Lossless Image Compression with Lossy Image Using Adaptive Prediction and Arithmetic Coding

Seishi Takamura and Mikio Takagi
Institute of Industrial Science, University of Tokyo

Abstract

Lossless gray scale image compression is necessary in many purposes, such as medical image, image database and so on. Lossy image is important as well, because of its high compression ratio. In this paper, we propose a lossless image compression scheme using a lossy image generated with JPEG-DCT scheme. Our concept is, send a JPEG-compressed lossy image primaly, then send residual information and reconstruct the original image using both the lossy image and residual information. 3-dimensional adaptive prediction and an adaptive arithmetic coding are used, which fully uses the statistical parameter of distribution of symbol source. The optimal number of neighbor pixels and lossy pixels used for prediction is discussed. The compression ratio is better than previous work and quite close to the originally lossless algorithm.

1 Introduction

Today there are many studies on image compression, particularly on lossy and very low bit rate compression. For image database, such high compression ratio is important for storage and also for quick transmission, but to deal with various kinds of users demand, lossless image transmission is indispensable.

In this paper, we propose an effective lossless compression algorithm for gray image using lossy compressed image. The lossy compression scheme uses the Joint Photographic Experts Group discrete cosine transform (JPEG-DCT) algorithm as the lossy coding algorithm.

First we search the similar pairs of pixels (*contexts*), according to their neighbor pixels. For such pixels which have contexts, we predict their values from the contexts and the neighbors. On the other hand, for each pixel which doesn't have its context pairs, we calculate the *edge level* according to the difference of adjacent pixel values. For each edge level of pixels, we calculate the predictive coefficients of linear combination under the least square error criterion. Not only the pixels which have already processed but also the pixels of the lossy image is used for prediction.

For every pixel, the difference between predicted value and real value is calculated, and the difference is converted to a non-negative value before being encoded, according to their distribution. In entropy coding stage, we use the arithmetic coding. It is made adaptive, and initial error distribution is given only by one parameter, which is specific for each edge level's statistical distribution. The pixels belonging to the different edge levels are encoded independently.

1068-0314/94 $3.00 © 1994 IEEE

Experimental results and good performance are shown. Like the other LPL(Lossy Plus Lossless) approaches, our compression ratio is less than that of originally lossless scheme, but the difference is slight. Of all things, however, users get the great merit that they can browse the image before the lossless decompression. Many such schemes have been proposed in the literature, but most of them treat the lossy image and its lossless residual as independent symbol source. One of the exceptions is Memon's algorithm[6]. We utilize the lossy data thoroughly, and much better result is obtained.

1.1 Pixel estimation

Normally the image data is scanned along the scan-line direction. In figure 1, current pixel is '■', processed pixels are '□'.

Ordinal pixel estimation method predicts the value of current pixel using $P_1 \ldots P_4$. Then calculate the prediction error $e = \hat{x} - x$. Normally the linear combination is used for prediction as follows, where $r_1 \ldots r_4$ are the coefficients. This is the extrapolative prediction.

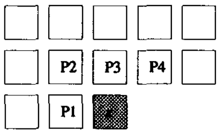

Figure 1: Current pixel and processed neighbor pixels

$$\hat{x} = r_1P_1 + r_2P_2 + r_3P_3 + r_4P_4 + C \tag{1}$$

Usually, the zero-order entropy of set $\{e\}$ is lower than that of set $\{x\}$. Therefore, after entropy coding scheme such as Huffman coding[1] or Arithmetic coding[2] or Lempel-Ziv coding[3], data size is reduced.

2 Our Method

Principally, we also use the linear combination like equation (1) for prediction, but the process is more adaptive than normal prediction method. And we use more neighbor pixels (up to ten), also using the pixels of lossy image and prediction error e is converted to another form before encoding.

2.1 Grouping the pixels

Each image pixel has different property under certain criterion. From a point of image compression, grouping similar pixels and encode them together causes effective result. For grouping the pixels, we use the Q value:

$$Q \equiv |P_2 - P_1| + |P_3 - P_1| + |P_4 - P_1| + |P_5 - P_1| \tag{2}$$

Using this Q value, we classify each pixel into several groups according to table 1.

Table 1: Grouping table

Q	0–1	2–4	5–8	9–16	17–32	33–64	65–128	129–
Group #	0	1	2	3	4	5	6	7

Figure 2: (a)Original image 'Girl' and (b)JPEG compressed image(quality value=5)

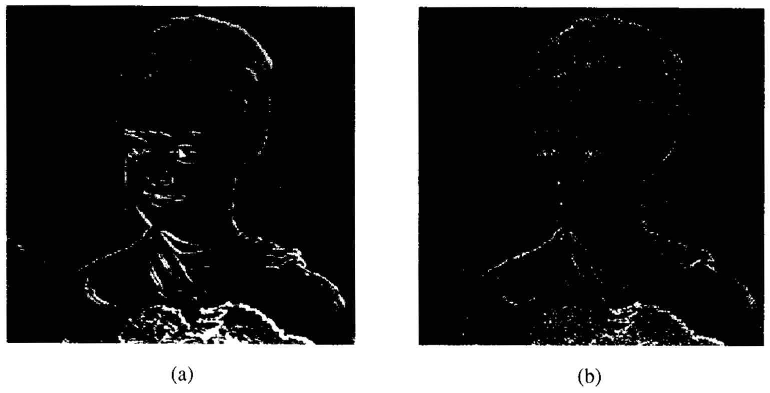

Figure 3: (a)Image of Q value, (b)Image of prediction error of simple prediction

Figure 3(a) shows the Q value and (b) shows the error of simple prediction. As can be seen from them, the value Q correlate closely with the prediction error. Therefore the prediction coefficients are calculated independently in each group.

2.2 Context search

Table 2 shows each group's final zero-order entropy of prediction result of image 'Girl'. Obviously, the upper groups are more difficult to be compressed than the lower groups. We use the context-based prediction method to deal with such upper groups.

The region where we search the similar area (we denote this *"context"*) is shown in figure 4. This is restricted within already processed pixels' area. The procedure is described below:

1. Scan the area and find points that satisfy
$$Q(x') > 70 \quad \text{and} \quad |Q(x) - Q(x')| < 10 \tag{3}$$
2. Within such points, find one that minimizes
$$C = ||b' - a'| - |b - a|| + ||c' - a'| - |c - a|| \tag{4}$$
3. If the C_{min} is smaller than 12, treat it as the context of the current point. Otherwise, return failure (not found).

2.3 Prediction

2.3.1 Prediction of normal groups

For each group, we predict the value of a current pixel by linear combination of its neighbors' pixel values. The coefficients are calculated by least square error method. The neighbor pixels used for prediction are shown in figure 5 ($P_1 \ldots P_{10}$). Number of pixels are variable (up to 10 pixels), and afterwards we will choose it optimumly. The priority is shown by the suffix number in the figure. The most effective number will be discussed later.

Table 2: Group v.s. Entropy (image 'Girl')

Group #	0	1	2	3	4	5	6	7
Entropy	3.0182	3.3405	3.5378	3.9016	4.4358	4.9533	5.5070	6.1044

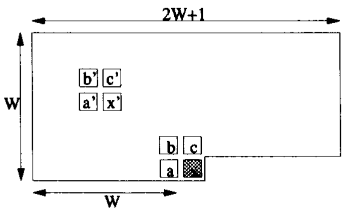

Figure 4: Context-search region

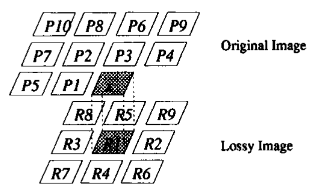

Figure 5: Pixels used for prediction

Here, some of the lossy pixels ($R_1 \ldots R_9$) got from JPEG-compressed image are also used. Using these pixels, something like interpolative prediction is achieved. This prediction contributes the reduction of compressed size.

2.3.2 Prediction of context

Under the criterion of condition (3) and (4), a pair of context have the similar shapes of height. Therefore, we predict the value $\hat{x}$ from $\{a', b', c', x', a, b, c\}$ (see figure 4). We also use the least square error estimation. The most different point from prediction of non-context pixels is, not only using the values of neighbor pixels but also using the closest context. This scheme is effective for continuous edges, because nearby an edge there is a sequence of similar looking pixels.

2.4 Error conversion

If each pixel has 8 bits, the prediction error $e(= \hat{x} - x)$ can have the real number between -255 and +255 (roughly). After prediction, e should be expressed as an integer. One easy way for conversion is, simply round the value off the integer (calculate $\lfloor e + 0.5 \rfloor$) and consider it as the 2's complement 8-bit number.

Our conversion algorithm is totally based on Taniguchi's method[5]. After this conversion, we can also get the 8-bit non-negative integer E. First we obtain the upper and lower bound of the group (max, min). Then, according to the figure 6, convert the actual pixel value into integer. (In this figure, if actual pixel is equal to 'max', $E = 9$.) For each group, we get the maximum and minimum pixel value and convert the prediction error respectively.

This conversion is reversible. If you get the predicted value and the converted number E (and also upper and lower bound), you can obtain the actual pixel value from similar numerical line.

2.5 Adaptive arithmetic coding

2.5.1 Fitting of the distribution of E

According to the experimental result, the distribution of E(figure 7(a)), after error conversion, looks very close to the right half of a Gaussian distribution:

$$f_{\sigma^2}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{x^2}{2\sigma^2}) \tag{5}$$

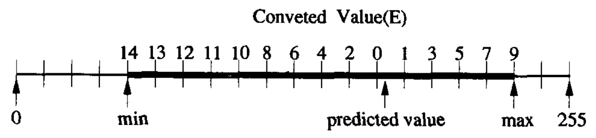

Figure 6: Algorithm of error conversion (example)

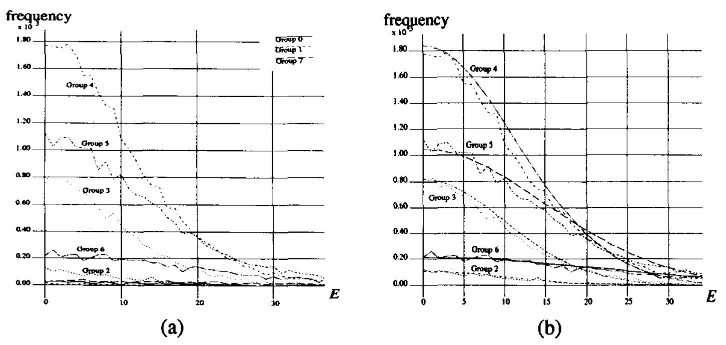

Figure 7: (a)Distribution of E (image='Moon'), (b)Fitting of E

where the variance is σ^2.

As this distribution is limited right half, σ^2 is equal to $\sum x^2/N$, where N is the number of samples. The fitting line (estimated frequency) of E's distribution is calculated $2Nf_{\sigma^2}(x)$.

Their graphs are shown in figure 7(b). You can see that using this Gaussian model, the distribution of E is approximated well.

2.5.2 Adaptive arithmetic coding

The probability density function of this fitting curve can be expressed by only one parameter, σ^2. This distribution is used to generate the initial distribution table for encoder (and also for decoder), instead of passing the large amount of actual frequency table. For this purpose, arithmetic coding is very suitable. For this purpose, our coder is made adaptive, each symbol from the data stream renews the frequency table. By using σ^2 and adaptive arithmetic coder, even small amount of data is coded with high coding rate.

3 Experimental Results

To generate a lossy image, we adopt the JPEG compression scheme. The reason is that JPEG is the standard scheme for still image compression and users can find JPEG-tools easily.

We use the tools named 'cjpeg' and 'djpeg', which is the products of Independent JPEG Group. To create the JPEG image,

```
cjpeg -quality quality-value -optimize filename
```

and to decompress the JPEG file,

```
djpeg jpegfile
```

is invoked. The option '-optimize performs optimization of entropy encoding parameters. It usually makes the JPEG file a little smaller. 'djpeg' has the option

Table 3: Effect of context search (N=9, NL=4, quality=5)

Image	without context search	with context search
Girl	4.61182	4.61169
Couple	4.05957	4.05481
Moon	5.04614	5.04419

'-blocksmooth', which performs cross-block smoothing, but from experimental results, it made the compression ratio worse, therefore this option is not used.

Parameters which are necessary for decompression are put together and also encoded by adaptive arithmetic coder. Only several percent compression is achieved, but better than doing nothing.

3.1 Effect of context search

We use three test images 'Girl', 'Couple' and 'Moon', which are chosen from SIDBA(Standard Image DataBase).

For these images, the bit rate of compressing with and without context search is compared. The results are shown in table 3. NL is the number of lossy pixels used for prediction. N is the number of lossless neighbor pixels.

3.2 Effect of the quality value on compression ratio

JPEG provides a fine tuning factor (*quality value*), which corresponds to different qualities of the compressed images. For typical image data, a low quality value such as 20 provides high compression with poor image fidelity. As the quality value increases the fidelity improves at the expense of compression ratio.

Figure 8 shows the bit rate and quality value. As the quality value decreases, the total bit rate decreases, too. Later we use the image quality value of 5, which is enough for undersanding the image roughly (see figure 2(b)).

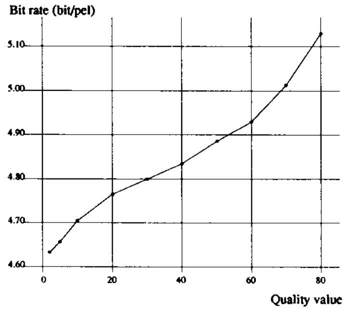

Figure 8: Bit rate V.S. Quality value (N=10, NL=9, image='Girl')

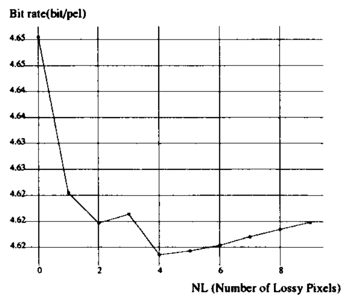

Figure 9: Bit rate V.S. NL (N=10, quality=5, image='Girl')

Table 4: Result of bit rate (quality=5, N=9, NL=4)

Image	Taniguchi	MAW (entropy)	Our method
Girl	4.494	5.10	4.612
Couple	3.978	4.56	4.055
Moon	5.019	—	5.046

3.3 Effect of using a lossy image

There might be a doubt that the lossy image looks similar with the original to our eyes, but differs much from a standpoint of pixel-value, therefore it hardly helps the prediction.

Figure 9 shows the bit rate and the number of lossy pixels used for prediction (NL). From this figure, it is known even the poor image (quality=5) helps the compression ratio considerably. The reason why the bit rate increases gradually where NL is greater than 4 is, that additional parameter (coefficients) is necessary for each pixel position. The optimum NL is 4. Moreover we conducted experiments to find the optimum number of N, and obtained 9.

3.4 Comparison with the other methods

Table 4 shows the result of the compression. Taniguchi's method[5] is originally lossless oriented one, therefore the results are slightly better than ours. In our method, N and NL are set optimally.

MAW method is proposed by Memon[6]. It also uses a lossy JPEG image to reconstruct a lossless image. Our result is about 0.5 bit/pel better than MAW, and 0.03 to 0.12 bit/pel worse than Taniguchi's method. As the MAW's result is given only by entropy, the difference of performance from ours might be bigger.

4 Conclusion

In this paper, we proposed the algorithm of lossless image compression. Unlike other literatures, we have discussed not only the entropy of residual, but how to encode it efficiently and the final size of the compressed product. This provides an attractive option in applications that have need for quick transmission on the one hand and exact reconstruction on the other. Furthermore, using this lossy image thoroughly, the total bit rate is considerably low.

Searching the contexts and using it for prediction is proved to work. We will research more effective way of using context.

Our algorithm is applicable for originally lossless oriented one (not using lossy image). In some images, we have obtained better results than Taniguchi's method. We are proceeding the investigation in this standpoint.

We are examining if there is a more suitable algorithm for lossy compression other than JPEG. And also speculating the more accurate error distribution fitting other than Gaussian model.

References

[1] D. A. Huffman: "A method for the construction of minimum redundancy codes", Proceedings of IRE 40, pp. 411–420, 1951

[2] J. J. Rissanen et al.: "Arithmetic coding", IBM Journal or Research and Development, 23(2), pp. 188–193, 1976

[3] J. Ziv and A. Lempel: "A Universal Algorithm for Sequential Data Compression", IEEE Transactions on Information Theory, IT-23(3), pp. 337–343, May 1997

[4] Paul G. Howard and Jefferey Scott Vitter: "New Methods for Lossless Image Compression Using Arithmetic Coding", Proceedings of Data Compression Conference '91, pp. 257–266, 1991

[5] Takayuki Taniguchi et al.: "Variable-Length-Code-Selective Reversible Predictive Coding for Multi-Level Images", The Transactions of the Institute of Electronics, Information and Communication Engineers, Vol.J70-B, pp.654–663, Jun. 1987

[6] Nasir D. Memon et al.: "Simple method for enhancing the performance of lossy plus lossless image compression schemes", Journal of Electronic Imaging 2(3), pp. 245–252, Jul. 1993

[7] Dmitry A. Novik: "Compression Through Decompression into Browse and Residual Images", Proceedings of 1993 Space and Earth Science Data Compression Workshop, NASA Conference Publication 3191, pp. 7–12, Apr. 1993