 2^l over both coordinates.

The hierarchical representation described above allows us to perform compression of the scale levels one after another beginning from the highest scale level $\mathbf{X}_{L-1}$; and at that we use pixels of more thinned (less detail) scale levels to interpolate pixels of less thinned (more detail) scale levels. Let us write down a formal compression procedure.

3. HIERARCHICAL IMAGE COMPRESSION

When compressing we process the scale levels of images one after another starting from the highest that is in the order $\mathbf{X}_{L-1}$, $\mathbf{X}_{L-2}$, ... $\mathbf{X}_1$, $\mathbf{X}_0$. The way of compression of the highest level $\mathbf{X}_{L-1}$ is not important since its contribution to the whole data set is very small (when number of the levels is larger than four this contribution is less than half a percent). So, in what follows we describe a compression procedure for a level $\mathbf{X}_l$, where $l < L-1$. It consists from the following steps (See also Fig. 2):

(1) Interpolation

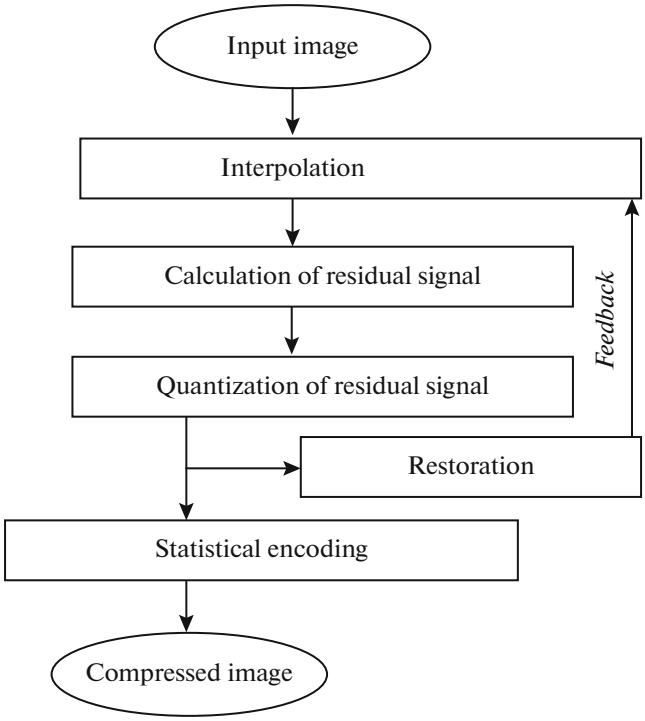

Fig. 2. Compression of hierarchical level $\mathbf{X}_l$ ($l < L-1$).

By the time of interpolation of the scale level $\mathbf{X}_l$ all the more thinned scale levels $\mathbf{X}_k, l < k \le L-1$ are already processed. Their union is the image $\{x_{l+1}(m,n)\}$ thinned with the step 2^{l+1}. For all the pixels of this image, we already know all the restored values $\bar{x}_{l+1}(m,n)$ that coincide with the values that will be obtained during decompression. Let $\overline{\mathbf{X}}_k$ denotes the set of the restored pixels corresponding to the scale level $\mathbf{X}_k$. Consequently, we perform interpolation of the pixels of the scale level $\mathbf{X}_l$ basing on the restored pixels that correspond to the already processed more thinned scale levels:

$$\hat{x}_l(m,n) = P\left(\bigcup_{k=l+1}^{L-1} \overline{\mathbf{X}}_k\right) = P\left(x_{l+1}(m,n)\right), \quad (2)$$

where $\hat{x}_l(m,n)$ are the interpolating values of the readings; $P\{\ldots\}$ is a function that specifies the interpolator.

(2) Calculation of residual signal

We calculate a residual signal that is a set of differences of true and predicted values of the readings (post-interpolation remainders):

$$f_l(m,n) = x_l(m,n) - \hat{x}_l(m,n). \quad (3)$$

(3) Quantization of residual signal

We quantize the obtained residual signal Eq. (3). At that, we replace each residual signal $f_l(m,n)$ by its quantized value

$$q_l(m,n) = Q_e\left(f_l(m,n)\right), \quad (4)$$

where $Q_e(..)$ is a function that defines a quantizer and $q_l(m,n)$ is the quantized residual signal (or simply the quantized signal).

(4) Restoration

At this step we use a quantized values $q_l(m, n)$ to calculate the restored values $\bar{x}_l(m,n)$ of pixels of the scale level l. For this purpose, we, first, have to calculate the restored values of the residual signal

$$\bar{f}_l(m,n) = Q_d\left(q_l(m,n)\right), \quad (5)$$

where $Q_d(..)$ is a dequantization function that defines approximate values of the residual signals from their quantized values; and second, to calculate the restored values themselves:

$$\bar{x}_l(m,n) = \bar{f}_l(m,n) + \hat{x}_l(m,n). \quad (6)$$

These restored values of pixels are identical to those that will be obtained during the decompression (as if the compression procedure includes the decompression procedure). We need the restored values for interpolation of the more thinned hierarchical levels $\mathbf{X}_k$, $0 < k < l$. At this stage of compression, we perform this operation for all the scale levels (except the scale level $\mathbf{X}_0$). Due to the above-described feedback (in other words, when using for interpolation the restored pixels in place of the input pixels; see Fig. 2) we can guarantee that during the compression and decompression the interpolator works in exactly the same way (it calculates the same values). This allows us to control the measure of inaccuracy.

(5) Statistical encoding

At this stage we perform the statistical encoding of the quantized signal $q_l(m, n)$. Since the in this signal distribution of probabilities is strongly non-uniform we can obtain a significant decrease of the data set. That is the result that should have been received after the image compression.

At this, we finished the general description of the algorithm of the hierarchical compression. To specify the compression method it is necessary to indicate algorithms of interpolation, quantization, and statistical encoding.

In the framework of our analysis we are concerned with the statistical encoding. Consequently, the interpolation algorithm is not so important for us. Next, for quantization we use an equidistant scale [4, 5]:

$$q_l(m,n) = \operatorname{sign}\left(f_l(m,n)\right)\left[\frac{\left|f_l(m,n)\right| + \varepsilon_{\max}}{2\varepsilon_{\max}+1}\right], \tag{7}$$

where [..] denotes the integral part of a number. This guarantees that the decompressed image differs from the input image by no more than the given value of the maximal error $\varepsilon_{\max}$:

$$\left|f_l(m,n)\right| = \left|x(m,n) - \bar{x}(m,n)\right| \le \varepsilon_{\max}. \tag{8}$$

Our algorithm of the statistical encoding we describe in what follows.

4. STATISTICAL ENCODING WHEN COMPRESSING IMAGES

Under the hierarchical compression, we apply the statistical encoding [6, 7] to the quantized residual signal $q_l(m,n)$. The distribution of probabilities in this signal is much alike the distribution of probabilities in the "not quantized" residual signal $f_l(m,n)$: it has a maximum at zero and quickly decreases symmetrically on the both sides of zero [4, 5]. However, the quantized signal differs from the "not quantized" since its distribution of probabilities is narrower due to the union of values when quantizing.

When encoding signals characterized by strongly non-uniform distributions of probabilities they often use variable-length codes [6]: short codes are assigned to frequently occurring symbols and longer codes to rare in occurrence symbols. In this case, the length of the code word defines the efficiency of the coder [7] that coincides with the average length of the code word. When measured in bits per reading it is

$$B = \sum_{i=0}^{N-1} p_i b_i, \tag{9}$$

where N is the number of possible values of the coded signal, p_i, b_i are probabilities and lengths of codes of coded values.

It is just the average length of a code word B we try to minimize when developing a coder. However, according to the Shannon–Hartley theorem [6] the average length of the code word has a theoretical limit. This means that it cannot be less the entropy [7]

$$H = -\sum_{i=0}^{N-1} p_i \log_2 p_i. \tag{10}$$

From the last two equations, it is easy to see that the average length of the code word reaches the theoretical limit (i.e. the entropy) if when developing the code we can achieve the length of the code equal to

$$b_i = -\log_2 p_i, \quad 0 \le i < N. \tag{11}$$

However, frequently in the case of the hierarchical compression the specific form of the distribution of probabilities of a coded (quantized) signal does not allows us to reach the length (11). This results in a loss of efficiency of the variable-length code for this signal. We have to discuss this effect in detail.

The more accurate the interpolator the larger the value of the given maximal error of quantization $\varepsilon_{\max}$ and consequently the more zero values are in the coded signal. In this case, the probability of zero p_0 can significantly exceed 0.5. The amount of information in the zero symbol is equal to $\left(-\log_2\left(1/p_0\right)\right)$; and it can be significantly less than one bit when $p_0 > 0.5$. In the same time, the variable-length coder assigns to this signal one-bit length code. This is the reason for the loss of efficiency of codding that increases when the error due to compression increases. To avoid the above-mentioned loss of efficiency in this paper we propose a two-stream algorithm for statistical encoding where we take into account a fraction of zeroth in the coded signal (Fig. 3).

The algorithm works as follows. If the probability of zero $p_0 \le 0.5$ there is no loss of the efficiency described above. In this case, when encoding we use the best variable-length coder that is the Huffman coder [16].

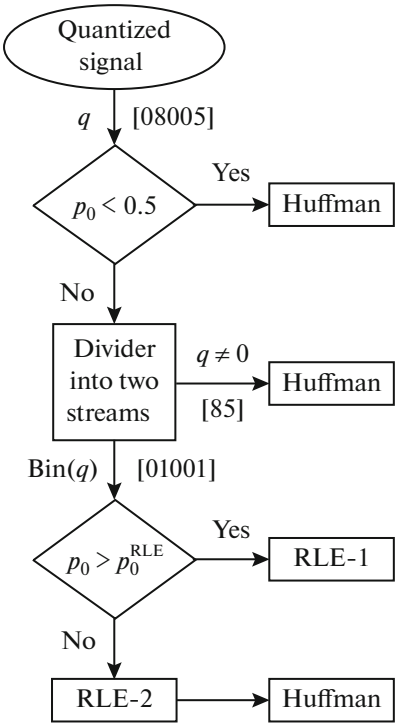

Fig. 3. Scheme of proposed two-stream coder (example of coded signal is in square brackets).

When the probability of zero $p_0 > 0.5$ we divide the coded signal into two streams. In the first stream, we send all the nonzero readings. In the second stream, we send a binary dating sequence [8] where we write ones in place of nonzero readings of the input stream and zeros in place of zero readings. Then we encode the streams independently.

In the first stream the symbols 1 and (−1) are the most probable. At that from the symmetry of the distribution of probabilities, it follows that the probability of each of this symbols is less than 0.5 and consequently we can use the Huffman algorithm that in this case does not loss its efficiency.

We can use different algorithms when encoding the second (binary) stream depending on the fraction of zero values. If the probability of zero does not exceed a certain threshold value $p_0 \le p_0^{RLE}$ (that is the probability of zero is large, but not too large) algorithms based on the run-length encoding (RLE) [7] and the Huffman algorithm [16] suit. Using these algorithms, we first perform a transition from the binary sequence to the sequence consisting from the enlarged RLE symbols of the form 1, 01, 001, .. , 00..01, 00..00. Unlike the standard RLE the number of different enlarged symbols N_{RLE} can be not only powers of two.

Next, in contrast to the usual RLE to each enlarged symbol we assign not a fixed length code, but the Huffman code. To avoid the above-discussed loss of efficiency due to excessively large probability of one of the symbols, we choose the parameter N_{RLE} satisfying the inequality

$$p_0^{N_{RLE}-1} < 0.5. \quad (12)$$

Because of the non-uniform distribution of probabilities of the enlarged symbols, such encoding is more effective comparing with the standard RLE; however, the codebook has to be transferred.

When the probability of zero $p_0 > p_0^{RLE}$ we use the standard RLE for the binary stream. Additional Huffman's encoding is not useful because there are too many enlarged symbols. Consequently the size of the codebook becomes too large and the coefficient of compression decreases.

To finish the description of our algorithm of the statistical encoding we note that one can use it not only for the hierarchical compression of images but also as a part of other compression methods generating a decorrelated signal with a strongly non-uniform distribution of probabilities.

5. ANALYSIS OF EFFECTIVENESS OF STATISTICAL CODING ALGORITHM

To analyze the effectiveness of the developed two-stream coder we compare it with the common encoding algorithms ARJ and ZIP [6, 7] using the specific data of the HGI method that is the quantized residual signal. Here as a measure of effectiveness, we use the degree of compression B (the set of compressed data in bits per reading). In Fig. 4, we show the typical results (for the image "Lena").

From this figure, we see that for the specific data of the HGI method our two-stream coder is 20% better comparing with the algorithms ARJ and ZIP.

6. CONCLUSIONS

We analyzed the problem of the loss of efficiency of the algorithms of the statistical encoding in the case of lossy compression. The proposed method allowed us to increase the efficiency of the statistical coder. As an example, we developed the two-stream algorithm for statistical encoding that constitute a part of the hierarchical compression method. We compared experimentally our algorithm with ZIP and ARJ coders in the framework of the hierarchical compression method. One can include the proposed algorithm into any compression method where it is necessary to encode decorrelated signals with non-uniform density of probabilities.

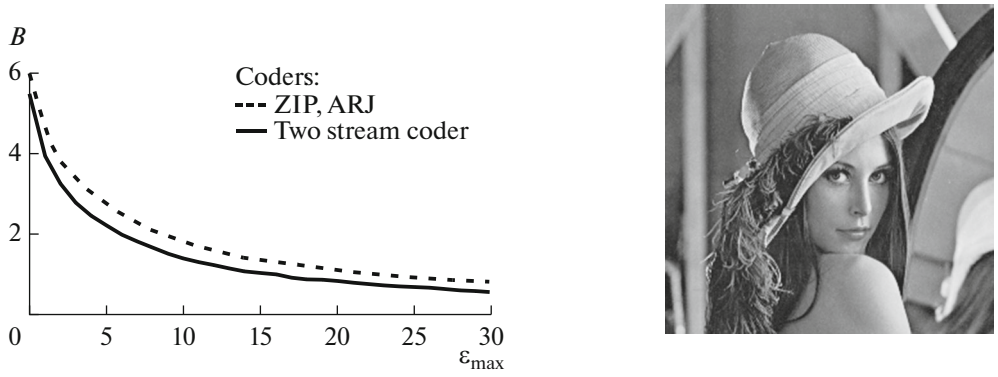

Fig. 4. Dependence of degree of compression B (in bits per reading) on maximal error ε_{max} for different coders on image "Lena".

ACKNOWLEDGMENTS

This work was financially supported by the Russian Science Foundation (RSF), grant no. 14-31-00014 "Establishment of a Laboratory of Advanced Technology for Earth Remote Sensing".

REFERENCES

1. Schowengerdt, R., *Remote Sensing: Models and Methods for Image Processing*, 3rd ed., Academic Press, 2007, p. 558, ISBN 978-0-12-369407-2.
2. Chang, C., *Hyperspectral Data Processing: Algorithm Design and Analysis*, Wiley Press, 2013, p. 1164.
3. Borengasser, M., *Hyperspectral Remote Sensing – Principles and Applications*, Borengasser, M., Hungate, W., and Watkins, R., CRC Press, 2004, p. 128.
4. Woods, E., *Digital Image Processing*, 3rd ed., Woods, E. and Gonzalez, R., Prentice Hall, 2007, p. 976.
5. Pratt, W., *Digital Image Processing*, 4th ed., Wiley, 2007, p. 807.
6. Sayood, K., *Introduction to data compression*, *The Morgan Kaufmann Series in Multimedia Information and Systems*, 4th ed., 2012, p. 743.
7. Salomon, D., *Data compression. The Complete Reference*, 4th ed., Springer-Verlag, 2007, p. 1118.
8. Chernov, A.V., *Computer Image Processing, Part II: Methods and Algorithms*, Chernov, A.V., Chernov, V.M., Chicheva, M.A., Fursov, V.A., Gashnikov, M.V., Glumov, N.I., Ilyasova, N.Yu., Khramov, A.G., Korepanov, A.O., Kupriyanov, A.V., Myasnikov, E.V., Myasnikov, V.V., Popov, S.B., Sergeyev, V.V., and Soifer, V.A., Eds., VDM Verlag, 2010, p. 584.
9. Plonka, G. and Tasche, M., Fast and numerically stable algorithms for discrete cosine transforms, *Linear Algebra and Its Appl.*, 2005, vol. 394, no. 1, pp. 309–345.
10. Wallace, G., The JPEG still picture compression standard, *Commun. ACM*, 1991, vol. 34, no. 4, pp. 30–44.
11. Gupta, V., Sharma, V., and Kumar, A., Enhanced image compression using wavelets, *Int. J. Res. Eng. Sci. (IJRES)*, 2014, vol. 2, no. 5, pp. 55–62.
12. Li, J., Image compression: The mathematics of JPEG-2000, *Modern Signal Processing*, *MSRI Publications*, 2003, vol. 46, pp. 185–221.
13. Gashnikov, M.V., Glumov, N.I., and Sergeyev, V.V., Compression method for real-time systems of remote sensing, *Proceedings of 15th International Conference on Pattern Recognition* (Barselona, 2000), vol. 3, pp. 232–235.
14. Gashnikov, M.V. and Glumov, N.I., Development and investigation of a hierarchical compression algorithm for storing hyperspectral images, *Opt. Mem. Neural Networks*, 2016, vol. 25, no. 3, pp. 168–179.
15. Ziv, J. and Lempel, A., A universal algorithm for sequential compression, *IEEE Trans. Inf. Theory*, 1977, vol. 23, no. 3.
16. Huffman, D., A Method for the construction of minimum-redundancy codes, *Proc. IRE*, 1952, vol. 40, no. 9, pp. 1098–1101.
17. Written Ian, H., Neal Radford, M., and Cleary John, G., Arithmetic coding for data compression, *Commun. ACM*, 1987, vol. 30, no. 6.
18. Gashnikov, M.V. and Glumov, N.I., Onboard processing of hyperspectral data in the remote sensing systems based on hierarchical compression, *Comput. Opt.*, 2016, vol. 40, no. 4, pp. 543–551.
19. Gashnikov, M.V., Minimizing the entropy of post-interpolation residuals for image compression based on hierarchical grid interpolation, *Comput. Opt.*, 2017, vol. 41, no. 2, pp. 266–275.