Efficient Lossless Still Image Coding Based on the Autoregressive Model

自己回帰モデルに基づく静止画像の高能率可逆符号化に関する研究

高村 誠之

Contents

List of Figures

List of Tables

Chapter 1

Introduction to Lossless Image Coding

BASED on today's ever growing progress of digital signal processing technology, the amount of image data transmission, such as for high definition TV broadcasting, for multimedia communication or storage for image databases, has become realistic and its demand is increasing. Simultaneously the characteristic of images to be transmitted or stored are gradually changing. They become to have finer spatial resolution, wider dynamic ranges which result in larger image amount. Therefore the importance of image compression, representing images with less data to save storage costs or transmission time and costs, is increasing.

Although the history of image compression goes long back to 1950's, there are ever more intent researches on this area even today because of the movements shown above.

Essentially there are two kinds of image compression methods: lossy and lossless, which is determined whether the original image and the reconstructed one coincide or not. Among image compression field, transform-based lossy image compression techniques have been ardently researched which results in JPEG, MPEG, H261 standards. On the other hand, lossless image compression, which recovers the exact original image from the compressed code and is required in many places such as

- medical area,
- astronomical area,
- image editing tools such as photo retouches or modify,
- high quality printing area,
- satellite remote sensing area

and so on, where even a slight deviation is sometimes significant and therefore any level of distortion is allowed. Obviously the lossless method has a drawback against a lossy method in terms of compression ratio. However relatively large-sized data handling has recently become practical since broadband network connection and large storage devices are now more common. Therefore not only those who require extremely high image quality but also those who have been putting up with the pictures of low fidelity only because of the code length restrictions come to use lossless image compression techniques.

Conceptually image compression techniques can be decomposed into two parts: "image modeling" and "entropy coding" which will be described in the following sections.

1.1 Image Models for Lossless Image Coding

Generally the image signal changes gradually, that is, there is a correlation between image pixels nearby. This means that it is possible to predict the value of a pixel using its

surrounding pixels somehow, i.e. an image contains a kind of redundancy. This is the core idea of image data compression.

For image coding purposes, there are several distinct categories, e.g. linear prediction, vector quantization, segmentation coding, transform coding, Markov model, etc. In this section, the models that are applicable for lossless image coding are described.

1.1.1 Autoregressive (AR) model

The most traditional method to remove the redundancy might be differential pulse code modulation (DPCM). This scheme takes the differences between one pixel and the next and encodes them, since in a continuous-tone image the differences are likely to be small. DPCM is a particular instance of a general class of coding known as *predictive coding.*

The model of these prediction methods is called "autoregressive model" in this paper.

1.1.2 Vector quantization

A straightforward application of rate-distortion theory to picture coding indicates that we assign a probability measure on the picture ensemble and then use an entropy coding to give smaller length codes to those pictures that are more likely. However, the number of possible pictures is so large that it is impossible to assign a probability measure on it, much less construct a code book.

Vector quantization scheme normally divides a picture into subpictures (called blocks) of smaller size and do the multi-dimensional quantization.

1.1.3 Wave model

"Wave model" is not the generally accepted term. Here I use this to represent the modeling of both "Transform coding" and "Subband coding".

Transform coding, which can be considered as a little bit more compromised version of the vector quantization, normally divides a picture into subpictures as the vector quantization scheme does. However, subpictures of reasonable size (e.g. 4×4 elements) yield a large code book. So instead, it takes the picture elements in the block and transforms them into certain coordinates that can be treated independently which are related to the probability measure on the original subpicture. The transformed coefficients are more independent, and are then quantized and coded for transmission. A truly optimum transform would result in the best picture quality using the least number of bits, but this criterion is difficult to specify quantitatively. A simpler criterion is to require that the transformed coefficients

be statistically independent, but this requires knowledge of higher order (higher than two) statistics of the images. For the alternative of this, we seek a transformation that results in *uncorrelated* coefficients. In this sense, the optimum transform is Karhunen-Loève transform (KLT).

It can be proved that for a first-order Markov process with exponential correlation, the best transform which can be computed by an algorithm similar to the fast Fourier Transform (FFT) is the discrete cosine transform (DCT)[1]. As the matter of course transmitting the DCT parameters is not necessary, and empirically the performances of KLT and DCT are very close to each other. Therefore DCT is adopted as the transform for the standard JPEG baseline scheme, MPEG, H261 etc.

On the other hand, subband coding such as a wavelet transform, which has been studied independently of transform coding, was pointed out to be equivalent to transform coding by Vetterli *et al*[2]. DCT, for example, can be represented by a filter bank, and vice versa, a filter bank can be interpreted as a linear transform. The major difference is that subband coding does not cause the *blocking artifacts*, visible discontinuities between adjacent subpictures.

1.1.4 Markov model

If the symbol occurrence probability depends only on the m preceding symbols, the source is called an *"m-th order Markov source"*. The series of m preceding symbols is called the *"state"*. If the source has N different kinds of symbols, the m-th order Markov model can be divided into N^m memoryless sources, the same as the number of possible states.

A Markov model approximation of digital images gives very low entropy, but requires a huge amount of storage to maintain the frequency table for encoding and decoding. Therefore, it has only been used for binary picture compression so far.

1.2 Entropy Coding

When the image redundancy according to the image model is removed, i.e. when the residual signals can be considered uncorrelated, there exists the statistical redundancy. Reducing the amount of such data without losing information is called *entropy coding*, and many algorithms are introduced. For example, Huffman coding[3], Lempel-Ziv coding[4], arithmetic coding[5] etc. After comparing these coding algorithms, I decided to adopt arithmetic coding because of its optimality concerning the code length or the code efficiency. Detailed discussion and the technique to reduce the code length will be shown

in chapter 2.

1.3 Motivation of the Study

In spite of the importance of the lossless image coding is increasing rapidly than ever, current techniques such as JPEG lossless mode or JBIG encoder is not very sufficient about compression ratio. The problems of those schemes are shown later in this section. The purpose of this study is to investigate the methods from numbers of aspects including image models and coding algorithms and so on, to compress an image as small as possible.

There are quite a few problems of former techniques from the views of modeling and coding.

The model of the image is too simple. JPEG lossless mode encoder predicts only one of seven fixed predictors using three pixels nearby. The characteristics of the image as autoregressive model must be fully used. Also the potential of other image models such as Markov model or wave model are investigated.

The coding process has more room for improvement. For example, normally prediction error signals are mixed and encoded together, which cannot adapt the unstationarity of the image. The mapping from the prediction error to an integer, normally just rounding off, is too simple. There is a way to decrease the entropy. Also the probability density function of the error signal, normally approximated with Laplacian pdf, is not appropriate from the point of optimal fitting criterion.

1.4 Organization of the Paper

This paper is constructed from six chapters. The organization is as follows.

In chapter 2 the entropy coding models are introduced and discussed. Four methods and logics to reduce the code length after image modeling is discussed in the second chapter. These methods are actually applied to the images based on certain image models through coding experiments on actual images from chapter 3 to chapter 5, which are concerned with three image models and evaluate the compression performances.

In chapter 3, image coding based on the autoregressive model is discussed. The application of adaptive filter (RLS, recursive least square) is also discussed, and the effect of performance is evaluated. Its application for the multi-channel image is shown in appendix A.

In chapter 4, Markov model image representation, which have not been applied for gray-scale image coding, is applied to the gray-scale image coding in harmony with the

autoregressive model.

In chapter 5, wave model or transform coding, which is originally a lossy technique, is utilized for lossless image coding. The lossy image data is used for quick-previewing of the image and reconstruction of lossless image. By this technology, users get the great merit that they can browse the image before the lossless decompression.

Finally in the chapter 6, the outcome of the study is summarized and discuss the results which will be derived from the discernment obtained in this study.

Chapter

Entropy Coding and Code Length Reduction

GENERALLY the image signal changes gradually. This means that it is possible to somehow predict the value of a pixel using its surrounding pixels. Standard techniques of lossless still image coding consist of two stages: pixel-value prediction (remove the redundancy) and prediction error coding. This chapter deals with improving the performance of the latter stage. That is, we investigate several algorithms which reduce the final code length of image prediction error signals. Compared to the other methods which consider better prediction or coding schemes for lossless image coding, this area has not been studied much yet especially in terms of quantitative evaluation and still has room for improvement, as will be shown in this chapter.

The structure of this chapter is as follows. After summarizing the theory and algorithms of entropy coding in section 2.1, algorithms for reducing 0th order entropy using several conversion techniques is discussed in section 2.2. In section 2.3, signal splitting which enables groups of signals with different variances to be encoded separately, in consequence reduces the code length, is discussed. In section 2.4, the successful formulation of prediction error distribution under theoretical code-length criterion is introduced.

2.1 Entropy Coding

When the image redundancy according to the image model were completely removed, there would exist the statistical redundancy. When this residual signals can be considered uncorrelated, the amount of the data is decreased effectively via *entropy coding techniques*.

The concept of entropy is derived from thermodynamics. As statistical mechanics was developed to explain the behavior of molecular systems, it was natural to link the results to earlier thermodynamic measures. It turned out to be a particularly interesting thermodynamic parameter, for it was found to be a measure of the degree of "disorder" of the molecular system. There are close relations between entropy to measure physical disorder and that to measure information. If a symbol occurs that is very improbable, we would therefore expect that the information transferred in coding this symbol would be large, and vice versa. This concept is expressed by the following relationship. The amount of information I transferred in coding a symbol of probability p is given by

$$I = -\log_2 p.$$

The entropy H is the average information per symbol, i.e. the average code length per symbol:

$$H = -\sum_s p(s) \log_2 p(s)$$

In stationary systems, the entropy provides a fundamental lower bound for the compression that can be achieved with a given alphabet of systems. Therefore it is a very important and convenient measure of the performance of a coding system.

To actually code the redundant symbol down (as close as) to its entropy, many kinds of coding algorithms such as followings are introduced.

- Run length coding
- Prefix coding
  Such as Huffman coding, Golomb and Rice coding, Onoe-Iwashita coding, Wyle coding, Hasler coding, etc.
- LZ coding and its relatives
- Arithmetic coding and its relatives

2.1.1 Huffman coding

Huffman coding procedure[3] is usually used to assign a code for each alphabet if a set of integer-length codes is to be selected. This procedure assigns shorter code for a more probable symbol, longer code for a more improbable symbol. This Huffman code is compact, i.e. for a particular set of symbols and probabilities, no other integer code gives shorter code amount in total. Although Huffman code is not a new technique, it is simple and suitable for fast coding. Therefore it is quite widely used so far.

2.1.2 Arithmetic coding

The approach of arithmetic coding[6] is quite different from that of Huffman coding and other prefix coding. It does not require integer-length codes as Huffman coding does, it converts one stream into one code, in a manner of speaking. Arithmetic coding can naturally change the symbol probability table any time, which is a desirable property in coding unstable source or different sources.

Among these coding methods, arithmetic coding outperforms the others in terms of the code length or the code efficiency. From the experiments, the length of arithmetic code yields very close to the entropy (the difference is normally less than 0.01%), hence this can be mentioned as almost optimal coder, as far as we know the exact probability of each signal. In this paper, we use this as an entropy coder, so the problem is narrowed down how to model, and how to estimate the probability and transmit the probability information.

2.2 Entropy Reduction

For the lossless reproduction of digital (quantized, grayscale) images, the prediction error signals do not have to be stored exactly as they are, and can be quantized (or mapped) to an integer. An easy, and most commonly used, way to convert is simply round the error value e off to an integer (calculate $\lfloor e + 0.5 \rfloor$) and consider it as a 2's complement number. Besides this method, several others are presented in this section.

Let P denote the prediction value which is obtained as a function of causal surrounding pixels and let T represent a pixel value.

2.2.1 Order of distance conversion

One commonly used method for converting the prediction error to an integer (or mapping (T, P) to an integer) is, increasingly ordering $|T - P|$ for all T and emitting the position of $|T_{\text{actual}} - P|$. We name this method 'order of distance conversion', and it sometimes decreases the entropy significantly (the performance will be evaluated later). Let us call this conversion ODC (order of distance conversion).

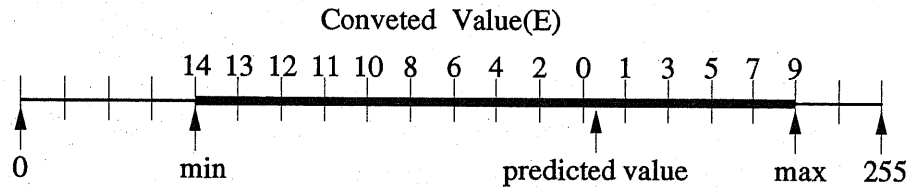


Figure 2.1: Order of distance conversion

Figure 2.1 explains the method visually. First we find the upper and lower bound of the pixel set (max, min). Then convert the actual pixel value into an non negative integer (In this figure, when the actual pixel is equal to 'max', $E = 9$.)

After the conversion, we can also get a non-negative integer E. This conversion is reversible, when the decoder gets the predicted value and the converted number E (and also the upper and lower bounds), the actual pixel value is found in the same way using the numerical line.

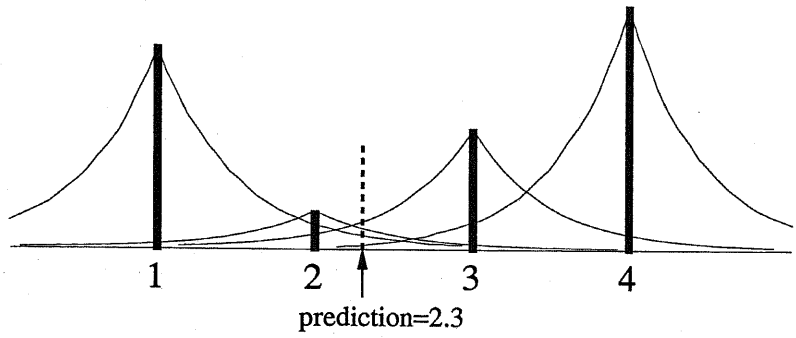


Figure 2.2: Histogram ($h(T)$) of each truth-value T (vertical-bars) and corresponding likelihood ($h(T)L_T(P)$).

2.2.2 Conversion with histogram and error distribution

Suppose we are paying attention to a certain pixel. Given a prediction value P for this pixel, the likelihood of its actual value, $L_P(T)$, is decreasingly distributed about P (commonly Gaussian or Laplacian distribution with mean P is used). Conversely, given a true value T, the distribution of its prediction value P, $L_T(P)$, becomes the bilaterally reversed $L_P(T)$.

But $L_T()$ does not reflect the pixel value probability. Therefore we apply the histogram information to $L_T()$. Let $h(x)$ denote the number of pixels with value x in the image. For each pixel value T, the likelihood of a certain prediction P is proportional to $h(T)$ and $L_T(P)$. Therefore, when a prediction P is given, the probable true pixel can be classified by comparing the value of T's function: $h(T)L_T(P)$.

We assume that $L_T(P)$ has a Laplacian probability density with mean T and variance σ^2, i.e.:

$$L_T(P) = \frac{1}{\sqrt{2}\sigma} \exp\left(-\sqrt{2}\,|P-T|\,/\sigma\right).$$

This is a realistic assumption which is supported by the literature.

The output of this method is a non-negative integer, which means the decreasing order of $h(T_{\text{actual}})L_T(P)$ among all $h(T)L_T(P)$. This number will be passed to the subsequent entropy coder. It is expected that the output signals concentrate toward zero, hence the entropy is anticipated to decrease.

For example, suppose we have a prediction value 2.3 (figure 2.2). The most likely truth-value with prediction value is '3', then '2', '1', '4', ... The output signal is the order of the truth-value.

Table 2.1: Comparison of 0th order entropy after conversion (unit=bit).

	crods	couple	girl	lena	zelda
raw	4.724	6.220	6.415	7.221	6.741
round off	5.432	4.315	4.683	4.164	4.956
ODC	5.416	4.304	4.680	4.159	4.952
CHED-A	2.455	3.756	3.743	4.151	4.420
CHED-B	2.454	3.754	3.745	4.151	4.395

Of course the shape of $L_T()$ is arbitrary, but the Laplacian distribution is suited for fast calculation because of the linearity of the index. The variance of $L_T()$ is obtained by scanning the whole image to calculate the variance of the prediction error.

As we are concerned with the ratios of $h(T)L_T(P)$, we have only to compare

$$h(T)\exp\left(-\sqrt{2}\,|P-T|\,/\sigma\right).$$

Hence if we calculate

$$\exp(-1/(\sqrt{2}/\sigma)),\ \exp(-2/(\sqrt{2}/\sigma)),$$
$$\exp(-3/(\sqrt{2}/\sigma)),\ \cdots$$

beforehand, the inside of comparison loop does not require exponential calculation.

Let us call this conversion CHED (conversion with histogram and error distribution). Note that this can be considered as the generalized version of ODC, which has a linear $L_T(P)$ and a constant $h(T)$.

2.2.3 Experimental results

We tested these methods on several 8-bit images. The histogram information is quantized down to 8-bits per level (which means 256 bytes of overhead). Table 2.1 shows the 0th order entropy of each method's output.

CHED-A uses a fixed histogram, CHED-B updates its histogram for each symbol. For some images, ODC reduces the entropy much more than rounding off, but CHED always outperforms ODC.

		f	
	c	b	d
e	a	X	

Figure 2.3: Surrounding pixels (a~f) of the noticing pixel (X).

2.3 Signal Splitting

If two or more symbol sources with different distributions are mixed together, the entire entropy is always larger than the average of the entropy of each source[7]. Conversely it means that when a source could be split into two or more sources which have different distributions without any additional information, the total code length could be shortened.

This fact is implicitly used in literatures, sometimes "estimation of the variance of the prediction error" in name. For example, Howard[8] proposes a method looking for the similar pixel area around the noticing pixel within an image, and calculate the variance of the prediction error among such areas. Kuroki *et al.*[9] simply calculate the variance from prediction errors of a certain size of surrounding area of the noticing pixel.

In this section several methods to achieve this are presented. All of these methods use the value calculated from the surrounding pixels (e.g. figure 2.3) to classify the current pixel. Let us name this value the *'roughness value'*. To calculate this roughness value, no additional information is needed for the decoder, because they are all taken from already-processed image area.

One method is, using surrounding pixels, simply calculate the value

$$\big(|a-b|+|c-a|+|c-b|+|d-a|+|e-b|+|f-b|+|b-d|\big)/20$$

and use this for classification. Let us call this simple calculation the 'ABS' method.

A slightly more complex one is, calculate the linear combination of

$$|a-b|, |a-c|, |a-e|, |b-c|, |b-d|, |b-f|$$

to estimate absolute actual prediction error $|E|$. The coefficients are obtained via the least square method. Let us call this calculation the 'LSE' method.

This is similar to Wu's method[10], which uses $|a-\hat{X}|$, $|b-\hat{X}|$, $|c-\hat{X}|$, ... where $\hat{X}$ is the prediction value of X.

Likewise $|a-\hat{a}|$, $|b-\hat{b}|$, $|c-\hat{c}|$, ... can be used for linear combination. Let us call this variant 'method Wu2'.

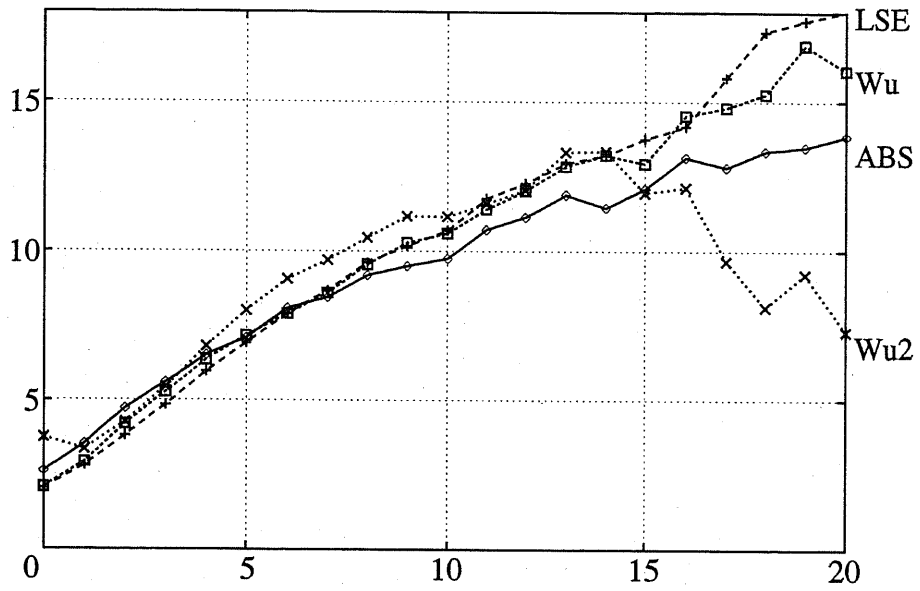


Figure 2.4: Relation of the roughness value (x-axis) and the sample standard deviation (y-axis).

2.3.1 Experimental results

In the experiment, all the methods use six surrounding pixels to calculate the roughness value, and 24 pixels to predict.

Figure 2.4 shows the results for 'lena'. Each algorithm yields high correlation. The ABS method takes around 30% less time than the other three.

The performances of each methods in terms of the contribution to the code length is shown in table 2.2. Each prediction error is classified into one of ten groups according to this roughness value. Then the theoretical code length according to the entropy is calculated and accumulated for each group, and shown in the table. This length is always shorter than the bulk length calculated without classification (the left column of the table). The contribution of this method is obvious, and as far as using six surrounding pixels, LSE works quite well. It is interesting that 'method Wu', which uses single p, outperforms 'method Wu2', which uses prediction value of each pixel.

Table 2.2: Code length comparison (unit=bytes, inside parentheses: reduction compared with 'no classification' in percentage).

	lena	girl	washdc1
No classification	134042	36975	115798
ABS	129401(3.46%)	35669(3.53%)	113758(1.76%)
LSE	128318(4.27%)	35433(4.17%)	112998(2.42%)
Wu	128351(4.25%)	35506(3.97%)	112971(2.44%)
Wu2	130915(2.33%)	35810(3.15%)	113940(1.60%)

2.4 Modeling of Prediction Error Distribution

Most image coding systems consist of image modeling and error coding. For lossless image compression, the AR (autoregressive) model is adopted quite often for performance reason. And the coding part consists of entropy coding of prediction error which is a penalty of the model approximation. As the efficiency of the coding stage is directly combined with the performance of the system, estimation of prediction error distribution plays an important role in image coding.

The pdf (probability density function) of the error signal is said to be fairly well approximated by the Laplacian pdf. This is widely used for image coding because of the good behavior for analytical calculations, but there are also many examples which disagree with this pdf[8]. Instead of this, the generalized Gaussian pdf was first proposed to fit the subband signal (DPCM for DC part) for images by Westerink *et al*[11].

From observation, this fits rather well for the tails of actual error signals' pdf, but not for the central blunt (often round) peak. Some papers refer to the quantized discrete distribution of prediction error to explain this rotundity partly, but not successfully at all, because integer quantization is the same as convolving $U(-0.5, 0.5)$ which influences very narrowly.

In this section, after examining the actual error distribution and see how traditional models fail to represent it in section 2.4.1, Then in section 2.4.2 we study on several pdf's which seem to fit the error distribution curves. Then clarify the assumptions on image characteristics in section 2.4.4, then derive a new formulation for prediction error pdf considering the effect of additive white Gaussian noise and quantization noise of each image pixel upon linear prediction error signals.. Finally in section 2.4.5, this model is evaluated through comparison with other analytic/ non-analytic functions using actual error signals from an image database. From the experimental results, our model fits the actual error distribution better than classical distributions or their quantized distributions.

2.4.1 Prediction error distribution

Typical distributions of prediction error signal are shown in figure 2.5 or figure 2.6. The shape looks like Mt. Fuji or a bell. Such a distribution is obtained not only from the direct linear image prediction error but also from the subband images in subband coding and also from the DCT coefficients of the image (DPCM data for the DC part, raw data for the AC part). These signals are all generated via linear combination of digital image pixels.

Followings are empirically observed as the common features for such data,

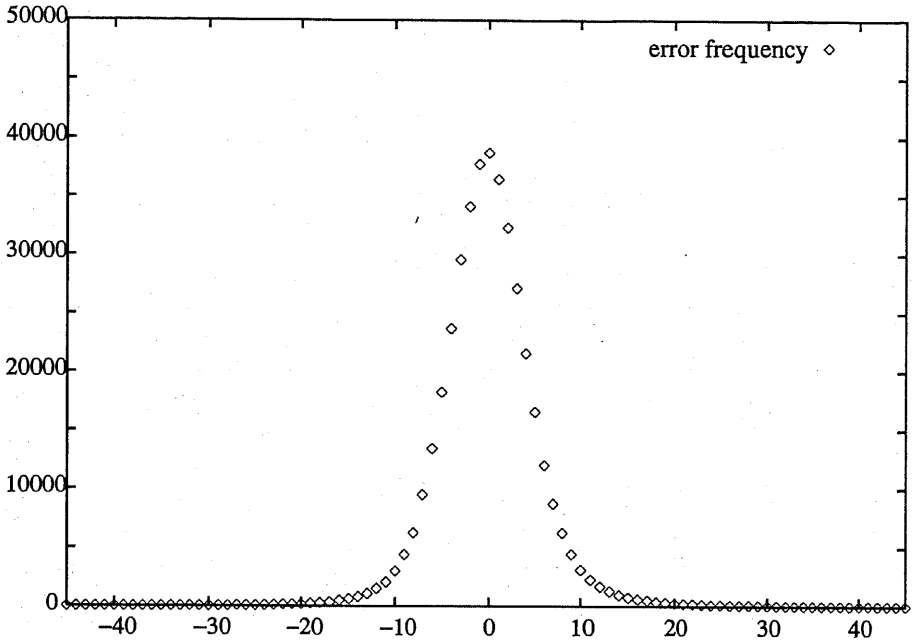


Figure 2.5: Typical prediction error distribution.

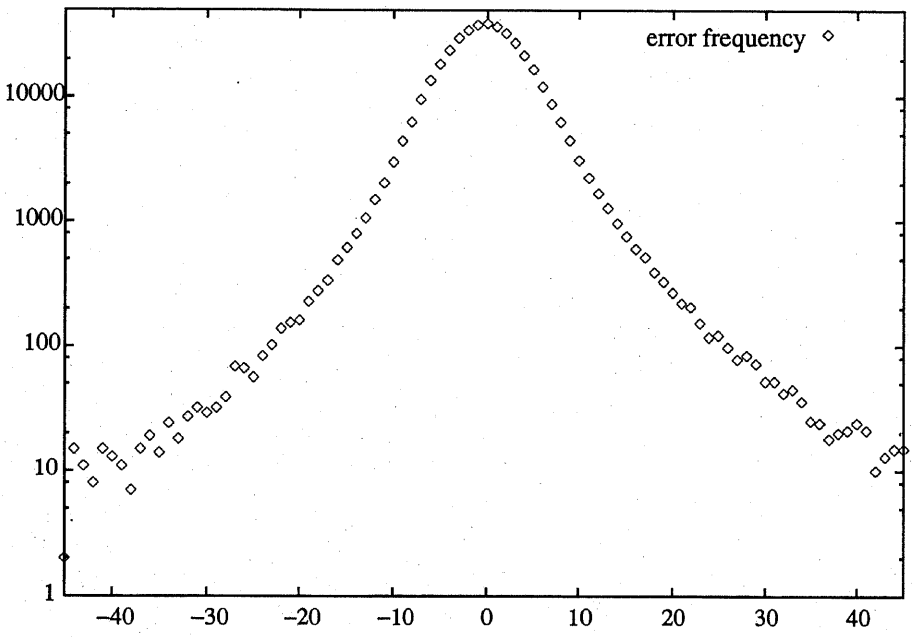


Figure 2.6: Typical prediction error distribution (log-scale).

- quasi-symmetric
- monotonically decrease from the center to both tails

From the first glance of the distribution graph, symmetric exponential curve (or Laplacian) and Gaussian curve may come up to mind. As the consequence, in the history of image coding, Laplacian pdf and Gaussian pdf have long been believed and used to formulate those graphs[12].

But unfortunately this is mere a myth; figure 2.7 and 2.8 show the Laplacian and Gaussian functions with the same variance and magnitude. Although the original error distribution exposes a smooth shape, the fitting result is not a satisfactory one at all. The aim of this section is investigate the mechanism of digital image prediction and formulate its error distribution[13].

2.4.2 Summary of typical density functions

Here listed some of the typical bell-shaped density functions with zero-mean, σ^2 variance. Most of them are examined for the actual error distribution fitting experiment in this section.

Uniform (or rectangular) pdf

$$U(-\sqrt{3}\,\sigma, \sqrt{3}\,\sigma) = \begin{cases} \dfrac{1}{2\sqrt{3}\,\sigma} & x \in (-\sqrt{3}\,\sigma, \sqrt{3}\,\sigma) \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

Gaussian (or Normal) pdf

$$\mathrm{G}(x) = N(0, \sigma^2) = \frac{1}{\sqrt{2\pi}\,\sigma} \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{2.2}$$

Laplacian (or two-sided exponential) pdf

$$L(x) = \frac{1}{\sqrt{2}\,\sigma} \exp(-\sqrt{2}|x|/\sigma) \tag{2.3}$$

Generalized Gaussian pdf

$$\mathrm{GG}(x) = \phi_p(x) = \frac{p}{2\Gamma(1/p)\gamma\sigma} \exp\left(-\left|\frac{x}{\gamma\sigma}\right|^p\right) \tag{2.4}$$

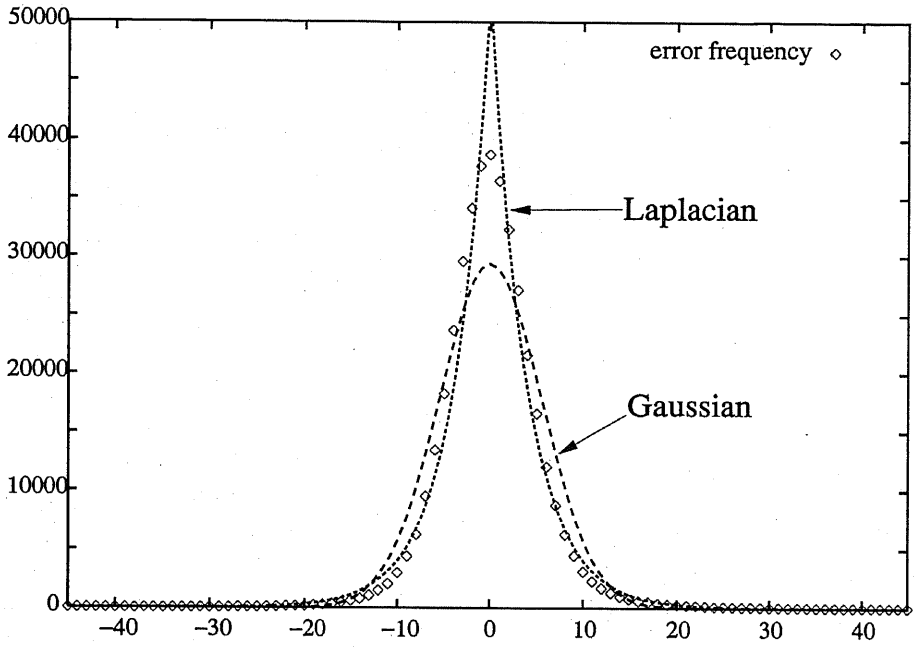


Figure 2.7: With Laplacian/ Gaussian distribution with the same variance.

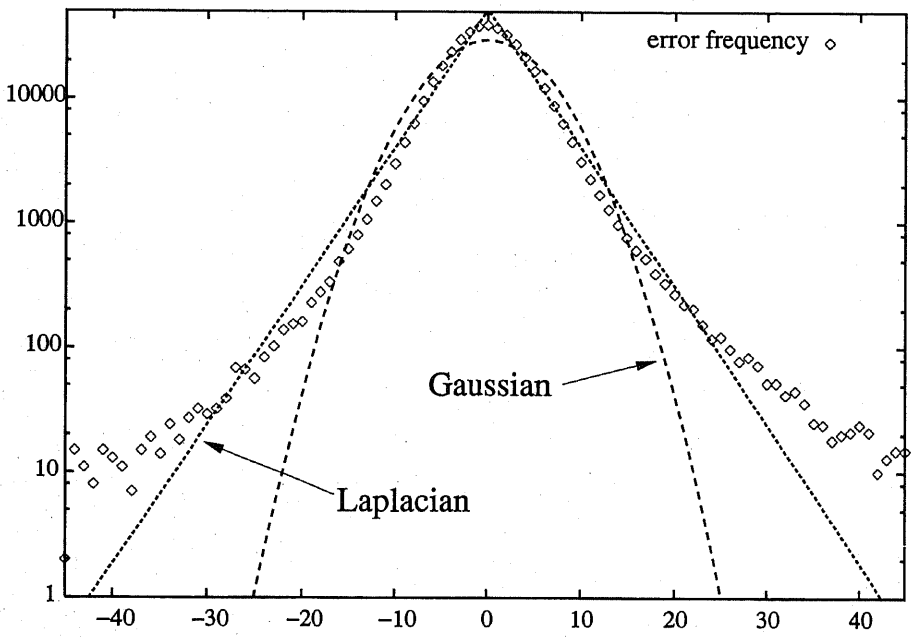


Figure 2.8: With Laplacian/ Gaussian distribution with the same variance (log-scale).

$$\gamma = \sqrt{\Gamma(1/p)/\Gamma(3/p)},$$
$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}\mathrm{d}t \qquad \text{for } x > 0.$$

p is called *the shape parameter*, σ^2 is the sample variance. This function can represent the Laplacian distribution (when $p = 1$), Gaussian distribution ($p = 2$) and uniform distribution ($p = \infty$). The graphs with different p's are shown in figure 2.9 and figure 2.10.

Gamma pdf

$$g_\alpha(x) = \frac{\beta^\alpha}{2\Gamma(\alpha)}|x|^{\alpha-1}\exp(-\beta|x|) \tag{2.5}$$
$$\beta = \frac{\sqrt{\alpha(\alpha+1)}}{\sigma}$$

The particular case of $\alpha = 1/2$ is sometimes referred as mere "gamma distribution"[14, 15]:

$$g_{1/2}(x) = \frac{\sqrt[4]{3}}{\sqrt{8\pi\sigma|x|}}\exp\left(-\frac{\sqrt{3}|x|}{2\sigma}\right) \tag{2.6}$$

When $0 < \alpha < 1$, this distribution has a singular peak at $x = 0$. This peak is very acute, as is clear from $\lim_{x\to 0} g(x) = +\infty$. When $\alpha = 1$, this is the same as Laplace distribution.

We tried to fit this gamma function for actual error distribution, but the peak is too acute to obtain good results compared with other fitting functions. Therefore this does not appear in the following experiments.

Figure 2.13 and figure 2.14 show the graphs of gamma pdf with various α. They are all zero-centered and of $\sigma^2 = 1$, too. The very acute peaks are observed and from log-scale graph (figure 2.14), it is understood that this is the same as Laplace pdf when $\alpha = 1$.

Cauchy pdf

$$\mathrm{C}_c(x) = \frac{1}{\pi}\frac{c}{c^2 + x^2} \tag{2.7}$$

where c is the extent parameter, this has heavier tails than those of the generalized Gaussian density (see figure 2.12). Because of the thick tail, its moments such as variance or mean amount infinity.

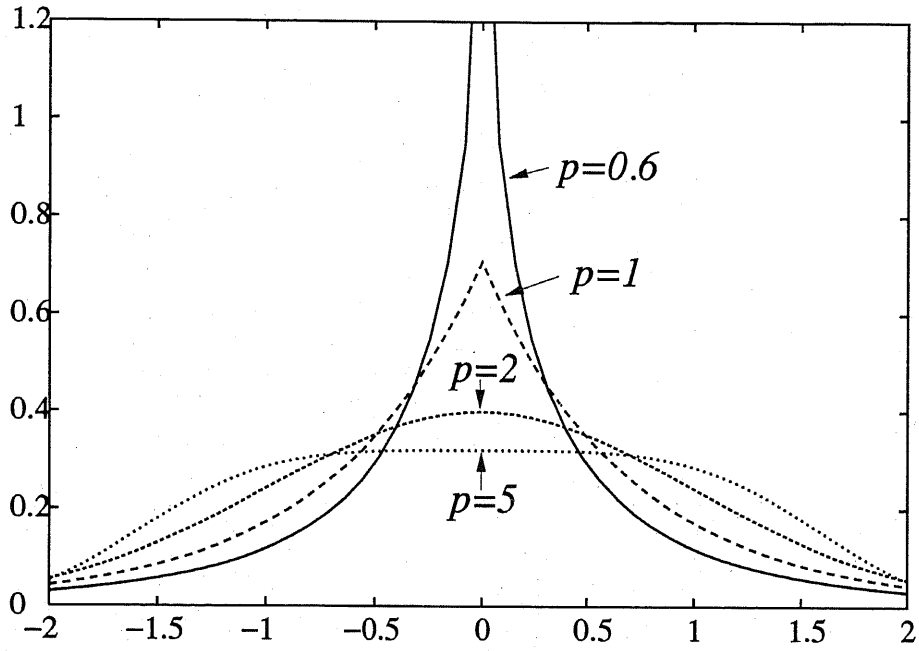


Figure 2.9: The generalized Gaussian pdf with various shape parameters.

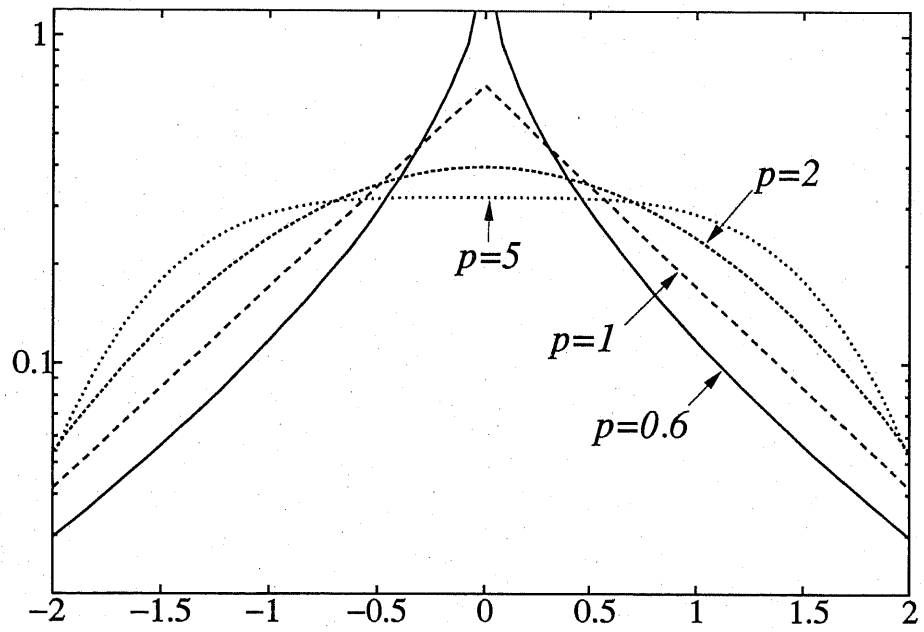


Figure 2.10: The generalized Gaussian pdf with various shape parameters (log-scale).

The graphs of these functions are shown in figure 2.11 and figure 2.12(log-scale). They are all zero-centered and of $\sigma^2 = 1$ except for Cauchy function, which is of $c = 0.3$.

2.4.3 On the generalized Gaussian pdf

For practical image coding, Laplace distribution works fairly well and therefore widely have been used practically[8, 16], but it is not the best choice unfortunately, which is shown in this section (e.g. figure 2.7 and 2.8). The desirable distribution is such a pdf that smoothly connect Laplacian and Gaussian distribution. It was initially introduced by Subbotin[17] in 1923, and consequently sometimes called Subbotin's density. This is what now widely called "the Generalized Gaussian pdf".

To author's knowledge the first literature which proposed the generalized Gaussian pdf for fitting DPCM error signal for subband image was introduced by Westerink *et al*[11]. Gamma pdf is tested by Bellifemine *et al.* for 2D-DCT coefficients[15] (not for the prediction error), with even worse results than Laplacian pdf.

In spite of Subbotin's successful generalization, this still does not fit the actual error distribution sufficiently, which will be quantitatively proved later in this section. But experimentally his pdf survives as the core of our proposal pdf, or our assumption 2 in section 2.4.4.

2.4.3.1 Estimation of the Shape Parameter p

In generalized Gaussian pdf, There is a simple relation between the variance σ^2, mean of the absolute values ($E[|x|]$), and the shape parameter p. From the definition of the generalized Gaussian function (equation(2.4)),

$$\frac{\sigma^2}{E^2[|x|]} = \frac{\Gamma(1/p)\Gamma(3/p)}{\Gamma^2(2/p)} \tag{2.8}$$

is easily derived. Hence, once a lookup-table of the right side value of equation(2.8) is made, p can be estimated from the table (see figure 2.15). This is a straightforward and quick estimation method[1]. I adopt this technique later in this section.

2.4.4 Formulation of prediction error pdf

In this section we make the following assumptions:

[1] In 1995 Sharifi *et al.* proposed this method[18]. However, the work offers nothing new, because Subbotin's original paper[17] already used it in 1923.

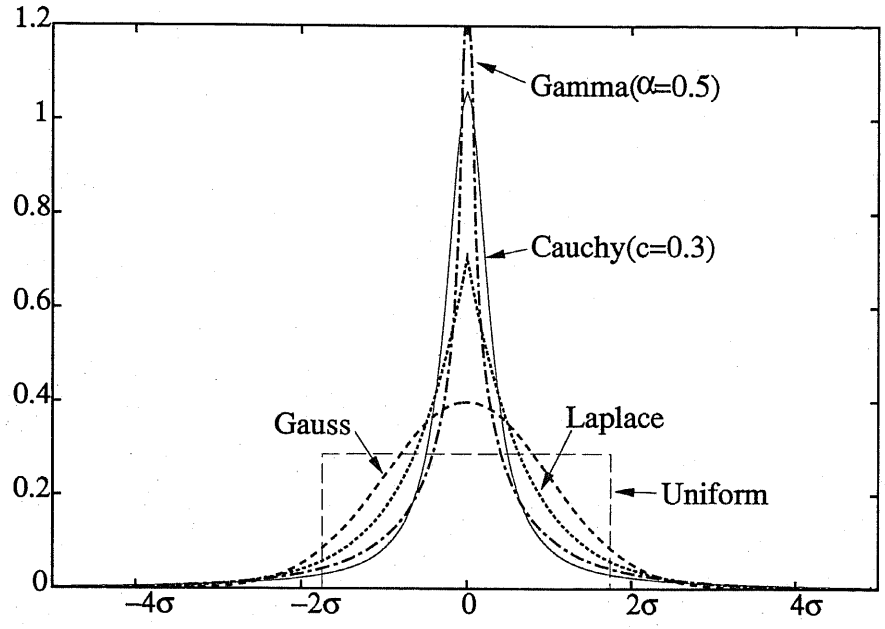


Figure 2.11: Comparison of typical bell-shaped pdf's. All zero-centered and of $\sigma^2 = 1$ except for Cauchy function, which is of $c = 0.3$.

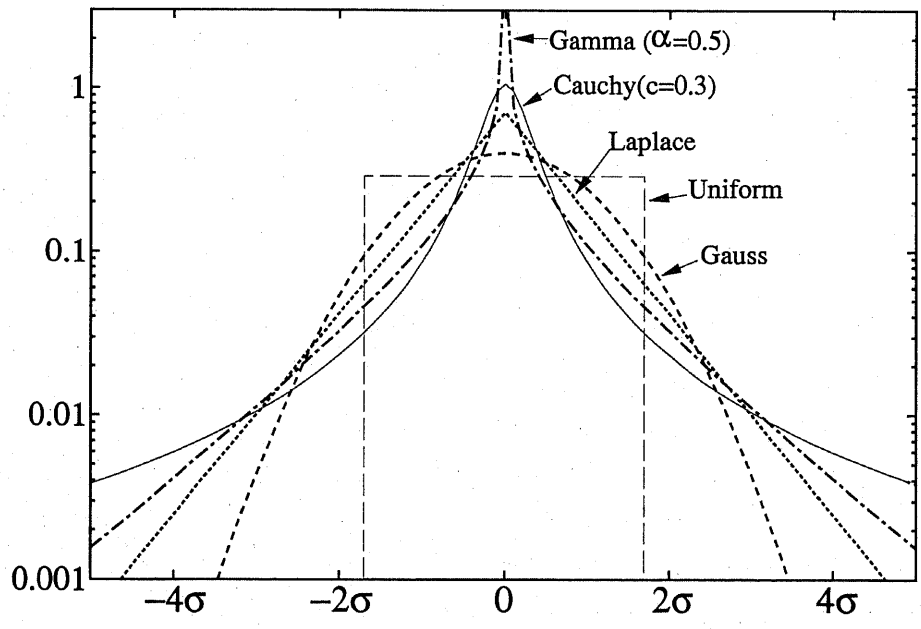


Figure 2.12: The same graphs as figure 2.11. (log scale of y-axis)

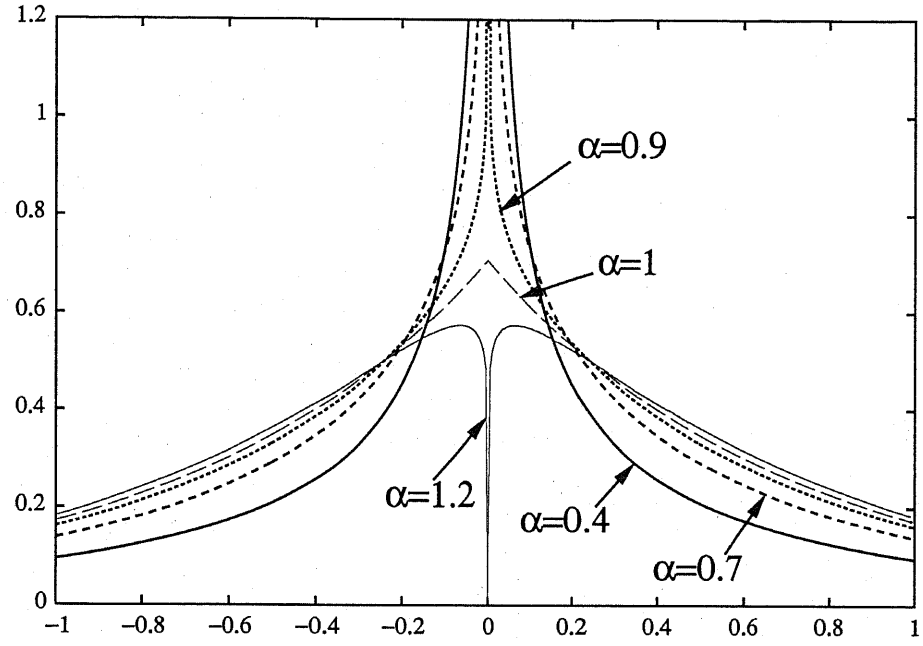


Figure 2.13: Graphs of gamma pdf with various α.

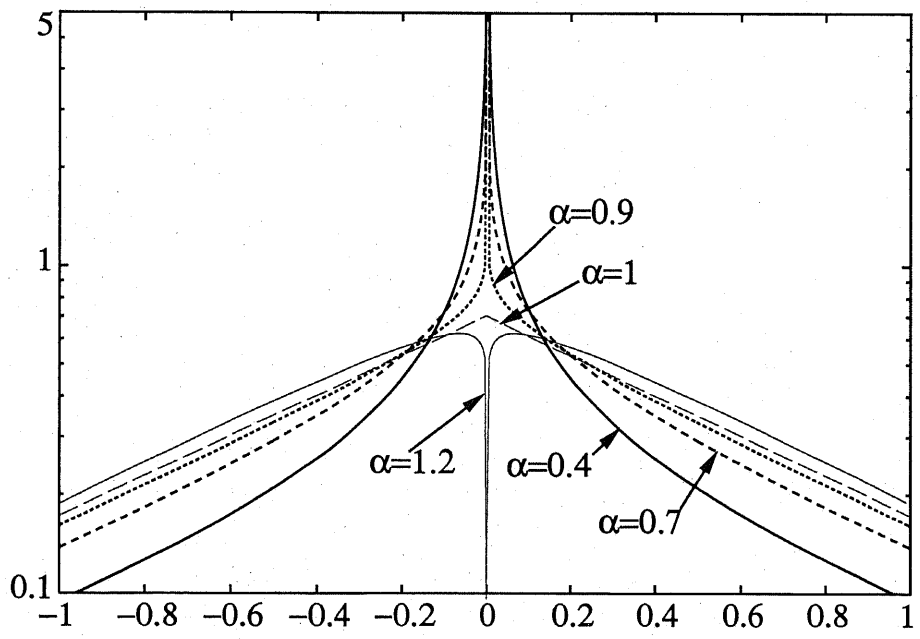


Figure 2.14: Graphs of gamma pdf with various α (log-scale).

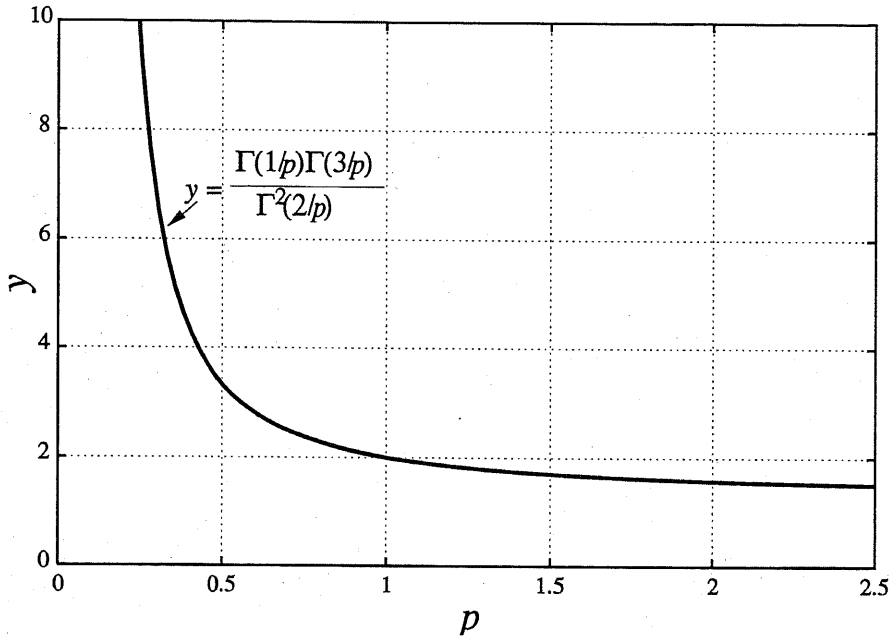


Figure 2.15: The graph of $y = \Gamma(1/p)\Gamma(3/p)/\Gamma^2(2/p)$.

Assumption 1: Each pixel in a digital image has additive white Gaussian noise with mean zero and the same deviation, and is uniformly quantized.

Assumption 2: The image signal can be approximated by the linear autoregressive model, and prediction error signals calculated from truth pixel values are distributed with the generalized Gaussian pdf.

Under assumption 1, digital pixel value X (integer) is represented as

$$X = \lfloor \dot{X} + w + 0.5 \rfloor , \tag{2.9}$$

where $\dot{X}$ denotes the *imaginary* true pixel value of X, w denotes the additive Gaussian noise on $N(0, \sigma_w^2)$. $N(\mu, \sigma^2)$ denotes the pdf of Gaussian (or normal) distribution with mean μ and variance σ^2.

As truncation (or quantization) is the same process as to add a random variable u taken from an uniform distribution $U(-0.5, 0.5)$, equation(2.9) becomes

$$X = \dot{X} + w + u. \tag{2.10}$$

Let N_X denote the number of surrounding pixels used for linear prediction, C_i denote the i-th prediction coefficient. This subscript i means the position used for the linear

prediction, for example,

$$X_i = \dot{X}_i + w_i + u_i \quad (i = 1 \cdots N_X), \tag{2.11}$$

$$\hat{X} = \sum_{i=1}^{N_X} C_i \dot{X}_i, \tag{2.12}$$

where $\hat{X}$ is the linear prediction value of the pixel X.

Then the assumption 2 can be described as

$$\dot{X} - \hat{X} = e. \tag{2.13}$$

e, a prediction error, is a random variable taken from a certain generalized Gaussian pdf: $P_e = \phi_{p,\sigma_e}$.

From equations (2.10), (2.11), (2.12) and (2.13), omitting the range of summation, the following are derived.

$$\begin{aligned} \dot{X} &= e + \hat{X} \\ X - w - u &= e + \sum_i C_i (X_i - w_i - u_i) \\ X - \sum_i C_i X_i &= e - \sum_i C_i w_i + w - \sum_i C_i u_i + u \\ &= e - w_{\text{sum}} - u_{\text{sum}}, \end{aligned} \tag{2.14}$$

where $w_{\text{sum}} = \sum_i C_i w_i - w$ and $u_{\text{sum}} = \sum_i C_i u_i - u$. Left side of equation(2.14) is what we obtain from a linear predictor as a prediction error. Hence with equation(2.14), the truncated linear prediction error signal ϵ is

$$\begin{aligned} \epsilon &= \left\lfloor X - \sum_i C_i X_i + 0.5 \right\rfloor \\ &= e - w_{\text{sum}} - u_{\text{sum}} + u_\epsilon \\ &= e + s, \end{aligned} \tag{2.15}$$

where u_ϵ is a random variable taken from $U(-0.5, 0.5)$ and $s = -w_{\text{sum}} - u_{\text{sum}} + u_\epsilon$.

Because of the Gaussianity of random variables w_i and w (assumption 1), w_{sum} is also a random variable with $N(0, \sum_i C_i^2 \sigma_w^2 + \sigma_w^2)$. By using *the central limit theorem*, when N_X is large enough (in our case around 20 or more), random variable $u_{\text{sum}} + u_\epsilon$ is approximately distributed with $N(0, \sum_i C_i^2 \frac{1}{12} + \frac{2}{12})$, because the variance of $U(-0.5, 0.5)$ equals $1/12$.

To validate the ground of approximation as Gaussian, I made the artificial convolution data (figure 2.16). It is observed that convolving only three or four uniform pdf's yields distinct bell-shape.

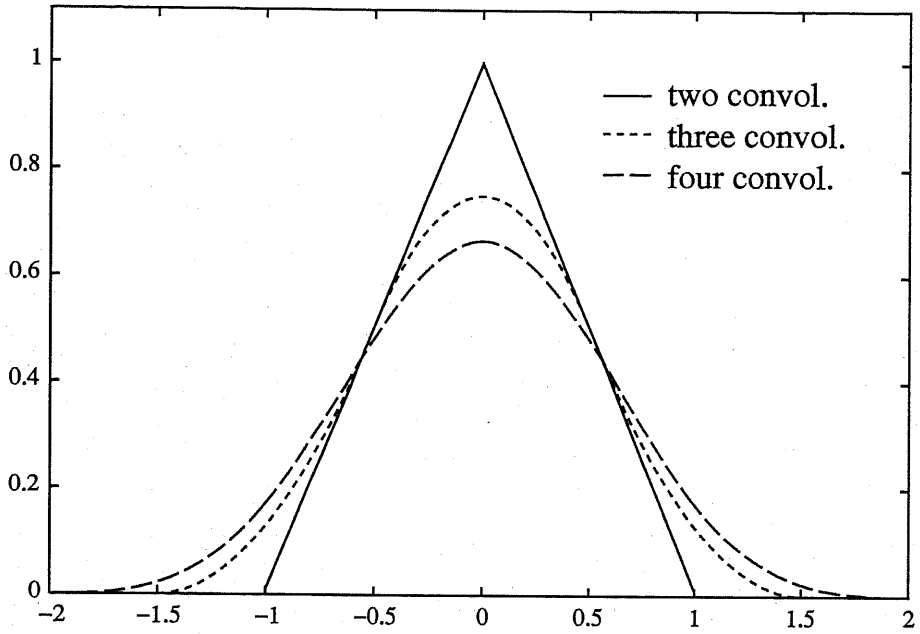


Figure 2.16: Convolution of several uniform pdf's (two, three, four $U(-0.5,-0.5)$'s)

Also figure 2.17 is the comparison of the graphs of convolution of $C_i u_i$ with actual

$$\begin{aligned} C_i \;=\; & (0.6200, 0.7182, -0.3047, 0.1252, -0.1687, -0.2952, 0.0732, \\ & 0.0391, -0.0619, 0.1085, -0.0056, -0.0041, 0.0666, 0.1283, \\ & -0.0258, 0.0230, -0.0469, 0.0032, 0.0067), \end{aligned}$$

which is obtained from SHD-PA085 blue component (explained later), and the Gaussian pdf with $\sigma^2 = \sum_i C_i^2/12$. The graph of convolution has a strong resemblance to Gaussian, hence s can be regarded as a "Gaussian" random variable with the pdf of

$$P_s(x) = N(0, \sigma_s^2), \tag{2.16}$$

where

$$\sigma_s^2 = \left(\sum_i C_i^2 + 1\right)\sigma_w^2 + \left(\sum_i C_i^2 + 2\right)\frac{1}{12}. \tag{2.17}$$

Finally from equation(2.15), the pdf of ϵ, $\mathrm{GG_G}(x)$ can be represented as the convolution of $P_e(x)$ and $P_s(x)$:

$$\mathrm{GG_G}(x) = \int_{-\infty}^{\infty} P_e(t)P_s(x-t)\mathrm{d}t, \tag{2.18}$$

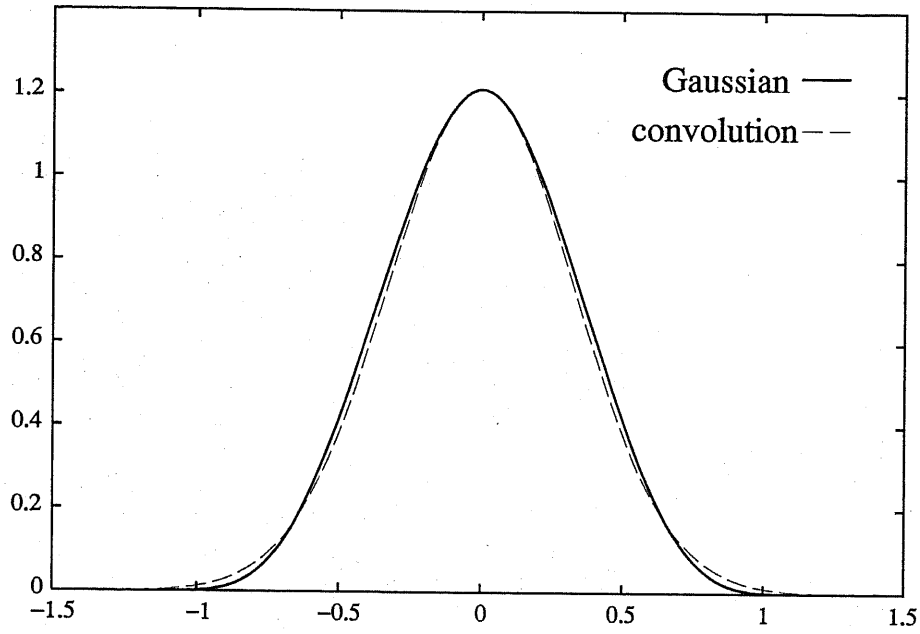


Figure 2.17: Convolution of 19 (= N_X)uniform pdf's with actual weights C_i, and the Gaussian pdf with the same variance.

where "GG" stands for the *"generalized Gaussian"* and subscript "$_G$" means *"with Gaussian convolution"*.

2.4.5 Experiments and discussions

In the previous section I have proposed the prediction error distribution model and formulated its pdf (equation(2.18)). In this section I will see how this model fits the actual data through experiments.

Criterion for the model evaluation

To know which fitting function works best, the criterion of fitness must be chosen carefully. Giving consideration on subsequent entropy coding, I adopt *"the theoretical code length"* according to the pdf fitting function f, i.e.:

$$I(f) = -\sum_s y_s \log_2 f(s) \quad \text{(bits)}, \tag{2.19}$$

	X_3	X_4	X_5
X_2	X_1	X	

Figure 2.18: Location of neighbor pixels $X_1 \ldots X_5$ to classify X

where y_s means the number of occurrences of symbol s, $f(s)$ is the fitting function which satisfies $\sum_s f(s) = 1$.

The minimum boundary of this criterion is given via calculating the entropy:

$$I_{\min} = -\sum_s y_s \log_2 \frac{y_s}{\sum_s y_s} \quad \text{(bits)} \tag{2.20}$$

Source splitting

The simple application of autoregressive prediction for an entire image yields the mixture of prediction error signals with different distributions. Such signals with different distributions are desirable to be coded separately, if possible, to reduce the total code length. In this section I use a simple source splitting method, which uses the *roughness* value r around a pixel, obtained from

$$r = |X_2 - X_1| + |X_3 - X_1| + |X_4 - X_1| + |X_5 - X_1|. \tag{2.21}$$

$X_1 \ldots X_5$ are shown in figure 2.18. Further discussion about the source signal splitting is in section 2.3.

Test images

Low noise-contaminated images with high quality and resolution are desirable for the experiment. I choose four images from SHD (Super High Definition) still images (V.3.0A) by NTT Optical Network Systems Laboratories, which have 12 bits dynamic range for each component of R, G and B and resolution of 4096×4096 pixels (figure 2.19–2.21). I also tested the gray image obtained from these R,G,B components according to

$$\text{gray value} = 0.2998\text{R} + 0.5868\text{G} + 0.1144\text{B}. \tag{2.22}$$

I used 1024×1024 cropped area for the small data set and 2048×2048 cropped area for the large data set, at the center of the image.

Figure 2.19: SHD image (PA085)

Figure 2.20: SHD image (SA002)

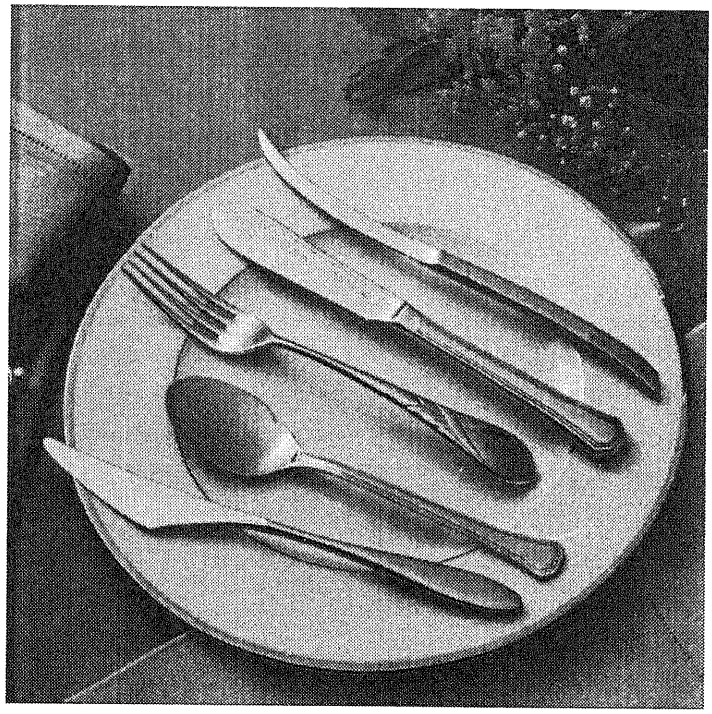


Figure 2.21: SHD image (SA003)

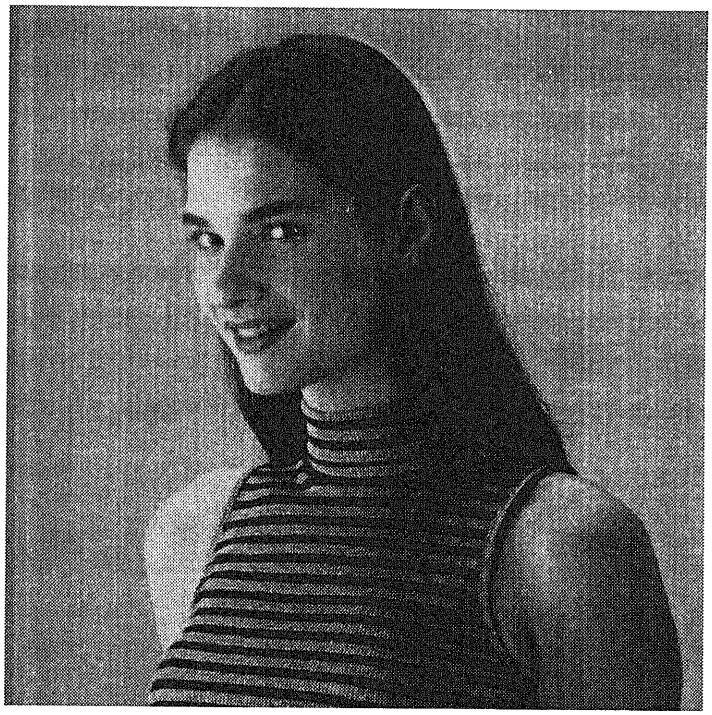


Figure 2.22: SHD image (PC010)

Table 2.3: Functions tested for fitting

GG_G	Generalized Gaussian distribution + Gaussian convolution (proposed method)
GG	Generalized Gaussian distribution
GG_T	Generalized Gaussian distribution + truncation
G	Gaussian distribution
L	Laplacian distribution
L_G	Laplacian distribution + Gaussian convolution
C	Cauchy distribution
C_G	Cauchy distribution + Gaussian convolution
S7...10	Natural cubic spline function (with 7..10 nodes for each)
logS7...10	Same as above, but fit for log-y-axis.

Fitting functions

The functions which I compared with the fitting performance of our model for the experiment are shown in table 2.3.

As prediction error is normally truncated before entropy coding, I also compare our model with GG_T, the discretized generalized Gaussian distribution:

$$GG_T(i) = \int_{i-0.5}^{i+0.5} GG(t)dt \tag{2.23}$$

This is the same process as convolving GG with a uniform pdf $U(-0.5, 0.5)$.

As convolving Gaussian with Gaussian yields also Gaussian pdf (i.e. G_G=G), "Gaussian distribution + Gaussian convolution" is not tested. And as spline function is not analytic and likely to represent the effect of Gaussian convolution, spline functions are tested without convolution, either.

Experimental results

By multi-dimensional search, I find the optimal (in terms of minimizing the criterion $I(f)$) sets of parameters (for GG_G: $(\mu_e, p, \sigma_e, \sigma_s)$, for S7: seven pairs of node coordinates and so on). To determine initial values of μ and σ_e, sample mean and root mean square of the error signal are used respectively. And to determine initial value of shape parameter p, I use the method described in section 2.4.3.1.

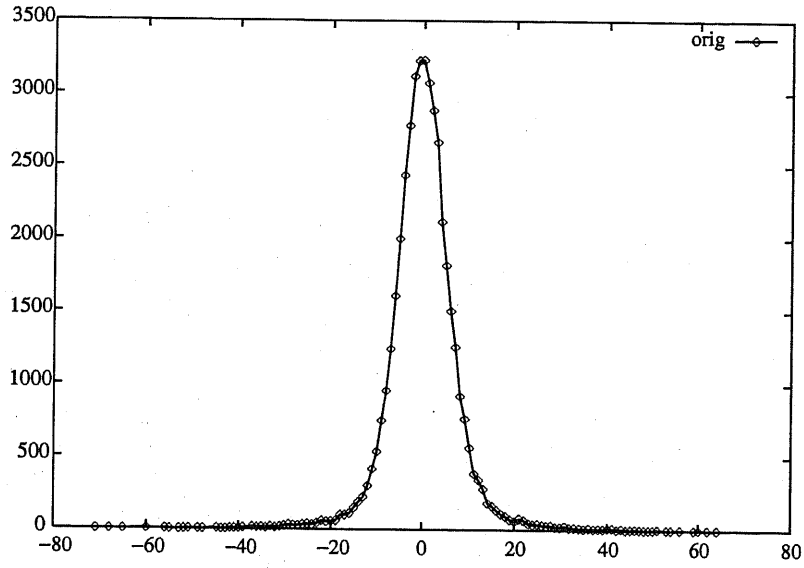


Figure 2.23: Example of error distribution, obtained from PA085 (green) image ($r \leq 20$).

Figure 2.23 and 2.24 show the error distribution of image 'PA085' (green component). Pixels which satisfy $r \leq 20$ (44062 pixels) are used and nineteen surrounding causal pixels are used for linear prediction (coefficients are calculated by least square error method). Table 2.4 shows the result of fitting for this data. Graphs of fitting results for four methods (GG_G, logS9, C_G and L) are shown in figure 2.25. It is obvious that traditional Laplacian pdf is not suited at all for this case. And table 2.5 shows the average performance of each fitting function applied for 16 images (i.e. RGB and gray components for PA085, SA002, SA003 and PC010) under the same condition (i.e. $r \leq 20$). In the tables, 'rate' means

$$\text{rate} = 100\left(\frac{I(f)}{I_{\min}} - 1\right). \tag{2.24}$$

The results for the same image (PA085 green), more rough area ($81 \leq r \leq 90$) are shown in figure 2.26, figure 2.27, table 2.6 and table 2.7. For this data, the feature of generalized Gaussian arises more than previous one. On the other hand Cauchy feature seems to decrease.

The tendency of the superiority of GG_G is even more apparent for large data set. Table 2.8 and 2.9 are the fitting results obtained from 2048×2048 area (four times wider than previously experimented area) of the same image data.

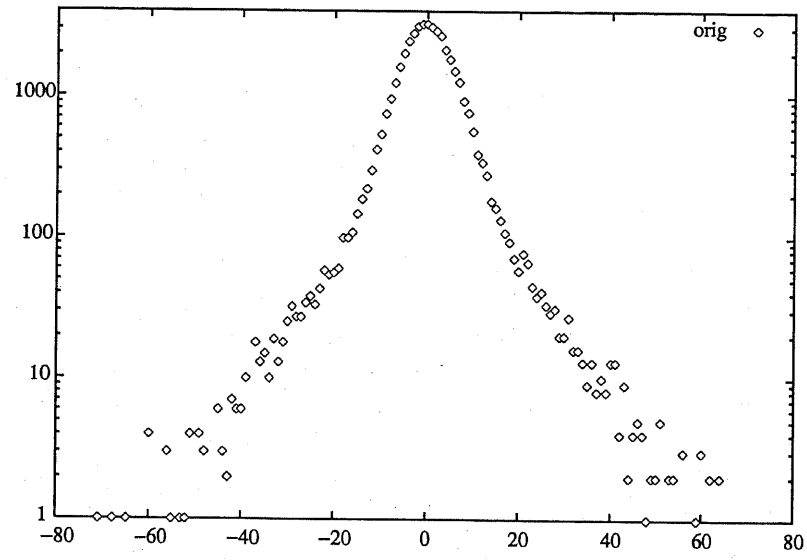


Figure 2.24: The same error distribution as figure 2.23, with logarithmic y axis.

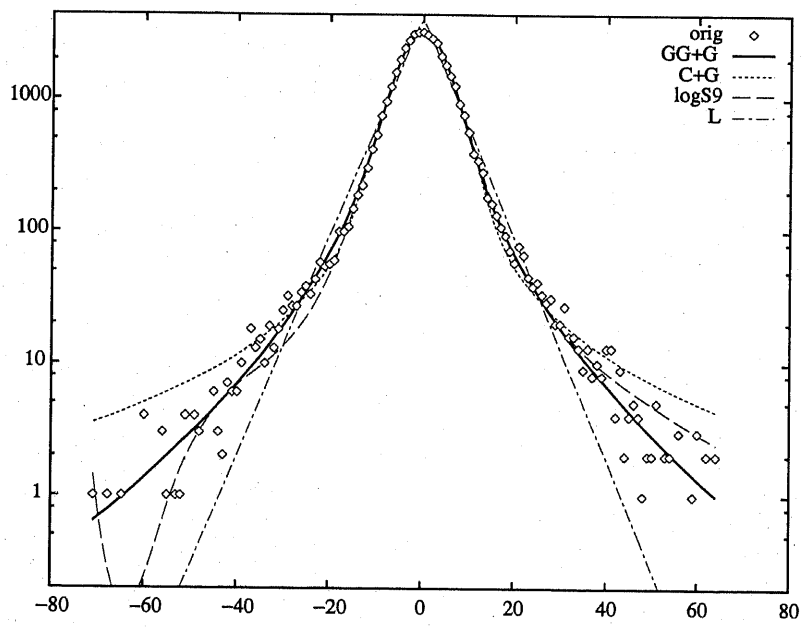


Figure 2.25: Fitting curves of GG_G, logS9, C_G and L_G, with logarithmic y axis.

Table 2.4: Fitting length comparison for PA085(green), $r \leq 20$, 44062 symbols obtained, entropy=4.795bits/ symbol, variance=7.645^2.

f	parameters	rate
GG_G	$\sigma_s = 4.15, p = 0.423, \sigma_e = 6.81, \mu = -0.0926$	0.0839
logS9		0.0973
logS8		0.1189
logS10		0.1833
C_G	$\sigma_s = 4.59, c = 1.24, \mu = -0.0790$	0.1951
logS7		0.2484
L_G	$\sigma_s = 2.14, \sigma_e = 6.66, \mu = -0.112$	0.4030
S10		0.4430
GG	$p = 1.07, \sigma_e = 7.11, \mu = -0.196$	0.5615
L	$\sigma_e = 7.27, \mu = -0.231$	0.5767
GG_T	$p = 0.869, \sigma_e = 7.65, \mu = -0.134$	0.7127
S9		0.8350
C	$c = 3.69, \mu = -0.170$	1.7933
G	$\sigma_e = 6.41, \mu = -0.0022$	1.9004
S8		2.3826
S7		3.6739

Table 2.5: Average fitting rate for 16 images ($r \leq 20$)

f	rate
logS9	0.4149
GG_G	0.4407
logS10	0.4698
logS7	0.5353
L_G	0.5542
GGvp	0.5962
C_G	0.6150
logS8	0.6317
GG_T	0.6900
L	0.7687
S10	0.9286
S9	1.0303
G	1.3231
S8	1.6904
C	1.7756
S7	2.3316

Table 2.6: Average fitting rate for 16 images ($81 \leq r \leq 90$)

f	rate
logS9	0.0768
GG_G	0.0824
GG	0.0853
GG_T	0.0858
logS10	0.0928
logS8	0.117
L_G	0.120
S10	0.122
logS7	0.133
S9	0.139
S8	0.194
S7	0.260
G	0.341
L	0.570
C_G	0.652
C	1.594

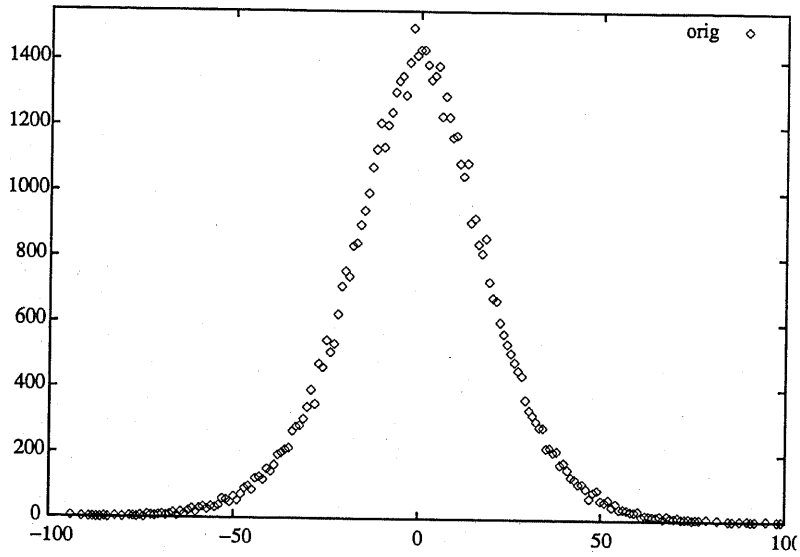


Figure 2.26: Example of error distribution, obtained from PA085 (green) image ($81 \leq r \leq 90$).

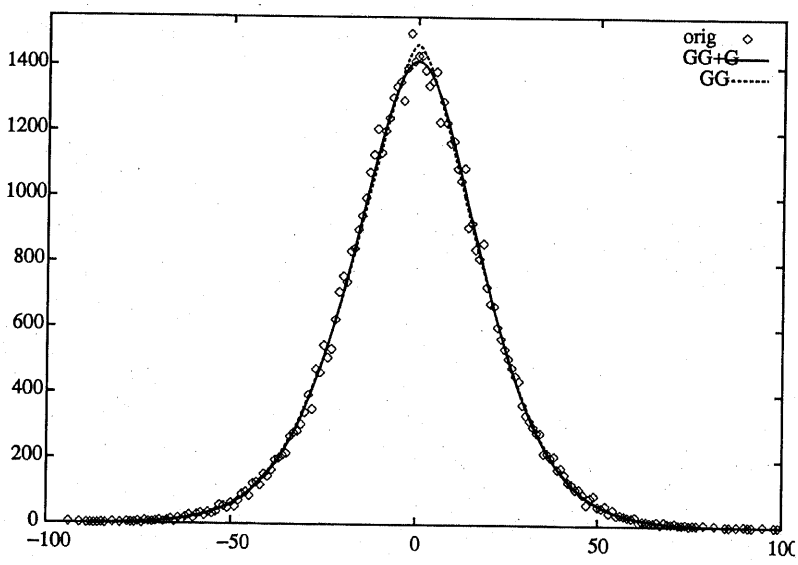


Figure 2.27: Fitting curves of GG_G and GG_G.

Table 2.7: Fitting length comparison for PA085(green), $81 \leq r \leq 90$, 63736 symbols obtained, entropy=6.357bits/ symbol, variance=20.05^2.

f	parameters	rate
logS9		0.04995
GG_G	$\sigma_s = 8.65, p = 1.36, \sigma_e = 18.1, \mu = -0.0527$	0.05238
logS10		0.05666
GG_T	$p = 1.56, \sigma_e = 20.0, \mu = -0.0604$	0.05894
GG	$p = 1.56, \sigma_e = 20.0, \mu = -0.0611$	0.05895
L_G	$\sigma_s = 13.0, \quad \sigma_e = 16.1, \mu = -0.0375$	0.06894
logS8		0.07463
logS7		0.08036
S9		0.09556
S10		0.09753
S8		0.11096
S7		0.16407
G	$\sigma_e = 20.0, \mu = 0.000648$	0.23552
C_G	$\sigma_s = 16.9, c = 4.90, \quad \mu = 0.00466$	0.49917
L	$\sigma_e = 22.0, \mu = 0.0436$	0.57323
C	$c = 12.9, \quad \mu = -0.165$	1.73777

Table 2.8: Average fitting rate for 16 images (large area, $r \leq 20$)

f	rate
GG_G	0.134238
logS9	0.319620
logS7	0.412924
C_G	0.459929
logS10	0.466318
L_G	0.488060
GG_T	0.529363
GG	0.555529
L	0.680457
logS8	0.823475
S10	1.567322
S9	1.868763
C	2.049568
S8	2.711409
S7	3.096082
G	3.447661

Table 2.9: Average fitting rate for 16 images (large area, $81 \leq r \leq 90$)

f	rate
GG_G	0.024387
GG_T	0.031011
GG	0.031695
logS10	0.043807
L_G	0.050955
logS9	0.056962
logS8	0.072420
logS7	0.075580
S10	0.131082
S9	0.207438
G	0.354114
L	0.489458
S8	0.542478
C_G	0.562473
S7	1.012669
C	1.838558

Discussions

In this section, a model of prediction error distribution of digital images which considers the effect of additive source noise and quantization noise upon linear prediction error distribution is investigated and formulated. This model successfully explains the central rotundity of the error distribution graph, which had not been explained by previous researches in this area.

Under the code-length criterion, the actual prediction error distributions are tried to fit, and our model fits better than other analytical functions such as the generalized Gaussian or the Cauchy pdf. We tried to use well-known χ^2 criterion and our model also gave the best fitting results.

Among spline functions, interpolating on $(x, \log y)$ coordinate yields obviously better results than on (x, y), and particularly nine-nodes (logS9) performs as well as our analytical model (GG_G). The main difference is that logS9 function requires nine pairs of $(x, \log y)$ coordinate values while GG_G function requires only four parameters. But for image compression, this function is proved to be quite useful.

But fundamental problems of error distribution modeling still remain. There is no theoretical reason that e in equation(2.13) is distributed with the generalized Gaussian pdf (assumption 2), which is a very interesting issue and no literatures succeed to prove this fact. However, taking the existence of additive noise and quantization noise into account is shown to be approximated by Gaussian convolution in this section. This fact is applicable not only for predictive coding as was seen in this chapter, but also for subband-image signal or for DCT coefficients of images. Because they are also the linear combinations of pixel values.

There is no warranty that the distribution is symmetric, either. But looking at the actual error distribution curve, this fact seems to be practically neglectable.

2.5 Adaptive Arithmetic Coding

We adopt Witten *et al.*'s arithmetic coder[5] for coding prediction error signals. Among encoders which uses only symbol's appearance probability, this coder performs the best. Once a precise frequency table is given, its code efficiency is very close to one[2]

Moreover we make this arithmetic coder adaptive, that is, update the internal frequency table on each signal input. This is because the actual probability table does not completely coincide fitting curves such as the ones discussed in section 2.4.

[2]In our experiment, efficiency= $(1 - 10^5) \sim (1 - 10^6)$.

Table 2.10: Comparison of the code efficiency with several encoders.

Method	Code efficiency
Proposed	0.9994
WNC(Arithmetic 1)	0.9779
RICE	0.9722
ADAP(Arithmetic 2)	0.9635
Huffman	0.8173
LZRW3A(LZ 1)	0.6209
LZRW1(LZ 2)	0.5604

Combining the optimal fitting techniques and adaptive arithmetic coder yields a powerful coder which constantly marks code efficiency of 0.998~0.999. Table 2.10 shows the actual code efficiency with several entropy coders for the prediction error signal (image=lena, after CHED-A conversion and fit by log-spline function). 'Our method' means the coder which generates initial frequency table with optimal fitting function and encodes it with an arithmetic coder (with frequency update). Because of the frequency update, the code efficiency is improved (initially ideal code efficiency is 0.9890) up to 0.9994.

2.6 Conclusion

The method for generating an integer from a prediction-value and truth-value is discussed in section 2.2 to yield the lower entropy than mere rounding. The source splitting algorithm, quantitatively evaluated in section 2.3, is relatively simple but contributes significantly towards shortening the code length.

In section 2.4, a model of prediction error distribution of digital images which considers the effect of additive source noise and quantization noise upon linear prediction error distribution is investigated and formulated. This model successfully explains the error distribution, which had not been explained by previous models. Under the code-length criterion, the actual prediction error distributions are tried to fit, and our model fits better than other analytical functions such as the generalized Gaussian or the Cauchy pdf.

The methods discussed here are all independent of each other, therefore they can be applied together. That is, for example, split the signal into several groups and for each group, convert using ODC, then find the optimal fitting function and finally pass it to the adaptive arithmetic coder.

Chapter

Image Coding Based on the Autoregressive Model

THE autoregressive model is one of the most fundamental model for signal processing field. Also as an image model, there is quite a few research. In this section, this autoregressive image model is introduced as a *modeler* or a *redundancy-remover* and its property is described, and its performance in terms of encoding performance is analyzed. Its variant, RLS filter, is also introduced for image coding. The application for multi-channel image coding is also shown.

3.1 The Autoregressive (AR) Image Model

In the signal processing theory, the autoregressive model considers the temporal correlation between signals, and a signal value at a certain time is represented by the weighted sum of the signal values of the past with an additive noise. For images, temporal signals are replaced by two-dimensional signal $i(x, y)$ and given by

$$i(x,y) = \sum_k \sum_l C_{k,l} i(x-k, y-l) + \omega(x,y), \quad (k,l) \in N \tag{3.1}$$

where N is the size of causal area, and ω can be hypothesized as an white noise when N is large.

Optimally speaking, the measure of the penalty of the model fitting is the entropy of prediction errors, and $C_{k,l}$ must be chosen to minimize it. But in practice, the sum of the squared errors

$$E = \sum_x \sum_y \left(i(x,y) - \sum_k \sum_l C_{k,l} i(x-k, y-l) \right)^2 \tag{3.2}$$

is used for this measure and it indeed works well. Such a pair of coefficients $C_{k,l}$ is obtained through solving *the normal equation*

$$\frac{\partial E}{\partial C_{k,l}} = 0. \tag{3.3}$$

3.2 Application of Two Dimensional RLS Method on Images

In the field of acoustic signal processing, adaptive filtering is widely used and produces remarkable results. There are two typical adaptive filtering methods, RLS (recursive least square) and LMS (least mean square) respectively.

RLS is one of the algorithms which completely consider autocorrelation matrix representing statistical property of the signal stream, and is optimal in terms of squared error criterion. It requires $O(3N^2)$ of computation order.

LMS updates the prediction coefficients while observing the prediction error and expects the coefficients to stochastically converge optimal. Although the convergence is not as quick as that of RLS, the computation order is $O(2N)$.

As for the image coding, the prediction coefficients, normally found from entire statistic property of the image, are fixed. It implicitly assumes that the image is a homogeneous random field, which is not often expectable. On the other hand, if we apply the adaptive optimal prediction method such as RLS for image prediction to follow the change of statistical property of the image, the compression performance is anticipated to improve. In this appendix, fast incremental calculation of two-dimensional RLS algorithm is described and its application and performance to image data is discussed.

3.2.1 Incremental solution for the optimal filter

23	19	17	14	18	22	24
21	11	9	6	10	12	20
15	7	3	2	4	8	16
13	5	1	X			

Figure 3.1: Pixels used for the prediction (in priority)

Let $r(x, y)$ denote image sequence (two-dimensional), vector $\vec{x}(m)$ the pixel array used for prediction (cf. figure 3.1), vector $\vec{\theta}(m)$ the prediction coefficients array. Hence the prediction value $\hat{x}$ becomes

$$\hat{x} = \vec{\theta}^t \vec{x}, \tag{3.4}$$

where $\cdot^t$ is the transpose operator. The dimension of these vectors is N.

Firstly, let's think of one-dimensional optimal filter (Wiener filter) which embodies one-dimensional RLS. To find the optimal prediction coefficient vector $\vec{\theta}$ in terms of least square error, define J as

$$J \triangleq \sum_W \left(r - \vec{\theta}^t \vec{x} \right)^2, \tag{3.5}$$

setting its partial derivative to zero,

$$\frac{\partial J}{\partial \vec{\theta}} = \vec{0}$$

hence

$$\underset{W}{E}[\vec{x}\vec{x}^t]\vec{\theta} = \underset{W}{E}[r\vec{x}]. \tag{3.6}$$

Equation(3.6) is called *"Wiener-Hopf normal equation"*. W is the model support region (window), $\underset{W}{E}[\cdot]$ is the expectation operator. For zero-mean data such as audio signals, $\underset{W}{E}[\vec{x}\vec{x}^t]$ is the autocorrelation matrix and $\underset{W}{E}[r\vec{x}]$ is the correlation vector between signal and actual output. Equation(3.6) is equivalent to the following equation:

$$\begin{aligned} \vec{\theta} &= \mathsf{P}\vec{b}, \\ \text{where } \mathsf{P} &\triangleq \left[\sum_W \vec{x}\vec{x}^t\right]^{-1} \\ \vec{b} &\triangleq \sum_W r\vec{x}. \end{aligned} \tag{3.7}$$

Solving equation(3.7) yields $\vec{\theta}$. The computational requirements for the calculation is

$$NN_W + N_W N(N+1) + N^3/2 = O(\frac{1}{2}N^3) \text{ times},$$

and it is not realistic to repeat this calculation (inverting matrices) for each movement of the window.

To incrementally obtain $\vec{\theta}(m+1)$ from $\vec{\theta}(m)$, let $W' \triangleq W(m+1)$ denote the new window after the shift. In case of one-dimensional RLS, both $W' \cap \overline{W}$ (the discarding region, 'out') and $\overline{W'} \cap W$ (the incoming region, 'in') have one signal for each, hence the update of $\vec{b}$ is

$$\vec{b}(m+1) = \vec{b}(m) + r_{\text{in}}\vec{x}_{\text{in}} - r_{\text{out}}\vec{x}_{\text{out}}, \tag{3.8}$$

on the other hand, the update of P is

$$\mathsf{P}(m+1) = \left[\mathsf{P}^{-1}(m) - \vec{x}_{\text{out}}\vec{x}_{\text{out}}^t + \vec{x}_{\text{in}}\vec{x}_{\text{in}}^t\right]^{-1} \tag{3.9}$$

and not as easy as the case of $\vec{b}$. So we apply the following matrix inversion lemma[19].

Lemma: $\left[\mathsf{M}^{-1} + \vec{x}\vec{x}^t\right]^{-1} = \mathsf{M} - \dfrac{\mathsf{M}\vec{x}\vec{x}^t\mathsf{M}}{1 + \vec{x}^t\mathsf{M}\vec{x}}$

Let

$$\mathsf{Q}^{-1} \triangleq \mathsf{P}^{-1}(m) + \vec{x}_{\text{in}}\vec{x}_{\text{in}}^t, \tag{3.10}$$

hence using the lemma gives

$$\mathsf{Q} = \mathsf{P}(m) - \frac{\mathsf{P}\vec{x}_{\text{in}}\vec{x}_{\text{in}}^t\mathsf{P}}{1 + \vec{x}_{\text{in}}^t\mathsf{P}\vec{x}_{\text{in}}}. \tag{3.11}$$

From equation(3.9),

$$\mathsf{P}(m+1) = \left[\mathsf{Q}^{-1} - \vec{x}_{\text{out}}\vec{x}_{\text{out}}^t\right]^{-1}. \tag{3.12}$$

Again using the lemma gives

$$\mathsf{P}(m+1) = \mathsf{Q} + \frac{\mathsf{Q}\vec{x}_{\text{out}}\vec{x}_{\text{out}}^t\mathsf{Q}}{1 - \vec{x}_{\text{out}}^t\mathsf{Q}\vec{x}_{\text{out}}}. \tag{3.13}$$

Then $\vec{\theta}(m+1)$ is found via equation(3.7) Note that no matrix inversion is operated in equation(3.13). The multiplication time required is decreased down to $3N^2 + 7N = O(3N^2)$.

3.2.2 Two-dimensional RLS and experimental results

For the case of two-dimensional RLS calculation, repeat the process above for all the signal pairs in $\overline{W'} \cap W$ and $W' \cap \overline{W}$.

If we use the window shown in figure 3.2, number of in-out-pairs in one horizontal movement is $S+1$, therefore the number of multiplication time is $(S+1)(3N^2+7N)$. The robust prediction is accomplished using larger S, and adaptive prediction is actualized with smaller S.

Figure 3.3 and 3.4 show the transition of the entropy according to the S and N.

3.3 JPEG Lossless Mode

JPEG Lossless mode basically is a predictive coder whose prediction mode is assigned by user. We used "crush" package[20] which performs the JPEG lossless mode for the comparison purpose. Using the pixels shown in figure 3.5, the predicted value of $X(=\hat{X})$ for each mode is shown in table 3.1.

Also crush has its own prediction mode 8, i.e. find the smallest of (|B-C|,|C-A|, |B+A-2C|), if |B-C| is smallest, choose A as $\hat{X}$; if |C-A| is smallest, choose B; if |B+A-2C| is smallest, choose C.

Crush adaptively encodes the difference between each sample and the predicted value. The compression ratio is always higher than those of GIF's or gzip's.

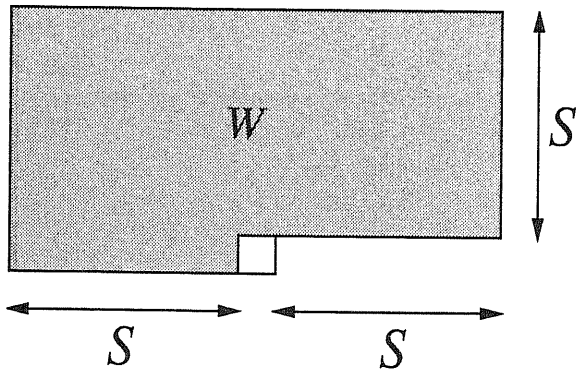


Figure 3.2: Two-dimensional window W

Table 3.1: Modes of JPEG lossless prediction (mode 1~3 are mere DPCM)

mode	4	5	6	7
$\hat{X}$	$A+B-C$	$A+(B-C)/2$	$B+(A-C)/2$	$(A+B)/2$

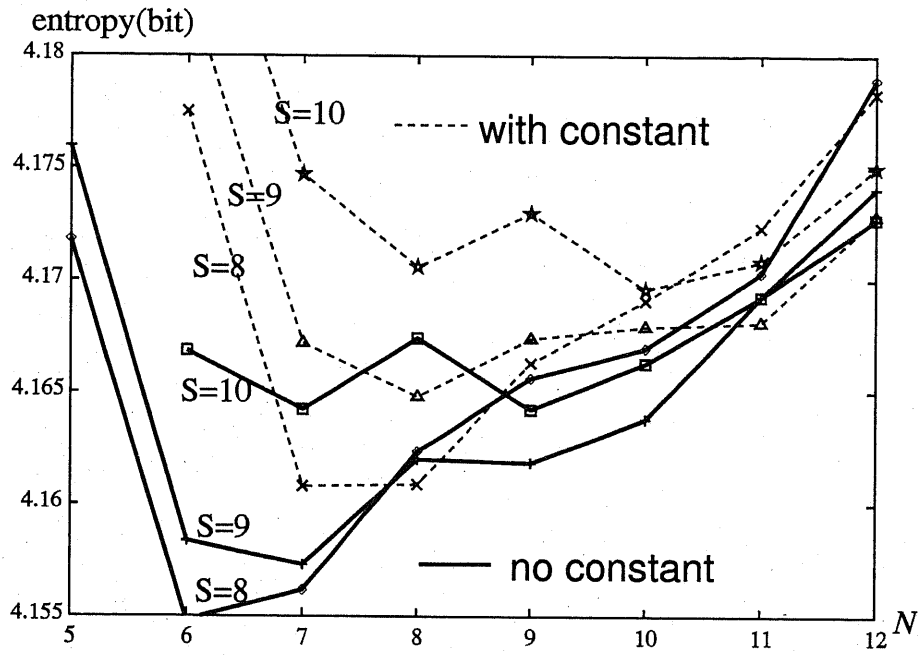
3.4 AR-based Prediction According to the Edge

Generally the image contains flat areas, edge areas, random or regular textures and so on. Therefore we classify the pixels into several categories based on the edge value and predict (and encode) independently. The detailed discussion is given in section 2.3.

To deal not only with edge intensity but with the edge direction, we classify the pixel X, by using surrounding pixels $a \sim f$. When $\max(|a-f|,\ |b-f|,\ |c-f|,\ |e-f|,\ |f+c-2b|,\ |b-c|,\ |d-c|)$ is smaller than a certain threshold, then g, the group number of X, is set to zero. Otherwise, g is decided according to the following:

$$g = \begin{cases} & |\cdot| < 0 & |\cdot| > 0 \\ |a-f| \text{ is max} & 1 & 2 \\ |b-f| \text{ is max} & 3 & 4 \\ |c-f| \text{ is max} & 5 & 6 \\ |e-f| \text{ is max} & 7 & 8 \\ |f+c-2b| \text{ is max} & 9 & 10 \\ |b-c| \text{ is max} & 11 & 12 \\ |d-c| \text{ is max} & 13 & 14 \end{cases}$$

This method enables the prediction adapting the local edge direction.



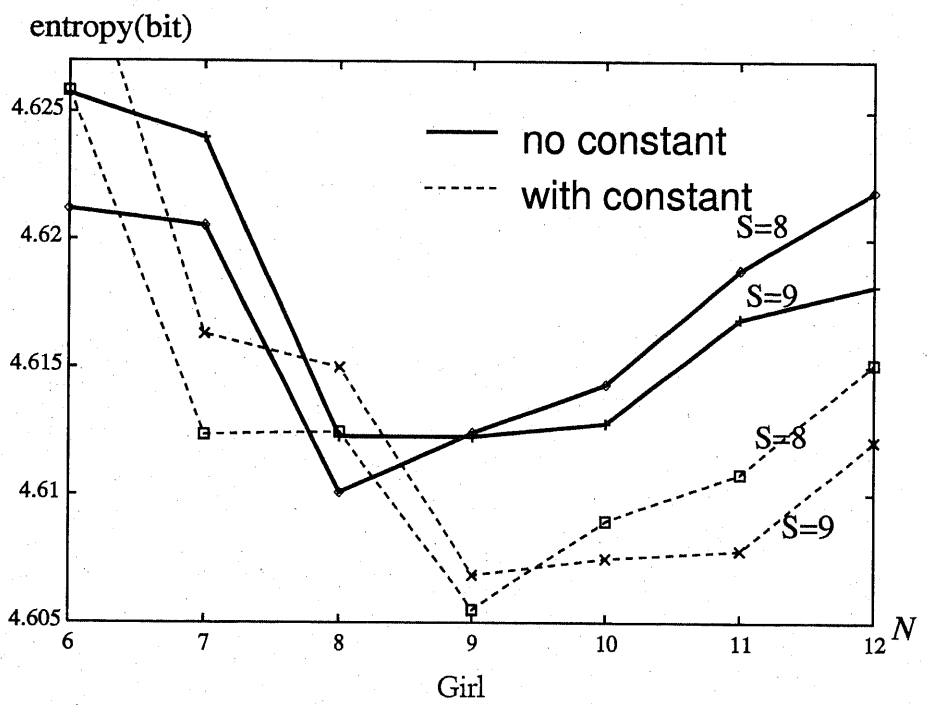
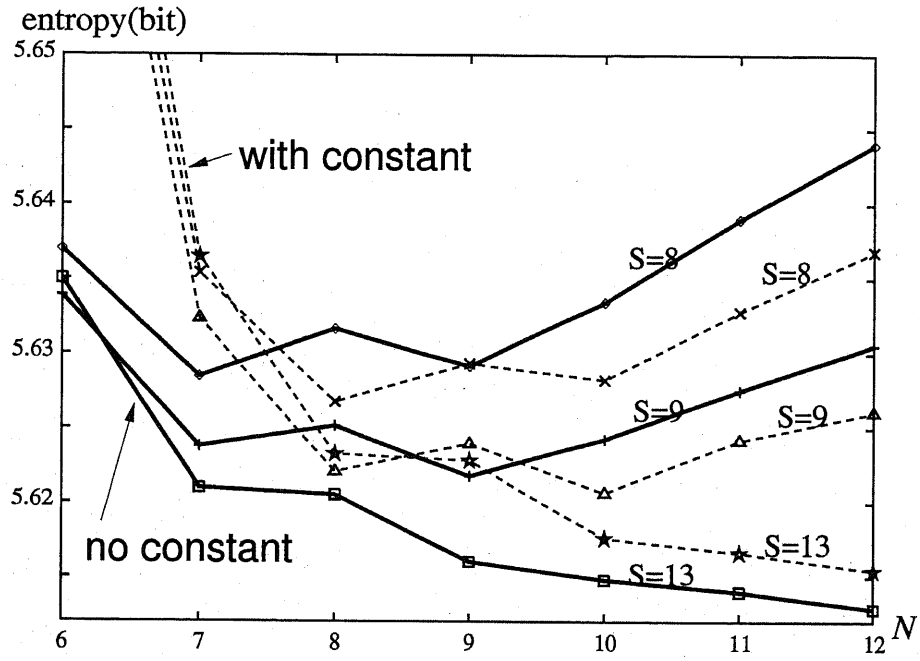
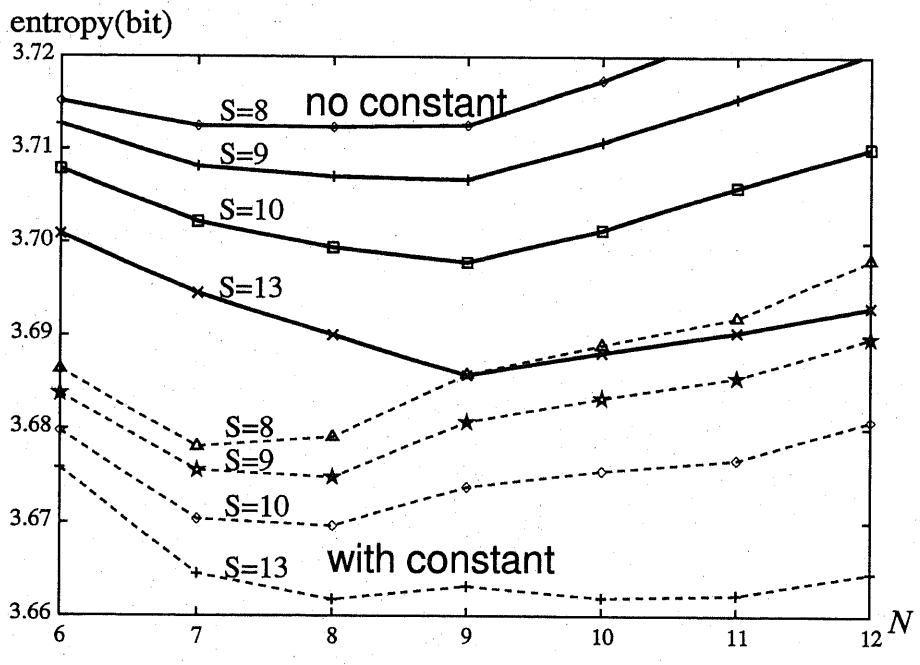


Figure 3.3: Transition of entropy according to S and N.



Aerial



WashDC-1

Figure 3.4: Transition of entropy according to S and N.

C	B
A	X

Figure 3.5: Pixels used for JPEG lossless prediction.

a	b	c	d
e	f	X	

Figure 3.6: Surrounding pixels $a \sim f$ used for classifying X

We implemented the adaptive prediction coder and compared with JPEG lossless methods. The parameters for the adaptive filter is $S = 10, N = 13$ (see section 3.2).

Table 3.2 shows the compression results. Image 'girl' in the first raw is 720×576 pixels JPEG standard image, while 'girl' in the fifth raw is taken from SIDBA (Standard image database), 256×256 pixels.

The approximate time for coding with RLS is, for boats∼goldhill (720×576 pixels), 600 seconds and for girl∼lady (256×256 pixels), 370 seconds.

The edge-adaptive method takes about 50 seconds for 720×576 images, and 14 seconds for 256×256 image.

3.5 Application to the Compression of Multi-Channel Image

The utilization of remote sensing satellite data such as NOAA-HRPT (High Resolution Picture Transmission) and GMS-VISSR (Visible and Infrared Spin Scan Radiometer) enables academic researchers in the field of meteorology oceanography, hydrology and vegetation to observe and analyze global dynamic phenomena and perform terrestrial observation.

In our Institute of Industrial Science at University of Tokyo, we daily receive such data, which amounts 100M byte for one transmission. We are trying to enable academic researchers to access our satellite image database via the network using multi-media tools. For this purpose more advanced image processing systems using parallel computers, efficient archiving systems and SeaWiFS satellite data receiving systems are under development.

Table 3.2: Comparison of the edge-adaptive prediction and RLS (unit: bit/ pixel)

Image	edge-adaptive	RLS	JPEG	Howard[8]
girl	3.865	3.782	4.981(8)	—
boats	4.112	3.781	4.185(8)	—
zelda	4.817	4.715	4.768(8)	—
goldhill	4.476	4.481	4.653(8)	—
girl	4.570	4.469	4.746(7)	—
moon	5.055	5.027	5.060(7)	—
aerial	5.606	5.580	5.822(7)	—
couple	4.022	4.039	4.094(8)	—
lady	4.413	4.348	4.648(7)	—
lena	4.569	4.414	4.898(7)	—
washdc1	3.565	3.647	3.709(7)	3.67
washdc2	2.751	2.822	2.880(7)	2.87
washdc3	3.200	3.267	3.368(7)	3.31
washdc4	4.021	4.055	4.247(7)	4.15
washdc5	4.368	4.388	4.586(7)	4.47
washdc6	0.776	0.728	0.920(4)	1.09
washdc7	3.527	3.547	3.670(7)	3.62

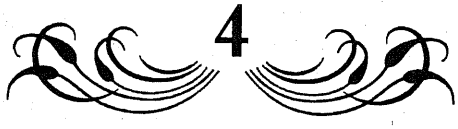
For image databases, a high compression ratio is important for the storage and also for rapid transmission, but to deal with various kinds of users demands lossless image transmission is indispensable[21, 22]. The precise implementation and its performance are discussed in appendix A.

3.6 Conclusion

In this section, the application of the autoregressive image model for coding is discussed. We proposed the coding method using edge-adaptive prediction and also the method using RLS filtering.

For relatively small images, edge-adaptive encoder has a drawback because it must emit prediction coefficients for each edge. But as far as the processing time is concerned, RLS is still slow, despite contriving not to calculate matrix inversion. Also RLS takes nearly the same time for decoding, because its dominant processing is updating the coefficients.

Chapter



4

Image Coding Based on Markov-AR Hybrid Model

MARKOV model approximation of digital images gives very low entropy, but requires a huge amount of storage to maintain the frequency table for encoding and decoding. Therefore, it has only been used for binary picture compression so far.

We have developed a high performance AR-based predictive lossless image coder as described in the previous chapter, and have combined this technique with a Markov model, which provides a better compression ratio for gray scale images.

The basic idea is, encode the first part (this size is variable) of an image using the normal AR-based encoder and concurrently use these data as a Markov model training set. Then the latter half will be encoded using the Markov model with the assistance of the AR-based coder.

4.1 Introduction

Lossless still image compression techniques have recently been highlighted. The most popular method for lossless compression of still images with more than 8 bits/ pixel is AR-based predictive coding like JPEG lossless mode, but performance limitations are beginning to appear.

On the other hand, the Markov model entropies of the images are attractively lower than the bit rate of predictive coders (e.g. table 4.2 in section 4.4), but a frequency table with a huge amount of data is necessary for encoding and decoding, therefore it has been unrealistic to apply the Markov model for image compression up to the present.

In this chapter, we propose using a Markov model for image compression and show that the performance of current lossless image compression techniques can be improved. Our method consists of Markov model encoding and classical AR-based encoding. It does not require the transmission of a huge frequency table. In addition, our AR-based coder itself outperforms the JPEG lossless coder.

Markov Model Image Representation for Image Coding

If the symbol occurrence probability depends only on the m preceding symbols, the source is called an *"m-th order Markov source"*. The series of m preceding symbols is called the *"state"*. If the source has N different kinds of symbols, the m-th order Markov model can be divided into N^m memoryless sources, the same as the number of possible states.

To encode such a Markov source, N^m different occurrence frequency tables are necessary. As each table has N words, total table size amounts to N^{m+1} words. At the moment,

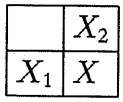


Figure 4.1: Support of a second order Markov model

Table 4.1: Entropy, bit rate of JPEG lossless coder and our method for each image(unit = bit). **0th E**: 0th order entropy, **M.M.E**: Second order Markov model entropy, **JPEG**: Bit rate of JPEG Lossless coder (by *'crush'* [20]).

Image	0th E.	M.M.E.	JPEG
girl	6.608	3.079	4.981(8)
boats	7.088	3.300	4.185(8)
goldhill	7.530	3.675	4.653(8)
zelda	6.741	2.990	4.768(8)
hotel	7.546	3.320	4.667(8)
washdc1	4.685	3.442	3.849(7)
washdc2	3.846	2.726	2.981(7)
washdc3	4.307	3.153	3.484(7)
washdc4	5.432	3.909	4.324(7)
washdc5	5.681	4.056	4.649(7)
washdc6	3.441	0.903	0.920(4)
washdc7	4.757	3.454	3.792(7)

current techniques using the Markov model only treat binary ($N = 2$) images for $m \simeq 10$. Needless to say, it is impractical to transmit the whole table for gray images ($N \simeq 2^8 = 256$) because of its astronomical size.

We use a two-dimensional second order Markov model whose state S is represented as $S(X_1, X_2)$, where X_1 and X_2 are as shown in figure 4.1. Although this model is very simple, the entropy of this model is quite low (1∼4 bits lower than 0th order entropy and 1∼2 bits lower than JPEG lossless coder, see table 4.1). which implies further compression possibilities.

a	b	c	d
e	f	X	

Figure 4.2: Neighbor pixels $a \sim f$ to decide the group of pixel X

4.2 AR-based Encoding Algorithm

In this section we explain our AR-based predictive coder which is used together with the Markov model image coder. The scheme consists of two main parts: pixel classification and source distribution fitting.

4.2.1 Pixel classification

We use two methods to classify each pixel. The first method is used to decide the *group* of each pixel and calculate the prediction coefficients respectively, and the second method calculates the *class* for each pixel, which is used to split the data stream of prediction error and to reduce entropy.

An important property of these methods is they can be done using only past data. Therefore the encoder have to transmit no information about *group* and *class* since the decoder can generate the same information independently.

Grouping according to local edge

Figure 4.2 shows X's neighbor pixels $a \sim f$ used to determine the group it belongs to.

The group number of X is decided according to which number among ($|a-f|$, $|b-f|$, $|c-f|$, $|e-f|$, $|f+c-2b|$, $|b-c|$, $|d-c|$) is the biggest. This method enables X to be distinguished according to its local edge direction. We calculate the prediction coefficients using the least square error method for each group respectively.

Classification according to the statistical property[23]

It is also possible to classify a pixel according to the value $Q = |e-f|+|b-f|+|c-f|+|d-f|$, which physically means the *"edge intensity around X"*. After obtaining this Q value, it is quantized properly and used as *class* number. Since this procedure is independent from the former classification method, they can be used together.

Effect of pixel grouping

For example, Gaussian sources with different variances should not be entropy coded together, but separately, because the latter always gives a shorter code. Therefore the source splitting improves the performance of compression if it is successful.

Our classification method is not directly related to the source property itself. But from the experimental results, the source is split into different statistical properties, which means an improvement in compression.

4.2.2 Improvement of coding efficiency

To encode signals effectively, a knowledge about its distribution is necessary. However sometimes the size of the table becomes significant compared to the code size. In our method, we find the best fit function of the error distribution and then transmit the parameters, which has negligible overhead so that the decoder can re-generate an approximation of the frequency table. Since the fitting curve is not perfect, the coder updates the internal table after encoding/ decoding each symbol, which is described next section.

Error conversion

Normally the prediction error is rounded off to an integer and encoded. In addition to this simple rounding, we also check to convert the prediction error in the following way and examine their entropy to obtain a smaller code. This conversion method is described in section 2.2.1.

Criterion of fitting

Giving consideration of subsequent entropy coding, the criterion of "better fitting" should be considered. We use "the theoretical information amount I" according to the probability function, i.e.:

$$I = \sum_s -\#(s) \log_2 f(s) \quad \text{(bits)}, \tag{4.1}$$

where $\#(s)$ means the number of occurrences of symbol s and $f(s)$ is the fitting function which satisfies $\sum_s f(s) = 1$.

Error distribution fitting

To find well approximated fitting, some different fitting curves, such as the *(Generalized) Gaussian function* (equation(2.4)) and *Spline function* (cubic natural spline) are attempted.

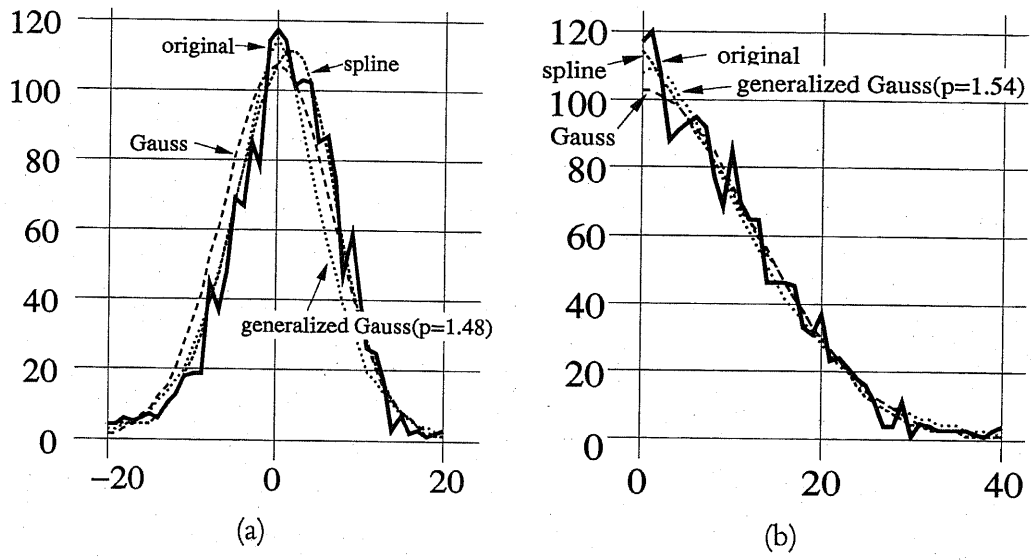


Figure 4.3: The fitting for the error distribution. (a): rounded prediction error (b): converted error E.

Sometimes the generalized Gaussian pdf does not fit the actual distribution well because of its asymmetricity. This is the reason why spline function is also tried.

Both simple prediction error distribution and E (converted value) distribution are tried to fit by these functions and the best one according to the criterion equation(4.1), which also considers the *overhead*, is chosen. This overhead is the number of the bits for each fitting function to represent its parameters such as node coordinates of spline, σ and p for $f_{p,\sigma}$. An example of fitting of a certain distribution is shown in figure 4.3. In this case, the generalized Gaussian pdf for E is chosen.

4.3 Image Coding using Learning Markov Model

In this section, we discuss the probability estimation method for a given Markov state.

There are two problems for the learning Markov model:

- Probability estimation for signal X
- Handling of the unknown signal X (when the state does not know signal X)

they are closely related each other. We use following Bayesian estimation method.

4.3.1 Markov model and Bayesian estimation

The symbol probability is assumed to be proportional to the symbol count. However, some symbols occur very infrequently. If a code word must be available for each symbol in the alphabet, the count for each symbol is typically started at 1. In an N-ary alphabet, therefore, the initial assumed probabilities are:

$$P(1) = P(2) = \cdots = P(N) = 1/N$$

which would give a code book of equal-length codes of length $\log_2(N)$. Of course, once the counts $C(M)$ are accumulated for each of the N symbols this estimate would be revised to:

$$P(M) = \frac{C(M)+1}{\sum_{k=1}^{N}(C(k)+1)}. \tag{4.2}$$

This is a particular form of Bayesian estimation in which no prior knowledge is assumed about symbol probabilities. In the limit of large symbol counts the effect of prior information about symbol probabilities becomes vanishingly small. This is usually the case in Huffman coding, in which the primary effect of an assumption of prior information is to reserve a code word for all symbols.

More generally, when there is prior information about symbol probabilities, the starting numerical factors can be adjusted to account for that information:

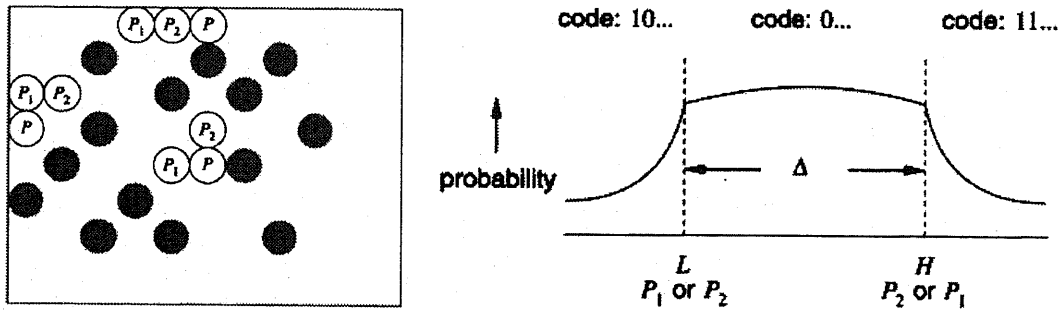
$$P(M) = \frac{C(M)+KB(M)}{\sum_{k=1}^{N}(C(k)+KB(k))}, \tag{4.3}$$

where $B(M)$ represents prior information about the expected probability for symbol M and K is a constant.

The probability estimation of second order Markov state frequency from first order Markov state, which is discussed in section 4.3.3, can be used instead of $B(M)$ in equation(4.3). However, in this case, $B(M)$ is updated at all times.

4.3.2 Dealing with unknown signals

In previous general discussion, every signal is assigned a non-zero probability. It is also possible to emit an special code for a signal of first appearance (unknown) and then emit some code to represent the signal consecutively in the middle of learning process. This special code can be considered as the sum of the probabilities of unknown signals. This problem is precisely discussed in section 4.3.6.



(a) (b)

Figure 4.4: A pixel's coding region depends on its two neighbors. (a) P's closest neighbors P_1 and P_2, (b) probability density of P. taken from [24].

4.3.3 Accelerative learning of Markov state using ML estimation

Howard[8] claims that distribution of P and idealy falls in graph of figure 4.4(b). When P is within the central region, its values are distributed almost uniformly (slightly convex). When P is outside the central region, its probability drops off sharply as the value moves further away.

But practically, probability density graph becomes like dotted lines of figure 4.5(b). They are noisy, however, rapid change of characteristics at the points of P_1 or P_2 is not observed.

From the point of ML estimation, state parameters X_1 (P_1 in figure 4.4) and X_2 (P_2) of second order Markov model are considered "indirect observation of X (P)". Hence derived

$$L(X|X_1, X_2) = L(X|X_1) \cdot L(X|X_2). \tag{4.4}$$

Let this equation apply to Markov model representation of images. Pixel likelihood of Markov state $(X_1, X_2) = (92, 96)$ for image "lena" and its estimation are shown in figure 4.5(a). It is observed that ML estimation fits well for practical density.

4.3.4 Learning Markov model

If all the frequency tables for Markov states of an image could be transmitted (without overhead), it could be encoded down to the amount calculated from Markov model entropy (cf. table 4.2) with help of arithmetic coder. But unfortunately the overhead for the frequency tale is not negligible.

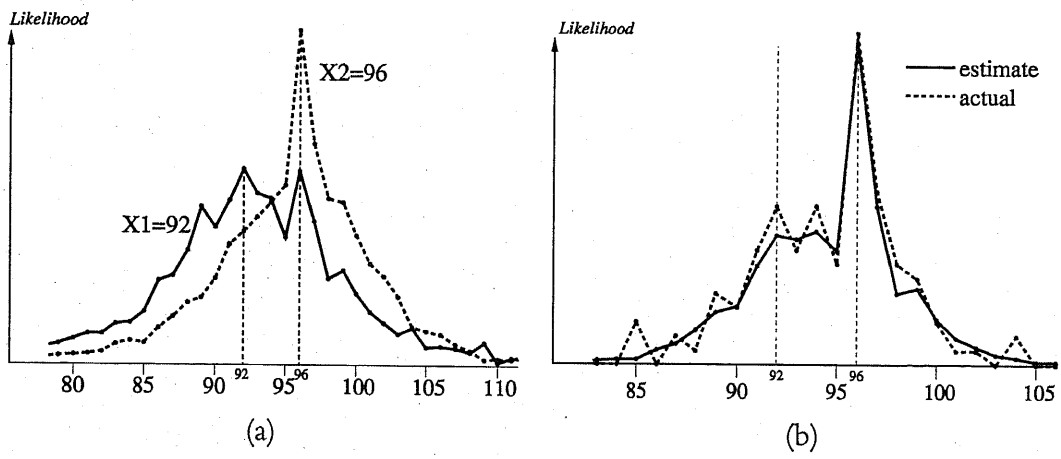


Figure 4.5: Likelihood graphs (a)$L(X|X_1 = 92)$ and $L(X|X_2 = 96)$, (b)$L(X|X_1 = 92, X_2 = 96)$ and its ML estimation. image=lena

The idea is encode the first image part of an image with a certain lossless coder (we use AR-based encoder in chapter 3), and use these processed (causal) pixels for the training of the Markov model. After the training we try to encode the latter part with the Markov encoder (see the following section for details). The coder is also trained while decoding process. Therefore there is no need to transmit the frequency table of each Markov state.

We perform Markov model learning with acceleration technique described in section 4.3.3. When the coder becomes *experienced enough* for a certain pixel, we encode its probability with arithmetic encoder. As the decoder can perform the same process, it is not necessary to transmit the information whether a pixel is AR-encoded (encode the linear prediction error) or Markov encoded, same as frequency table transmission.

4.3.5 Choice of Markov encoding or AR-based encoding

To judge if the encoder is experienced enough or not for encoding aa certain pixel, we observe the occurrence of each state. If the occurrence is larger then a certain threshold, we encode the pixel with Markov model encoder.

After a certain number of occurrence of the state and therefore judged as experienced, it is still possible that the pixel has not occurred for this state. To deal with this zero-probability problem, we keep a certain amount of probability for this case. Such a pixel is AR-encoded after emitting this probability.

4.3.6 Appearance of an unknown symbol

In compensation of no table transmission, a key question is how to deal with in the coding process of the latter image part. It is "the appearance of an unknown symbol s", because the decoder has no means of detecting it. Therefore some means of informing the decoder must be provided.

To solve this problem, we prepare the special signal *"Unknown"*. Each frequency table prepares room for this signal. In encoding, when the coder detect the new symbol, "Unknown" is transmitted. Suppose A is the fixed frequency for this symbol, "theoretical code size for the model" H_A is

$$H_A = \sum_S \sum_{s \in S} -\#(X) \log_2 \frac{\#(s)}{\#(S) + A} \quad \text{(bit)}, \tag{4.5}$$

where S is a state, s is a symbol and $\#(\cdot)$ means the element number of the set. Note that $H_A|_{A=0}$ is equal to the Markov model entropy. If the state S is unknown or the symbol is unknown, we use the normal prediction coding methods using the neighbor pixels (see section 4.2) to recover the value of current pixel.

We use arithmetic coder which is suited to encode dynamically-varying probability fractions with high coding efficiency. The evaluation of the other trial of letting the decoder know when a new symbol appears and the comparison with JBIG will also be shown in section 4.4.1.

4.3.7 Other possibility

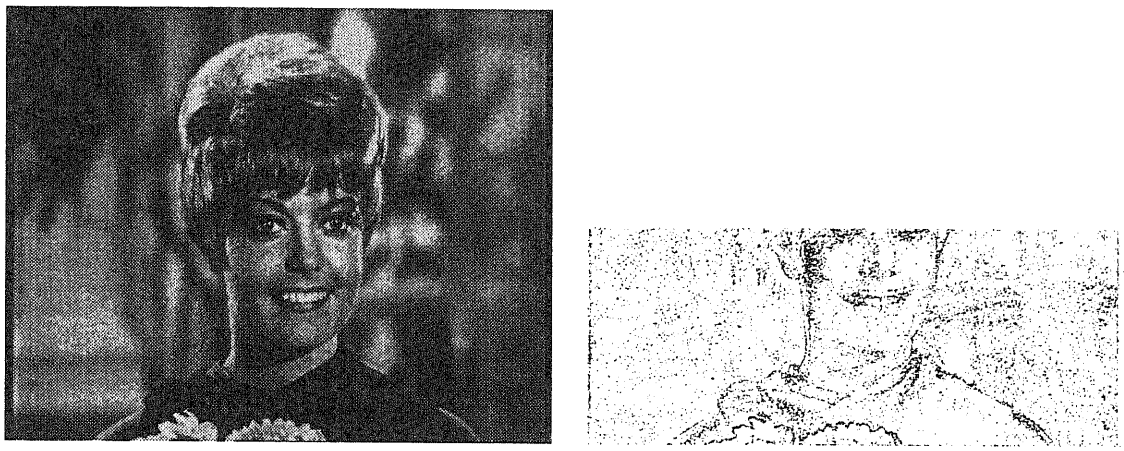
Figure 4.6(a) shows the original image "zelda" and the black pixels in figure 4.6(b) shows the occurrence points of the unknown symbols. The first half (50%) of the image is used for the initial training. There are 12551 occurrences within the latter half (207360 pixels).

It is also possible to inform the decoder about the new symbol by encoding the binary figure like figure 4.6(b), since with such binary figures the decoder can know where the new symbol occurs.

We adopt the JBIG coder to compress such data. This image is compressed to 8551 bytes by JBIG coder while the calculated overhead of our method is 7518 bytes.

4.3.8 Our Markov-AR hybrid coding scheme

figure 4.7 shows the flow of our image coder using Markov model with help of AR-based coder.



(a) (b)

Figure 4.6: (a) Sample image "zelda". (b):The positions of the new occurrence of the symbols in image (black dots).

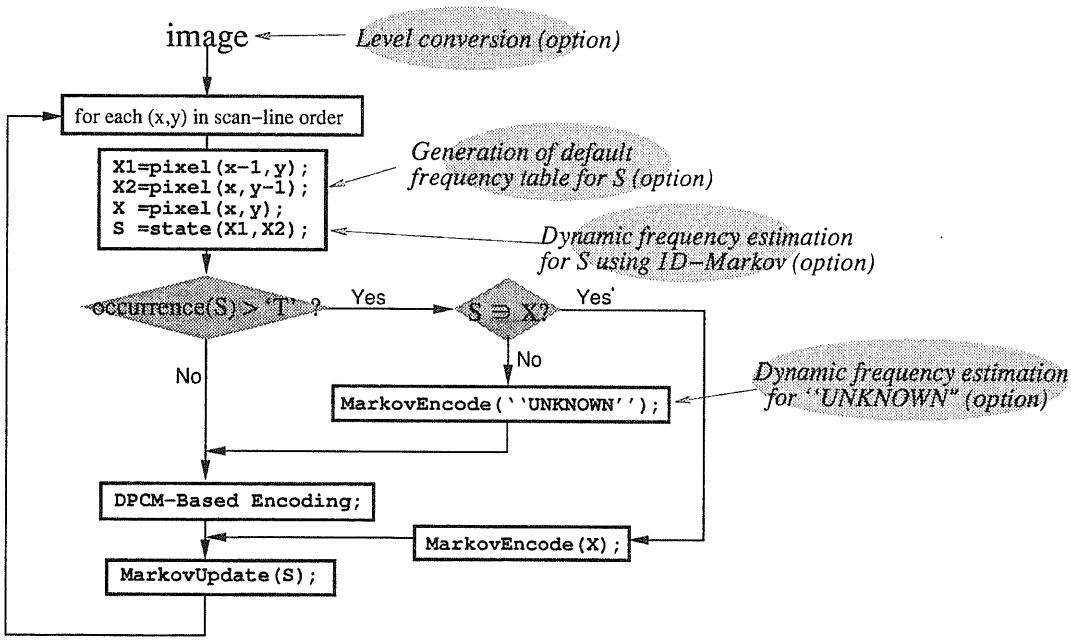


Figure 4.7: Image coding with use of Markov model

4.4 Experimental Results

The compression performance of our coder with varying training sizes is tested on several gray-scale images and compared with other lossless compression schemes. We used 24 neighbor pixels for AR-based linear prediction, the results are shown in table 4.2. The percentage "0%" means trying to compress all the image with Markov encoder except the points where the new states or new symbols appear. "100%" means whole-AR encoding.

The smallest bit-rate for each image is underlined. The first 5 images are 720×576 pixels, "Washdc1..7" are multi-band LANDSAT images with 512×512 pixels.

Table 4.2: Entropy, bit rate of JPEG lossless coder and our method for each image(unit = bit). **M.M.E**: Second order Markov model entropy, **JPEG**: Bit rate of JPEG Lossless coder (by *'crush'* [20]). The numbers inside the parentheses is the predictor number, **percentage**: Ratio of the first DPCM-encoded training part. **overhd**: Overhead size (bytes) for noticing the occurrence of new symbols, **JBIG**: Code size (bytes) for informing the same information by the JBIG coder.

Image	M.M.E.	JPEG	0% (overhd:JBIG)	50% (overhd:JBIG)	100%
girl	3.079	4.981(8)	3.653 (23396:25571)	4.136 (10589:11771)	4.861
boats	3.300	4.185(8)	4.465 (45978:35521)	4.371 (34060:22798)	4.107
goldhill	3.675	4.653(8)	4.943 (62949:46104)	4.720 (27073:20430)	4.450
zelda	2.990	4.768(8)	3.501 (20151:22735)	4.091 (7518:8551)	4.770
hotel	3.320	4.667(8)	4.719 (29953:41969)	4.582 (14960:18471)	4.352
washdc1	3.442	3.849(7)	3.748 (9074:10309)	3.652 (2989:3588)	3.543
washdc2	2.726	2.981(7)	2.881 (4107:5402)	2.817 (1249:1802)	2.738
washdc3	3.153	3.484(7)	3.390 (7105:8357)	3.286 (2330:2842)	3.183
washdc4	3.909	4.324(7)	4.434 (20084:18670)	4.206 (7315:7052)	4.007
washdc5	4.056	4.649(7)	4.816 (27274:22968)	4.567 (10226:9097)	4.351
washdc6	0.903	0.920(4)	0.948 (413:712)	0.894 (171:334)	0.938
washdc7	3.454	3.792(7)	3.747 (9652:10825)	3.617 (3141:3721)	3.511

4.4.1 Representation of new symbol appearance

To substantiate our method to deal with the occurrence of new symbols, the theoretical overhead of the extra information necessary for representing the occurrence of new symbol

according to equation(4.5) is calculated.

To notify the appearance of Unknown signal, JBIG coder can be used as mentioned in section 4.3.7. Experimental results are shown within the parentheses in table 4.2. Sometimes the JBIG coder outperforms our method in regard to new symbol treatment.

4.5 Conclusion

In this chapter, the algorithm for continuous tone still image compression using a Markov model is proposed. The coder always outperforms JPEG lossless coding schemes by 0.2~1.3 bits/ pixel.

In our method, the predictive coding part and the Markov model coding part are independent, therefore it is possible to combine more powerful predictive coders.

Although it is clear for some images (girl, zelda) that the Markov model coding improves the performance, there are still some images for which it does not work well.

Increasing the order of Markov model to third degree or more may be an interesting topic for further research.

Markov model image representation offers not only the pixel probability but also pixel value prediction. In appendix B, this issue is discussed and the minimum entropy prediction algorithm which enables efficient non-linear prediction is proposed.

Chapter

Image Coding Based on Transform-AR Hybrid Model

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 chapter, we propose a lossless image compression scheme using a lossy image generated with JPEG-DCT scheme. The basic concept is, first send a JPEG-compressed lossy image primarily, then send residual information and reconstruct the original image using both the lossy image and residual information. Three-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.

5.1 Lossy Plus Lossless (LPL) Image Coding

Lossy image coding scheme can yield quite small size of compressed data, which is convenient for image browsing, or transmission. Even though the reconstructed image has distortion, it does contain some amount of information about the original image which helps lossless reconstruction. Utilizing the lossy image, the receiver may be able to reconstruct the complete original image with some additional information. This is the idea of *'lossy plus lossless'* image coding.

Formerly, the schemes of LPL coding simply encode the difference between lossy reconstructed image and the original image[25].

In this chapter, a lossless compression algorithm for gray image using lossy compressed image is presented. The lossy compression scheme uses the JPEG discrete cosine transform (JPEG baseline) algorithm as the lossy coding algorithm.

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[26]. We utilize the lossy data thoroughly, and much better result is obtained.

5.2 Outline of the Scheme

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 does not 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

Table 5.1: 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

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 actual value is calculated, and it 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.

5.2.1 Our method

Principally, we use the linear prediction. We use up to ten neighbor pixels, also using the pixels of lossy image and prediction error e is converted to another form before encoding.

5.2.2 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:

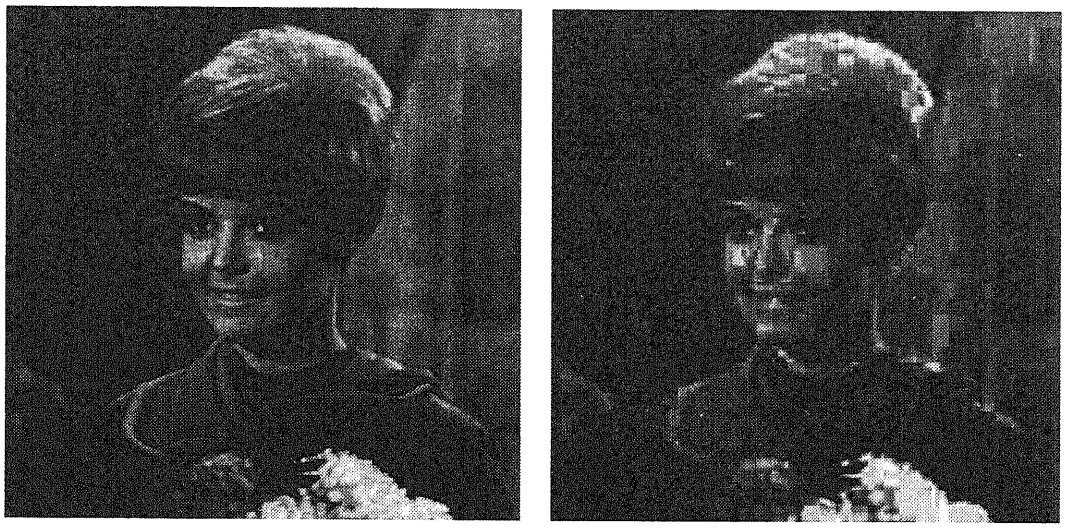
$$Q \equiv |P_2 - P_1| + |P_3 - P_1| + |P_4 - P_1| + |P_5 - P_1| \tag{5.1}$$

Using this Q value, we classify each pixel into several groups according to table 5.1.

Figure 5.2(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.

5.2.3 Context search

Table 5.2 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.



(a) (b)

Figure 5.1: (a)Original image 'Girl' and (b)JPEG compressed image(quality value=5)

(a) (b)

Figure 5.2: (a)Image of Q value, (b)Image of prediction error of simple prediction

Table 5.2: 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

The region where we search the similar area (we denote this *"context"*) is shown in figure 5.3. This is restricted within already processed pixels' area. The procedure is described below:

1. Scan the area and find points that satisfy

$$Q(x') > 70 \quad \text{and} \quad |Q(x) - Q(x')| < 10 \tag{5.2}$$

2. Within such points, find one that minimizes

$$C = ||b' - a'| - |b - a|| + ||c' - a'| - |c - a|| \tag{5.3}$$

3. If the $C_{\min}$ is smaller than 12, treat it as the context of the current point. Otherwise, return failure (not found).

5.3 Linear Prediction of Pixels

5.3.1 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 5.4 ($P_1 \ldots P_{10}$). Number of pixels are variable (up to 10 pixels), and afterwards we will choose it optimally. The priority is shown by the suffix number in the figure. The most effective number will be discussed later.

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

5.3.2 Prediction of context

Under the criterion of condition (equation(5.2)) and (5.3), 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

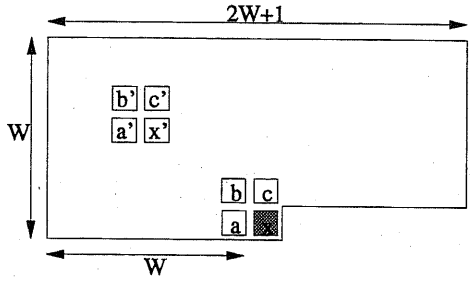


Figure 5.3: Context-search region

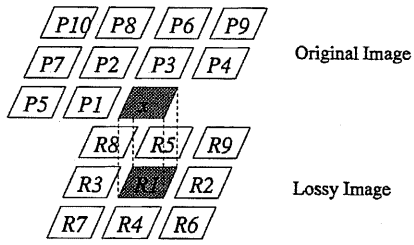


Figure 5.4: Pixels used for prediction

figure 5.3). 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.

5.3.3 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.

The conversion algorithm used here is different from this, and called 'ODC' method (described in section 2.2.1). After this conversion, we get an 8-bit non-negative integer E. The conversion is reversible, i.e., when a predicted value and a converted number E (and also upper and lower bound) are given, the actual pixel can be calculated.

5.3.4 Fitting of the distribution of E

According to the experimental results, the distribution of E (figure 5.5(a)), after error conversion, looks close to the right half of a Gaussian distribution.

As this distribution is limited right half, σ^2 is equal to $\sum x^2/N$, where N is the number of samples.

Their graphs are shown in figure 5.5(b). It can be seen that using this Gaussian model, the distribution of E is approximated well enough. More precise discussion on prediction error distribution is given in section 2.4.

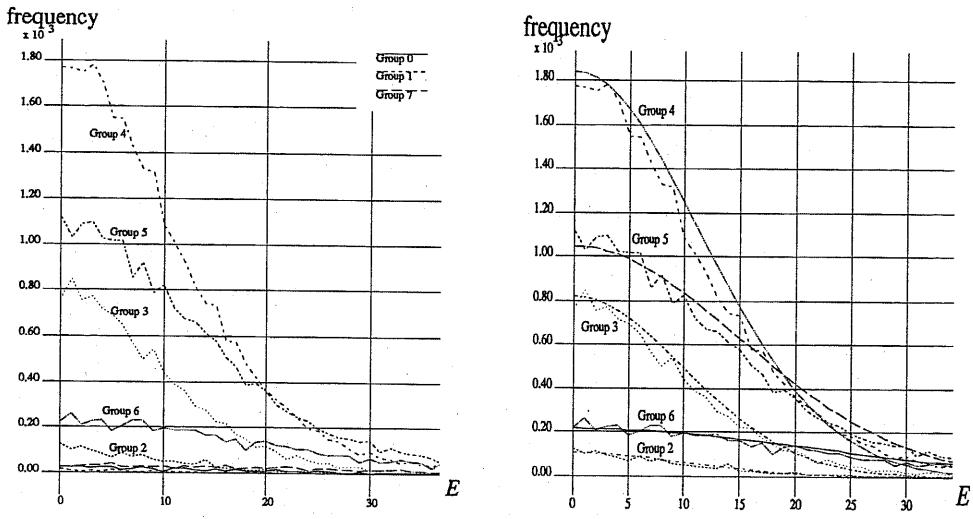


Figure 5.5: (a)Distribution of E (image='Moon'), (b)Fitting of E with Gaussian pdf

5.3.5 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.

5.4 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,

Table 5.3: 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

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 '-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.

5.4.1 Effect of context search

We use three test images 'Girl', 'Couple' and 'Moon', which are chosen from SIDBA(Standard Image DataBase). They are all of 256×256 resolution, 8bits/pixel.

For these images, the bit rate of compressing with and without context search is compared. The results are shown in table 5.3. NL is the number of lossy pixels used for prediction. N is the number of lossless neighbor pixels.

5.4.2 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 5.6 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 5.1(b)).

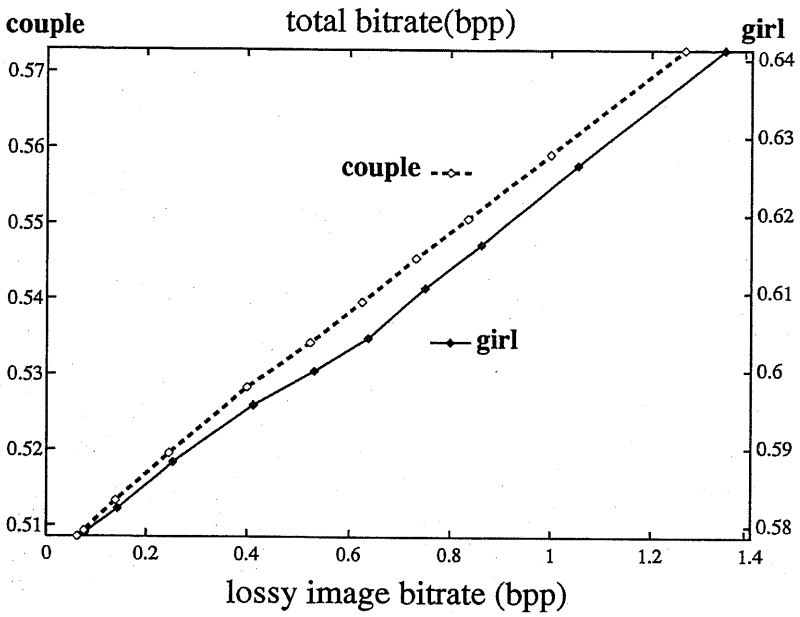


Figure 5.6: Total bit rate V.S. lossy image bit rate (N=10, NL=9, image=girl,couple)

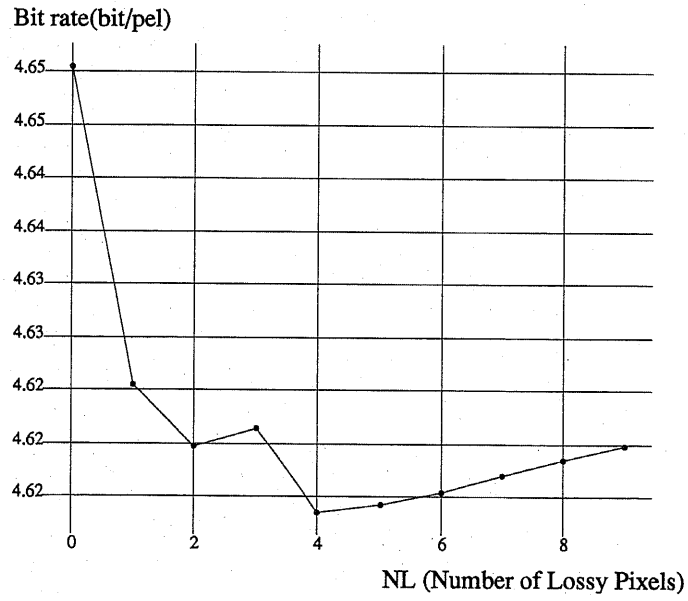


Figure 5.7: Bit rate V.S. NL (N=10, quality=5, image='girl')

5.4.3 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 5.6 shows the transition of total bit rate of compressed product according to lossy image quality (in its bit rate). This means that the more less-fidelity image is used for coding, the better the compression becomes. They have an obvious linear relation, however, the slope is not steep (about 1/20).

Figure 5.7 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.

5.4.4 Comparison with the other methods

Experimental results and comparison with the other methods are shown in table 5.4. MAW method is proposed by Memon[26]. It also uses a lossy JPEG image to reconstruct a lossless image.

For our method, we use NL=9, N=10. As the results of MAW is not of bit-rate but entropy, therefore the actual bit-rate must be larger. In spite of that, thanks to the entropy reduction techniques described in chapter 2, our result yields 0.8~0.5bpp shorter code than MAW and 0.02~0.5bpp shorter than JPEG lossless coder. Taniguchi's method[23] is originally lossless oriented, therefore the results are slightly (0.03~0.12 bpp) better than ours.

5.5 Conclusion

In this chapter, we proposed the algorithm of lossless image compression using lossy reconstructed image. 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.

Like the other LPL 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

Table 5.4: Result of LPL coding(unit: bit/pixel). In 'JPEG', parentheses mean the best predictor number. In 'MAW' and 'Our method', parentheses mean the bit-rate of used lossy image.

image	Taniguchi	gzip -9	JPEG	MAW	Our method
girl	4.494	5.690	4.746(7)	5.10(0.28)	4.612(0.144)
couple	3.978	5.365	4.094(8)	4.53(0.29)	4.055(0.140)
moon	5.019	6.563	5.060(7)	—	5.046(0.102)
lady	—	6.757	4.648(7)	—	4.506(0.131)
house	—	5.894	4.708(4)	4.84(0.30)	4.130(0.159)
tree	—	6.515	5.694(5)	5.92(0.48)	5.102(0.262)

that they can browse the image before the lossless decompression.

It is possible that there is a more suitable lossy compression scheme other than JPEG.

Chapter 6

Conclusion

In this paper, the lossless coding method based on the autoregressive model is investigated from the points of image model and modeling penalty coding. The contribution of this work are in two areas: improve the performance of entropy coding especially for prediction error signals, and the hybrid application of autoregressive model with the other models such as wave model or Markov model yielding higher compression.

In chapter 2, several methods to decrease the code length is discussed and quantitatively discussed. And the error distribution modeling described in section 2.4 successfully formulates the distribution function, and contributes much for the reduction of code length. Such a distribution is obtained not only from the direct linear image prediction error but also from the subband images in subband coding and also from the DCT coefficients of the image (DPCM data for the DC part, raw data for the AC part), whose signals are all generated via linear combination of digital image pixels. Our theory is generally applicable for such distributions.

In chapter 3, image coding based on the autoregressive model is discussed. The application of adaptive filter (RLS) is also discussed, and the effect of performance is evaluated. Its application for the satellite NOAA multi-channel image coding (appendix A) will contribute for more powerful satellite database system.

In chapter 4, Markov model image representation, which have not been applied for gray-scale image coding, is applied to the gray-scale image coding in harmony with the autoregressive model. Through combining two completely different image models, new possibility of hybrid coding is revealed. Also the non-linear pixel predictor minimizing the entropy based on Markov model is proposed in appendix B.

In chapter 5, wave model or transform coding, which is originally a lossy technique, is utilized for lossless image coding. The lossy image data is used for quick-previewing of the image and reconstruction of lossless image. By this technology, users get the great merit that they can browse the image before the lossless decompression.

Although the history of image coding goes long back to 1950's, the demand and importance of efficient lossless image coding is going to rise even more because of the ever enhancing communication infrastructure, and this study is anticipated to be located as a framework for such a technology.

Acknowledgment

This work can not be completed without being supervised by professor Mikio Takagi. His profound advice based on extensive knowledge always helped me struggling my way. Also his generous attitude toward the students enabled me to try anything I want to.

During my time in Takagi laboratory, I was deeply influenced by assistant Munekazu Sakamoto. His willingness that allows no compromise gives me an introspective attitude consciously or subconsciously. I think that helped me make rigid theories and algorithms and prudent data analysis.

I would like to express my regards to associate professor Masaru Kitsuregawa, his assistants and his laboratory students for preparing a splendid computer environment. Particularly I asked Stephen Davis, a doctoral student, to correct my English papers over and over, and he always accepted them willingly. A considerable amount of my works in English is owing to him.

Mr. Masao Aizu, working for DNP company and studying as a doctoral student simultaneously at our laboratory, gave me a lot of highly suggestive visions backed with his profound experiences on image coding through daily small talks or laboratory meetings.

I appreciate very much for making warm atmosphere in our laboratory for my colleagues Mr. Nakagawa, Mr. Asanobu Kitamoto, Mr. Jun Takakura, Mr. Yuji Matsumoto, Mr. Masayuki Suzuki, Mr. Eiji Ikoma, Mr. Yuichiro Ohtsuki, Mr. Ido Daiji and Ms. Chen Pei for supporting me personally.

I would like to thank the secretaries of Takagi laboratory: Ms. Kasuga, Ms. Tanaka, Ms. Ishizeki, Ms. Tsunoda, Ms. Kikuchi and Ms. Suwa so much for always doing miscellaneous duties readily to promote an environment for study.

To have an opportunity to study at the institute of industrial science, which has a very well maintained computer room and network systems and also a good library, is very fortunate for me. I thank all those people who dedicated for the environment of this

campus.

I do not know how I can ever express my gratitude for my parents. They approved of me to proceed a doctoral course, supported me financially and personally.

Bibliography

[1] Anil K. Jain: **"A Fast Karhunen-Loeve Transform for a Class of Random Processes"**, *IEEE Trans. Commn.*, Vol. COM-24, No. 9, pp. 1023–1029, September 1976.

[2] Martin Vetterli and Didier le Gall: **"Perfect Reconstruction FIR Filter Banks: Some Properties and Factorizations"**, *IEEE Trans. ASSP.*, Vol. 37, No. 7, pp. 1057–1071, July 1989.

[3] D. A. Huffman: **"A method for the construction of minimum redundancy codes"**, *Proceedings of IRE*, Vol. 40, No. 10, pp. 1098–1101, September 1951.

[4] J. Ziv and A. Lempel: **"A Universal Algorithm for Sequential Data Compression"**, *IEEE Transactions on Information Theory*, Vol. IT-23, No. 3, pp. 337–343, May 1977.

[5] Ian H. Witten, Radford M. Neal, and John G. Cleary: **"Arithmetic Coding for Data Compression"**, *Communications of the ACM*, Vol. 30, No. 6,, June 1987.

[6] J. J. Rissanen *et al.*: **"Arithmetic coding"**, *IBM Journal or Research and Development*, Vol. 23, No. 2, pp. 188–193, 1976.

[7] Dong Wook Kang, Su Won Kang, and Choong Woong Lee: **"Entropy reduction of symbols by source splitting and its application to video coding"**, *Signal Processing: Image Communication*, Vol. 6, No. 5, pp. 471–478, October 1994.

[8] P. G. Howard: **"The design and analysis of efficient lossless data compression systems"** PhD thesis, Brown University, Department of Computer Science, June 1993.

[9] Nobutaka Kuroki *et al.*: **"Lossless Image Compression by Two-Dimensional Linear Prediction with Variable Coefficients"**, *IEICE Trans. Fundamentals.*, Vol. E75-A, No. 7, pp. 882–889, July 1992.

[10] Xiaolin Wu: **"Context Selection and Quantization for Lossless Image Coding"**, In *Proceeding of Data Compression Conference 95*, pp. 453, Utah USA, March 1995.

[11] P. H. Westerink, J. Biemond, and D. E. Boekee: **"Subband coding of color images"**, In J. W. Woods, editor, *Subband Image Coding*, pp. 198–205. Kluwger Academic, 1991.

[12] J. B. O'Neal Jr.: **"Predictive Quantizing Systems (Differential Pulse Code Modulation) for the Transmission of Television Signals"**, *The Bell System Technical Journal*, pp. 689–721, May–June 1966.

[13] 高村誠之、高木幹雄: **"雑音を考慮した画像信号の線形予測誤差分布モデル"**, 電子情報通信学会春季大会, March 1996 発表予定.

[14] N. S. Jayant and Peter Noll: **"Digital Coding of Waveforms** Principles and Applications to SPeech and Video" Signal Processing. Prentice-Hall, 1984.

[15] F. Bellifemine, A. Capellino, A. Chimienti, R. Picco, and R. Ponti: **"Statistical analysis of the 2D-DCT coefficients of the differential signal for images"**, *Signal Processing: Image Communication*, Vol. 4, No. 2, pp. 477–488, 1992.

[16] G. R. Kuduvalli and Rangaraj M. Rangayyan: **"Performance Analysis of Reversible Image Compression Techniques for High-Resolution Digital Images"**, *IEEE Trans. Medical Imaging*, Vol. 11, No. 3, pp. 430–445, September 1992.

[17] M. Th. Subbotin: **"On the Law of Frequency of Error"**, *Matematicheskii Sbornik*, Vol. 31, pp. 296–301, 1923.

[18] K. Sharifi and A. Leon-Garcia: **"Estimation of Shape Parameter for Generalized Gaussian Distributions in Subband Decompositions of Video"**, *IEEE Trans. Circuits & Syst. for Video Tech.*, Vol. 5, No. 1, pp. 52–56, February 1995.

[19] A.M.Tekalp *et al.*: **"Fast Recursive Estimation of the Parameters of a Space-Varying Autoregressive Image Model"**, *Transactions on ASSAP*, Vol. 33, No. 2, pp. 469–472, April 1985.

[20] 'crush' package is available via anonymous ftp at dftnic.gsfc.nasa.gov: disk$moe:[anonymous.files.software.unix.crushv3]: crush_v3_tar.Z.

[21] 高村誠之、高木幹雄: **"衛星 NOAA 画像の高能率可逆圧縮"**, In *Proceedings of* 画像符号化シンポジウム *94*, 9-18, October 1994.

[22] Seishi Takamura and Mikio Takagi: **"Lossless Compression of NOAA-AVHRR Satellite Data"**, In *Proceedings of Science Information Management and Data Compression Workshop 94*, pp. 65–73, Meryland USA, September 1994.

[23] 谷口隆行 他: **"高階調画像の可変長符号選択形可逆予測符号化方式"**, 信学論, Vol. J70-B, pp. 654–663, June 1987.

[24] Ian H. Witten *et al.*: **"Managing Gigabytes"** Van Nostrand Reinhold, 1994.

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

[26] Nasir D. Memon *et al.*: **"Simple Method for Enhancing the Performance of Lossy Plus Lossless Image Compression Schemes"**, *Journal of Electronic Imaging*, Vol. 2, No. 3, pp. 245–252, July 1993.

[27] S. R. Tate: **"Band ordering in Lossless Compression of Multispectral Images"**, In *Proceedings of Data Compression Conference 94*, pp. 311–320, Utah USA, March 1994.

[28] Anil K. Jain: **"Image Data Compression: A Reveiew"**, *Proceedings of the IEEE*, Vol. 69, No. 3, pp. 349–391, March 1989.

[29] 有馬哲、石村貞夫: 『多変量解析のはなし』, 東京図書, October 1987.

[30] 鈴木義一郎: 『情報量規準による統計解析入門』, 講談社, April 1995.

[31] 中川徹、小柳義夫: 『最小二乗法による実験データ解析 プログラム SALS』, 東京大学出版会, May 1982.

[32] Howard Kaufman *et al.*: **"Estimation and Identification of Two-Dimensional Images"**, *IEEE trans. Automatic Control*, Vol. 28, No. 7,, July 1983.

[33] John G. Cleary and Ian H. Witten: **"Data Compression Using Adaptive Coding and Partial String Matching"**, *Transactions on Communications*, Vol. 32, No. 4,, April 1984.

[34] M. Kim *et al.*: **"NOAA Data Compression Using a Multi-Length DPCM Code and a Variable-Length Code"**, In *Proc. 11th Asian Conf. Remote Sensing*, pp. P–1, 1990.

[35] Ying Wang: **"A Set of Transformations for Lossless Image Compression"**, *IEEE Trans. Image Processing*, Vol. 4, No. 5, pp. 677–679, May 1995.

[36] 張埼、河野隆二、今井秀樹: **"濃淡画像の算術符号のためのモデル化と確率テーブルの初期値の決定法"**, 信学技報, Vol. IT92, No. 68,, 1992.

[37] 宇津井修 他: **"自然画像の可逆符号化に関する研究"**, 画像符号化シンポジウム *94*, 5-17, pp. 103–104, October 1994.

[38] 伊東晋: 『画像情報処理の基礎』, 東京理科大学出版会, 1986.

[39] 原島博監修: 『画像情報圧縮』, オーム社, 1992.

[40] 高木幹雄、下田陽久監修: 『画像解析ハンドブック』, 東京大学出版会, 1992.

[41] Seishi Takamura, Toshihiro Nemoto, and Mikio Takagi: **"Remote Sensing Data Receiving and Processing"**, In *International Seminar on Space Informatics and Sustainable Development: Grassland Monitoring and Management*, Ulaan Baatar, Mongolia, June 1995.

[42] Seishi Takamura and Mikio Takagi: **"A Hybrid Lossless Compression of Still Images Using Markov Models and Linear Prediction"**, In *Proceedings of 8th ICIAP'95*, pp. 203–208, San Remo ITALY, September 1995.

[43] Seishi Takamura and Mikio Takagi: **"Lossless Image Compression with Lossy Image Using Adaptive Prediction and Arithmetic Coding"**, In *Proceedings of Data Compression Conference 94*, pp. 166–174, Utah USA, March 1994.

[44] Seishi Takamura and Mikio Takagi: **"A Framework for the Coding of Image Prediction Error"**, In *Proceedings of Picture Coding Symposium 96*, March 1996 prearranged.

[45] 高村誠之、高木幹雄: **"静止濃淡画像の学習型マルコフモデル予測に関する検討"**, 電子情報通信学会春季大会, D-284, pp. 10, March 1995.

[46] 高村誠之、高木幹雄: **"自己回帰モデルとマルコフモデルによる静止多階調画像の可逆符号化"**, In *Proceedings of*画像符号化シンポジウム *95*, pp. 127–128, October 1995.

[47] S.Takamura and M.Takagi: **"Recognition of Handwritten Layout Drawings"**, In *Proceedings of ICAIP'93*, Bari ITALY, September 1993.

[48] M.Takagi and S.Takamura: **"Recognition of Handwritten Layout Drawing"**, In *Proceedings of CVVC'94*, October 1994.

[49] S.Takamura, T.Nemoto, and M.Takagi: **"Remote Sensing Data Receiving and Processing"**, In *Proceedings of International Seminar on Space Informatics and Sustainable Development: Grassland Monitoring and Management*, June 1995.

Publication List

Society Paper

1. S.Takamura, M.Takagi: *"A Modeling of Linear Prediction Error Distribution of Digital Images"*, Journal of Visual Communication and Image Representation (submitted)
2. S. Takamura and M. Takagi: *"A Study on the Application of Markov Model to Lossless Image Coding"*, Signal Processing: Image Communication (submitted)

International Conferences

1. S.Takamura, M.Takagi: DCC'94 *"Lossless Image Compression with Lossy Image Using Adaptive Prediction and Arithmetic Coding"*, pp. 166–174, Mar. 1994
2. S.Takamura, M.Takagi: Proc. Science Information Management and Data Compression Workshop *"Lossless Compression of NOAA-AVHRR Satellite Data"*, pp. 65–73, Sep. 1994
3. S.Takamura, M.Takagi: 8th ICIAP *"A Hybrid Lossless Compression of Still Images Using Markov Model and Linear Prediction"*, pp.203–208, Sep. 1995
4. S.Takamura, M.Takagi: Picture Coding Symposium, PCS'96 *"A Framework for the Coding of Image Prediction Error"* Mar. 1996 (prearranged)
5. S.Takamura, T.Nemoto, M.Takagi: International Seminar on Space Informatics and Sustainable Development: Grassland Monitoring and Management *"Remote Sensing Data Receiving and Processing"* Jun. 1995

Domestic Conferences (in Japanese)

1. 高村誠之, 高木幹雄: PCSJ94 「衛星 NOAA 画像の高能率可逆圧縮」, 9-18, Oct. 1994
2. 高村誠之, 高木幹雄: 電子情報通信学会春季大会 「静止濃淡画像の学習型マルコフモデル予測に関する検討」, D-284, Mar. 1995
3. 高村誠之, 高木幹雄: PCSJ95 「自己回帰モデルとマルコフモデルによる静止多階調画像の可逆符号化」, Oct. 1995
4. 高村誠之, 高木幹雄: 電子情報通信学会春季大会 「雑音を考慮した画像信号の線形予測誤差分布モデル」, D-236, Mar. 1996 (prearranged)

Other Presentations

1. 川口, 高村, 相田, 齊藤: 電子情報通信学会秋季大会 「UNIX における脅威のモデル化と防御法」, A-148, Oct. 1991
2. 高村, 高木: 第 45 回情報処理学会秋季大会「雑音に強い線分抽出手法」, 2J-5, Oct. 1992
3. 高村, 高木: 電子情報通信春季大会 「線分の太さ推定を用いた線分抽出手法」, D-541, Mar. 1993
4. 高村, 高木: 東京大学生産技術研究所電気談話会報告 「手書きレイアウト図面の認識」 Vol.43, No.8, May 1993
5. 高村, 佐藤: C Magazine, 「画像処理特集 画像処理プログラミング」 ソフトバンク, Jul. 1993
6. S.Takamura, M.Takagi: CAIP'93 *"Recognition of Handwritten Layout Drawings"* Sep. 1993
7. 高村, 高木: 電子情報通信学会 技術報告「擾乱の多い図面認識手法の一提案 ～手書きレイアウト図面への応用」, PRU93-70, Oct. 1993
8. M.Takagi, S.Takamura: CVVC'94 *"Recognition of Handwritten Layout Drawing"* Oct. 1994
9. 高村, 高木: 第 47 回情報処理学会秋季大会「手書きレイアウト図面の認識」, 4L-8, Oct. 1993
10. 笠原, 高村, 高木: 電子情報通信学会春季大会「Hough 変換を用いた地中探査レーダ画像からの埋設管の検出」, D-417, Mar. 1994
11. 高村, 高木: 生産研究 「手書きレイアウト図面認識」 Vol.46 No.3 pp.32-35, Mar.

1994

12. 高村, 佐藤: C Magazine, 「特集 3 次元コンピュータグラフィクス」 ソフトバンク, Jun. 1994

Appendix A

NOAA AVHRR Image Compression

A high-performance lossless compression system for satellite NOAA data is developed and discussed in this chapter. The data is called HRPT (high resolution picture transmission) data, and consists of around 93% AVHRR (advanced very high resolution radiometer) multi-channel image data and 7% of miscellaneous data. In compressing the image portion, we classify each pixel into 10 different groups and apply a multi-channel prediction and a non-linear error conversion. The entropy coder is an arithmetic coder which is adaptive and regenerates the approximation of the statistical properties of the source as an initial probability table. To compress the non-image part, we used the general compressor (gzip). From experimental results, the original information is compressed down to 25% $\sim$ 40%.

A.1 Introduction

The remotely sensed NOAA satellite HRPT data provide very useful and important information in meteorology, oceanography and many other scientific fields. To date there have been many studies on image compression, particularly on lossy and very low bit rate compression. For image databases, a high compression ratio is important for storage and also for rapid transmission, but to deal with various kinds of users demands lossless image transmission is indispensable.

Also for this HRPT data, we must store them as they are. But one difficulty of this data is its size: one datum has more than 90MBytes, and we receive 5 $\sim$ 8 data each day. Reducing the size of the stored data is desired by both archiver and receiver.

The dominant part of this HRPT data is AVHRR (advanced very high resolution radiometer) image data. So we utilize the property of multi-channel 2-dimensional data of AVHRR data for compression. In the literature, some approaches of lossless compression (ex.[34, 27]) have been presented, but they are not very efficient in terms of compression ratio. For our database purposes, the compression ratio is more important than compression time, because decompression is a more common operation than compression.

In this appendix we propose a method to losslessly compress the HRPT data which is somewhat computationally expensive but compresses much better.

A.2 The Satellite NOAA and Its HRPT Data

The meteorological satellite NOAA-11 and NOAA-12 go around the earth at the average altitude of 810km in about 101.2 minutes. They have AVHRR sensor on board, which has five channels covering the wavelength from 0.55μm (channel 1, visible) to 12.5μm

<table>
<tr><td colspan="2">Header(ID,time,etc.)</td><td>1500 bytes</td><td rowspan="11">One line of HRPT data
22180 bytes</td></tr>
<tr><td>Channel 1</td><td>2 bytes</td><td rowspan="10">One line of
AVHRR data
20480 bytes</td></tr>
<tr><td>2</td><td>2 bytes</td></tr>
<tr><td>3</td><td>2 bytes</td></tr>
<tr><td>4</td><td>2 bytes</td></tr>
<tr><td>5</td><td>2 bytes</td></tr>
<tr><td>Channel 1</td><td>2 bytes</td></tr>
<tr><td>2</td><td>2 bytes</td></tr>
<tr><td colspan="2">⋮
repeat 2048 × 5 times</td></tr>
<tr><td colspan="2">Footer(synchronize)</td><td>200 bytes</td></tr>
</table>

Table A.1: HRPT data format of one line

(channel 5, infra red). Reception of the observation data requires about 13 minutes when it is at its highest orbit and one transmission yields $3,000 \sim 4,000$ data lines. The word size for HRPT data is 10bits, but for simplicity of handling, our receiving system represents it by sixteen bits (two bytes). The 6 MSB's are padded with '0's. Therefore each HRPT data amounts to 90Mbytes. We receive $5 \sim 8$ HRPT data a day, so the total amount in a year exceeds 1TByte.

Each line of HRPT data is independent from all others, and contains 2048 pixels. The structure is shown in table A.1.

Figure A.1 shows an example of an AVHRR image obtained from NOAA-HRPT data.

A.3 Coding Algorithm

In short, our method for compressing AVHRR image portion is a kind of predictive coding such as DPCM. But we use many kinds of techniques to reduce its entropy and to code efficiently.

A.3.1 Noise-line and non-imagery part treatment

Usually the AVHRR data contains noise lines(e.g. figure A.2). The number of noise lines in each datum is independent from all others. Such lines should be detected and removed from the image array, put together and compressed using a non-imagery compression

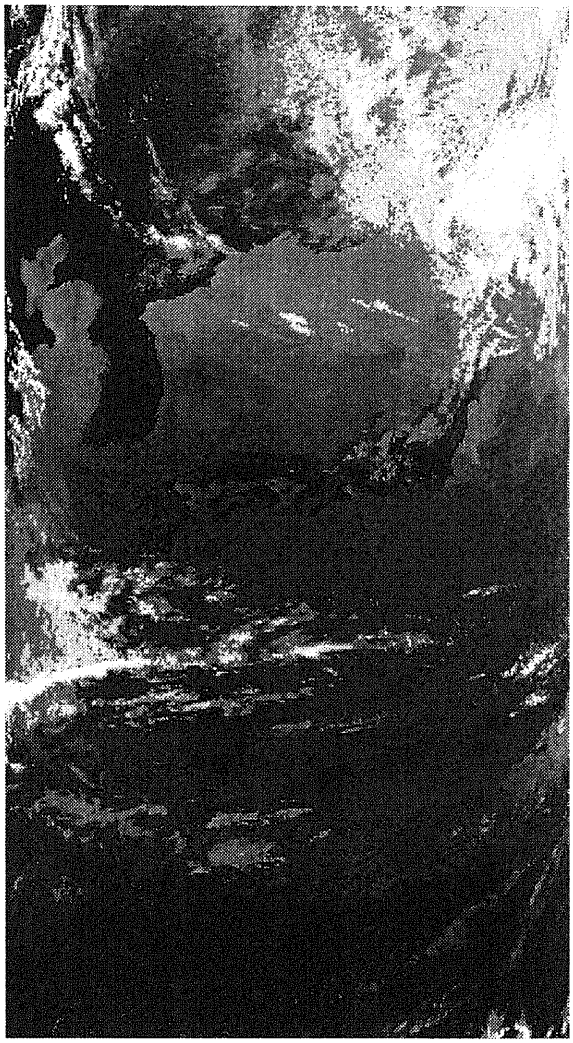


Figure A.1: Example of AVHRR image (Channel 4, 2048 × 3736)

method (gzip).

To detect the noisy lines, we use a simple but reliable criterion. First we check the *ID* bytes in the header area. If the ID of a line is irregular, we regard the line as noisy. If a line passes this test, we calculate the variance of the difference between horizontally adjacent pixels in channel 1. If the variance is greater than a certain threshold (we use 1×10^7), we regard it as a noisy line. From experimental results, the first ID check is noise sensitive enough (see figure A.3 and figure A.4). We just use this second check to be doubly sure. The time for this check is significantly shorter than the total processing time.

The HRPT data contains about 7% of non-imagery data such as the fixed ID code, time stamp, fixed synchronizing code and so on (see table A.1). The fixed or predictable

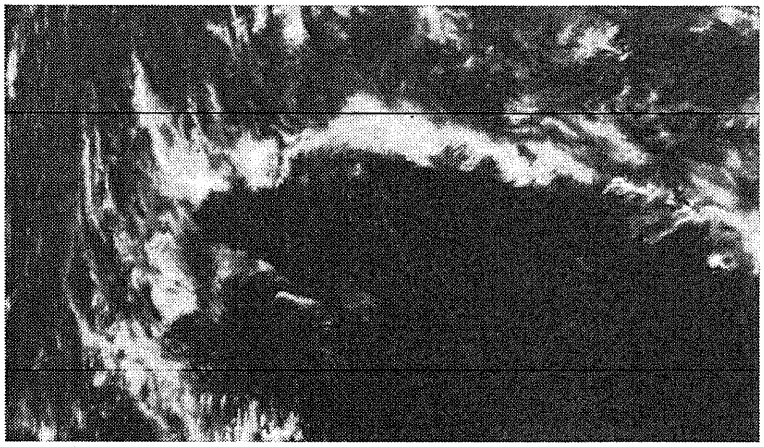


Figure A.2: Example of noise lines in AVHRR data (A magnified part of channel 3, 544×314). The two black horizontal lines are the noisy lines.

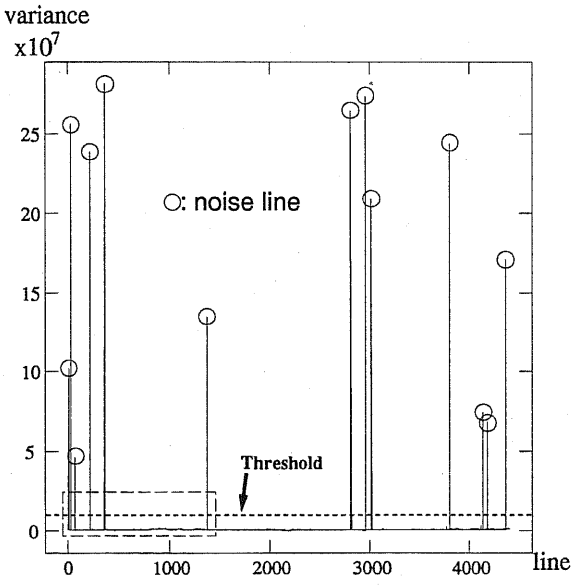


Figure A.3: Calculated variance for each line. Circles: detected noise lines, dotted line: threshold, rectangle with broken line: magnified area in figure A.4

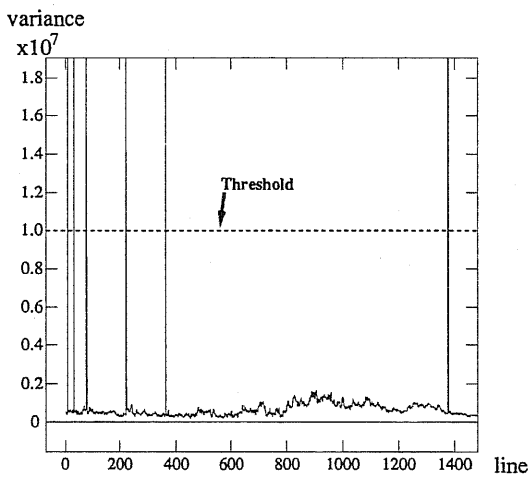


Figure A.4: Magnified part of figure A.3

Q	0–1	2–3	4–7	8–15	16–31	32–63	64–127	128–255	256–511	512–
Group #	0	1	2	3	4	5	6	7	8	9

Table A.2: Grouping table

portion is cut off. The remaining unpredictable portion and the noise lines are compressed separately from the image data. They are passed to gzip and compressed. Gzip is invoked with the '`-9`' option, which specifies the best compression.

This process is HRPT data dependent.

A.3.2 Pixel classification

Each image pixel has different properties under certain criteria. From the point of image compression, grouping similar-propertied-pixels and encoding them respectively generates effective results. The more generalized discussion is given in section 2.3.

Here we use the following Q value for grouping the pixels.

$$Q \equiv |P_2 - P_1| + |P_3 - P_1| + |P_4 - P_1| + |P_5 - P_1| \tag{A.1}$$

The position of the pixels ($P_1 \ldots P_5$) are shown in figure A.5. This can also be calculated during decoding process, because only the upper or left pixels are used in equation(A.1). Using this Q value, we classify each pixel into several groups according to table A.2.

A.3.3 Multi-channel prediction

For each classified group, we predict the value of the current pixel using linear combination of its neighbors' pixel values. The coefficients are calculated by the least square error method and use a constant to let the mean error be zero.

The neighbor pixels used for prediction are shown in figure A.5 ($P_1 \ldots P_{10}$). For the pixels in the first channel, we use these 10 neighboring pixels.

The already decoded pixels are used to predict the pixels in the next channel ($R_1 \ldots R_9$ in figure A.5). Using these pixels, something like interpolative prediction is achieved. This prediction contributes to the compression.

It is possible to use more than one of the previous channels if they are already decoded, but this increases processing time. We consider that looking for the optimal encoding order is more important. This is what we are investigating now.

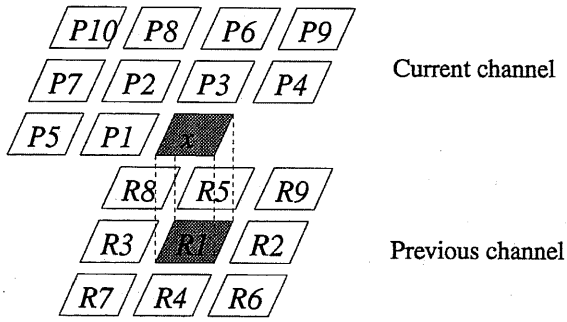


Figure A.5: Pixels used for prediction

A.3.4 Error conversion

The conversion method used here is just a 10-bit version of the 'ODC' method described and evaluated in section 2.2.1. After this conversion, we can also get the 10-bit non-negative integer E.

A.3.5 Distribution fitting and entropy coding

In section 2.4, we derived the formulation of error distribution as the convolution of Gaussian and generalized Gaussian pdf. It is expected to better approximate the error signal obtained from AVHRR data than the Gaussian or Laplacian distribution. But particularly for AVHRR images, the distribution of E is well approximated by the Gaussian distribution[22, 21]. As this is quicker, we adopted this Gaussian pdf. This distribution is used to generate the initial probability table for the encoder and decoder. The Gaussian distribution requires only one parameter — the variance — to be regenerated.

Figure A.6 shows the graph of E vs. normalized distribution (probability) for each group (0...9). They do not exactly have the Gaussian shape, but for approximation and initial distribution generation, Gaussian curve fitting works well to reduce the code size. And it is clearly seen that from this figure, the curve of lower group (lower Q value) has more accurate peak (less variance) than that of upper group.

For the entropy coding, we adopt an arithmetic coder, because it has very effective performance and it is easy to make it adaptive.

A.4 Experimental Results

We programmed the compression program in C, on an HP9000/735. In this section the compression performance of our method and the time needed.

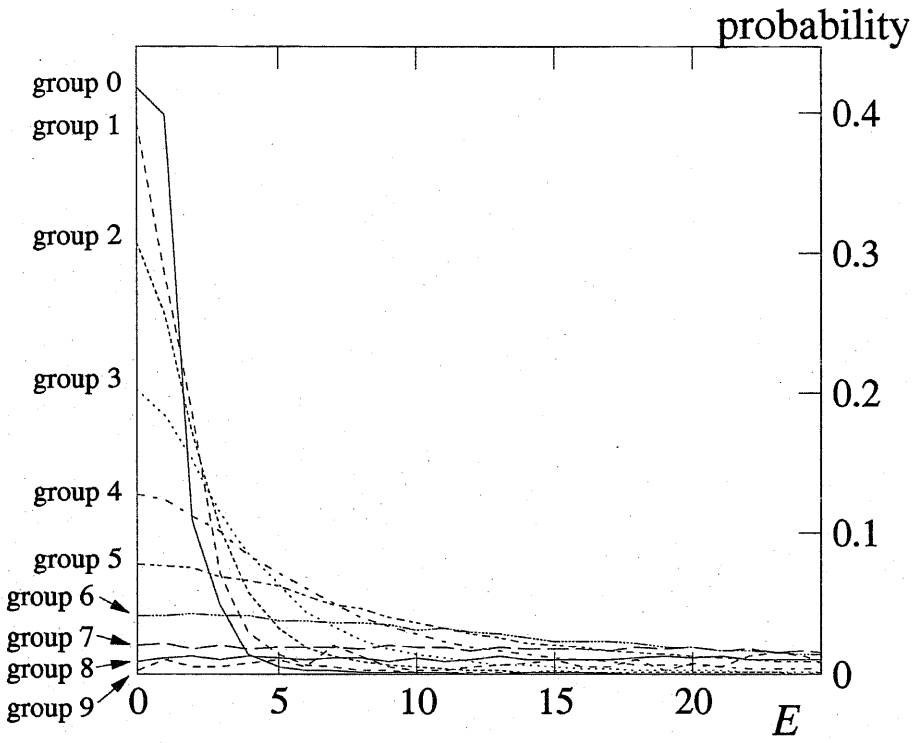


Figure A.6: E vs. probability curve

Table A.3 lists the compression results made on 11 HRPT data obtained in December 1993. The column "stored size" means the whole file size of archived HRPT data in bytes, line-number × 22180 (the size of HRPT single line). The column "original size" means the actual amount of bits from NOAA satellite in bytes, "stored size" ×10/16. This difference is because we store one HRPT word (ten bits) by two bytes (see section A.2). This value is used to calculate the column "compression ratio" (C.R.), i.e. "original size" divided by "compressed size".

In table A.4 the comparison with 'gzip -9' is shown. Also the compression ratio is given by the ratio of original size to compressed size.

A.5 Conclusion

We proposed an effective method of lossless compression of NOAA HRPT images. Our method accomplishes the compression ratio of around 3 to 4. It actually means that in our receiving system the amount of HRPT data is reduced down to around one fifth. Though we do not use the same image as in other experiments found in the literature, the

Date	lines	stored size	original size	C.R.	time(sec)
Dec.4, 15JST 1993	4400	97592000	60995000	3.219	4586
Dec.5, 15JST 1993	3390	75190200	46993875	3.506	3621
Dec.5, 16JST 1993	3023	67050140	41906337	3.458	3134
Dec.6, 15JST 1993	4400	97592000	60995000	3.219	4655
Dec.6, 16JST 1993	3607	80003260	50002037	3.245	3801
Dec.7, 14JST 1993	4289	95130020	59456262	3.277	4535
Dec.7, 16JST 1993	4023	89230140	55768837	3.309	4247
Dec.8, 14JST 1993	4358	96660440	60412775	3.338	4567
Dec.9, 14JST 1993	4087	90649660	56656037	3.194	4245
Dec.9, 16JST 1993	4400	97592000	60995000	3.189	4600
Dec.9, 17JST 1993	3053	67715540	42322212	4.713	3167

Table A.3: Results of compression ratio (C.R.) and processing time

Date	lines	C.R.(gzip)	C.R.(proposed) and time(sec)
Apr.2, 15JST 1993	4400	1.204	3.018 (3303)
May 7, 14JST 1993	3187	1.188	3.066 (2381)
May 20, 15JST 1993	4400	1.137	2.730 (3297)
May 13, 20JST 1994	3267	1.686	4.280 (2522)

Table A.4: Compression comparison with gzip and time

compression ratio by Kim's method[34] is around 2, and Tate's method for AVHRR data is around 2.7[27].

It is possible that the encoding order of channels has effect on compression ratio. Currently we encode five channels simply in channel order(1, 2, 3, ...), and we only use the previous channel's pixels for prediction. Tate reports that for multi-spectral image, a well-chosen encoding order performs as well as optimal order[27]. He also reports that the effect of channel ordering makes slight difference especially for AVHRR data. We will seek for the optimal order and investigate if it is applicable to our data, and also examine using more channels for prediction.

It usually takes about one hour to compress a single HRPT datum. On the other hand, decompression is around six times faster. As mentioned in the introduction, the compression ratio matters more than the compression time, but this time might be considered too long. Therefore we are thinking of faster, more efficient and less redundant encoding algorithms.

Appendix B

Nonlinear Prediction with Markov Model

TO losslessly encode still grayscale images, the methods like JPEG lossless mode which linearly predict the pixel using surrounding causal pixels, are mainly used, and yield good results. These methods can be classified as autoregressive model. On the other hand, applying Markov model for images yield attractively lower entropy than autoregressive model. Its drawback is that it requires a huge amount of storage to maintain the frequency table for encoding and decoding. Hence, it has only been used for binary picture compression so far.

B.1 Introduction

In this appendix, I propose learning Markov model for grayscale image and several prediction algorithms based on this model. Then compare with traditional linear prediction in terms of entropy of prediction error. Experimental results show the potential of nonlinear pixel predictor based on Markov model image representation.

Learning Markov Model

The Markov model described here is precisely discussed in section 4.1. Briefly speaking, we use a two-dimensional second order Markov model whose state S is represented as $S(X_1, X_2)$, where X_1 and X_2 are as shown in figure B.1. Although this model is very simple, the entropy of this model is much lower (1-2 bit/ pixel) than the bit rate of former lossless compression scheme, which implies further compression possibilities.

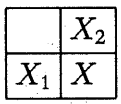


Figure B.1: Support of a second order Markov model

Our learning Markov model predictor basically starts without preconceived knowledges. Processing the pixels in scan-line order with help of linear predictor, it grows the frequency table of its Markov states. By doing this, there is no need to transmit a huge Markov state information, and the encoder and decoder can have identical frequency tables.

When the appearance frequency of a certain Markov state exceeds a certain threshold T, our encoder predicts the value of the pixel using the methods described in the next section.

B.2 Prediction Algorithms

We tested five prediction schemes. They all predict values of pixels using the said frequency table, except for "Linear prediction".

Linear prediction Using twelve surrounding causal pixels prediction value is calculated as their linear combination. The coefficients are decided by minimum square error method.

Nearest value prediction Among the values memorized in the said frequency table, find one which is the closest from the actual pixel value. Its "probability interval" and prediction error is output.

Maximum likelihood prediction If there is a convention between encoder and decoder that "choose the one whose frequency is the highest", there is no need to send the "probability interval", although the increase of prediction error code can be foreseen.

Mean value prediction Mean value (expected value) of the pixel calculated from the said frequency table can be output as the prediction. It is not necessary to send the "probability interval" either, like "linear prediction" and "Maximum likelihood prediction".

Minimum entropy prediction When the sum of *"code for prediction value P"* and *"code for prediction error E"* could be minimized, such a P would be optimum in terms of output code length.

The probability of P can be obtained from said frequency table, but the problem is how to know *"the pdf of E"*. From our experiments, it is known that E is strongly concentrated around 0. For that reason, we initially generate the probability table proportional to Laplacian distribution with mean 0 and variance 1, and successively update the table.

Let $p_{\text{diff}}(E)$ denote the pdf of prediction error E and $p_{\text{Markov}}(X)$ the probability of X obtained from said Markov frequency table. Then

$$\begin{aligned} \text{the information amount of prediction } X &= -\log_2(p_{\text{Markov}}(X)) \text{ bit,} \\ \text{and the information amount of error } E &= -\log_2(p_{\text{diff}}(E)) \text{ bit.} \end{aligned}$$

Hence the sum of them, $I = X + E$, equals $-\log_2(p_{\text{Markov}}(X)p_{\text{diff}}(E))$bit.

Minimizing I is equivalent to maximizing its argument $p_{\text{Markov}}(X)p_{\text{diff}}(E)$. As they both are represented by raised integers, this check is done quickly by integer multiplications. Therefore logarithmic calculation is not necessary at all.

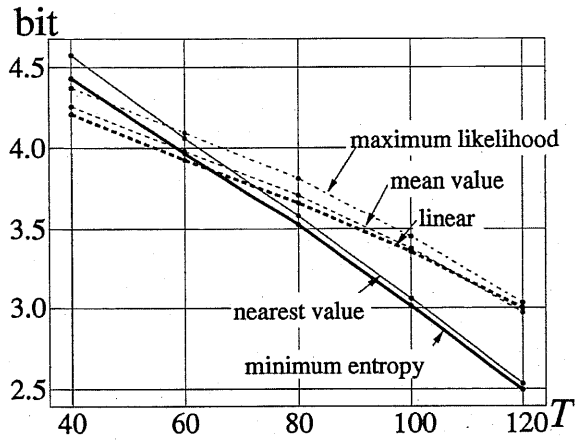


Figure B.2: Prediction (error) entropy. image=goldhill

B.3 Experimental Results

Figure B.2 to B.4 show the experimental results for the images 'goldhill', 'zelda' and 'hotel' (720x526). The vertical axis means the sum of the entropies of prediction value (probability range) and the prediction error. The horizontal axis means the threshold T. To make the conditions the same, the linear predictor only processed the pixels which other Markov predictor could predict (i.e. the pixels whose Markov state frequency exceeded the threshold T).

From the figure, the 'minimum entropy predictor' gives the best results among the Markov-type predictors, followed by 'nearest value predictor'. The ratio of the 'Markov-predicted' pixels in the whole image is, 16.5% for goldhil at $T = 80$, 74.2% for zelda at $T = 40$.

B.4 Conclusion

Here we propose learning Markov model which requires no frequency information, and several pixel prediction method using this model. From the experimental results, the possibility of nonlinear high-performance predictor is assured.

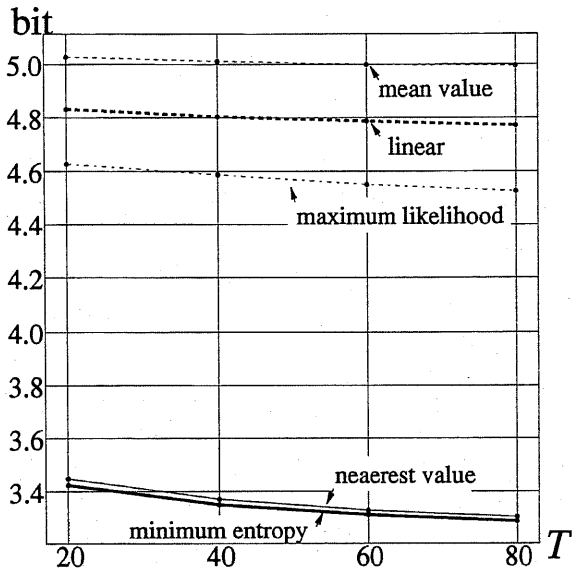


Figure B.3: Prediction (error) entropy. image=zelda

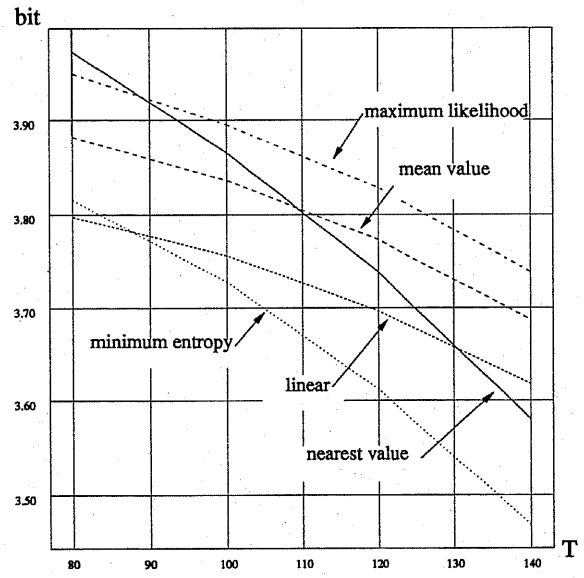


Figure B.4: Prediction (error) entropy. image=hotel