國 立 交 通 大 學

電機與控制工程學系

博 士 論 文

具邊界前瞻之非失真影像預測編碼技術

Predictively Encoded Techniques with Edge-look-ahead for Lossless Compression of Images

研 究 生：高立人

指導教授：林源倍　博士

中 華 民 國 九十七 年 九 月

具邊界前瞻之非失真影像預測編碼技術

Predictively Encoded Techniques with Edge-look-ahead for Lossless Compression of Images

研 究 生：高立人　　Student：Lih-Jen Kau

指導教授：林源倍　　Advisor：Yuan-Pei Lin

國 立 交 通 大 學

電 機 與 控 制 工 程 學 系

博 士 論 文

A Dissertation

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

in

Electrical and Control Engineering

September 2008

Hsinchu, Taiwan, Republic of China

中 華 民 國 九十七 年 九 月

具邊界前瞻之非失真影像預測編碼技術

研究生：高立人　　指導教授：林源倍 博士

國立交通大學電機與控制工程學系（研究所）博士班

摘　要

非失真影像編碼在許多場合皆有其應用之需求性；例如醫學影像編碼、遠端感測，以及影像壓縮等。在進行非失真影像編碼過程中，如何有效地移除統計累贅至今仍是訊號源編碼研究領域的一個主要挑戰課題。因此已有許多關於非失真影像編碼的方法被提出。其中有部分的研究係使用可還原小波轉換。然而由各項文獻中可以發現，使用轉換編碼所得到的結果往往卻不若以空間域之預測編碼結合環境建模（predictive coding with context modeling in spatial domain）所得到之結果來的好。

在本篇論文中，我們將針對近年來非失真影像編碼之概況做一簡介。此外我們亦將提出一基於線性預測之架構並以最小平方法進行此預測器係數之修正。由於使用最小平方法進行預測器係數之修正具有所謂邊界導向之特性（edge-directed characteristic），因此對於位處影像邊界附近像素之預測具有非常良好的效果。然而若將整張圖像皆以最小平方法進行預測器係數之修正勢將導致極高之複雜度，因此我們提出當編碼過程遭遇影像之邊界時才以最小平方法進行預測器係數之修正；如此運算複雜度將可大幅地降低。為了能夠於編碼過程中事先偵測影像邊界之存在與否，我們提出了一個非常簡易、有效而且僅使用已掃瞄/編碼像素（causal pixels）的邊界偵測法。如此一來我們的系統便可以預知影像邊界存在與否，並在遭遇邊界時，事先以最小平方法進行預測器係數之修正以防範較大預測誤差之發生。在我們所提出的方法中僅使用到已掃瞄/編碼像素來進行預測編碼；因此並無需傳送額外之資訊。經由實驗證明，我們所提出的方法能夠有效地在預測結果與運算複雜度之間取得良好的平衡點。此外我們也透過大量的實驗並針對當前非失真影像壓縮領域最先進的預測器（predictor）與編碼器（coder）進行比較來證明所提出方法之可用性。

除上述所提出具邊界前瞻（edge-look-ahead）能力之非失真預測編碼架構外，我們發現較大的預測誤差通常好發於影像中具有邊界之處。因此，我們在本篇論文中提出一創新概念，亦即利用控制工程的技術來改進影像中位於邊界附近像素之預測結果。之所以有這樣的想法是因為我們瞭解控制系統的目的本就是希望系統的輸出能夠準確地遵循所輸入之控制命令；因此在目的上與預測編碼是一致的。此外影像中的邊界（亦即像素的急速變化）亦可視為控制系統中的步階命令（step command）。基於前述的觀察使我們產生了嘗試以控制方法來改進預測編碼效能的想法。為了實現這樣想法，我們也以 Takagi 以及 Sugeno 兩位學者所提出之模糊類神經網路實做了一適應性預測編碼架構。此外我們也將控制領域中經常被使用的 Proportional Controller（P 型控制器）實現於此一模糊類神經網路中以強化影像中位於邊界像素之預測結果。我們發現這樣的作法對於預測編碼的效能改善確實是有所幫助的；雖然目前的改進效果並不是那麼地顯著，但是這樣的概念的確為非失真影像預測編碼領域開啟了一個全然不同的問題思考以及解決方式。

Abstract

Lossless image coding is required by many applications, such as medical imaging, remote sensing, and image archiving. It has remained a major challenge to source coding community for the difficulty of removing statistical redundancy effectively and efficiently. Therefore, many approaches have been proposed for lossless compression of images. Among proposed approaches, some of which are based on reversible transform coding, like integer wavelet transformation. However, we find in literatures that the results obtained by using transform coding are typically inferior to that of obtained by predictively encoded techniques with context modeling in spatial domain.

In this dissertation, an introduction on recent advances in lossless image coding will be given. Moreover, we will propose an approach based on linear predictive coding with least-squares (LS) optimization for the adaptation of predictor coefficients. The LS-based adaptive predictor, for its edge-directed characteristic, has been shown to be useful for the prediction of pixels around boundaries. Instead of performing LS adaptation in a pixel-by-pixel manner, we adapt the predictor coefficients only when an edge is detected so that the computational complexity can be significantly reduced. For this, we propose a simple yet effective edge detector using only causal pixels. This way, the proposed system can look ahead to determine if the coding pixel is around an edge and initiate the LS adaptation in advance to prevent the occurrence of a large prediction error. Furthermore, only causal pixels are used for estimating the coding pixels in the proposed encoder; no additional side information needs to be transmitted. As we will see later in the experiments, a very good trade-off between

prediction results and the computational complexity can be obtained with the proposed approach. Besides, extensive experiments as well as comparisons to existing state-of-the-art predictors and coders will be given to demonstrate the usefulness of the proposed approach.

In addition to the proposed edge-look-ahead approach, we find a large prediction error can usually take place for pixels around boundaries. Therefore, we also propose in this dissertation a novel concept of using control technologies to improve prediction result for pixels around boundaries. This idea comes from the fact that the purpose of a control system is to follow the input command as precisely as possible, which has the same objective with predictive coding. Moreover, an edge or a boundary can be regarded as a step command in control system. The above observations lead to the idea of solving this problem using control technologies. To realize this idea, we also implement an adaptive predictor using Takagi-Sugeno fuzzy neural network (TS-FNN). Moreover, the widely used proportional controller in control theory is applied implicitly in the consequent part of the network as a compensator to enhance the prediction result around edges. The effectiveness of the proposed novel approach, though not very conspicuous at present, can be further improved if a more sophisticated compensator is applied, and what's more, we have brought up an idea of solving this problem in a quite different aspect for lossless compression of images.

誌　謝

回首民國七十六年時母親引領著我來到交大控制工程系報到；從那一刻起，交大控制系這個名字便已伴隨著我開始了人生的重要旅程。當時的情景依然如此地鮮明在目，然而轉瞬卻已過了二十一個年頭。在這二十一個年頭當中，立人在交大控制系歷經了人生最重要的大學時代、專任助教工作、碩士學位以及甫完成的博士學位等不同階段。一直以來，立人在系裡面受到了許多師長的照顧與提攜；這點點滴滴，立人都感激並謹記在心。

在立人博士班就學期間承蒙林源倍教授不棄，願收入門下加以指導並進行論文研究。林源倍教授治學極為嚴謹；在林源倍教授之指導下，立人得以培養謹慎的研究態度與研究方法。若無老師不厭其煩地反覆指導實驗進行與論文寫作，立人絕無法達成今日之目標。此間立人亦因工作之故，必須往返花蓮新竹兩地，對於自身之研究進度多未能如期完成；然而老師非但未曾加以責難，尚且加以關懷立人之家庭與工作概況，亦為師亦為友；此份包容與提攜之情，使立人甚感慚愧與感激。

在研究期間，實驗室歷屆學弟妹們的相互鼓勵與扶持使我能走過研究的低潮。特別是建樟，常常受到我的麻煩，最是辛苦。由於實驗室學弟妹成員眾多無法一一列出，在此一併致上立人的感謝與祝福。

在最後階段的博士論文口試期間，系上的鄧清政教授、林進燈教授以及吳炳飛教授，清大電機系的陳博現教授以及中研院資訊所的黃文良教授慨然允諾擔任口試委員並撥空前來給予立人指導，斧正論文，使此一論文能更臻於完備，在此向老師們致上最深的感謝。口試畢後各位老師所給予高度的評價更使立人感到惶恐；立人也深知自己的耕耘不夠；這份肯定更代表未來需要加倍地努力。

立人於民國八十七年返回花蓮大漢技術學院任職至今，期間前後任校長張國照教授以及康自立教授不以立人駑鈍，屢屢委以重任；此份提攜之情，立人點滴感受在心。此外兩位校長的智慧以及謙謙君子的學者風範更是我後生晚輩所需效法學習的對象。

岳父林添龍先生，岳母張春崇女士在立人與內人林純雯女士婚後，持續給予最大的支持，使我們能夠順利地建立起屬於一家人的溫暖天地。此外岳父母尚且需忍受兩個小外孫需索無度的玩具購買要求，真是難為他兩老了。

立人與兄樹人雖幼年失怙，然母親黃牡丹女士一肩挑起重擔，獨立扶養我兄弟二人長大成人，提供我們衣食無缺、沒有憂慮的生活並接受高等教育；慈恩浩蕩，山高水長。母親自我兄弟二人踏入社會後，秉持「施比受更有福」的理念，開始投身社會關懷以及全職志工的行列。多年以來不求回饋地付出，而能贏得鄰里的敬重，亦曾當選全國

好人好事代表，獲頒八德獎。古諺云：「積善人家慶有餘」；相信這也是我兄弟二人多年以來能遇見許多師長、朋友提攜扶持的原因吧。年已古稀的母親不忍立人忙於學校行政、教學工作，同時又有進修學位的壓力，原應含飴弄孫、享清福的母親還需每日哄騙我們家裡那兩個小惡魔，照顧他們洗澡、吃飯、就寢還要受氣。凡此種種，做為不孝兒子的我要向您說一聲：「媽媽您辛苦了，媽媽我愛您」。

在立人成長的過程中，雖然沒有父親的榜樣可以學習，但是母親以及兄長樹人一路的引導，使立人在面對重大挑戰與困難時猶如一盞指引的明燈。正所謂「長兄如父」，大哥的擔子也總是特別地重，從小以來您就是我的榜樣，謝謝您。

這八年來，除進修博士學位外，也完成了終身大事。九十二年與九十五年間，長子高暉以及次子高旭亦分別報到。內人純雯在我擔任行政工作並進修博士學位期間分擔我家那兩隻小犬的照顧工作，使我沒有後顧之憂。時常身兼兩職的內人也使孩子們瞭解到慈母也有威嚴的一面，不可放肆。辛苦妳了純雯，謝謝妳讓我有了一個完整的家。

學生生涯階段雖然已告一段落，但是真正的挑戰（研究）才要開始。面對眼前一片開闊的天際，我將帶著大家的祝福繼續在學術研究的路上前行、飛翔。

最後謹以此論文獻給關心立人的師長、朋友以及我的家人，謝謝大家。

高立人

97 年 09 月 9 日

於新竹

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Lossless image coding is required by many applications, such as medical imaging, remote sensing, and image archiving. It has remained a major challenge to source coding community for the difficulty of removing statistical redundancy effectively and efficiently. In this chapter, we will give a short description on the differential encoding technique, which is widely used in image and speech encoding system. Moreover, a review on recent advances in lossless image coding will also be introduced. After that is an overview of the proposed predictive coding scheme. Finally, the chapter outline of the dissertation will be given.

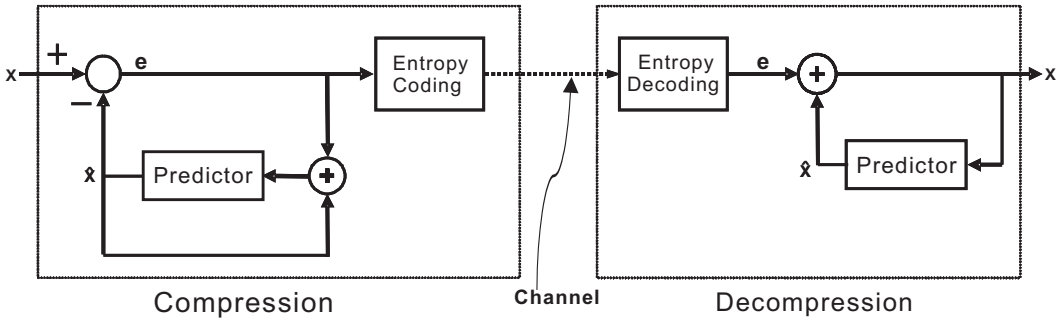


Figure 1.1: Basic block diagram of a lossless differential encoding system.

1.1 Advances in Lossless Image Coding

There have been great advances in lossless image coding recently [1]-[31]. Some of which are based on reversible wavelet transformation using lifting structure [6]-[10]. By using integer wavelet transformation, lossless to near-lossless compression as well as progressive reconstruction of image data can be achieved [6]-[10]. However the compression results obtained with the use of integer wavelet transformation are typically inferior to that of obtained by predictively encoded techniques [17].

The predictive coding scheme, known as the differential pulse code modulation (DPCM), is used in a wide variety of applications such as image and speech compression for ease of implementation [1]. Due to the high correlation between successive image samples, the differential encoding technique removes the inter-pixel redundancy by encoding the difference between successive image samples rather than the samples themselves. Since the difference between samples is expected to be smaller than the actual sampled amplitudes, fewer bits are required to represent the difference. Thus, the differential encoding removes the inter-pixel redundancies and encodes only the new information between closely spaced image pixels.

For lossless compression of images, we show in Fig. 1.1 the basic block diagram of a differential encoding system (predictive coding system). As can be seen in Fig. 1.1, the lossless predictive coding system is composed of two major blocks; the predictor and the entropy coder. In order that a lower first-order entropy and hence a lower actual bit rate can be obtained, many researches on the design of an effective and efficient predictor for removing the statistical redundancy among coding pixels have been proposed. Among which, adaptive predictors with context modeling are often used to accommodate the varying statistics of coding images [11]-[31]. Besides, adaptive prediction is achieved in most of the coders by using multi-predictor structures [11]-[22]. Among which, the CALIC coding system [14], a state-of-the-art lossless coder proposed for JPEG-LS, uses a gradient adjusted predictor (GAP). Based on the gradient of neighboring pixels, one out of a set of seven predictors is chosen. The LOCO-I coder [15], an algorithm motivated by CALIC [14] and standardized into JPEG-LS, uses a median edge detector (MED) to choose one of three predictors for current prediction. In [16], adaptive prediction is achieved by choosing one out of a set of predictors that minimizes the energy of prediction errors in a specified cluster of causal pixels, and the predictor coefficients of the selected predictor are then updated by applying gradient descent rule.

In [17]-[20], multi-pass prediction is introduced. With multiple passes, progressive transmission of lossless and near-lossless coding of image data can be achieved. Besides, the encoder can form a 360 degree prediction [19] or perform a global image analysis [20] by using multi-pass prediction. A highly complex two-pass coder called TMW has been proposed in [20]. Using multiple linear predictors and global image analysis, the TMW system can achieve lower bit rates than existing coders for most images. While achieving

very low bit rates, the computational cost is regarded as prohibitive in TMW [20]. Recently, a fuzzy logic-based adaptive DPCM algorithm called FMP [21] is proposed. The FMP presents a competitive, and in some cases superior result than TMW but with a lower computational cost. Though FMP is effective in removing the statistical redundancy, it still takes minutes.

In the context of optimal predictors, the minimum mean square error estimate of Y given observations $X_1, X_2, \cdots, X_n$ is $E\{Y|X_1, X_2, \cdots, X_n\}$, generally a nonlinear function. Therefore, there have been many results using neural networks as nonlinear estimators [22]-[24]. Neural network based predictors perform well in slowly varying areas. However, there can be large prediction error around boundaries [32]. The result can be improved using additional hidden layers or hidden neurons, but this incurs a drastic increase in complexity [23], [33].

The performance of predictive image coding scheme highly depends upon the effectiveness of the predictor used in the coding process. Most of the image predictors perform very well in slowly varying areas. However, large prediction errors can take place around edges and boundaries, and this has become a major problem to be conquered so far. Intuitively, the prediction results can be improved if we can foresee the existence of an edge and then predict along the edge orientation. However, the design of a robust edge detector and the analysis of edge orientation are difficult problems themselves, let alone to predict along the edge orientation. Recently, linear predictors adapted by least-squares (LS) optimization have been proposed as an efficient approach to accommodate varying statistics of coding images [25]-[31]. Among which, the EDP [26] pointed out that the superiority of LS adaptation is in its edge-directed property. That is, the LS-based predictor can adjust the prediction support along the edge orientation automatically during

the adaptation process. With the edge-directed property, LS-based adaptive predictor performs very well for pixels around boundaries. For complexity consideration, performing the LS adaptation process in a pixel-by-pixel manner is regarded as prohibitive. Therefore, the EDP [26] proposed initiating the LS optimization process only when the prediction error is beyond a preselected threshold such that the computational complexity can be reduced. The EDP [26] has made a noticeable improvement over the state-of-the-art lossless coder CALIC [14]. On the other hand, we know that the normal equations provide the key for LS adaptation, and some fast algorithms, Cholesky decomposition for example, can be applied in the LS adaptation process. Therefore, the complexity in solving the normal equations itself is not a problem. Nevertheless, the computational cost for the construction of normal equations is rather high. Thus, an algorithm for the fast construction of normal equations has been proposed in [25] so that the computational cost for LS adaptation process can be reduced significantly.

1.2 The Proposed approach

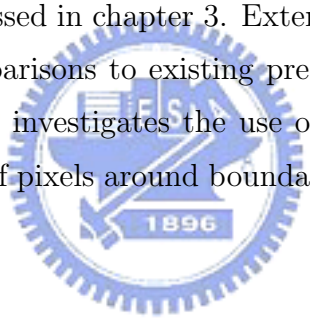
It is known that many coding methods are more efficient with some images than others. In particular, run-length coding is very useful for coding areas of little changes. Adaptive predictive coding achieves high coding efficiency for fast changing areas like edges. In this dissertation, we propose a switching coding scheme that will combine the advantages of both Run-length and Adaptive Linear Predictive coding (RALP). There are other switching methods that achieve very low bit rates [20]-[23]. However the results are usually obtained with a very high computational complexity [20]-[23]. On the contrary, the proposed RALP coder can achieve a very good coding efficiency but still with a moderate computational complexity. In the proposed approach, the run-length encoder is used for pixels in slowly varying areas; otherwise an LS-based adaptive predictor is used. The LS-based predictor has been shown to be very useful for the prediction of pixels around an edge [26], [27]. Moreover, we adapt the predictor coefficients only when an edge is detected or when the prediction error is beyond a pre-selected threshold so that the computational cost can be significantly reduced [27]. To do this, we use a simple and efficient edge detector that uses only causal pixels, i.e., pixels that have already been coded. This way, the predictor can look ahead if the coding pixel is around an edge and initiate the LS adaptation process beforehand to prevent the occurrence of a large prediction error. With the proposed switching structure, very good prediction results can be obtained in both slowly varying areas and pixels around boundaries. Some preliminary results regarding the proposed LS-based predictor with edge-look-ahead can be found in [27].

For prediction error refinement, the so-called context modeling technique [14], [19] is used in the proposed system. The compensated error has a

narrower histogram and hence a lower first-order entropy. As we will see in the experiments that the switching structure combined with edge-look-ahead prediction as well as automatic error modeling renders the proposed RALP highly adaptable and very feasible under limited resources. A very good trade-off between coding efficiency and computational complexity can be achieved. Comparisons with existing state-of-the-art LS-based predictors can also be found in our experiments.

1.3 Outline of the Dissertation

The rest of the paper is organized as follows. Chapter 2 introduces the proposed LS-adaptive predictor with edge-look-ahead. The entropy coding of prediction error is addressed in chapter 3. Extensive experiments of the proposed method and comparisons to existing predictors and coders are given in chapter 4. Chapter 5 investigates the use of control technologies to enhance prediction result of pixels around boundaries. A conclusion is given in Chapter 6.



Chapter 2

Least-squares Based Adaptive Predictor with Edge-look-ahead

In this chapter, details on the proposed least-squares (LS) based adaptive predictor will be addressed. First of all, we will give an overview on the proposed predictive coding system. Secondly, an illustrative example will be given to manifest the edge-directed property of LS adaptation. In order that the proposed system can foresee the existence of an edge, we will introduce in this chapter the proposed causal edge detector. After that, a detailed description on the LS adaptation process will be given. Finally, the so-called bias cancelation technique for prediction error refinement will be addressed.

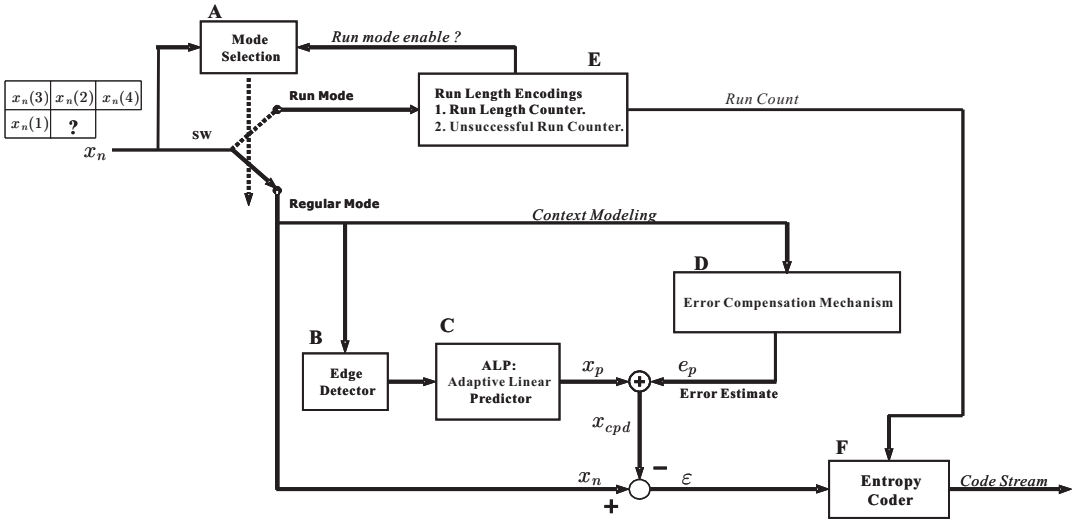


Figure 2.1: Proposed RALP coding system.

2.1 An Overview of the Proposed System

Many coding methods are more efficient with some images than others. In particular, run-length coding is very useful for coding areas of little changes. Adaptive predictive coding achieves high coding efficiency for fast changing areas like edges. In this dissertation, we propose a switching coding scheme (as shown in Fig. 2.1) that will combine the advantages of both Run-length and Adaptive Linear Predictive coding (RALP). For pixels in slowly varying areas, run-length coding is used; otherwise least-squares (LS) based adaptive predictive coding is used. Instead of performing LS adaptation in a pixel-by-pixel manner, we adapt the predictor coefficients only when an edge is detected so that the computational complexity can be significantly reduced. For this, we use a simple yet effective edge detector using only causal pixels. This way, the proposed system can look ahead to determine if the coding pixel is around an edge and initiate the LS adaptation in advance to prevent

the occurrence of a large prediction error. With the proposed switching structure, very good prediction results can be obtained in both slowly varying areas and pixels around boundaries. Furthermore, only causal pixels are used for estimating the coding pixels in the proposed encoder; no additional side information needs to be transmitted.

The proposed RALP system, as shown in Fig. 2.1, has two operation modes, run mode and regular mode. The "*mode selection*" block (Fig. 2.1) determine if the current pixel is in an local area of little changes. If it is, the run mode is triggered and the current pixel is encoded using run-length encoding. If not, the regular mode is assumed and the pixel is encoded using predictive coding.

Run mode

It is known that the run-length coding is most efficient for the encoding of consecutive pixels with identical grey values. The case that consecutive pixels are identical can usually occur in an artificial image or in slowly varying areas of a natural image. Therefore, we use the run-length coding in the proposed RALP system for the encoding of pixels in an area of little changes. If the four pixels $x_n(1), \ldots, x_n(4)$ in Fig. 2.2 are identical, the run mode is switched on and the run-length is encoded using an arithmetic coder with an alphabet set of $\{0, 1, \ldots, 20\}$. The "0", called escape symbol in the proposed approach, is used to indicate an unsuccessful run and should be encoded if the grey value of the coding pixel x_n and that of $x_n(1), \ldots, x_n(4)$ are distinct so that the decoder can also make a right decision and quit the run mode automatically. This time, the regular mode is used for the encoding of the current pixel. It is noted that the run mode can also be broken by ends of lines, in which case the encoder returns to the regular mode, i.e., the regular mode is assumed for

the first pixel of every line. Moreover, the encoding of an escape symbol can cause penalty and degrade the coding efficiency. Therefore, we record the number of times of run mode triggered and the times of unsuccessful run. If the percentage of unsuccessful run is greater than a predefined threshold, the run mode is disabled and not to be used for the rest of the coding process. It is noted that all the pixels used for mode selection are causal and the decoder can reproduce the same decisions without any side information.

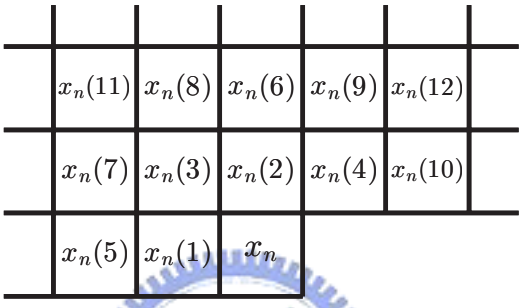


Figure 2.2: The ordering of pixels for prediction inputs.

Regular mode

In the regular mode, pixels are encoded using predictive coding. Predictive coding can be very efficient for the removal of statistical redundancy between neighboring pixels in slowly varying areas. However, there can have a large prediction error around boundaries. In this paper, we will use LS optimization to update the predictor coefficients on the fly so that the predictor can adapt itself to the varying statistics [25]-[28]. It is known that the LS-based adaptive predictor is an efficient approach to improve the prediction result around boundaries for its edge-directed property [26]-[28]. However, a pixel-by-pixel adaptation of the predictor coefficients is computationally expensive and often not necessary [25]-[28]. Therefore, we will initiate adap-

tation only when the prediction is inadequate, which is around an edge. For this, an "*edge detector*" is used to look ahead and determine if the coding pixel is around an edge so that the predictor can adapt itself beforehand to prevent the occurrence of a large prediction error. In the regular mode, the prediction error is further refined using *error compensation*. That is, the predictor output x_p is added by a correction term e_p (as shown in Fig. 2.1) to get a compensated prediction $x_{cpd} = x_p + e_p$. The amount of compensation e_p is determined through an error modeling mechanism. The refined error signal $\varepsilon = x_n - x_{cpd}$ can then be entropy encoded using conditional arithmetic coding to produce the coded bit stream.

In the proposed encoder, only causal pixels, i.e., pixels that have already been coded, are used for estimating the coding pixels; no additional side information needs to be transmitted. Moreover, the proposed RALP coder is symmetric, meaning that the decoder has the same predictor switch as the encoder, and performs prediction and error compensation just like the encoder. Therefore, the actual pixel value can be reconstructed in the decoder with the received bit stream of refined errors. Details of the individual components of the system are introduced in subsequent sections.

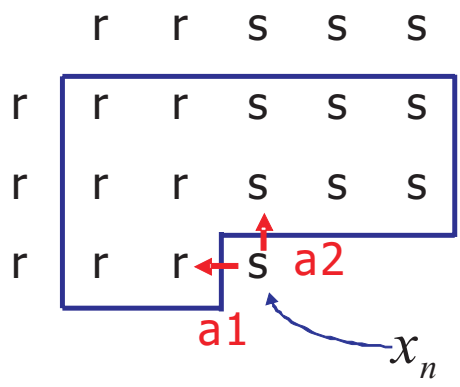


Figure 2.3: Area that contains a vertical edge.

2.2 Edge-directed Characteristic of LS adaptation

In predictive coding schemes, the effectiveness of any adaptive predictor depends upon its capability of adapting from slowly varying areas to edge regions [26]. Intuitively, the prediction results can be improved if we can foresee the existence of an edge and then predict along the edge orientation. However, the design of a robust edge detector and the analysis of edge orientation are difficult problems themselves, let alone to predict along the edge orientation. In contrast, the LS-based adaptation provides an elegant way of approximating the optimal orientation adaptive prediction due to its edge-directed property [26].

To illustrate the edge-directed property of LS adaptation, a simple example will be given so that the relationship between the prediction support and the edge orientation can be quantitatively analyzed. For this, we show in Fig. 2.3 an example in which the current pixel is along a sharp vertical edge, i.e., $|r - s| \gg 0$. For simplicity, we only consider the second-order predictor $\hat{x}_n = a_1 x_{west} + a_2 x_{north} = a_1 r + a_2 s$, in which case the training window has

12 elements (Fig. 2.3). To find the least-squares solution of the second-order predictor coefficients, i.e., a_1 and a_2, we start with the following equation, which is constructed by using the 12 training pixels in raster scan order.

$$\begin{bmatrix} r & r \\ r & r \\ r & s \\ s & s \\ s & s \\ r & r \\ r & r \\ r & s \\ s & s \\ s & s \\ r & r \\ r & r \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} r \\ r \\ s \\ s \\ s \\ r \\ r \\ s \\ s \\ s \\ r \\ r \end{bmatrix}, \tag{2.1}$$

where a_1 and a_2 are the predictor coefficients to be determined. Obviously, (2.1) is an over determined system. To find the LS solution of (2.1), the transpose of the coefficient matrix in (2.1) is left multiplied to both sides of itself, and we can get

$$\begin{bmatrix} 6r^2+2rs+4s^2 & 6r^2+6s^2 \\ 8r^2+4s^2 & 6r^2+2rs+4s^2 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 6r^2+6s^2 \\ 6r^2+2rs+4s^2 \end{bmatrix}. \tag{2.2}$$

By some simple operations, we find the optimal LS solution for the predictor coefficients are given by

$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}. \tag{2.3}$$

The edge-directed characteristic of LS adaptation can be best observed from (2.3) where the prediction support has been adjusted along the edge orientation (i.e., vertical orientation in this case) automatically. For a horizontal or arbitrarily oriented edges, the edge-directed property can also be verified by using similar mathematical derivations [26]. As we will see later in the experiments, the LS-based adaptive predictor, having the edge-directed

property, can have a very good performance for pixels around boundaries, and the edge-directed characteristic of LS adaptation will be utilized in the proposed approach.

2.3 Proposed Edge Detector

In the regular mode, we use a LS-based adaptive predictor to accommodate the varying statistics of the image. To save computations, the predictor is adapted only when prediction error is large or likely to be large. For a pixel around an edge, prediction error is usually large and adaptation is needed. To determine whether the coding pixel is around an edge, we use a simple yet effective edge detector in this paper. It should be noted that conventional edge detectors, e.g., "Sobel" operator, can not be applied here because they use non-causal pixels, i.e., pixels yet to be encoded.

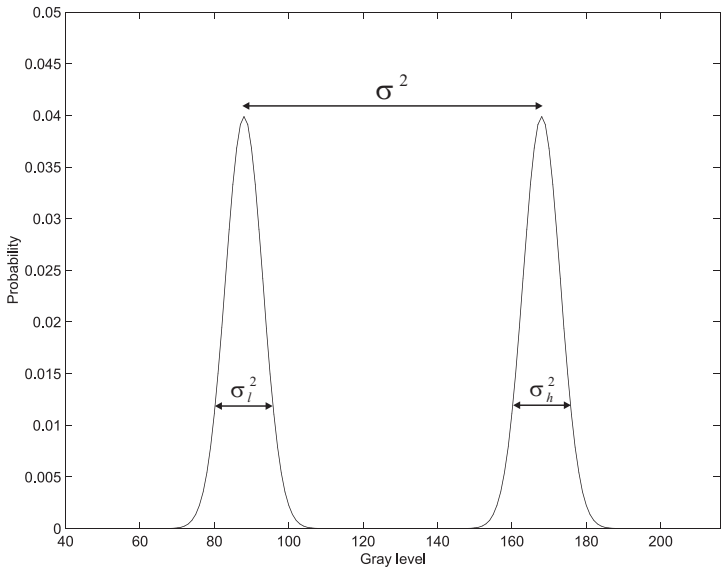


Figure 2.4: A typical histogram of an area that contains an edge.

We observe that an area that contains an edge usually has a large variance. Furthermore, the histogram of such an area tends to have two peaks, one on each side of the mean value (Fig. 2.4). We will use these two observations to determine the existence of an edge. Moreover, the set of the four pixels $\kappa =$

$\{x_n(1), \ldots, x_n(4)\}$ in Fig. 2.2 are used for the detection. The mean $\bar{x}$ and variance σ^2 of the set κ are calculated. Furthermore, the four pixels can be divided into two groups, the pixels with gray levels higher than $\bar{x}$ in group κ_h and the rest in group κ_l. We also compute the respective variance σ_h^2, σ_l^2 of the pixels in κ_h and κ_l (Fig. 2.4).

A pixel around an edge is likely to have a large σ^2 but small σ_h^2 and σ_l^2. We determine whether the coding pixel is around an edge if the following two conditions are both satisfied,

$$\sigma^2 \geq \gamma_1, \quad \text{and} \quad \sigma^2 \geq \gamma_2 \, (\sigma_h^2 + \sigma_l^2). \tag{2.4}$$

It is noted that the second condition in (2.4) is included because a region with uniformly distributed gray values also results in a large σ^2. Therefore, the switch first examines if $\sigma^2 \geq \gamma_1$ when a large σ^2 is detected then the switch checks the second inequality in (2.4). In this paper, the LS adaptation process in the regular mode is activated whenever the two conditions in (2.4) are satisfied. We have found through experiments that $\gamma_1 = 100$ and $\gamma_2 = 10$ work very well and these values will be used throughout this dissertation. It should also be noted that the run mode will be triggered when $\sigma^2 = 0$, i.e., the case that $x_n(1), \ldots, x_n(4)$ are identical. In this case, we do not have to check the conditions in (2.4). As we will see later in experiments that the proposed detector is very effective in detecting edges although only four pixels are used. Moreover, since we use only causal pixels for the detection of an edge, the decoder can perform the same edge detection operation and switches on the LS adaptation process.

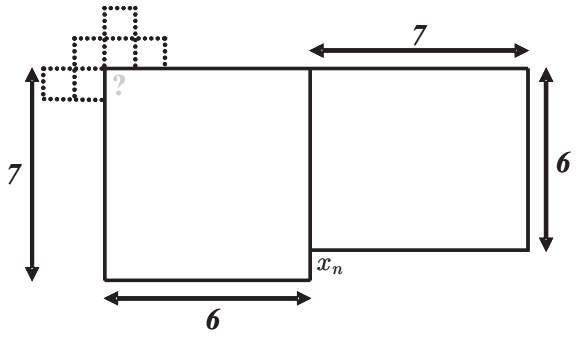


Figure 2.5: The online training regions for the proposed predictor.

2.4 Least-squares Based Adaptive Prediction

In the regular mode, the LS adaptation process is activated whenever the two conditions in (2.4) are both satisfied or when the prediction error is greater than a predefined threshold θ. The corresponding predictor inputs for different prediction orders are shown in Fig. 2.2 where the ordering of pixels is based on the distance to the pixel to be encoded. In this paper, the predicted value x_p of the coding pixel x_n is a linear combination of its causal neighbors given by

$$x_p = \sum_{k=1}^{N} a(k)x_n(k), \tag{2.5}$$

where N is the prediction order, $x_n(k)$ is the kth nearest neighbor of x_n and $a(k)$ is the corresponding predictor coefficient. It is noted that we do not use any training set for the optimization of initial predictor coefficients in this paper. The initial coefficients for the proposed predictor are equally weighted, i.e., the coefficients $a(k)$ for the Nth-order predictor are $1/N$ respectively.

The training area for LS adaptation process of the coding pixel is shown in Fig. 2.5. Suppose we have M pixels in the training area, our objective is

to find a least-square solution for the system

$$\mathbf{Pa} = \mathbf{y}, \tag{2.6}$$

where

$$\mathbf{P} = \begin{bmatrix} x_{n-1}(1) & x_{n-1}(2) & \dots & x_{n-1}(N) \\ x_{n-2}(1) & x_{n-2}(2) & \dots & x_{n-2}(N) \\ \vdots & \vdots & \ddots & \vdots \\ x_{n-M}(1) & x_{n-M}(2) & \dots & x_{n-M}(N) \end{bmatrix},$$

$$\mathbf{a} = \begin{bmatrix} a(1) \\ a(2) \\ \vdots \\ a(N) \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} x_{n-1} \\ x_{n-2} \\ \vdots \\ x_{n-M} \end{bmatrix}.$$

The optimal $\mathbf{a}$ that minimizes the square errors $\|\mathbf{y}-\mathbf{Pa}\|_2^2$ can be obtained by solving the normal equations [35]

$$\mathbf{P}^T\mathbf{Pa} = \mathbf{P}^T\mathbf{y}. \tag{2.7}$$

There are well-developed numerical approaches to solve (2.7). For the case that $\mathbf{P}$ has full rank; i.e., rank N, $\mathbf{P}^T\mathbf{P}$ is nonsingular and positive definite [35], [36]. The normal equations will have a unique solution $\mathbf{a} = (\mathbf{P}^T\mathbf{P})^{-1}\mathbf{P}^T\mathbf{y}$. In this case, the *Cholesky decomposition*, which requires only half the usual number of multiplications than alternative methods, can be used to solve (2.7) [35], [36].

If $\mathbf{P}$ is defective; i.e., rank $< N$, $\mathbf{P}^T\mathbf{P}$ fails to be positive definite and the *singular value decomposition* (SVD) can be used to solve (2.7) [35], [36] . The positive definite property of $\mathbf{P}^T\mathbf{P}$ can be easily examined in the process of *Cholesky decomposition* [36].

In the proposed approach, updated predictor coefficients are used for current prediction and passed on to the next coding pixel. For non-edge pixels, we use the stored prediction coefficients of the four nearest causal

Figure 2.6: The image "Lennagrey".

neighbors to generate four prediction values and take their average as the final prediction result. This manner, the predictor can resist against moderate salt-and-pepper noise. To summarize, the pseudo code of the proposed algorithm is given in appendix A.

2.5 Prediction Error Refinement

It is known that the prediction error $x_n - x_p$ in the regular mode can be further refined by learning from previous predictions, i.e., the so-called bias cancelation technique [14], [15], [19]. To do this, we define the *compound context* v of a coding pixel as,

$$v = \{x_n(1), \ldots, x_n(6), e_n(1), \ldots, e_n(4)\}, \tag{2.8}$$

where $x_n(i),\ i = 1, \ldots, 6$ are as shown in Fig. 2.2 and $e_n(i),\ i = 1, \ldots, 4$ are the uncompensated prediction errors corresponding to $x_n(1), \ldots, x_n(4)$ respectively. We have incorporated prediction errors in error modeling because the amount of compensation is likely to be related to the prediction errors of neighboring pixels.

In the proposed RALP system, error modeling is achieved by performing context clustering with a fixed number of contexts. For this, a set of initial contexts are generated off-line using the image "Lennagrey" (Fig. 2.6) for the clustering process. For each coding pixel, the compound context v is assigned to one of the existing contexts using mean absolute error (MAE) distance measure and the corresponding context is then modified accordingly in the coding process. By classifying coding pixels with similar context into the same group, the amount of compensation e_p for the current prediction can be estimated by calculating the sample mean of prediction errors in that group. Therefore, the value e_p to be used in compensating the current prediction is given by

$$e_p = \frac{S}{N}, \tag{2.9}$$

where S is the prediction errors accumulated in the context which v belongs to, and N is the number of members in that context. With the correction

term e_p, we now form a more refined prediction $x_{cpd} = x_p + e_p$. The compensated error $\varepsilon = x_n - x_{cpd}$ has a narrower histogram and hence a lower first-order entropy. In the regular mode, the refined error ε is then entropy encoded using a conditional arithmetic coder to produce the bit stream [28], [34].

2.6 Concluding Remarks

In this chapter, we have introduced the proposed switching structure for lossless compression of images. Moreover, a detailed description on the proposed LS-based adaptive predictor with edge-look-ahead, i.e., the core of the regular mode, is also given. In order that the adaptive predictor in regular mode can look ahead if the coding pixel is around an edge, we also propose in this research an edge detector using only causal pixels. Summarizing, the proposed approach has the following properties:

1. The run mode is triggered if the four pixels $x_n(1), x_n(2) \ldots x_n(4)$ in Fig. 2.2 are identical. The run length is then entropy encoded using an arithmetic coder to produce the bit stream.

2. An escape symbol have to be encoded to indicate an unsuccessful run if the value of the coding pixel x_n and that of $x_n(1), \ldots, x_n(4)$ are distinct. However, the encoding of an escape symbol can cause penalty and degrades the coding efficiency. Therefore, the run mode is disabled once the percentage of unsuccessful run is beyond a predefined threshold.

3. The proposed LS-based adaptive predictor, for its edge-directed property, is very useful for the prediction of pixels around boundaries.

4. The proposed edge detector uses only causal pixels, i.e., pixels that have been coded. Therefore, the decoder can perform the same operation just like the encoder and determine if the coding pixel is around an edge without any side information.

5. The LS-based adaptation process in regular mode is initiated only when an edge is detected or when the prediction error is greater than a prede-

fined threshold so that the computational complexity of the LS adaptation process can be significantly reduced.

6. With the proposed approach, the results obtained are are very close to those with pixel-by-pixel LS adaptation, but with a much reduced complexity. A very good trade-off between the prediction result and the computational complexity can be obtained as we will see later in Chapter 4.

Chapter 3

Entropy Coding of Prediction Errors

In the proposed approach, the refined error in regular mode is then entropy encoded through a conditional arithmetic coder to produce the bit stream. For entropy coding of the refined error signal ε, we borrow the concepts of "*error sign flipping*", "*error remapping*", and "*histogram tail truncation*" in [14], [15], [19] so that the coding efficiency in actual bit rates can be further improved. All the details on how the entropy coder works will be addressed in this chapter.

3.1 Conditional Entropy Coding

It is known that the coding efficiency can be further improved with the use of conditional probability models. By classifying similar prediction errors in the same group, the coding efficiency can be improved by sharpening the error histogram in each group and thus a smaller conditional entropy can be obtained. To perform the classification, we have to define an error strength estimate Δ so that both the encoder and decoder know exactly which group does the prediction error belongs to or which probability model should be used. In this dissertation, we define the error strength estimate of the coding pixel to be $|e_p|$, that is

$$\Delta = |e_p|. \tag{3.1}$$

It is noted that the e_p, introduced in section 2.5, is used in compensating the prediction error. Moreover, it is also available in both the encoder and the decoder. Therefore, we have applied the use of $|e_p|$ as the error strength estimate.

By conditioning on the error strength estimate Δ, we can quantize refined errors into classes of different variances [14], [19], [26]. Moreover, the error histogram in each quantization bins can be sharpened and thus a smaller conditional entropy can be obtained. Therefore, we are using the conditional probability model $P(\varepsilon|\Delta)$ instead of $P(\varepsilon)$ for entropy coding of refined errors. Furthermore, to find the optimal number of quantization bins as well as an optimal quantization for Δ so that the conditional entropy can be minimized, we define a cost function $C(Q, \varepsilon, \Delta)$ as

$$\begin{aligned} C(Q,\varepsilon,\Delta) &= E\{\, H(\,\varepsilon \,|\, Q(\Delta))\,\} \\ &= -\sum_{Q(\triangle)=1}^{N} \left\{ P(Q(\triangle)) \sum_{\varepsilon} P(\varepsilon \mid Q(\triangle)) Log_2 \left[P(\varepsilon \mid Q(\triangle)) \right] \right\}, \end{aligned} \tag{3.2}$$

where $Q(.)$, which maps Δ into one of the N quantization bins, is the quantizer to be designed, and N is the number of quantization bins to be decided. With such cost function, we are calculating the first-order entropy of ε assigned in each quantization bins, i.e., $H(\ \varepsilon \mid Q(\Delta))$, and then take expectation on the entropy $H(\ \varepsilon \mid Q(\Delta))$.

To minimize the cost function in (3.2), we use a set of training pairs (ε, Δ) recorded in the prediction process of the fourteen test images [20] in Chapter 4. By performing dynamic programming off-line, the final number of quantization bins is found to be three and the approximately optimized quantization of Δ is found to be $[0,1], (1,55], (55,\infty)$. Though the quantization may be suboptimal, the compression result is satisfactory as the simulation results in Chapter 4 will demonstrate. Using the above quantization bins, one out of a set of three probability models is chosen for the entropy coding of ε based on the value of error strength estimate Δ as below

$$\begin{cases} using \quad pmf\,1, \quad if \quad 0 \leq \triangle \leq 1 \\ using \quad pmf\,2, \quad if \quad 1 < \triangle \leq 55 \\ using \quad pmf\,3, \quad otherwise \end{cases} \tag{3.3}$$

3.2 Further Manipulations on the Refined Errors

In order that a further coding gain on the actual bit rates can be obtained, the following techniques are applied to process the refined errors prior to they are entropy encoded.

3.2.1 Error Sign Flipping

The concept of error sign flipping in [14], [15], [19] is also used in this paper for coding the refined error signal ε. Since e_p, the amount for prediction error compensation, is an average result of past history or experiences, it is very likely that the refined error ε has the same sign as e_p. Therefore, the refined error ε is encoded according to the following equation,

$$\begin{cases} encode\ -\varepsilon, & if\ e_p < 0 \\ encode\ \varepsilon, & otherwise. \end{cases} \tag{3.4}$$

It is noted that all the pixels used in the proposed coding system are causal, the decoder can calculate the error estimate e_p just like the encoder. Therefore, the decoder can reconstruct the sign flipped errors successfully when the image is to be decompressed.

3.2.2 Error Remapping

The range of the refined error ε is $[-255, 255]$. In general, a probability model with a set of 511 symbols should be used for the entropy coding of prediction errors. However, they can only take on values in the range $[-x_{cpd}, 255 - x_{cpd}]$ [14], [15], [19]. Therefore, we can use the following error remapping before it is entropy encoded so that the symbols to be encoded will fall in the range

of $[-128, 127]$, i.e., a set with 256 symbols.

$$\begin{cases} \text{Encode } \varepsilon \text{ by } (\varepsilon + 256), & if \quad \varepsilon < -128 \\ \text{Encode } \varepsilon \text{ by } (\varepsilon - 256), & if \quad \varepsilon \geq 128 \end{cases} \tag{3.5}$$

In this case, the number of symbols used is reduced to 256, which further improves the coding gain in the entropy coding stage. Since all the pixels used are causal and the decoder performs prediction and compensation just like the encoder, the predicted value x_p, the error estimate e_p and thus the compensated prediction, i.e., $x_{cpd} = x_p + e_p$, can be calculated in the decoder. Therefore, the decoder can reconstruct the remapped ε by

$$\begin{cases} \text{Reconstruct } \varepsilon \text{ by } (\varepsilon + 256), & if \quad \varepsilon < -x_{cpd} \\ \text{Reconstruct } \varepsilon \text{ by } (\varepsilon - 256), & if \quad \varepsilon > (255 - x_{cpd}). \end{cases} \tag{3.6}$$

3.2.3 Histogram Tail Truncation

With the quantization bins in (3.3), the error histogram in each quantization bin for the image "Lennagrey" (Fig. 2.6) are plotted in Fig. 3.1. The curve of Bin3 is not shown because no error strength estimate falls in that region. As can be seen, most of the refined errors fall in the region around 0. Though seldom occur, the count of occurrence for those away from 0 are initialized to be 1 in case they do occur. This operation degrades the performance of entropy coding. To conquer this problem, we use the concept of *histogram tail truncation* in [14].

The probability distribution of the prediction error is usually a two-sided Laplacian distribution [14], [19]. Instead of remapping the error into a one-sided monotonically decreasing probability distribution in [14], the error histogram tail are truncated symmetrically in each quantization bin. The cut off region for quantization bins are chosen to be $[-25, 25]$, $[-48, 48]$ and $[-128, 127]$ such that over 99% of the refined errors in each quantization bin are within the truncated regions. With histogram tail truncation, an error

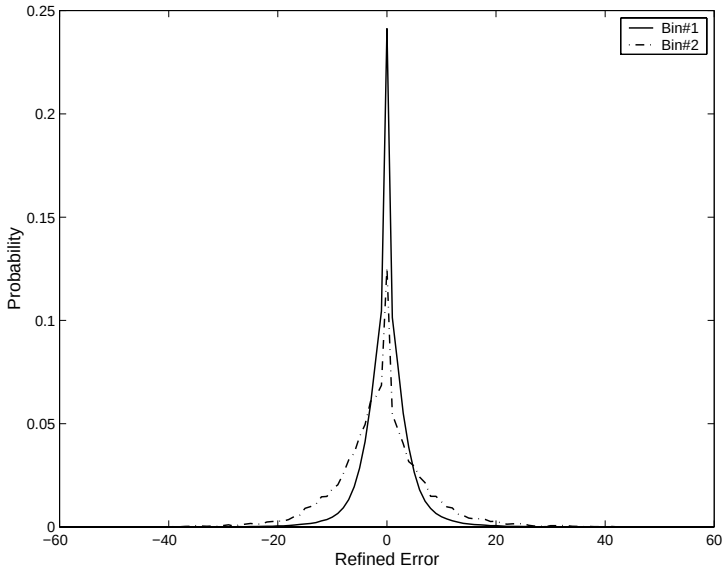


Figure 3.1: Histogram of errors in quantization bins for image "Lennagrey". (using a sixth-order LS-based predictor with $\gamma_1 = 100$, $\gamma_2 = 10$)

30 in Bin1, for example, is encoded first to be 25 using $pmf\,1$, followed by 5 using $pmf\,2$.

3.3 Concluding Remarks

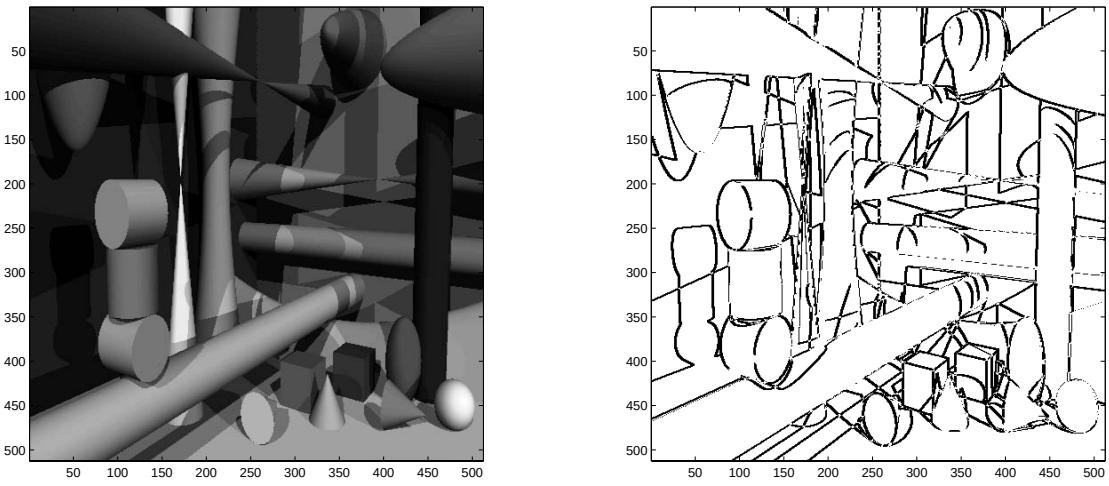
In this chapter, we have introduced the entropy coding stage of the proposed system. In order that a lower actual bit rates can be obtained, we use a conditional arithmetic coder in the proposed system. To do this, we define an error strength estimate $\Delta = e_p$, and then we find an approximately optimal quantization for Δ by using dynamic programming off-line. In this dissertation, the number of quantization bins for Δ is found to be three, and the approximately optimal quantization is shown in (3.3). By conditioning on the error strength estimate Δ, the refined error is then entropy encoded using one out of the three probability models as in (3.3). Besides, the entropy decoder also knows which probability model should be used if the received bit stream is to be decoded.

In addition to the conditional arithmetic corder, the concept of "error sign flipping", "error remapping", and "histogram tail truncation" are also applied in the proposed system before the refined error is entropy encoded so that a further coding gain on actual bit rates can be obtained. We would like to mention again that all the pixels used in the encoder are causal. Therefore, the decoder can reconstruct the remapped and sign flipped error just like the encoder without any side information. The proposed entropy coder determines the coding performance of actual bit rates, and the usefulness of the proposed entropy coding stage can be found in Chapter 4, where comparisons with existing state-of-the-art lossless image coders will be given.

Chapter 4

Experiments

In this chapter, we evaluate the performance of the proposed lossless image codec. Extensive experiments as well as comparisons to existing state-of-the-art predictors and coders will be given to demonstrate the usefulness of the proposed system. All the test images used in the experiments are from TMW [20]. We will first demonstrate the usefulness of the proposed edge detector and then the error compensation mechanism in the regular mode. After that, we will demonstrate the usefulness of the proposed edge-look-ahead approach in the regular mode. Then, the entropy and bit rate performance of the system is presented. Finally, the computational complexity of the proposed system will be discussed.

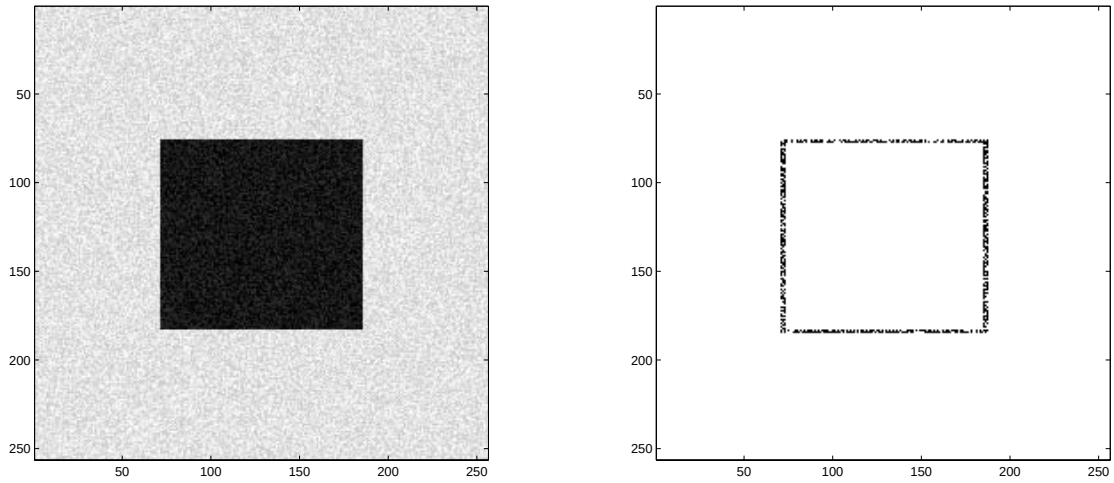


(a) (b)

Figure 4.1: (a) The image "Shapes". (b) Pixels for which (2.4) is satisfied in the image "Shapes".

4.1 The Edge Detector

To demonstrate the effectiveness of the proposed edge detector, we use the image "Shapes" (Fig. 4.1(a)), an artificial image with many edges and lines. The pixels that satisfy the two conditions in (2.4) are marked in Fig. 4.1(b). We can see from Fig. 4.1(b) that the edge detector has successfully picked out the pixels around edges. To test the robustness of the detector, we apply the edge detector to the image "Noisesquare" (Fig. 4.2(a)), an image with salt-and-pepper noise. The pixels picked out by the edge detector are as shown in Fig. 4.2(b). We see from Fig. 4.2(b) that the edge detector is robust to moderate salt-and-pepper noise. In addition to artificial images, we also apply the edge detector to "Lennagrey", a natural image that is shown in Fig. 2.6. As can be seen in Fig. 4.3, the pixels around the edges have been picked out successfully.



(a) (b)

Figure 4.2: (a) The image "Noisesquare". (b) Pixels for which (2.4) is satisfied.

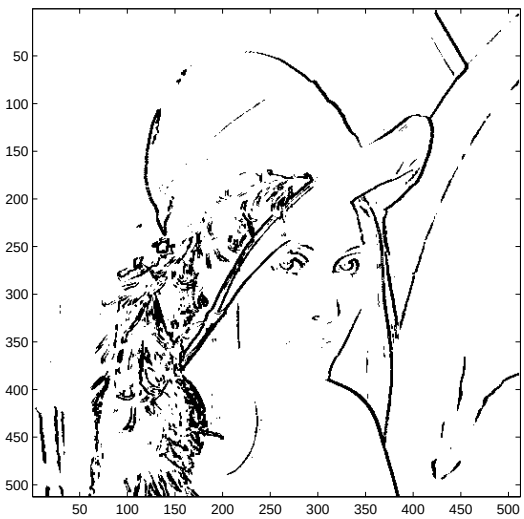


Figure 4.3: Pixels for which (2.4) is satisfied in the image "Lennagrey" ($\gamma_1 = 100$, $\gamma_2 = 10$).

-1	-2	-1
0	0	0
1	2	1

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

Figure 4.4: Masks used in "Sobel" operator. (a) The mask G_x for the computation of vertical derivatives. (b) The mask G_y for the computation of horizontal derivatives.

4.1.1 Comparison with "Sobel" Operator

In the proposed approach, we use raster scan for the encoding of image pixels, i.e., the pixels are encoded in a sequence from left to right and top to bottom. In order that the decoder can perform the same edge detection as the encoder, non-causal pixels, i.e., pixels that lies below and to the right of the coding pixel, can not be used because they are not available in the decoder at the moment. Therefore, we have indicated in section 2.3 that conventional edge detectors, e.g., "Sobel" operator, can not be applied here because they use non-causal pixels. That is also why we have to propose an edge detector that uses only causal pixels in this paper.

It is noted that the "Sobel" operator defines two masks G_x and G_y (as in Fig. 4.4) for the computation of vertical and horizontal derivatives respectively. Moreover, the gradient of the pixel is define by,

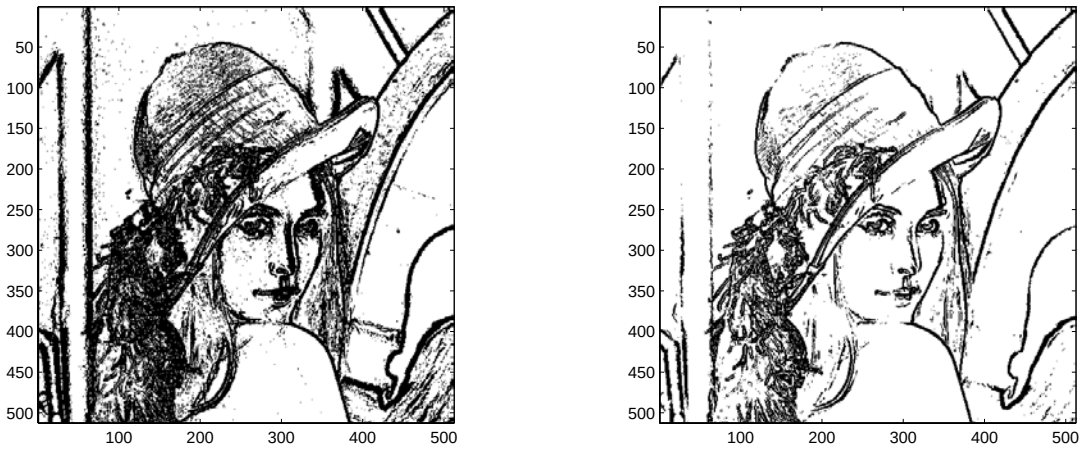
$$\nabla f = |G_x| + |G_y|. \tag{4.1}$$

A pixel is detected as around an edge in "Sobel" operator if the following equation is satisfied.

$$\nabla f \geq threshold. \tag{4.2}$$

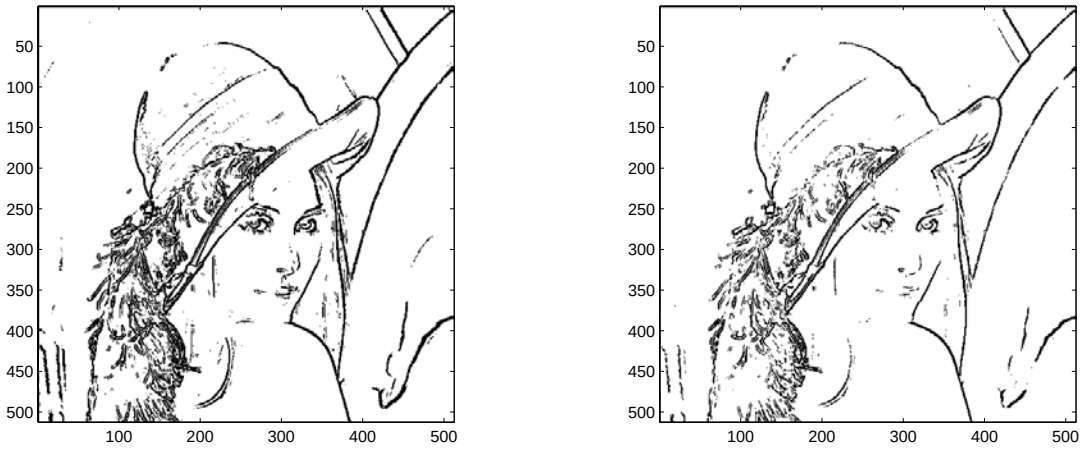
As can be seen in (4.2), we have to define a gradient threshold for the edge detection when using "Sobel" operator. To compare the effectiveness of the

proposed edge detector with "Sobel" operator, we use the image "Lennagrey" as the test image. We show in Fig. 4.5 to Fig. 4.6 the image of pixels that is detected as around an edge using "Sobel" operator with threshold varies from 50 to 200 respectively. The image of pixels that is detected as around an edge by using the proposed edge detector is shown in Fig. 4.3. As can be seen in Fig. 4.6(b) and Fig. 4.3, the image of pixels that is detected as around an edge using the proposed edge detector is very similar to that of obtained by using the non-causal "Sobel" operator. The proposed edge detector is very effective in detecting edges although only four causal pixels are used. The Matlab program for the "Sobel" operator in this experiment is given in Appendix B.



(a) (b)

Figure 4.5: Pixels detected as around an edge in image "Lennagrey" when the Sobel operator is used. (a) threshold= 50. (b) threshold= 100.



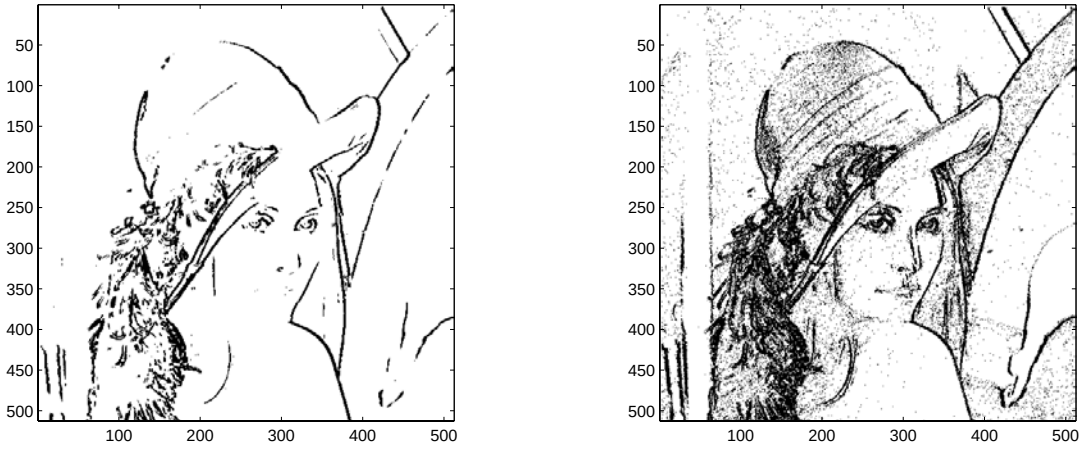
(a) (b)

Figure 4.6: Pixels detected as around an edge in image "Lennagrey" when the Sobel operator is used. (a) threshold= 150. (b) threshold= 200.

4.1.2 Texture context for Edge Detection

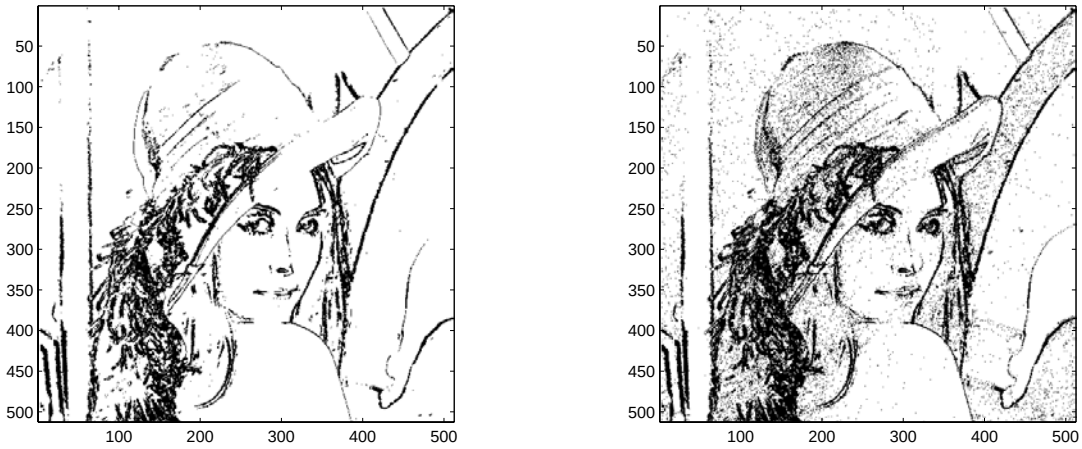
In this experiment, we will show that a larger context for the detection of an edge is not necessary. For this, we construct four sixth-order LS based predictors each with an edge detectors that uses 4, 6 ,8 and 10 pixels respectively. The pixels used in the four constructed edge detectors are selected in an order as defined in Fig. 2.2. Again, the image "Lennagrey" is used for the experiment. To show that the proposed 4-point edge detector is sufficient for the detection of an edge, we compare the improvement on the prediction result and the complexity increased when the number of pixels used for the edge detector varies from 4 to 10. In this experiment, the run mode is disabled; only the regular mode is used. Moreover, the LS adaptation process is activated when an edge is detected or when the prediction error is greater than a predefined threshold.

We show in Fig. 4.7.(a) to Fig. 4.10.(a) the image of pixels that is detected as around an edge by using the four constructed edge detectors. Besides, the corresponding pixels for which the LS adaptation is activated are shown in Fig. 4.7.(b) to Fig. 4.10.(b) respectively. Moreover, we show in Table 4.1 the first-order entropies of the uncompensated prediction errors and the percentage of pixels that activates the LS adaptation when different context is used for edge detection. The execution times of the proposed edge-look-ahead predictor with different context edge detection are also recorded in the last column of Table 4.1. As can be seen in Table. 4.1, the percentage of pixels that activates the LS adaptation process by using the proposed 4-point edge detector is 17.90% and that of obtained by the use of a 10-point edge detector is 21.13%. There is an increase of about 3.2% in performing LS adaptation, which means an increased computational complexity. However, the prediction results (in terms of first-order entropy) almost remain unchanged as can be seen in Table 4.1; only marginal improvement can be obtained. Therefore, the use of a larger context for the detection of an edge is not necessary and the proposed 4-point context is sufficient for the edge detection process.



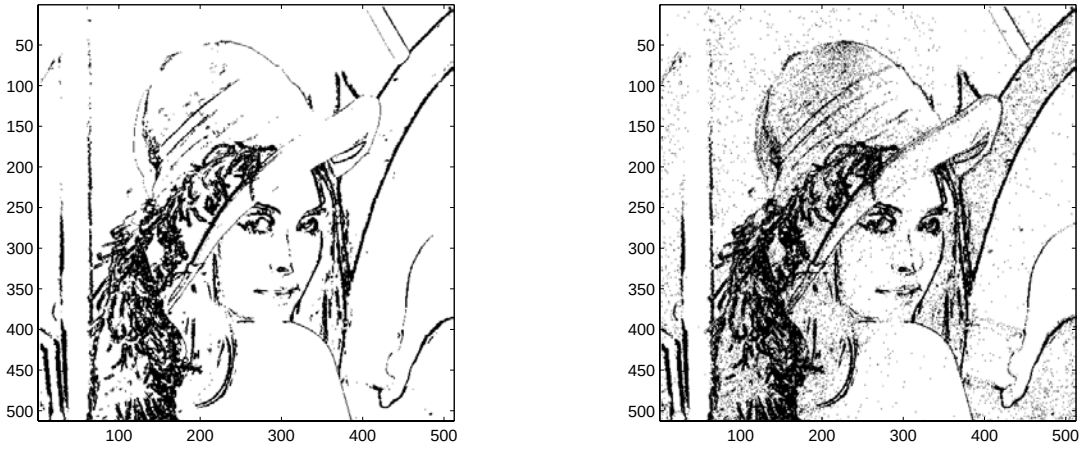
(a) (b)

Figure 4.7: 4-point edge detector for image "Lennagrey". (a) Pixels detected as around an edge. (b) Pixels for which LS adaptation is used.



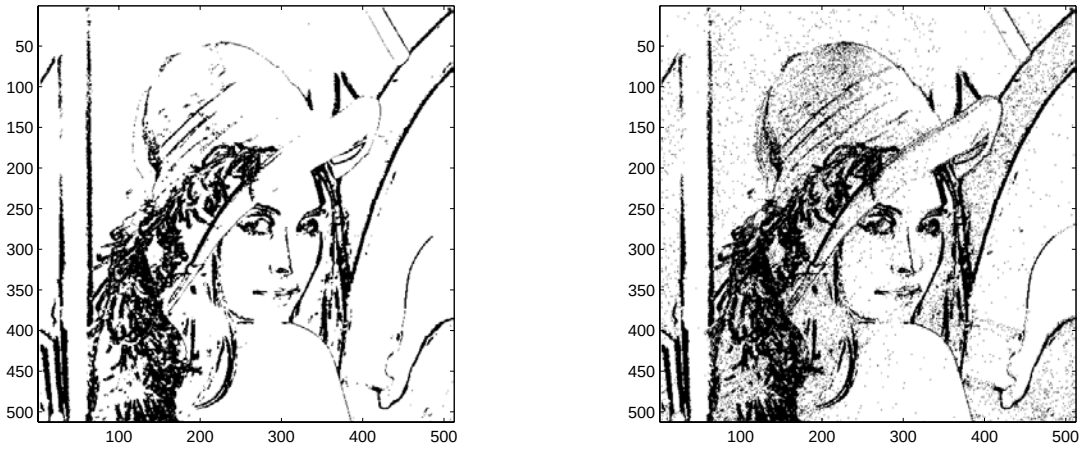
(a) (b)

Figure 4.8: 6-point edge detector for image "Lennagrey". (a) Pixels detected as around an edge. (b) Pixels for which LS adaptation is used.



(a) (b)

Figure 4.9: 8-point edge detector for image "Lennagrey". (a) Pixels detected as around an edge. (b) Pixels for which LS adaptation is used.



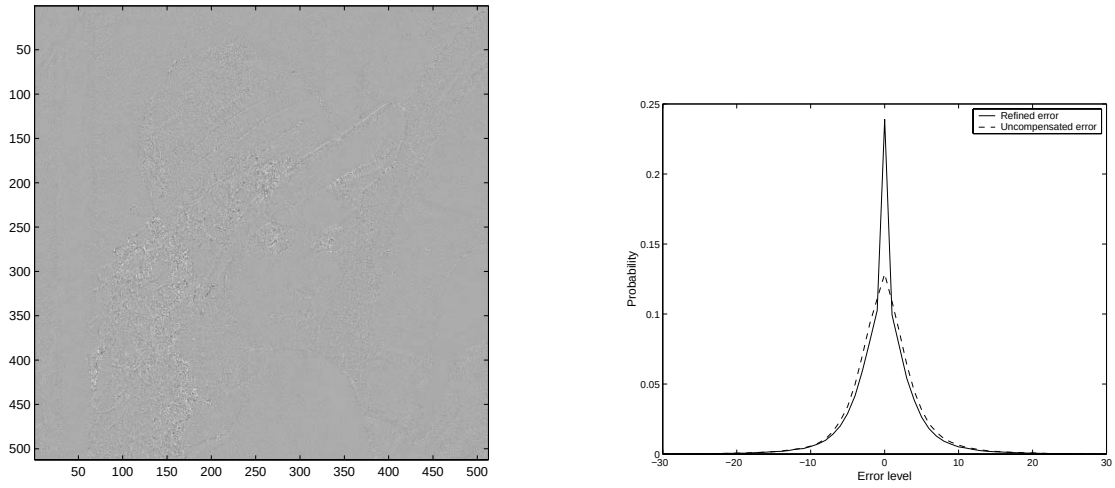
(a) (b)

Figure 4.10: 100-point edge detector for image "Lennagrey". (a) Pixels detected as around an edge. (b) Pixels for which LS adaptation is used.

Table 4.1: First-order entropies of uncompensated prediction errors and percentage of pixels that activate the LS adaptation when different context is used for the edge detection. (Run on a P4-1.4GHz machine; without doing error compensation and entropy coding).

Number of pixels used in edge detector	% of LS adaptation	First-order Entropy	Execution Time (in seconds)
4	17.90	4.1975	3.39
6	18.25	4.1971	3.53
8	18.67	4.1969	3.55
10	21.13	4.1941	3.72

4.2 Error compensation



(a) (b)

Figure 4.11: (a) Image of refined errors using the proposed approach for "Lennagrey". (using a sixth-order LS-based predictor with $\gamma_1 = 100$, $\gamma_2 = 10$) (b) Histogram of prediction errors for image "Lennagrey".

To demonstrate the usefulness of the proposed error compensation mechanism in regular mode, we use a sixth-order LS-based predictor. The image "Lennagrey" (Fig. 2.6) is used as the test image. The compensated prediction errors for image "Lennagrey" is shown in Fig. 4.11(a). As can be seen in Fig. 4.11(a), the proposed approach performs very well around edges and the statistical redundancy has been removed effectively. Fig. 4.11(b) shows the histograms of prediction error with and without error compensation. We can observe the usefulness of the proposed automatic context modeling for error refinement from Fig. 4.11(b). The histogram of the refined error has a narrower peak. The first-order entropy for the compensated error and the uncompensated error is 3.97 bits and 4.20 bits respectively.

In addition to the natural image "Lennagrey", the image "Airplane"

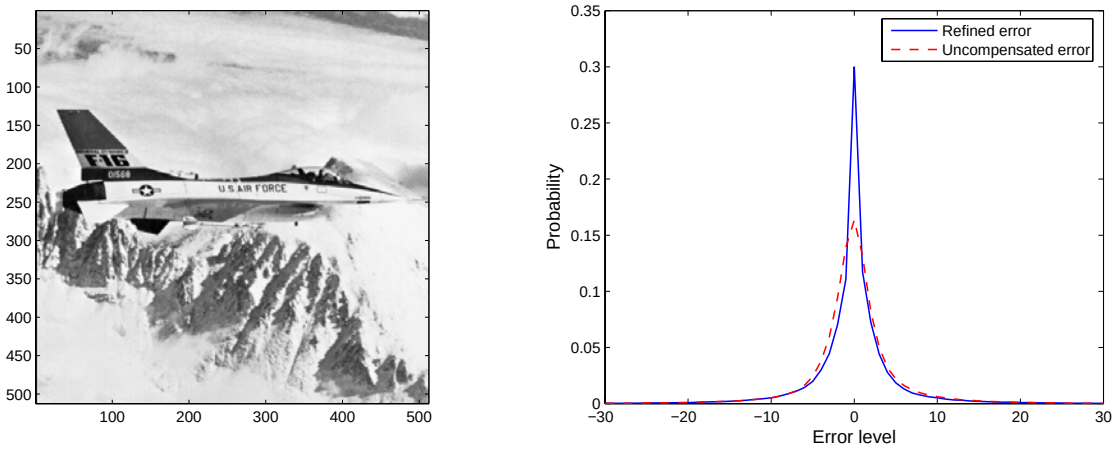
(Fig. 4.12.(a)), "Goldhill" (Fig. 4.14.(a)) and "Peppers" (Fig. 4.16.(a)) are also used to demonstrate the usefulness of the proposed error compensation mechanism in regular mode. It is noted that all the results are obtained by using a sixth-order predictor with $\gamma_1 = 100$, $\gamma_2 = 10$ and $\theta = 8$.

For the image "Airplane", we show in Fig. 4.13 the image of prediction errors with and without doing error compensation. As can be seen in Fig. 4.13.(b), the proposed approach performs very well around edges and the statistical redundancy has been removed effectively. The usefulness of the proposed error compensation mechanism can be best observed from Fig. 4.12.(b), in which the histogram of prediction errors with and without error compensation are shown. The refined error has a much narrower peak than that without error compensation. In Fig. 4.12.(b), the first-order entropy for the compensated error is 3.82 bits and 4.11 bits for the uncompensated error.

As another example, we use the image "Goldhill" in Fig. 4.14.(a). The image of prediction errors with and without error compensation are shown in Fig. 4.15. As can be seen in Fig. 4.15.(b), the proposed error compensation mechanism performs very well around edges. For comparison purpose, we also show in Fig. 4.14.(b) the histogram of prediction errors. The histogram of refined errors is much narrower than that without doing error compensation. In Fig. 4.14.(b), the entropies corresponding to the two histograms are respectively 4.40 bits (Refined errors) and 4.60 bits (Uncompensated errors).

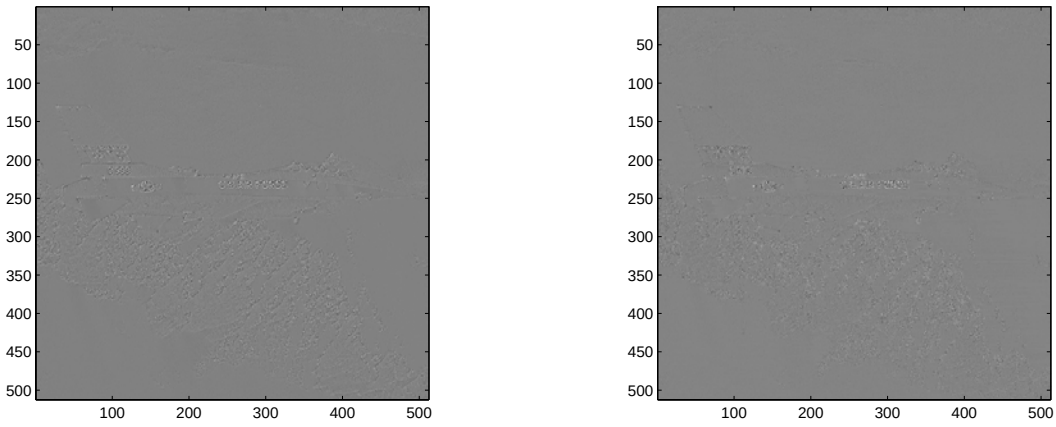
Moreover, we use the image "Peppers" in Fig. 4.16.(a) to demonstrate the effectiveness of the proposed error compensation mechanism. The images of prediction errors with and without doing error compensation are shown in Fig. 4.17.(a) and (b) respectively. As can be seen in Fig. 4.17.(b), the proposed approach performs very well for pixels around edges. Again, we also

shown in Fig. 4.16.(b) the histogram of prediction errors with and without doing error compensation. Obviously, the histogram of refined errors is much narrower than that without doing error compensation. In Fig. 4.16.(b), the first-order entropy are respectively 4.27 bits (Refined errors) and 4.45 bits (Uncompensated errors).



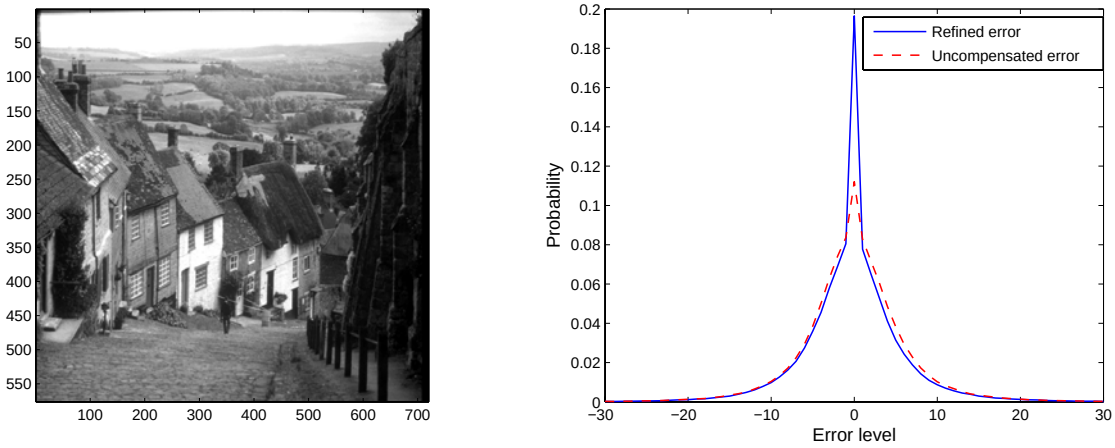
(a) (b)

Figure 4.12: (a) The image "Airplane". (b) Histogram of prediction errors for image "Airplane".



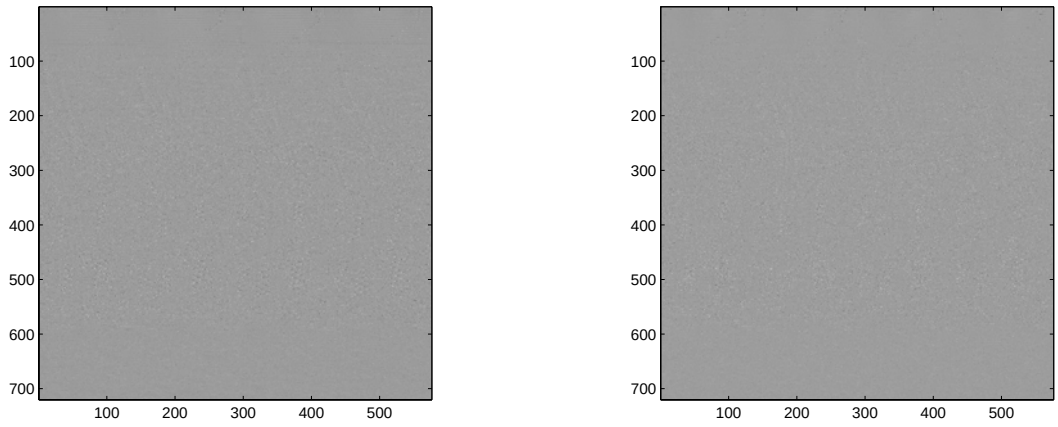
(a) (b)

Figure 4.13: (a) The image of uncompensated errors for "Airplane". (b) Image of refined errors for "Airplane".



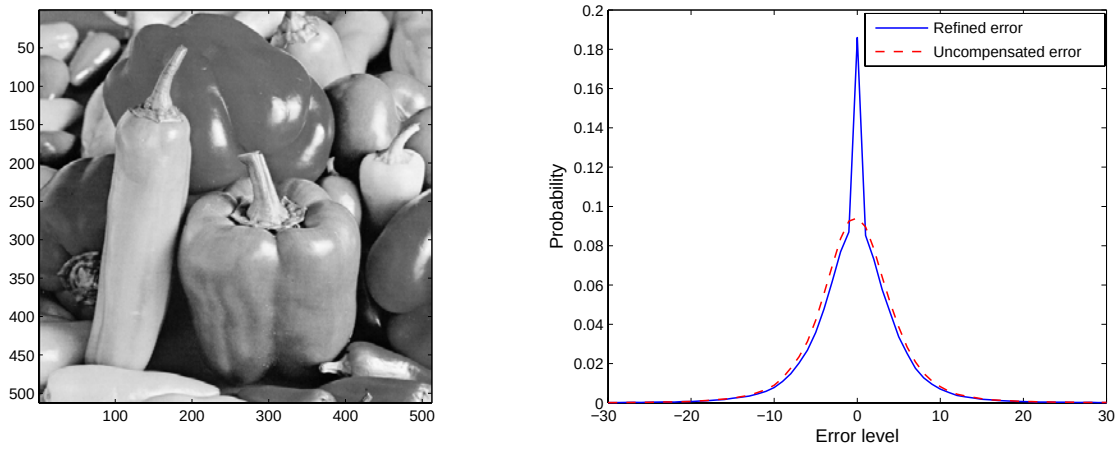
(a) (b)

Figure 4.14: (a) The image "Goldhill". (b) Histogram of prediction errors for image "Goldhill".



(a) (b)

Figure 4.15: (a) The image of uncompensated errors for "Goldhill". (b) Image of refined errors for "Goldhill".



(a) (b)

Figure 4.16: (a) The image "Peppers". (b) Histogram of prediction errors for image "Peppers".

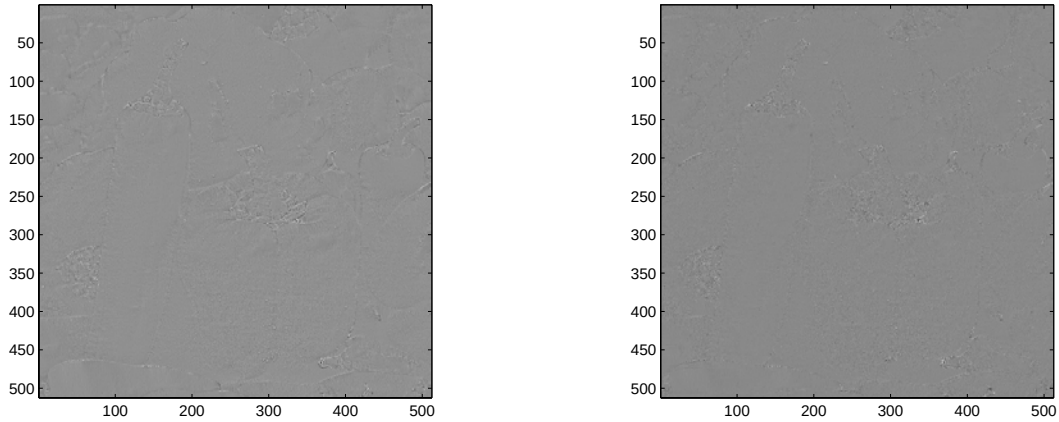


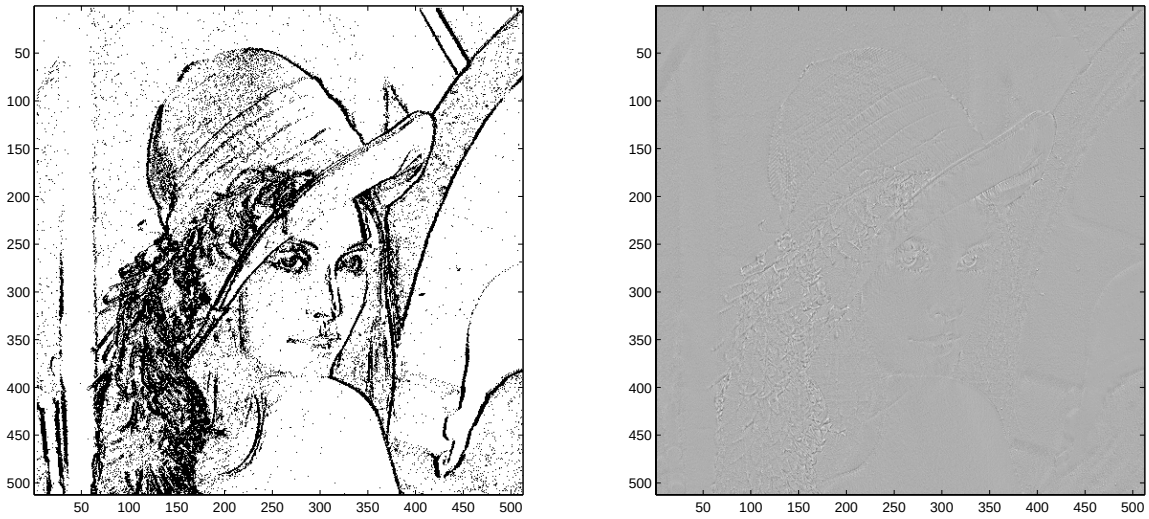
Figure 4.17: (a) The image of uncompensated errors for "Peppers". (b) Image of refined errors for "Peppers".

Table 4.2: Compression ratio and the running time (in seconds, on a P3-1.06GHz machine) of the constructed coder vary with different prediction order using the proposed approach.

Image	N=4		N=6		N=8		N=10	
	Compression Ratio	Run Time	Compression Ratio	Run Time	Compression Ratio	Run Time	Compression Ratio	Run Time
Baboon	1.35	3.82	1.38	6.80	1.38	12.37	1.38	14.21
Lena	1.82	3.09	1.84	4.22	1.84	5.71	1.84	6.95
Lennagrey	2.01	3.04	2.03	3.79	2.03	5.18	2.03	5.84
Peppers	1.86	3.03	1.88	3.86	1.88	4.59	1.89	6.06
Barb	1.88	3.38	1.95	4.80	1.96	9.07	1.97	8.62
Barb2	1.73	5.41	1.77	8.28	1.77	14.33	1.77	15.26
Boats	2.11	4.82	2.15	6.02	2.16	8.43	2.17	9.18
Gold Hill	1.82	4.76	1.83	6.58	1.84	8.30	1.84	10.97
Average	1.82	3.92	1.85	5.54	1.86	8.50	1.86	9.64

4.3 Order of predictor

The order of the predictor affects coding gain in the regular mode. We list in Table 4.2 the compression ratio for order $N = 4, 6, 8, 10$ with the run mode disabled. The execution time is also listed in the Table. As can be seen in Table 4.2, the compression ratio quickly saturates when the prediction order is greater than six. Moreover, the increases in the execution time does not justify the use of a predictor with order higher than six. Therefore, the use of a sixth-order predictor in the regular mode is a proper choice and this will be used in the design of a lossless image coder afterward for comparison with existing state-of-the-art lossless image coders.



(a) (b)

Figure 4.18: (a) Pixels for which LS adaption is used in the proposed edge-look-ahead predictor for the image "Lennagrey". (using a sixth-order predictor with $\gamma_1 = 100$, $\gamma_2 = 10$) (b) Image of uncompensated prediction errors using the proposed edge-look-ahead approach for "Lennagrey".

4.4 Effectiveness of the Edge-look-ahead mechanism

The usefulness of the proposed predictor with edge-look-ahead can be demonstrated through the following experiment. We construct two sixth-order LS based predictors for the regular mode; one with the use of the proposed edge-look-ahead mechanism and the other performs LS adaptation in a pixel-by-pixel manner. Then we compare the performance of the two predictors. In this experiment, the run mode is also enabled and the image "Lennagrey" in Fig. 2.6 is used for this comparison.

For the predictor with edge-look-ahead, the pixels for which LS adaption is activated are shown in Fig. 4.18(a). Overall, about 17% of pixels activate

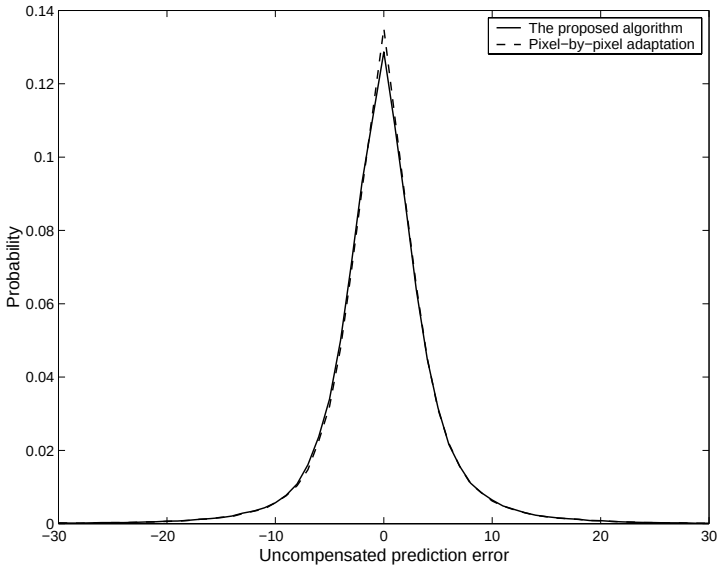


Figure 4.19: Histogram of uncompensated prediction errors for the proposed approach and that of a pixel-by-pixel adaptation. (both using a sixth-order predictor)

the LS adaptation process. The image of uncompensated prediction errors $(x_n - x_p)$ and the corresponding histogram are shown in Fig. 4.18(b) and Fig. 4.19 respectively. For comparison, we also show in Fig. 4.19 the histogram of uncompensated prediction error when the LS adaptation process is performed in a pixel-by-pixel manner. The histogram using the proposed approach is very close to that with pixel-by-pixel adaptation although only 17% of pixels activate the LS adaptation process. The proposed edge-look-ahead approach has made a good tradeoff between prediction efficiency and computational complexity. Indeed, the entropies corresponding to the two histograms in Fig. 4.19 are respectively 4.20 bits (proposed approach) and 4.18 bits (adapted in a pixel-by-pixel manner).

Table 4.3: First-order entropies of prediction errors. (Only the regular mode is used in the proposed algorithm; the run mode is disabled.)

Image	MED	GAP	EDP				Proposed Algorithm				Pixel by Pixel Optimization			
			N=4	N=6	N=8	N=10	N=4	N=6	N=8	N=10	N=4	N=6	N=8	N=10
Baboon	6.28	6.22	6.04	6.01	6.00	5.99	6.03	5.99	5.98	5.98	6.03	5.99	5.98	5.98
Lena	4.90	4.75	4.64	4.60	4.59	4.58	4.58	4.53	4.53	4.51	4.58	4.53	4.53	4.51
Lennagrey	4.56	4.40	4.32	4.26	4.24	4.22	4.24	4.20	4.19	4.15	4.22	4.18	4.17	4.15
Peppers	4.95	4.78	4.55	4.52	4.51	4.50	4.48	4.45	4.44	4.43	4.47	4.43	4.43	4.43
Barb	5.21	5.15	4.67	4.44	4.40	4.35	4.52	4.36	4.30	4.25	4.46	4.31	4.26	4.21
Barb2	5.19	5.06	4.93	4.80	4.79	4.78	4.90	4.77	4.75	4.75	4.88	4.75	4.74	4.74
Boats	4.31	4.29	4.20	4.14	4.12	4.10	4.16	4.10	4.07	4.05	4.07	4.00	3.97	3.96
Gold Hill	4.72	4.70	4.64	4.60	4.59	4.58	4.64	4.60	4.59	4.59	4.63	4.58	4.57	4.57
X-ray Neck	N/A	N/A	N/A	N/A	N/A	N/A	3.21	3.17	3.16	3.16	3.19	3.16	3.15	3.15

4.5 Comparisons to existing predictors

Table 4.3 gives comparisons of uncompensated prediction errors for a set of eight test images in first-order entropies. We compare with existing linear and nonlinear predictors for prediction order $N = 4, 6, 8, 10$. In this experiment, the run mode of the proposed system is disabled; only the regular mode is used so that we can make a fair comparison. The results of a median edge detector (MED) [15], a gradient adjusted predictor (GAP) [14] and an edge directed predictor (EDP) with different orders are taken from [26]. As can be seen in Table 4.3, the proposed system can remove the statistical redundancy efficiently. It achieves noticeable improvement when compared with MED and GAP predictor. The proposed predictor also gives lower entropies when compared with those of EDP [26]. Moreover, the results of the proposed approach are very close to those with pixel-by-pixel LS adaptation.

In addition to the eight natural images, we also apply the proposed approach to a medical image "Neck" taken by X-ray (Fig. 4.20). The first-order

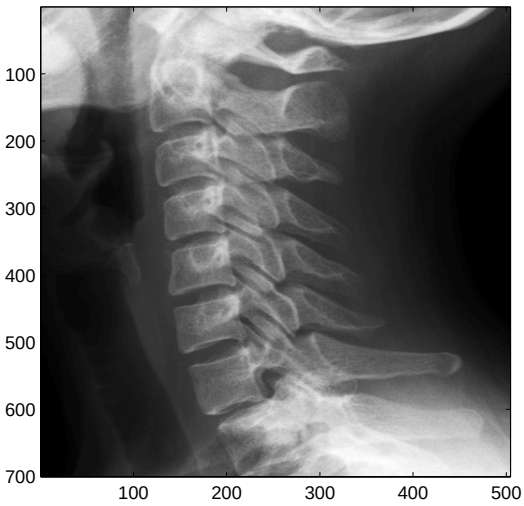
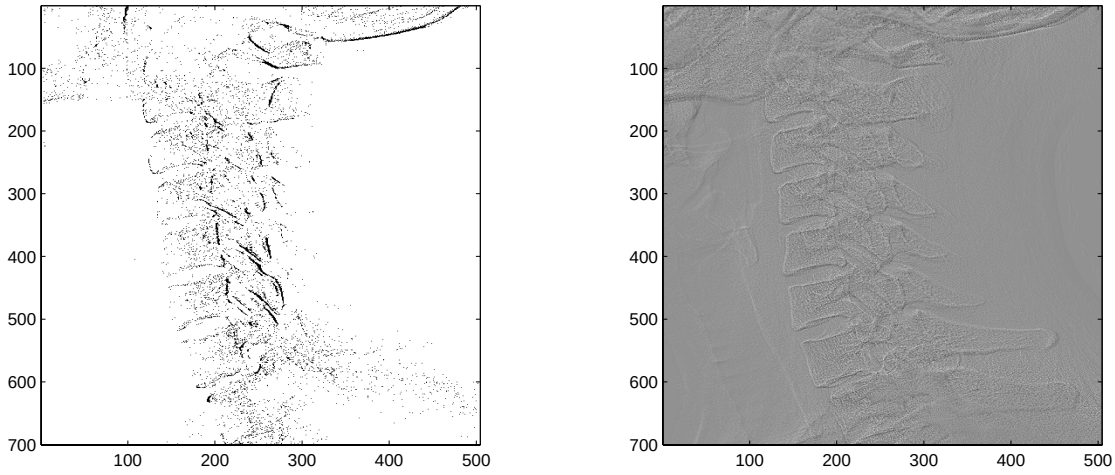


Figure 4.20: The X-ray image "Neck".

entropies of the image "Neck" obtained by using the proposed approach with different prediction order and that of obtained by using pixel-by-pixel LS adaptation are list in the last row of Table 4.3. Besides, pixels for which LS adaptation is used for the image "Neck", and the image of uncompensated prediction error obtained with the proposed approach by using a sixth-order predictor are shown in Fig. 4.21(a) and 4.21(b) respectively. As can be seen in Fig. 4.21(b), the proposed approach performs very well for the medical image "Neck" around boundaries with only about 2.7% of pixels activate the LS adaptation process.



(a) (b)

Figure 4.21: (a) Pixels for which the LS adaptation is used in the proposed edge-look-ahead predictor for the X-ray image "Neck". (using a sixth-order predictor) (b) Image of uncompensated prediction error for the X-ray image "Neck".

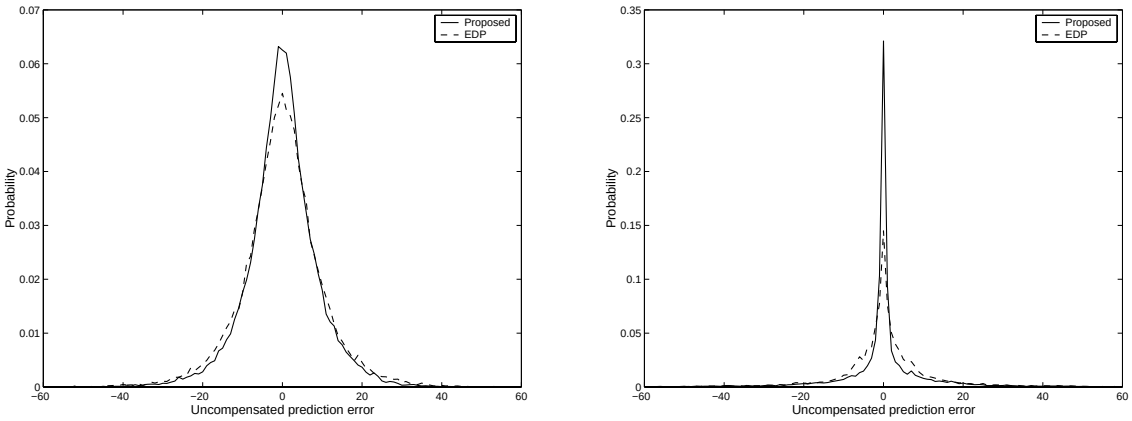
Table 4.4: Percentage of pixels performing LS adaption and the resulting first-order entropy by varying the variance threshold γ_1 in the proposed approach (The image "Lennagrey" is used for the test with $\gamma_2 = 10$, $\theta = 10$ for all cases).

Parameter	Number of Pixels with LS Adaptation	Percentage with LS Adaptation (A)	Percentage Detected as around an Edge (B)	Percentage with LS Adaptation in slowly varying areas (C=A-B)	First-order Entropy
γ_1=100	37692	14.38	11.47	2.91	4.176
γ_1=200	28205	10.76	6.87	3.89	4.184
γ_1=300	24117	9.20	4.79	4.41	4.188
γ_1=400	21819	8.32	3.58	4.75	4.190
γ_1=500	20342	7.76	2.75	5.01	4.193

4.6 LS adaptation

In this section, we investigate how the prediction performance (in terms of entropy) varies with the variance threshold γ_1 in (2.4) for LS adaptation. We construct a tenth-order predictor with edge-look-ahead for the experiment, and the image "Lennagrey" (Fig. 2.6) is used for the test. For LS adaptation, we use the same training area as defined in EDP [26]. Moreover, we set $\gamma_2 = 10$ and $\theta = 10$ for all cases in the experiment. By varying the variance threshold γ_1, the number of pixels that activates LS adaptation also changes. The experimental results using the proposed approach with various γ_1 are shown in Table 4.4.

We observe from Table 4.4 that a small γ_1 may results in a small entropy at the expense of an increased number of pixels performing LS adaptation process. The percentage of pixels regarded as around an edge increases as γ_1 decreases, but the percentage of pixels that activates the LS adaptation in slowly varying areas almost remain unchanged. This can be best observed from the fifth column, which is obtained by subtracting the fourth column from the third column, of Table 4.4, and we can find that about 3% to 5% of pixels in slowly varying areas will activates the LS adaptation. Therefore, the improvement on the entropy in the proposed approach is mainly around edges.



(a) (b)

Figure 4.22: (a) Histogram of uncompensated prediction errors for the pixels in Fig. 4.3. (b) Histogram of uncompensated prediction errors for those pixels in Fig. 4.1(b).

4.7 Further insight on the proposed edge-look-ahead mechanism

To highlight the usefulness of the proposed edge-look-ahead mechanism for LS-based predictor, we compare the proposed approach with the state-of-the-art LS-based EDP predictor in [26]. As in [26], we use a tenth-order predictor. With EDP [26], the percentage of pixels activating LS adaptation is 9.87% and the resulting entropy is 4.22bpp. The results of the proposed edge-look-ahead approach are given in Table 4.4. As can be seen in the rows for which γ_1 varies from 300 to 500 (the last three rows of Table 4.4), the number of pixels performing LS adaptation is fewer than that with EDP approach but still have lower entropy than EDP.

To gain a further insight, we look into the case with $\gamma_1 = 100$. The histogram of uncompensated prediction errors for those pixels that are considered as around an edge (i.e., pixels for which (2.4) are satisfied) is shown

in Fig. 4.22(a). For comparison, we also show the histogram of uncompensated prediction error if EDP is used for those pixels instead. As can be seen in Fig. 4.22(a), the histogram with the proposed approach is much narrower than that with EDP. The proposed approach has a smaller prediction error than EDP does around edges. Indeed, the entropies corresponding to the two histograms in Fig. 4.22(a) are respectively 5.12 bits (proposed) and 5.32 bits (EDP).

As another example, we use the artificial image "Shapes" (Fig. 4.1(a)) for the same experiment. The pixels for which (2.4) is satisfied, i.e., pixels detected as around an edge, have been shown in Fig. 4.1(b). The histogram of uncompensated prediction errors using the proposed approach for those pixels in Fig. 4.1(b) is shown in Fig. 4.22(b). The histogram with the proposed approach is much narrower than that with EDP; the proposed approach has a smaller prediction error than EDP does around edges. The entropies corresponding to the two histograms in Fig. 4.22(b) are respectively 4.51 bits (proposed) and 5.28 bits (EDP).

As can be seen in Fig. 4.22(a) and Fig. 4.22(b), the proposed system has achieved a noticeable improvement over the use of EDP around edges. Moreover, the improvement is very distinct for images with many edges and lines. As indicated in the title of EDP [26], the EDP is designed mainly for natural images and it is noted that most of the areas vary slowly in natural images. The EDP initiates the LS adaptation process only after the prediction error is beyond a preselected threshold so that the computational complexity can be reduced. Nevertheless, an abrupt change in the image pixel, e.g., an edge or a line, may results in a large prediction error with EDP. Compared with EDP, the proposed system can look ahead to determine if the coding pixel is around an edge and initiate the LS adaptation process

beforehand to prevent the occurrence of a large prediction error. With a moderately increased computational complexity in detecting the existence of an edge, the proposed system has achieved a noticeable improvement around edges than that with the EDP approach.

Table 4.5: Comparisons with existing lossless image coders (in bits/sample). The fifth column is the execution time of the proposed approach (on a P3-1.06GHz machine).

Image	First-oreder Entropy	RALP	% of LS adaptation	seconds	JPEG-LS [15]	CALIC [14]	EDP* [26]	TMW [20]
Airplane	3.82	3.71	16.6	3.75	3.82	3.74	N/A	3.60
Baboon	5.91	5.81	63.8	6.82	6.04	5.88	5.81	5.73
Balloon	2.56	2.55	4.0	4.50	2.90	2.83	N/A	2.66
Barb	4.15	4.12	33.6	4.80	4.69	4.32	4.11	4.09
Barb2	4.57	4.51	37.7	8.02	4.69	4.53	4.52	4.38
Boats	3.84	3.75	18.1	6.07	3.93	3.83	3.80	3.61
Camera	4.42	4.24	26.7	1.18	4.31	4.19	N/A	4.10
Couple	3.63	3.63	15.8	1.01	3.70	3.61	N/A	3.45
Gold Hill	4.41	4.32	23.6	6.81	4.48	4.39	4.39	4.27
Lena	4.36	4.35	23.5	4.27	4.61	4.48	4.40	4.30
Lennagrey	3.97	3.95	17.4	3.82	4.24	4.11	4.02	3.91
Noisesquare	5.35	5.37	56.7	1.60	5.68	5.44	N/A	5.54
Peppers	4.27	4.27	18.2	3.97	4.51	4.42	4.35	4.25
Shapes	1.87	1.52	7.1	1.94	1.21	1.14	N/A	0.76
X-ray Neck	2.87	2.82	2.7	3.23	N/A	N/A	N/A	N/A
Average	4.08	4.01	25.92	4.18	4.20	4.07	4.43	3.90

(* For the EDP method, the bit rates for some of the images are not available and the average is computed only for those that are available. The X-ray image "Neck" is not taken into account for the average values listed in the last row of the table.)

4.8 Comparisons to existing state-of-the-art coders

Table 4.5 gives the actual bit rates of proposed RALP coder, JPEG-LS [15], CALIC [14], EDP [26] and TMW [20] for a set of fourteen test images. In Table 4.5, the results of CALIC, EDP and TMW are taken directly from [21] and those of the JPEG-LS are obtained using the code given in the website of LOCO-I [15]. All the bit rates of the proposed algorithm are obtained using the same parameters described in previous sections, i.e., sixth-order predictor with a four-point texture context edge detector, and no individual optimization is performed. Besides, we show in the second column and the fourth column respectively the first-order entropies of the compensated prediction errors and the percentage of pixels performing LS adaptation using the proposed approach. Moreover, we also show in the fifth column the execution time of the proposed coder so that we can get a picture on the runtime performance of the proposed approach. It should be noted that some of the results in EDP are denoted by "N/A" because they are not reported in the paper of EDP [26] and [21]. Therefore, the set of images included in the average is different for that column than any other. Table 4.5 shows that RALP has lower bit rates than JPEG-LS in thirteen out of the fourteen test images and outperforms CALIC [14] in eleven of fourteen test images. Encouragingly, the proposed RALP achieves lower bit rates than the highly complex TMW in two images, "Balloon" and "Noise square". It should be noted that the proposed bit rate performance for the artificial image "Shapes" is inferior to that of obtained by CALIC [16] and JPEG-LS [17]. For this, we can see from Fig. 4.1.(a) that the image "Shapes", an artificial image with edges and lines, can be segmented into many slowly varying areas in which the run-length coding is most efficient to be used. In the meanwhile, we also

know that the run-length encoding is applied in the coder of JPEG-LS [15] and RALP [28]. Furthermore, the CALIC [14] also uses a "Ternary entropy coder" for the so-called "binary mode" or uniform regions. Therefore, we think the bit rate performance for artificial images, like "Shapes", can be improved if a more sophisticated run-length coder or an extra entropy coder with a small symbol set is applied for slowly varying areas in the proposed approach.

Table 4.6: Percentage of pixels performing LS adaption and the number of pixels performing *Cholesky decomposition* and *Singular Value Decomposition (SVD)*. (Only the regular mode is used; the run mode is disabled.)

Image	Proposed linear predictor with edge-look-ahead											
	N=4			N=6			N=8			N=10		
	%	Cholesky	SVD	%	Cholesky	SVD	%	Cholesky	SVD	%	Cholesky	SVD
Baboon	65.1	170545	0	64.7	169588	0	64.7	169651	0	64.5	169203	0
Lena	24.4	63989	16	23.9	62681	52	23.9	62568	76	23.4	61136	83
Lennagrey	18.2	47757	14	17.9	46856	29	17.9	46754	42	17.4	45607	32
Peppers	19.0	49789	0	18.4	48362	0	18.3	47940	0	18.1	47567	0
Barb	35.8	93825	0	34.8	91254	0	34.5	90552	0	34.3	89946	0
Barb2	39.7	164570	0	38.7	160509	0	38.8	160888	0	38.8	160829	0
Boats	19.3	79992	65	18.8	78030	0	18.5	76863	21	18.3	76036	36
Gold Hill	24.8	102857	1	24.2	100169	0	24.0	99628	0	23.9	99086	0
Average	30.8	96666	12.0	30.2	94681	10.1	30.1	94356	17.4	29.9	93676	18.9

4.9 Computational complexity

It is noted that the construction of normal equations requires roughly $MN^2/2$ multiplications, where M is the number of training pixels and N is the prediction order [35]. Numerically, the normal equations (2.7) can be solved by *Cholesky decomposition* or *Singular Value Decomposition (SVD)* depending on the rank of $\mathbf{P}$ in (2.6). When $\mathbf{P}$ is full-ranked, $\mathbf{P}^T\mathbf{P}$ is nonsingular and positive definite. For a positive definite matrix, the *Cholesky decomposition* can be used, and it requires only $N^3/6$ multiplications to solve (2.7), which is about half the usual number of multiplications required by alternative methods [35], [36]. Only when $\mathbf{P}$ is not full-ranked, *SVD*, which requires much higher computations, is needed. Fortunately, our experiments show that in most cases $\mathbf{P}$ has full rank. This can be seen in Table 4.6, where we have listed the percentage of pixels performing LS adaption and the number of pixels performing *Cholesky Decomposition* or *SVD* for predictors with differ-

Table 4.7: Operation counts for edge detector in (2.4).

Operation	Compare	ADD/SUB	MUL/DIV	Square
Edge detection	≤ (N+2)	≤ (4N-2)	≤ 7	≤ (N+3)
"N" is the number of pixels used in edge detection. In this paper, N= 4.				

ent orders. Indeed, this is because pixels around boundaries usually have a large variation in gray levels and thus the matrix $\mathbf{P}$ in (2.6) is seldom rank deficient. Therefore, most of the computations take place in forming the normal equations (2.7) rather than solving them. For this, an inclusion and exclusion method for fast construction of the $\mathbf{P}^T\mathbf{P}$ matrix is proposed in [25]. The algorithm in [25] utilize the overlapped training area between successive coding pixels. Therefore, the fast algorithm can only be used in a pixel-by-pixel adaptation manner and can not be applied in the proposed approach as we activate the LS adaptation process only when necessary.

For the proposed edge detector, the operation counts for each coding pixel in the edge detection process are listed in Table 4.7. It should be noted that there is no need to check both of the two inequalities in (2.4) for every pixel. Only when the first inequality holds then we check the second condition. Therefore, the actual computational cost is lower than what is listed in Table 4.7. Though edge detection incurs a slight increase in computations, the overall complexity is reduced significantly when compared with that of pixel-by-pixel adaptation approach. The proposed approach has achieved a very good trade-off between runtime performance and prediction efficiency.

4.10 Concluding Remarks

In this chapter, we have evaluated the performance of the proposed lossless image codec. Extensive experiments as well as comparisons to existing state-of-the-art predictors and coders are also given to demonstrate the usefulness of the proposed system. In summary, we can make the following concluding remarks:

1. We have investigated in this chapter the usefulness of the proposed edge detector. As can be seen in our experiments, the proposed edge detector is very effective in detecting edges and robust to images with moderate salt-and-pepper noise although only four causal pixels are used.

2. The effectiveness of the error compensation mechanism in regular mode is also evaluated in this chapter. Our experiments show this very useful that further improves the bit rates by, on average, 0.2bpp in test images.

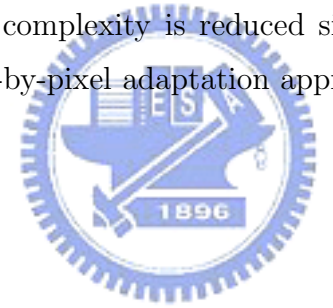
3. The prediction order affects the coding gain. In our experiments, we find the compression ratio quickly saturates when the prediction order is greater than six. Moreover, the use of a higher prediction order also means an increase in computational complexity. Therefore, we have found the use of a sixth-order predictor is a proper choice.

4. With the proposed edge-look-ahead approach, we activate the LS adaptation process only when an edge is detected or when the prediction error is beyond a predefined threshold. As can be seen in our experiments, the results obtained by using the proposed approach are very close to those with pixel-by-pixel LS adaptation, but with a significantly reduced complexity. The proposed edge-look-ahead approach is

more feasible under limited resources.

5. In this chapter, we also compare the proposed system with existing state-of-the-art predictors and coders. As can be seen in Table 4.3 and Table 4.5, comparisons on first-order entropies and actual bit rates have demonstrated the superiority of the proposed system.

6. A detailed analysis on the computational complexity of the proposed system is also given in this chapter. For LS adaptation process, fast algorithms, like *Cholesky decomposition*, can be used in most of the cases depending on the defectiveness of the matrix $\mathbf{P}$ in (2.6). Besides, we also list in Table 4.7 the operation counts of the proposed edge detector. Though edge detection incurs a slight increase in computations, the overall complexity is reduced significantly when compared with that of pixel-by-pixel adaptation approach.



Chapter 5

Enhancing the Predictive Coding Efficiency with Control Technologies

There has been great interest in applying predictive coding to lossless compression of images. Predictive coding is very useful for removing statistical redundancy among pixels in slowly varying areas. However, there can be large prediction errors for pixels around boundaries. In this chapter, we introduce techniques commonly used in control systems to enhance the coding efficiency of predictive coding. Actually, the predictive coding system, which calculates the system output based on the texture context of the coding pixel, behaves just like a multi-input single-output system with the predictor itself can be taken as the system model. Besides, the prediction error is usually feedback for the adaptation of predictor coefficients so that the difference between the desired and the actual output, i.e., the so-called error signal, for consecutive pixels can be minimized. When compared with the purpose of a control system, which is to follow the system command as precisely as possible, we find the objective of both systems are the same. Moreover, an edge or a boundary among image pixels can be regarded as a step command

in control systems. These observations lead to the idea of using control technologies to improve prediction result for pixels around boundaries. To realize this idea, we use an adaptive Takagi-Sugeno fuzzy neural network (TS-FNN) as the predictor for its advantages of fast convergence and parallel computation. Furthermore, the widely used **proportional controller** (P-controller) in control system is implemented implicitly in the consequent part of the network so that the prediction error can be further compensated for pixels around boundaries. We find in experiments that the proposed approach can have a very good prediction result even without using any online training area for network adaptation process. This makes the proposed system more feasible under limited resources, and a very good run time performance can be obtained. Finally, comparisons to existing state-of-the-art lossless predictors and coders will be given to highlight the advantages of the proposed novel approach.

5.1 Introduction

The performance of predictive image coding scheme highly depends upon the effectiveness of the predictor used in the coding process. Most of the image predictors perform very well in slowly varying areas. However, large prediction errors can take place around edges and boundaries, and this has remained a major problem in predictive coding schemes so far. Intuitively, the prediction results can be improved if we can foresee the existence of an edge and then predict along the edge orientation. However, the design of a robust edge detector and the analysis of edge orientation are difficult problems themselves, let alone to predict along the edge orientation. Recently, some approaches propose the use of a linear predictor adapted by least squares (LS) optimization during the coding process [25]-[31]. Among which, the EDP [26] pointed out that the superiority of LS adaptation is in its edge-directed property. That is, the LS-based predictor can adjust the prediction support along the edge orientation automatically during the adaptation process. With the edge-directed property, LS-based adaptive predictor performs very well for pixels around boundaries. On the other hand, we know that the normal equations provide the key for LS adaptation, and some fast algorithms, "Cholesky decomposition" for example, can be applied in the LS adaptation process. Therefore, the complexity in solving the normal equations itself is not a problem. Nevertheless, the computational cost for the construction of normal equations is rather high. Thus, a pixel-by-pixel LS adaptation process for predictive image coding is regarded as prohibitive. For this, an edge-look-ahead approach is proposed [27], [28]. The edge-look-ahead approach proposes a causal edge detector and initiates the LS adaptation process only when the coding pixel is around an edge or when the prediction error is beyond a predefined threshold. With the edge-

look-approach, the advantage of edge-directed property in LS-based predictor can be fully exploited. Moreover, a noticeable reduction in computational complexity can also be obtained when compared with that of pixel-by-pixel LS adaptation approach.

In order that the highly complex statistical redundancy can be removed more effectively, some of the results are obtained by using fuzzy logic or neural network as the nonlinear predictor [21]-[23]. Theoretically, the use of nonlinear predictors can get better prediction performance than that of obtained by using linear predictors. However, we find in literatures that fuzzy logic or neural network based nonlinear predictors usually come along with a very high computational complexity for inherent nonlinear characteristics and the time-consuming online adaptation process. Furthermore, the improvement on the prediction result obtained seems does not justify the use of such a highly complex nonlinear predictor when the run time performance is taken into consideration.

In this chapter, we propose an approach performing nonlinear prediction based on a four-layered Takagi-Sugeno fuzzy neural network (TS-FNN) [37]-[45]. The TS-FNN is applied in the proposed approach as the nonlinear predictor for its advantages in parallel computation, universal approximation and fast convergence [41]-[45]. With priori knowledge on the partitioning of input space, the network structure can be constructed with neurons provided with a set of Gaussian membership functions. Moreover, suitable fuzzy rules can also be derived and recruited by measuring the membership degrees so that a nonlinear prediction model is developed. To make the proposed system adapted to the varying statistics, the prediction error of the coding pixel is feedback to update the connection weights and the parameters in the membership layer. In order that the high computational complexity incurred in

the network learning process can be avoided, the number of training pixels used during the network adaptation process is decreased substantially and even not used in the proposed TS-FNN approach. Moreover, all the operations used in the proposed TS-FNN based predictor are linear except the the Gaussian membership degree measurement in second layer. This makes the proposed approach more feasible under limited resources and a very good run-time performance can be obtained.

Regarding to the problem of encountering a large prediction error for pixels around boundaries in most of the predictive coding schemes, we propose in this chapter a novel idea based on commonly used control technologies. Actually, the design of a controller is to follow the system command as precisely as possible, which has the same objective with predictive coding. Moreover, an edge or a boundary, i.e., an abrupt change among neighboring pixels, can be regarded as a step command in control systems. The above observations lead to the idea of enhancing the prediction result with control technologies. In this chapter, both the horizontal and vertical deviations around the coding pixel are calculated such that we can know the existence of an edge. The calculated deviations are then used as the inputs of P-controller compensators which are implemented implicitly in the network weights connecting the rule layer and the output layer, i.e., the consequent part of a fuzzy rule. As we will see in the experiment that the proposed P-controller compensator is very useful in removing the inter-pixel redundancy and can further improve the entropies of prediction errors.

Novelty of the proposed approach

In this chapter, we propose the use of a TS-FNN based nonlinear predictor for lossless coding of images. In order a very good run-time performance can

be obtained, the number of training pixels for network adaptation process is decreased substantially and even not used in the proposed approach. For pixel around edges, we propose a novel approach by applying the commonly used P-controller in control system as a compensator so that the prediction error can be further reduced. It is noted that the conventional error modeling technique, a statistical approach, can also be used for further reduction in entropies. With the proposed adaptive predictor and the P-controller compensation mechanism render the proposed approach very effective in removing statistical redundancy and feasible for real time applications.

The rest of the chapter is organized as follows. Section 5.2 gives an overview of the proposed TS-FNN based coding system. A detailed description on the prediction and the network adaptation process are also addressed in this section. Besides, a brief introduction on the conditional entropy coder applied in the proposed coding system is given in Section 5.3. The experimental results are shown in section 5.4 to demonstrate the effectiveness of the proposed system. Finally, a concluding remark is given in Section 5.5.

5.2 Proposed TS-FNN based coding system

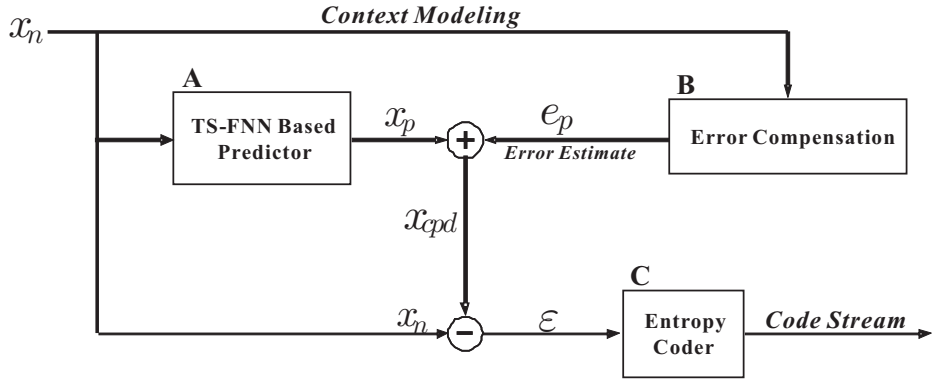


Figure 5.1: Proposed TS-FNN based coding system.

Fig. 5.1 shows the block diagram of the proposed TS-FNN based predictive coding system. As can be seen in Fig. 5.1, the proposed system is composed of three major blocks, the "TS-FNN based predictor", the "error compensation mechanism", and the "entropy coder". The TS-FNN based predictor is a four-layered linear-like network with Gaussian membership degree measurement and linear combinations. Besides, the network parameters are adapted in the coding process by using back propagation learning algorithm. As we will see in succeeding sections, the proposed system performs nonlinear prediction but with much reduced computational complexity just like a linear predictor. In the proposed system, the predictor output x_p is further refined through an error compensation mechanism, i.e., the so-called bias cancelation technique, to get a compensated prediction $x_{cpd} = x_p + e_p$. The refined error $\varepsilon = x_n - x_{cpd}$ is then entropy encoded through a conditional arithmetic coder to produce the bit stream [34]. It should be noted that all the pixels used in the proposed approach are causal, i.e., pixels previously encoded. Therefore, the decoder can reconstruct the image pixel from the re-

ceived bit stream by performing the same prediction and error compensation just like the encoder without any side information.

5.2.1 The TS-FNN based adaptive predictor

In this chapter, the predictor is implemented by using the TS-type fuzzy neural network (TS-FNN) proposed by Takagi and Sugeno [37]-[39]. It is noted that a highly complex nonlinear model can be easily described in the TS-FNN system with a small number of fuzzy "If-Then" rules [37]-[39]. In general, the kth rule, R_k, of the TS-FNN model can be expressed by

$R_k:$ If $z_1(t)$ is F_{1k} ...and $z_i(t)$ is F_{ik} Then

$$y_k(t) = A_k x(t) + B_k u(t) \; for \; k = 1, 2, \ldots, L, \quad (5.1)$$

where $z_1(t), \ldots, z_i(t)$ are the input variables, $F_{1k}, \ldots, F_{ik}$ are the fuzzy quantifiers (fuzzy sets) associated with corresponding input variables, $y_k(t) \in R^p$ represents the output of the kth rule, t denotes the current discrete time index, L denotes the number of rules, $A_k \in R^{p\times n}, x(t) \in R^n, B_k \in R^{p\times m}$, and $u(t) \in R^m$ are the consequent parts of the rule.

The proposed TS-FNN based predictor is shown in Fig. 5.2. As can be seen in Fig. 5.2 , the proposed network has four layers. The nodes in Layer1 (the input layer) transmit the input signal to their output directly, i.e., it plays the role of signal buffering. In the proposed approach, there are two nodes in the input layer ($i = 2$). Layer2 is the "membership layer". In this layer, the number of nodes associated with corresponding input variable is set to be three ($j = 3$). Layer3 is the "rule layer". Since the interconnection topology between nodes in layer2 is fully connected, we have nine (i.e., j^i) nodes (i.e., rules) in layer3. Layer4 is the "output layer". The output layer has only one node in the proposed approach, and its output is the prediction

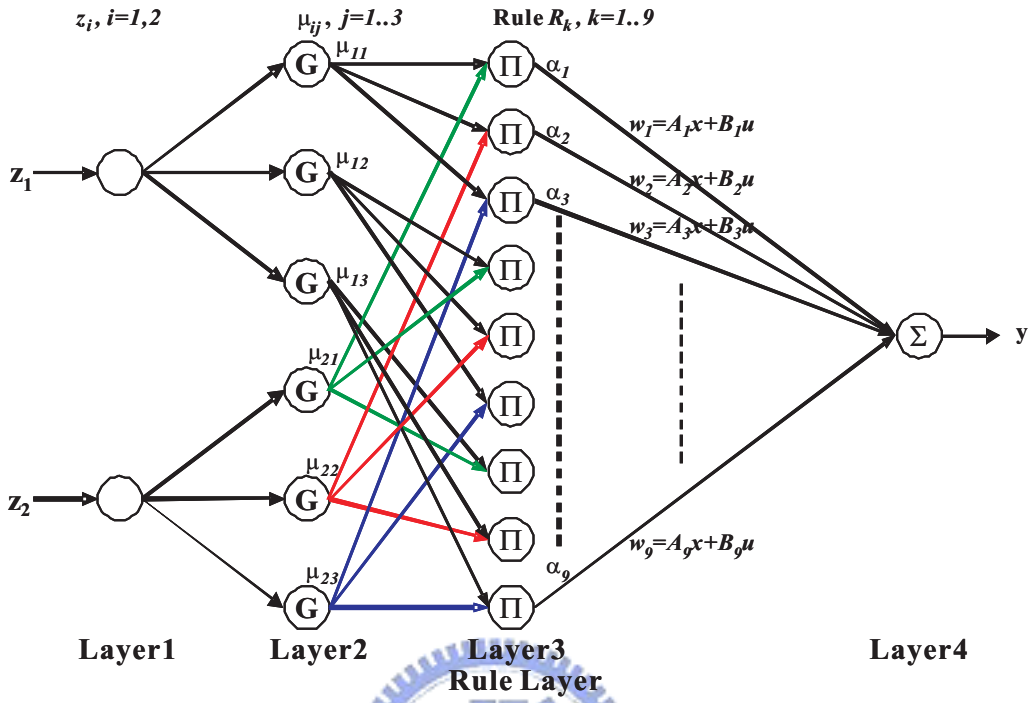


Figure 5.2: Proposed TS-FNN predictor.

value of the coding pixel. It is noted that the pixel values are normalized by 255 before they are feed into the system, suppose the input image has 8 bits per pixel, and the output of the network is scaled by 255 and then rounded to the nearest integer as the prediction value.

For notation convenience, the net input to the jth node in layer l and the corresponding activation function are denoted by $net_j^{(l)}$ and $f_j^{(l)}$ respectively, and the output of jth node in layer l, is given by $O_j^{(l)} = f_j^{(l)}(net_j^{(l)})$. A brief description about the proposed TS-FNN based predictor is given below.

Fig. 5.3 defines the texture context of the coding pixel. Besides, the input vector $z(t)$ of the proposed network is given by

$$z(t) = [z_1(t),\ z_2(t)]^T = [x_1 - x_3,\ x_2 - x_3]^T, \tag{5.2}$$

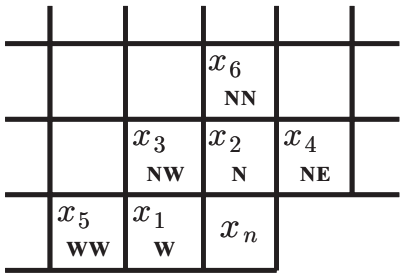


Figure 5.3: Texture context around the coding pixel.

where t denotes the discrete coding sequence. Obviously, z_1 measures the strength of vertical deviation (i.e., horizontal edge detection), while z_2 is for that of horizontal deviation (i.e., vertical edge detection). It should be noted that all the pixels used should be normalized by 255 before feed into the network in order not to fall into the saturation region during the network learning process.

Layer1: Input layer

The nodes in this layer just perform the buffering operations. That is, the input variables z_1 and z_2 are transmitted to layer2 directly. Therefore, the output of the ith node in layer1 is given by

$$O_i^{(1)} = z_i \quad \text{for } i = 1, 2. \tag{5.3}$$

Layer2: Membership layer

In this layer, the activation function of the jth node associate with the ith input, z_i, in layer1 is denoted by $f_{ij}^{(2)}$. Each node in this layer performs the membership degree measurement of a Gaussian function, and its output, $\mu_{ij} = O_{ij}^{(2)}$, specifies the degree to which the given input z_i satisfies the fuzzy

quantifier $f_{ij}^{(2)}$, that is,

$$\mu_{ij} = O_{ij}^{(2)} = f_{ij}^{(2)}(z_i) = exp\{-\frac{(z_i - m_{ij})^2}{\sigma_{ij}^2}\} \quad \text{for } i = 1, 2, \text{ and } j = 1, 2, 3, \tag{5.4}$$

where m_{ij} and σ_{ij} denotes the center and the width (i.e., standard deviation) of the Gaussian membership function respectively, and the subscript ij indicates the jth node associated with the ith input z_i in layer1.

Layer3: Rule layer

The links in this layer are used to implement the antecedent matching. The matching operation (or the fuzzy "**AND**" aggregation) is chosen as the simple "**product**" operation instead of "**min**" operation. Therefore, each node in this layer multiplies incoming signals and sends the product to its output. The output of the kth node in this layer represents the firing strength of the kth rule, and is given by

$$\alpha_k = O_k^{(3)} = \mu_{1p} * \mu_{2q} = net_k^{(3)} \quad \text{for } k = 1, 2, \ldots, 9,$$
$$p = 1, 2, 3, \text{ and } 1 \leq q = k - 3(p-1) \leq 3. \tag{5.5}$$

Layer4: Output layer

This layer performs the defuzzification process to get the numerical output. As can be seen in Fig. 5.2, the output of each node in layer3 (i.e., the firing strength α_k) is first weighted by a factor w_k, and then summed up together as the net input of layer4. The connection weight w_k, which represents the output action for the kth rule, is given by

$$w_k = A_k x(t) + B_k u(t) \quad \text{for } k = 1, 2, \ldots, 9, \tag{5.6}$$

where $x(t) = [x_1, x_2, \ldots, x_6]^T$ is composed of the six causal neighbors around the coding pixel, $A_k = [a_k^1, a_k^2, \ldots, a_k^6]^T$ can be regarded as the sixth-order predictor coefficients associated with the kth rule, $u(t) = [u_1, u_2]^T = [x_1 - x_3, x_2 - x_3]^T = z(t)$ is used for horizontal and vertical edge detection, and the vector $B_k = [b_1, b_2]^T$ is the weighting coefficients associated with the kth rule for the vector $u(t)$. As we know that a large prediction error can take place around an edge. Besides, an edge among image pixels during the coding process can be regarded as a step command in control system. Moreover, we find both the predictive coding and a control system have the same objective to minimize the difference between the desired and the actual output, i.e., the so-called error signal. These observations lead to the motivation of enhancing the prediction result with control technologies. For this, the term $B_k u(t)$, which is called "**P-controller**" in control system, is applied in (5.6) for prediction error suppression. It should be noted that the proposed P-controller compensator is quite different to the so-called "error modeling" or "bias cancelation technique" in [14], [19], and both of which can be applied jointly for further refinement of prediction errors.

The output of the TS-FNN system is then a linear combination of the consequent part. That is, the final output of the predictor network is a weighted summation of the individual output of the rules in layer3 (Fig. 5.2), and is given by

$$y(t) = net^{(4)} = \sum_{k=1}^{9} \alpha_k w_k = \sum_{k=1}^{9} \alpha_k (A_k x + B_k u). \tag{5.7}$$

As can be seen in (5.7), the output of the proposed TS-FNN based predictor is implemented in an un-normalized manner which has the advantages of a faster training rate and a much simpler input/output sensitivity equation [41], [44], [45]. In addition, it should be noted again that all the pixels

used are normalized by 255 before they are feed into the proposed predictor network. Therefore, the predictor output should be scaled by 255 this time and bounded in the range of $[0, 255]$ as the prediction value.

5.2.2 Network Adaptation process

In order that the proposed predictor network can adapt itself to the varying statistics of the image to be coded. The network parameters are updated continuously using the gradient descent method during the coding process. To do this, an online training area is commonly used [22]-[24]. However, the use of an online training area also means an increase in computational complexity. Therefore, we have investigated in this chapter the use of an online training area composed of only the four causal pixels $x_1, x_2, \cdots, x_4$ (Fig. 5.3) and even that of without using any online training area. Though with a substantially decreased number of training pixels, the prediction performance of proposed approach is still very good as we will see later in the experiments.

To adapt the parameters of the proposed predictor network during the coding process, we first define a cost function (or error function) $E(t)$ as in (5.8).

$$E(t) = \frac{1}{2}(d(t) - y(t))^2, \tag{5.8}$$

where $d(t)$ denotes the desired output and $y(t)$ denotes the actual output of the proposed TS-FNN system. The back propagation (BP) learning algorithm is used to minimize the error function, and the parameter w is adapted by

$$w(t+1) = w(t) + \eta(-\frac{\partial E(t)}{\partial w(t)}), \tag{5.9}$$

where $w = [m, \sigma, A_k, B_k]^T$, and η is the learning rate. We will first derive the adaptation rules for the parameters in the consequent part, i.e., the links

between layer3 and layer4. After that, the adaptation equations for those parameters in layer2 will be given.

A-1. Consequent part: Derivation for Δa_k^r

In the consequent part, coefficients of the nine sixth-order predictors and the P-controller compensators are to be updated. We will first derive the adaptation rules for the coefficients of the nine sixth-order predictors. The adaptation for a_k^r, the rth coefficient of the sixth-order predictor associated with the kth rule (or kth predictor) is given by

$$a_k^r(t+1) = a_k^r(t) + \Delta a_k^r(t) \quad \text{for } r = 1, 2, \ldots, 6, \text{ and } k = 1, 2, \ldots, 9, \quad (5.10)$$

where $\Delta a_k^r(t)$ is the correction term for $a_k^r(t)$ given by

$$\begin{aligned}
\Delta a_k^r(t) &= -\eta_A \frac{\partial E}{\partial a_k^r} \\
&= -\eta_A \left[\frac{\partial E}{\partial O^{(4)}}\right]\left[\frac{\partial O^{(4)}}{\partial net^{(4)}}\right]\left[\frac{\partial net^{(4)}}{\partial a_k^r}\right] \\
&= -\eta_A \left[\frac{\partial E}{\partial y}\right]\left[\frac{\partial y}{\partial net^{(4)}}\right]\left[\frac{\partial net^{(4)}}{\partial a_k^r}\right] \\
&= \eta_A (d-y)\left[\frac{\partial net^{(4)}}{\partial a_k^r}\right] \\
&= \eta_A (d-y)\frac{\partial}{\partial a_k^r}\left[\sum_{j=1}^{9} \alpha_j (A_j x + B_j u)\right] \\
&= \eta_A (d-y)\alpha_k x_r \\
&= \eta_A \delta^{(4)} \alpha_k x_r,
\end{aligned} \quad (5.11)$$

where $\delta^{(4)}$ is the *error signal* of layer4 (i.e., output layer) defined by

$$\delta^{(4)} = -\frac{\partial E}{\partial net^{(4)}} = -\left[\frac{\partial E}{\partial O^{(4)}}\right]\left[\frac{\partial O^{(4)}}{\partial net^{(4)}}\right] = (d-y). \quad (5.12)$$

In general, (5.11), the correction term for $a_k^r(t)$, is represented in the following form

$$\Delta a_k^r(t) = \eta_A \delta^{(4)} \alpha_k x_r + \tau \Delta a_k^r(t-1), \tag{5.13}$$

where τ is the momentum term to accelerate the convergence speed. Using (5.13), we can rewrite (5.10) as

$$a_k^r(t+1) = a_k^r(t) + \eta_A \delta^{(4)} \alpha_k x_r + \tau \Delta a_k^r(t-1) \quad \text{for } r = 1, 2, \ldots, 6, \text{ and } k = 1, 2, \ldots, 9. \tag{5.14}$$

A-2. Consequent part: Derivation for Δb_k^s

With a similar approach for that of a_k^r, the adaptation rule for b_k^s, i.e., the sth coefficient of the compensator associated with the kth rule, is given by

$$b_k^s(t+1) = b_k^s(t) + \Delta b_k^s(t) \quad \text{for } s = 1, 2, \text{ and } k = 1, 2, \ldots, 9, \tag{5.15}$$

where $\Delta b_k^s(t)$ is the correction term for $b_k^s(t)$ given by

$$\begin{aligned}
\Delta b_k^s(t) &= -\eta_B \frac{\partial E}{\partial b_k^s} \\
&= -\eta_B \left[\frac{\partial E}{\partial y}\right] \left[\frac{\partial y}{\partial net^{(4)}}\right] \left[\frac{\partial net^{(4)}}{\partial b_k^s}\right] \\
&= \eta_B (d - y) \left[\frac{\partial net^{(4)}}{\partial b_k^s}\right] \\
&= \eta_B (d - y) \frac{\partial}{\partial b_k^s} \left[\sum_{j=1}^{9} \alpha_j (A_j x + B_j u)\right] \\
&= \eta_B (d - y) \left[\alpha_k u_s\right] \\
&= \eta_B \delta^{(4)} \alpha_k u_s.
\end{aligned} \tag{5.16}$$

As for $\Delta a_k^r(t)$, a momentum term $\tau \Delta b_k^s(t-1)$ is added to (5.16) so that the convergence speed can be accelerated. Therefore, the correction term for the

$b_k^s(t)$ can be rewritten as

$$\Delta b_k^s(t) = \eta_B \delta^{(4)} \alpha_k u_s + \tau \Delta b_k^s(t-1). \tag{5.17}$$

Using (5.17), (5.15) can be rewritten as

$$b_k^s(t+1) = b_k^s(t) + \eta_B \delta^{(4)} \alpha_k u_s + \tau \Delta b_k^s(t-1) \quad \text{for } s = 1, 2, \text{ and } k = 1, 2, \ldots, 9. \tag{5.18}$$

B-1. Primary part: Derivation for Δm_{ij}

In the primary part, the mean m_{ij} and the standard deviation σ_{ij} of the six Gaussian membership functions in layer2 have to be updated. We will first derive the adaptation rules of the six Gaussian means, and then the adaptation rules for the six standard deviations will be given.

The adaptation rule for m_{ij}, i.e., the mean of the jth Gaussian membership function associated with the ith input variable, is given by

$$m_{ij}(t+1) = m_{ij}(t) + \Delta m_{ij}(t) \quad \text{for } i = 1, 2, \text{ and } j = 1, 2, 3, \tag{5.19}$$

where

$$\begin{aligned} \Delta m_{ij}(t) &= -\eta_m \frac{\partial E}{\partial m_{ij}} \\ &= -\eta_m \left[\frac{\partial E}{\partial y}\right] \left[\frac{\partial y}{\partial net^{(4)}}\right] \left[\frac{\partial net^{(4)}}{\partial m_{ij}}\right] \\ &= \eta_m (d - y) \left[\frac{\partial net^{(4)}}{\partial m_{ij}}\right], \end{aligned} \tag{5.20}$$

where $\frac{\partial net^{(4)}}{\partial m_{ij}}$, by applying chain rule, can be expressed as

$$\begin{aligned}
\frac{\partial net^{(4)}}{\partial m_{ij}} &= \sum_k \left[\frac{\partial net^{(4)}}{\partial O_k^{(3)}}\right]\left[\frac{\partial O_k^{(3)}}{\partial net_k^{(3)}}\right]\left[\frac{\partial net_k^{(3)}}{\partial O_{ij}^{(2)}}\right]\left[\frac{\partial O_{ij}^{(2)}}{\partial m_{ij}}\right] \\
&= \sum_{k=1}^{9} \Phi_k^{(ij)} \left[\frac{\partial net^{(4)}}{\partial O_k^{(3)}}\right]\left[\frac{\partial O_k^{(3)}}{\partial net_k^{(3)}}\right]\left[\frac{\partial net_k^{(3)}}{\partial O_{ij}^{(2)}}\right]\left[\frac{\partial O_{ij}^{(2)}}{\partial m_{ij}}\right] \\
&= \sum_{k=1}^{9} \Phi_k^{(ij)} w_k \frac{\alpha_k}{\mu_{ij}} \frac{\partial}{\partial m_{ij}} \left\{ exp\left[-\frac{(z_i - m_{ij})^2}{\sigma_{ij}^2}\right]\right\} \\
&= \sum_{k=1}^{9} \Phi_k^{(ij)} w_k \frac{\alpha_k}{\mu_{ij}} \mu_{ij} \frac{2(z_i - m_{ij})}{\sigma_{ij}^2} \\
&= \frac{2(z_i - m_{ij})}{\sigma_{ij}^2} \left\{ \sum_{k=1}^{9} \Phi_k^{(ij)} w_k \alpha_k \right\},
\end{aligned} \tag{5.21}$$

where

$$\Phi_k^{(ij)} = \begin{cases} 1 & \text{if } k = [j - (2-i)]\, r^{(2-i)} + [v - (i-1)]\, r^{(i-1)} \\ & \qquad \text{for } v = 1, 2, 3, \text{ and } r = 3 \\ 0 & \text{otherwise.} \end{cases} \tag{5.22}$$

By substituting (5.21) back into (5.20), we have

$$\begin{aligned}
\Delta m_{ij}(t) &= \eta_m \frac{2(z_i - m_{ij})}{\sigma_{ij}^2} \left\{ \sum_{k=1}^{9} \Phi_k^{(ij)} (d - y) w_k \alpha_k \right\} \\
&= \eta_m \frac{2(z_i - m_{ij})}{\sigma_{ij}^2} \left\{ \sum_{k=1}^{9} \Phi_k^{(ij)} \delta_k^{(3)} \alpha_k \right\},
\end{aligned} \tag{5.23}$$

where $\delta_k^{(3)}$ is the error signal associated with the kth node in layer3 defined

by

$$\begin{aligned}
\delta_k^{(3)} &= -\frac{\partial E}{\partial net_k^{(3)}} \\
&= -\left[\frac{\partial E}{\partial net^{(4)}}\right]\left[\frac{\partial net^{(4)}}{\partial O_k^{(3)}}\right]\left[\frac{\partial O_k^{(3)}}{\partial net_k^{(3)}}\right] \\
&= -\left[\frac{\partial E}{\partial y}\right]\left[\frac{\partial y}{\partial \alpha_k}\right] \\
&= (d-y)\frac{\partial}{\partial \alpha_k}\left[\sum_{j=1}^{9}\alpha_j w_j\right] \\
&= (d-y)w_k \\
&= \delta^{(4)} w_k.
\end{aligned} \tag{5.24}$$

The $\Phi_k^{(ij)}$ in (5.22) describes the connectivity of the nodes in layer3 which is associated with the ijth node in layer2. That is, if the kth node in layer3 has one of its incoming signal from the ijth node in layer2, then $\Phi_k^{(ij)} = 1$; otherwise $\Phi_k^{(ij)} = 0$. Combing (5.23) with the momentum term for convergence speed acceleration, we can rewrite (5.19) as

$$m_{ij}(t+1) = m_{ij}(t) + \eta_m \frac{2(z_i - m_{ij})}{\sigma_{ij}^2}\left\{\sum_{k=1}^{9}\Phi_k^{(ij)}\delta_k^3\alpha_k\right\} + \tau\Delta m_{ij}(t-1)$$
$$\text{for } i = 1, 2, \text{ and } j = 1, 2, 3. \tag{5.25}$$

B-2. Primary part: Derivation for $\Delta\sigma_{ij}$

The adaptation rule for σ_{ij}, i.e., the standard deviation of the jth Gaussian membership function associated with the ith input variable, is given by

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \Delta\sigma_{ij}(t) \quad \text{for } i = 1, 2, \text{ and } j = 1, 2, 3, \tag{5.26}$$

where

$$\begin{aligned}\Delta\sigma_{ij}(t) &= -\eta_\sigma \frac{\partial E}{\partial \sigma_{ij}} \\ &= -\eta_\sigma \left[\frac{\partial E}{\partial y}\right]\left[\frac{\partial y}{\partial net^{(4)}}\right]\left[\frac{\partial net^{(4)}}{\partial \sigma_{ij}}\right] \\ &= \eta_\sigma (d-y)\left[\frac{\partial net^{(4)}}{\partial \sigma_{ij}}\right],\end{aligned} \tag{5.27}$$

where $\frac{\partial net^{(4)}}{\partial \sigma_{ij}}$, by applying chain rule, can be expressed as

$$\begin{aligned}\frac{\partial net^{(4)}}{\partial \sigma_{ij}} &= \sum_k \left[\frac{\partial net^{(4)}}{\partial O_k^{(3)}}\right]\left[\frac{\partial O_k^{(3)}}{\partial net_k^{(3)}}\right]\left[\frac{\partial net_k^{(3)}}{\partial O_{ij}^{(2)}}\right]\left[\frac{\partial O_{ij}^{(2)}}{\partial \sigma_{ij}}\right] \\ &= \sum_{k=1}^{9} \Phi_k^{(ij)}\left[\frac{\partial net^{(4)}}{\partial O_k^{(3)}}\right]\left[\frac{\partial O_k^{(3)}}{\partial net_k^{(3)}}\right]\left[\frac{\partial net_k^{(3)}}{\partial O_{ij}^{(2)}}\right]\left[\frac{\partial O_{ij}^{(2)}}{\partial \sigma_{ij}}\right] \\ &= \sum_{k=1}^{9} \Phi_k^{(ij)} w_k \frac{\alpha_k}{\mu_{ij}} \frac{\partial}{\partial \sigma_{ij}}\left\{exp\left[-\frac{(z_i-m_{ij})^2}{\sigma_{ij}^2}\right]\right\} \\ &= \sum_{k=1}^{9} \Phi_k^{(ij)} w_k \frac{\alpha_k}{\mu_{ij}}\left[\mu_{ij}\frac{2(z_i-m_{ij})^2}{\sigma_{ij}^3}\right] \\ &= \frac{2(z_i-m_{ij})^2}{\sigma_{ij}^3}\left\{\sum_{k=1}^{9} \Phi_k^{(ij)} w_k \alpha_k\right\}.\end{aligned} \tag{5.28}$$

By substituting (5.28) back into (5.27), we have

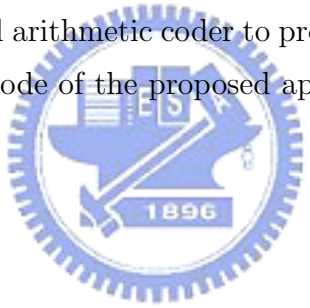
$$\begin{aligned}\Delta\sigma_{ij}(t) &= \eta_\sigma \frac{2(z_i-m_{ij})^2}{\sigma_{ij}^3}\left\{\sum_{k=1}^{9} \Phi_k^{(ij)}(d-y) w_k \alpha_k\right\} \\ &= \eta_\sigma \frac{2(z_i-m_{ij})^2}{\sigma_{ij}^3}\left\{\sum_{k=1}^{9} \Phi_k^{(ij)} \delta_k^{(3)} \alpha_k\right\}.\end{aligned} \tag{5.29}$$

Using (5.29) together with a momentum term, (5.26), the adaptation rule for

σ_{ij}, can be rewritten in the following form

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \eta_\sigma \frac{2(z_i - m_{ij})^2}{\sigma_{ij}^3} \left\{ \sum_{k=1}^{9} \Phi_k^{(ij)} \delta_k^{(3)} \alpha_k \right\} + \tau \Delta\sigma_{ij}(t-1)$$
$$\text{for } i = 1, 2, \text{ and } j = 1, 2, 3. \quad (5.30)$$

In this chapter, the updated network parameters are used for the prediction of next coding pixel. In addition to the predictor adaptation process, we also know that the prediction error can be further refined through an error compensation mechanism. For this, we use the proposed error modeling technique in [28] so that the the prediction error $x_n - x_p$ can be further compensated. The compensated error $\varepsilon = x_n - x_{cpd}$ has a narrower histogram and hence a lower first-order entropy. The refined error ε is then entropy encoded using a conditional arithmetic coder to produce the bit stream [34]. To summarize, the pseudo code of the proposed approach is given in Appendix C.



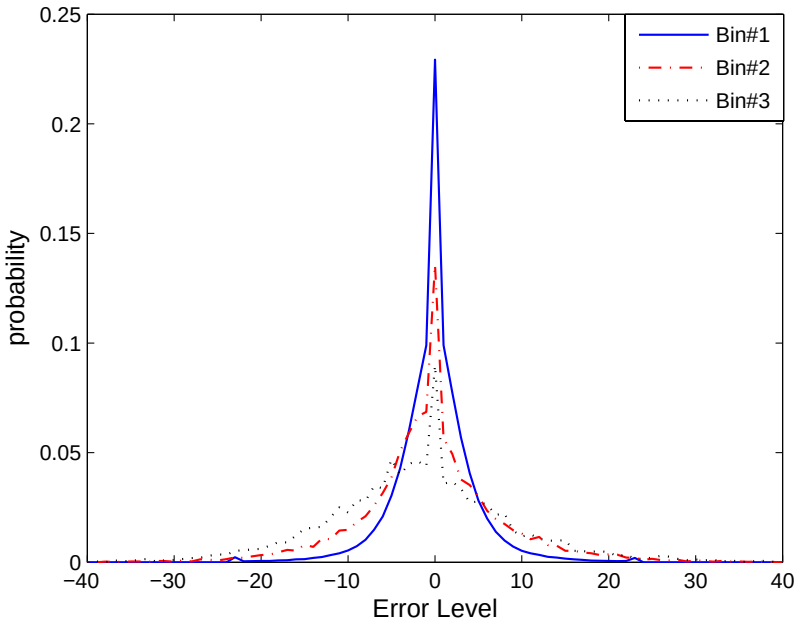


Figure 5.4: Histogram of refined errors in quantization bins for image "Lenna-grey."

5.3 Entropy Coding of Prediction Errors

In the proposed approach, the refined error ε is then entropy encoded through a conditional arithmetic coder. For this, we use the same approach introduced in Chapter 3, and the error strength estimate Δ of the coding pixel defined here is also the same as that of in Chapter 3. The approximately optimal quantization of Δ in the proposed approach is found to be $[0, 1], (1, 3], (3, 90], (90, \infty)$. With this quantization, one out of a set of four probability models is chosen for the entropy coding of ε based on the value

of error strength estimate Δ.

$$\begin{cases} use\ pmf\,1, & if\ 0 \leq \triangle \leq 1 \\ use\ pmf\,2, & if\ 1 < \triangle \leq 3 \\ use\ pmf\,3, & if\ 3 < \triangle \leq 90 \\ use\ pmf\,4, & otherwise. \end{cases} \tag{5.31}$$

In order that the coding efficiency in actual bit rates can be further improved, the techniques including "error sign flipping", "error remapping", and the "histogram tail truncation" are also applied in the proposed approach prior to it is entropy encoded. With the quantization bins in (5.31), the error histogram in each quantization bin for image "Lennagrey" (Fig. 2.6) are plotted in Fig. 5.4. The curve of Bin4 is not shown because no error strength estimate falls in that region. With the "histogram tail truncation" technique, the cut off region for the quantization bins are selected to be $[-23, 23]$, $[-41, 41]$, $[-59, 59]$ and $[-128, 127]$ such that over 99% of the refined errors in each quantization bin are within the truncated regions. By using the histogram tail truncation, an error 30 in Bin1 for example, is encoded first to be 23 using $pmf\,1$, followed by 7 using $pmf\,2$.

5.4 Experiments

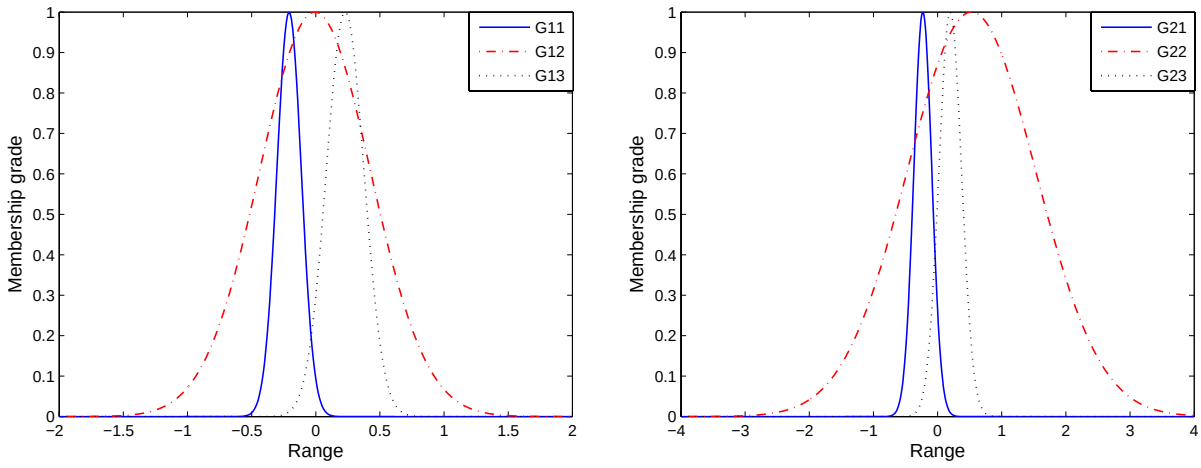
In this section, performance of the proposed TS-FNN based lossless image coding system will be evaluated. Comparisons to existing state-of-the-art linear and nonlinear predictors and coders are also given. All the test images used in the experiments are from TMW [20]. We will first demonstrate the effectiveness of the proposed TS-FNN based predictor and the implicitly implemented "P-controller" compensator. After that, the usefulness of the error compensation mechanism that uses the so-called bias cancelation technique will be demonstrated. Finally, the bit rate performance and the computational complexity of the proposed system will be discussed.

5.4.1 The network parameters

Table 5.1: Initial parameters for the six Gaussian membership functions in Layer2.

$m_{11} = -0.208$	$\sigma_{11} = 0.135$
$m_{12} = -0.006$	$\sigma_{12} = 0.612$
$m_{13} = 0.229$	$\sigma_{13} = 0.208$
$m_{21} = -0.230$	$\sigma_{21} = 0.199$
$m_{22} = 0.528$	$\sigma_{22} = 1.418$
$m_{23} = 0.197$	$\sigma_{23} = 0.250$

In this chapter, initial values of the parameters that are used in the proposed approach are listed in Table 5.1 to Table 5.4. These settings are obtained by using the image "Lennagrey" (Fig. 2.6) as the predictor input iteratively off-line so that the network parameters are well trained beforehand. Table 5.1 shows initial values of the center and standard deviation for the six Gaussian membership functions in layer2. To be clear, the six Gaussian membership distributions are also plotted in Fig. 5.5. The initial



(a) (b)

Figure 5.5: The six membership functions in layer2. (a) associated with input variable z_1. (b) associated with input variable z_2.

coefficients of the nine sixth-order predictors and the "P-controller" compensators are as shown in Table 5.2 and Table 5.3 respectively. In Table 5.3, some of the initial coefficients for the "P-controller" compensator appear to be zero just because they are too small to be represented under limited display precision.

Parameters used for network adaptation process are summarized in Table 5.4. As can be seen in Table 5.4, the learning rate for different parameters are defined separatively, while the momentum coefficient used are the same. Moreover, the learning rates and the momentum coefficient in Table 5.4 are chosen empirically, and we find them perform very well in our experiments. Therefore, these values will be used through out this chapter.

Table 5.2: Initial parameters for matrix A. (i.e., coefficients of the nine sixth-order predictors)

a_{11}= -0.287	a_{12}= -0.489	a_{13}= 0.225	a_{14}= 0.224	a_{15}= 0.098	a_{16}= 0.191
a_{21}= 0.195	a_{22}= -0.135	a_{23}= 0.247	a_{24}= -0.120	a_{25}= -0.245	a_{26}= -0.208
a_{31}= 0.258	a_{32}= 0.600	a_{33}= 0.154	a_{34}= -0.318	a_{35}= -0.003	a_{36}= -0.323
a_{41}= -0.369	a_{42}= 0.773	a_{43}= 0.152	a_{44}= -0.013	a_{45}= -0.023	a_{46}= -0.412
a_{51}= 0.267	a_{52}= 0.160	a_{53}= 0.111	a_{54}= 0.536	a_{55}= -0.006	a_{56}= 0.113
a_{61}= 0.220	a_{62}= 0.833	a_{63}= -0.237	a_{64}= -0.529	a_{65}= -0.107	a_{66}= -0.237
a_{71}= 0.199	a_{72}= 0.774	a_{73}= 0.148	a_{74}= -0.902	a_{75}= -0.249	a_{76}= -0.068
a_{81}= 0.567	a_{82}= 0.045	a_{83}= -0.254	a_{84}= 0.084	a_{85}= 0.050	a_{86}= -0.379
a_{91}= -0.222	a_{92}= -0.211	a_{93}= -0.169	a_{94}= 0.347	a_{95}= -0.046	a_{96}= 0.018

5.4.2 The training area for network adaptation

Further to the network adaptation process, there is one more thing that should be mentioned is the number of training pixels and the number of training cycles selected in the proposed approach. The use of a moderate training area around the coding pixel can adapt the predictor network to the local statistics. However, a large number of training pixels and training cycles can incur a drastic increase in computational complexity. In this chapter, the image pixels are encoded in an order of raster scan, i.e., from left to right and from top to bottom. Therefore, we choose in this chapter the training area to be the four nearest causal pixels $x_1, x_2, \ldots, x_4$ in Fig. 5.3. We also find in our experiments that a larger training area than the four pixels defined above has only minor improvement on the prediction results. Based on this observation, we even try not to use any training area in the network adaptation process, and the prediction results obtained are still very good as we will see later in the following experiments. Furthermore, the training

Table 5.3: Initial parameters for matrix B. (i.e., parameters of the nine "P-controller" compensators)

$b_{11} = 0.000$	$b_{12} = 0.000$
$b_{21} = 0.004$	$b_{22} = 0.004$
$b_{31} = 0.002$	$b_{32} = 0.002$
$b_{41} = 0.001$	$b_{42} = 0.001$
$b_{51} = 0.003$	$b_{52} = 0.003$
$b_{61} = -0.001$	$b_{62} = -0.001$
$b_{71} = 0.000$	$b_{72} = 0.000$
$b_{81} = 0.000$	$b_{82} = 0.000$
$b_{91} = 0.000$	$b_{92} = 0.000$

Table 5.4: Learning rates and momentum for network adaptation process.

Learning rate for matrix A in the connection from Layer3 to Layer4	$\eta_A = 0.01$
Learning rate for matrix B in the connection from Layer3 to Layer4	$\eta_B = 0.8$
Learning rate for the mean of Gaussian member functions	$\eta_m = 0.05$
Learning rate for the standard deviation of Gaussian member functions	$\eta_\sigma = 0.05$
Momentum used in parameters updating process	$\tau = 0.2$

cycle for network adaptation is set to be one in our experiments so that the run time performance of the proposed approach can be very good.

5.4.3 The "P-controller" based compensation mechanism

The usefulness of the proposed "P-controller" compensator, i.e., the term $B_k u$ in (5.6), can be evaluated through the following experiment. In this experiment, the fourteen test images in Table 5.5 will be used as the predictor input. We first set the B matrix in the consequent part to be a zero vector (denoted as Type 1), and then calculate the first-order entropies of uncompensated prediction error for the fourteen test images. After that, the prediction process is repeated again but with the component $B_k u$ in existence

Table 5.5: The usefulness of the proposed "P-controller" compensator. (i.e., the term $B_k u$)

Image	Original Entropy	Case 1: without online training area				Case 2: with 4-pixel online training area			
		B Absent (A1)	B Present (B1)	Difference (C1) = (A1) - (B1)	Percentage (C1)/(A1)	B Absent (A2)	B Present (B2)	Difference (C2) = (A2) - (B2)	Percentage (C2)/(A2)
Airplane	6.71	4.09	4.07	0.01	0.3%	4.09	4.08	0.01	0.3%
Baboon	7.36	6.11	6.10	0.01	0.1%	6.11	6.07	0.04	0.6%
Balloon	7.35	2.99	2.98	0.01	0.3%	2.97	2.96	0.01	0.4%
Barb	7.47	4.89	4.81	0.07	1.5%	4.83	4.72	0.11	2.3%
Barb2	7.48	4.90	4.87	0.03	0.6%	4.90	4.89	0.01	0.1%
Boats	7.10	4.23	4.17	0.06	1.4%	4.23	4.16	0.07	1.7%
Camera	7.01	4.86	4.85	0.01	0.2%	4.94	4.89	0.05	1.0%
Couple	6.39	4.12	4.07	0.05	1.2%	4.08	4.06	0.02	0.5%
Goldhill	7.53	4.64	4.64	0.00	0.0%	4.65	4.64	0.01	0.2%
Lena	7.59	4.67	4.67	-0.00	0.0%	4.67	4.68	-0.01	-0.2%
Lennagrey	7.45	4.32	4.32	-0.00	0.0%	4.33	4.34	-0.00	-0.1%
Noisesquare	5.72	5.44	5.43	0.01	0.2%	5.48	5.46	0.01	0.2%
Peppers	7.59	4.58	4.56	0.02	0.4%	4.55	4.53	0.01	0.3%
Shapes	6.74	2.55	2.48	0.07	2.7%	2.41	2.36	0.05	1.9%
Average	7.11	4.46	4.43	0.03	0.6%	4.44	4.42	0.03	0.6%

this time (denoted as Type 2). Moreover, the experiment can be classified into two cases, one with no online training area (denoted as Case 1), and the other with the predefined four-pixel online training region (denoted as Case 2).

In Table 5.5, the results obtained by setting the B matrix to be a zero vector (Type 1) are shown in the columns denoted as A1 and A2, and that of obtained by using the proposed "P-controller" compensator (Type 2) are shown in the columns denoted as B1 and B2 respectively. For comparison purpose, we also show in the last column of each case (Table 5.5) the percentage of improvement between the two types. As can be seen in Table 5.5, the proposed "P-controller" compensator brings up a conspicuous improvement from 1.4% to 2.7% on the first-order entropy of the three images "Barb", "Boats", and "Shapes". That is, the proposed approach is very useful for

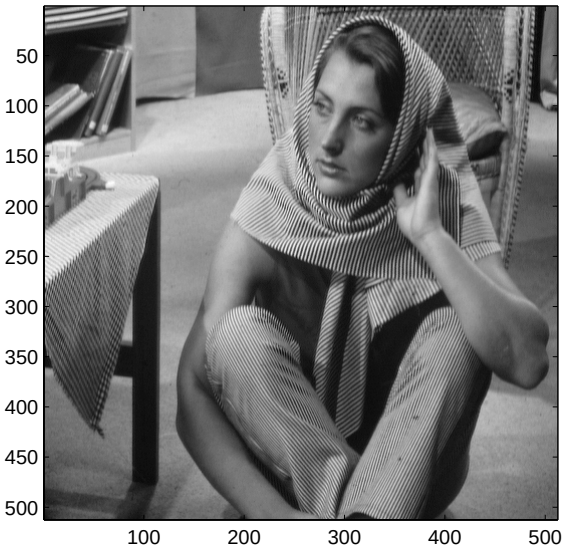
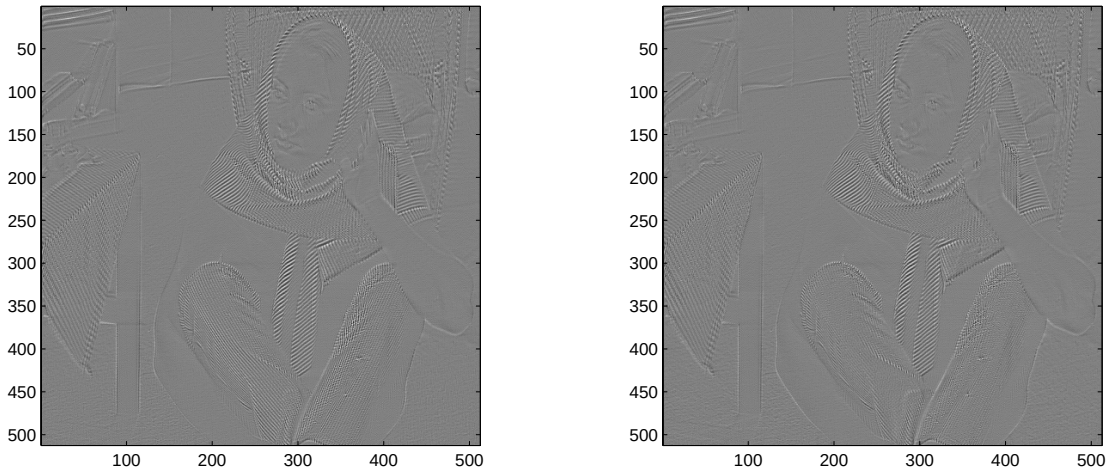


Figure 5.6: The image "Barb."

images with many edges and lines.

While expected to be very useful for pixels around edges and boundaries, the proposed "P-controller" compensator results in a minor degradation on the prediction result for the two images, "Lena" and "Lennagrey". It is noted that all the initial network parameters used in this chapter are generated off-line from the image "Lennagrey". Therefore, the results shown for the image "Lena" and "Lennagrey" are quite interesting. One possibility for explaining the degradation is the over-training of network parameters, and the cause of the fact has to be further investigated.

To look further into the usefulness of the proposed "P-controller" compensation mechanism, we use the image "Barb" in Fig. 5.6 as the test image. The images of uncompensated prediction error obtained with B matrix set to



(a) (b)

Figure 5.7: Uncompensated prediction error for image "Barb." (a) with B matrix absent. (b) with B matrix in presence.

a zero vector (Type 1) and in presence (Type 2) are shown in Fig. 5.7(a) and 5.7(b) respectively. In Fig. 5.7, both the two types perform online adaptation using the predefined four-pixel training area. The histograms of uncompensated prediction error for the two types in Fig. 5.7 are shown in Fig. 5.8. As can be seen in Fig. 5.8, the uncompensated prediction error obtained by using the proposed "P-controller" compensator has a narrower histogram, and hence a lower first-order entropy. Indeed, the entropies of the two histograms in Fig. 5.8 are 4.83bpp (Type 1) and 4.72bpp (Type 2) respectively.

5.4.4 Comparisons to existing predictors

Table 5.6 shows the comparison results of uncompensated prediction errors with state-of-the-art lossless image predictors for a set of eight test images in first-order entropies. The results of the median edge detector (MED) [15],

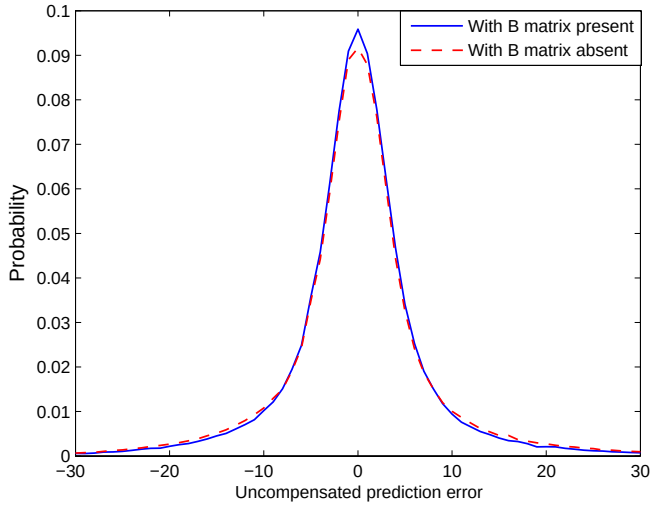


Figure 5.8: Histogram of uncompensated prediction error for the image "Barb." (both with the four-pixel online training area)

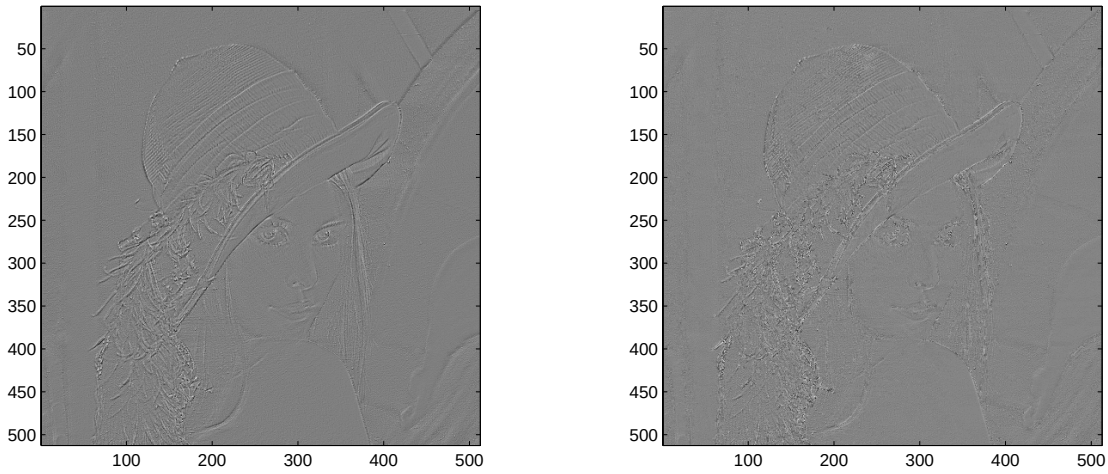
the gradient adjusted predictor (GAP) [14], the sixth-order edge directed predictor (EDP) [26], and the sixth-order LS-based edge-look-ahead approach [27] are taken from [26] and [27] respectively. The results of the proposed approach, which are obtained with the component $B_k u$ in existence, are listed in the last two columns (Case 1 and Case 2) of Table 5.6. As can be seen in Table 5.6, the proposed TS-FNN based predictor outperforms the MED and GAP predictors in all the test images, and is compatible with that of reported by the sixth-order EDP and the sixth-order LS-based edge-look-ahead approach, which justifies the usefulness of the proposed predictor approach.

Table 5.6: Comparisons on first-order entropy with existing state-of-the-art predictors.

Image	Original Entropy	MED	GAP	Sixth-Order EDP	Sixth-Order LS-Based Edge-look-ahead	Proposed	
						Without Local Training	With Local Training
Baboon	7.36	6.28	6.22	6.01	5.99	6.10	6.07
Lena	7.59	4.90	4.75	4.60	4.54	4.67	4.68
Lennagrey	7.45	4.56	4.40	4.26	4.20	4.32	4.34
Peppers	7.59	4.95	4.78	4.52	4.45	4.56	4.53
Barb	7.47	5.21	5.15	4.44	4.36	4.81	4.72
Barb2	7.48	5.19	5.06	4.80	4.77	4.87	4.89
Boats	7.10	4.31	4.29	4.14	4.11	4.17	4.16
Gold Hill	7.62	4.72	4.70	4.60	4.61	4.64	4.64
Average	7.46	5.02	4.92	4.67	4.63	4.77	4.75

5.4.5 The error compensation mechanism

The usefulness of the error compensation mechanism (Fig. 5.1) will be evaluated in this part. In this chapter, the error compensation mechanism uses the so-called "bias cancelation" technique for further refinement of the prediction errors [14], [15], [19], [28]. It should be noted that the conventional error compensation mechanism discussed here, a statistical approach, is quite different to the proposed "P-controller" compensator implemented implicitly in the consequent part of the predictor network. To demonstrate the effectiveness of the proposed error compensation mechanism, the image "Lennagrey" (Fig. 2.6) is used for the experiment. The uncompensated and refined prediction error for the image "Lennagrey" are shown in Fig. 5.9(a) and 5.9(b) respectively. Moreover, the corresponding histogram are as shown in Fig. 5.10. As can be seen in Fig. 5.10, the histogram of refined error has a narrower peak and hence a lower entropy. Actually, the entropy for the two error images are respectively, 4.116bpp (refined) and 4.335bpp (uncompensated).



(a) (b)

Figure 5.9: Prediction errors for the image "Lennagrey." (a) Uncompensated. (b) Refined.

5.4.6 Comparisons to existing state-of-the-art coders

To demonstrate the usefulness of the proposed coding system on actual bit rates, we also compare the proposed approach with state-of-the-art lossless image coders. Table 5.7 gives actual bit rates by JPEG-LS [15], CALIC [14], EDP [26], RALP [28] and TMW [20] for a set of fourteen test images. In Table 5.7, the results of JPEG-LS, CALIC, EDP, RALP and TMW are taken directly from [28]. All the bit rates of the proposed algorithm are obtained using the same parameters with no individual optimization. Besides, the bit rates of the proposed approach are obtained with the "P-controller" compensator in presence. We also show in the second column and the fifth column respectively the first-order entropies of the compensated prediction errors using the the proposed approach. Moreover, the execution time (in

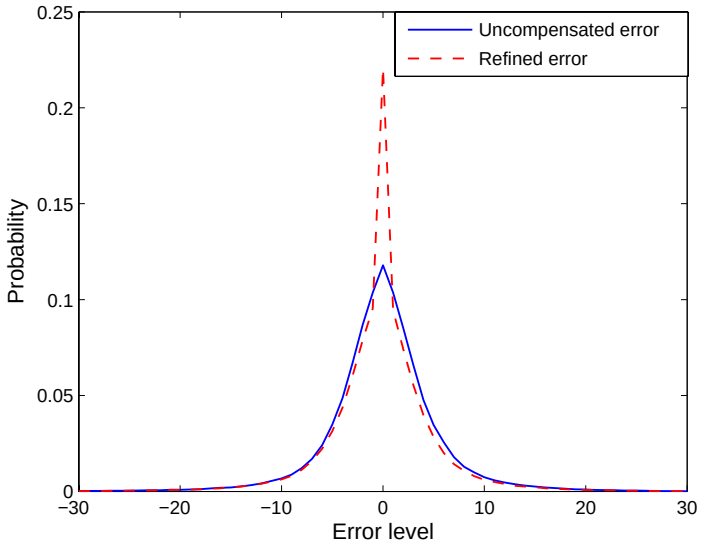


Figure 5.10: Histogram of uncompensated and refined prediction errors for the image "Lennagrey."

seconds; on a Pentium 1.4GHz machine) of the proposed coder are also listed in the fourth and seventh column so that we can get a picture on the runtime performance. It should be noted that some of the results in EDP are denoted by "N/A" because they are not reported in [26]. Therefore, the average value for EDP is different to that of for others.

As can be seen in Table 5.7, the proposed TS-FNN based approach outperforms JPEG-LS [15] in almost all of the test images, and has lower bit rates than CALIC [14] in eight out of the fourteen test images. Encouragingly, the proposed approach achieves lower bit rates than the highly complex TMW in two images, "Balloon" and "Noise square". We notice that the proposed bit rate performance for the image "Shapes" (Fig. 4.1) is inferior to that of obtained by JPEG-LS [15], CALIC [14], and RALP [28]. For this, we can see from Fig. 4.1 that the image "Shapes", an artificial image with many

Table 5.7: Comparisons on actual bit rates with existing state-of-the-art lossless image coders. (run on a P4-1.4GHz machine with memory 512MB)

Image	Proposed TS-FNN based coding system						JPEG-LS [15]	CALIC [14]	EDP [26]	RALP [28]	TMW [20]
	Without online training area			With 4-pixel online training area							
	First Oreder Entropy	Bit rate	seconds	First Oreder Entropy	Bit rate	seconds					
Airplane	3.80	3.69	3.36	3.79	3.68	7.88	3.82	3.74	N/A	3.71	3.60
Baboon	6.02	6.00	3.97	6.00	5.97	8.31	6.04	5.88	5.81	5.81	5.73
Balloon	2.57	2.56	4.94	2.55	2.54	11.88	2.90	2.83	N/A	2.55	2.66
Barb	4.60	4.54	3.67	4.54	4.49	7.94	4.69	4.32	4.11	4.12	4.09
Barb2	4.68	4.62	5.72	4.71	4.66	15.92	4.69	4.53	4.52	4.51	4.38
Boats	3.91	3.80	5.36	3.90	3.78	12.27	3.93	3.83	3.80	3.75	3.61
Camera	4.61	4.40	1.00	4.61	4.39	2.05	4.31	4.19	N/A	4.24	4.10
Couple	3.76	3.73	1.03	3.75	3.73	2.31	3.70	3.61	N/A	3.63	3.45
Gold Hill	4.44	4.36	5.58	4.45	4.36	12.59	4.48	4.39	4.39	4.32	4.27
Lena	4.49	4.46	3.56	4.50	4.47	9.44	4.61	4.48	4.40	4.35	4.30
Lennagrey	4.11	4.07	3.52	4.12	4.08	10.08	4.24	4.11	4.02	3.95	3.91
Noisesquare	5.32	5.35	0.98	5.40	5.44	2.11	5.68	5.44	N/A	5.37	5.54
Peppers	4.38	4.35	3.55	4.35	4.33	8.00	4.51	4.42	4.35	4.27	4.25
Shapes	1.97	1.86	3.31	1.90	1.79	11.30	1.21	1.14	N/A	1.52	0.76
Average	4.19	4.13	3.54	4.18	4.12	8.72	4.20	4.07	4.43	4.01	3.90

(* For the EDP method, the bit rates for some of the images are not available and the average is computed only for those that are available.)

edges and lines, can be segmented into many slowly varying areas in which the run-length coding method is most efficient to be used. In the meanwhile, we also know that the run-length encoding is applied in the coder of JPEG-LS [15] and RALP [28]. Furthermore, the CALIC [14] also uses a "Ternary entropy coder" for the so-called "binary mode" or uniform regions. Therefore, we think the bit rate performance for artificial images, like the image "Shapes", can be improved if a more sophisticated run-length coder or an extra entropy coder with a small symbol set is applied for slowly varying areas in the proposed approach.

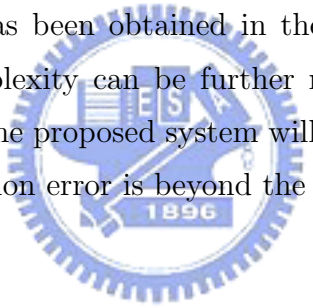
As to the computational complexity is compared, the JPEG-LS [15] that

uses the division-free MED predictor as the core of the image coder has the lowest complexity. The CALIC coding system [14] applies the use of the GAP as the core predictor. The GAP calculates the gradient of a coding pixel and chooses one out of a set of seven predictors based on the calculated gradient. Therefore, the complexity of the CALIC coding system is higher than that of JPEG-LS. It is noted that both the coefficients of GAP and MED predictors are fixed during the coding process. On the other hand, both the EDP [26] and RALP [28] use a least squares (LS) adapted predictor as the core of the coding system. We know that the normal equations provides the key for the solution of LS optimization problem. Moreover, fast algorithms, like *Cholesky decomposition*, can be used for the problem solving based on the singularity of the constructed normal equations [35], [36]. Therefore, the major complexity of the LS adaptation process in EDP and RALP will be in constructing the normal equations rather than solving them [25]-[28]. A detailed analysis on complexity of the LS-based adaptive predictor can be found in [27] and [28]. Obviously, the computational complexity of EDP and RALP are little higher than that of JPEG-LS and CALIC.

5.4.7 Computational complexity of the proposed approach

When reviewing the state-of-the-art lossless image coders, we can find that some of which are designed for pursuing the ultimate compressibility regardless of the computational complexity encountered [20], [21]. In this chapter, the proposed TS-FNN based approach is designed to be practical and feasible under limited resources. In the proposed system, the prediction errors are used to update the network parameters with back propagation learning algorithm during the prediction process. Moreover, the learning cycle is set

to be one for each coding pixel such that the computational complexity can be reduced. Unlike other neural network based image predictors, the number of online training pixels is decreased substantially and even not used in the proposed approach so that the network adaptation process can be accelerated. As can be seen in previous sections and the pseudo code, the major nonlinear operation in the proposed system is the calculation of the six Gaussian membership degree, i.e., (5.4), in the membership layer. In the network adaptation process, only addition, multiplication and division operations are needed ((5.12), (5.14), (5.18), (5.25), (5.30)). To get a picture about the run-time performance, we also listed in Table 4.5 the execution times of the proposed system (on a Pentium 1.4GHz machine). As can be seen in Table 4.5, a very good trade-off between the computational complexity and the prediction results has been obtained in the proposed system. Indeed, the computational complexity can be further reduced if we can define an error threshold so that the proposed system will not activate the adaptation process until the prediction error is beyond the pre-selected threshold.



5.5 Concluding Remarks

In this chapter, we have proposed the use of a TS-FNN based predictor for lossless coding of images. As the number of online training pixels for the network adaptation process is decreased substantially, a very good run-time performance can be achieved in the proposed approach. For pixels around edges, we propose a novel approach by implementing implicitly the commonly used "P-controller" compensator in control system into the network weights so that the prediction result can be improved. It is noted that the proposed "P-controller" compensator is quite different to the so-called "bias cancelation" technique applied in most of the lossless image coders. As can be seen in our experiments, the use of the TS-FNN based predictor and the "P-controller" compensator render the proposed approach highly adaptable to the varying statistics of coding images. Besides, comparisons to existing state-of-the-art lossless image predictors and coders have demonstrated the usefulness of the proposed approach, and what's more, we have brought up an idea of enhancing the predictive coding efficiency in a quite different aspect.

Chapter 6

Conclusion

In this dissertation, a switching coding scheme that combines the advantages of both run-length and adaptive linear predictive coding is proposed. For pixels in slowly varying areas, run-length coding is used; otherwise an LS-based adaptive predictor is used. We find in experiments that the LS-based adaptive predictor performs very useful around boundaries for its edge-directed property. Instead of performing LS adaptation in a pixel-by-pixel manner, we adapt the predictor coefficients only when an edge is detected or when the prediction error is greater than a predefined threshold so that the computational complexity can be significantly reduced. For this, we propose a simple yet effective edge detector using only causal pixels. This way, the proposed system can look ahead to determine if the coding pixel is around an edge and initiate the LS adaptation in advance to prevent the occurrence of a large prediction error. With the proposed switching structure, very good prediction results can be obtained in both slowly varying areas and pixels around boundaries. Moreover, only causal pixels are used for estimating the coding pixels in the proposed encoder; no additional side information needs to be transmitted. When compared with the pixel-by-pixel LS adaptation, the proposed approach can achieve a noticeable reduction in complexity with

only a minor degradation in entropy; a good tradeoff between computational complexity and prediction results has been obtained. Furthermore, comparisons to existing state-of-the-art lossless image predictors and coders have demonstrated the superiority of the proposed system.

In addition to the proposed edge-look-ahead approach. We find a large prediction error can usually take place for pixels around boundaries. Therefore, we also propose a novel idea of using control technologies to improve the prediction result of pixels around boundaries. To realize this idea, we have implemented an adaptive predictor based on Takagi-Sugeno fuzzy neural network. Moreover, the widely used proportional controller in control theory is applied implicitly in the consequent part of the network as a compensator to enhance the prediction result around edges. The effectiveness of the proposed novel approach, though not very conspicuous at present, can be further improved if a more sophisticated compensator is applied, and what's more, we have brought up an idea of solving this problem in a quite different aspect for lossless compression of images.

Bibliography

[1] Khalid Sayood, *Introduction to Data Compression*, Morgan-Kaufmann, 2^{nd} Edition, pp. 139-163, 2000.

[2] K. R. Rao and J. J. Hwang, *Techniques & Standards For Image, Video & Audio Coding*, Prentice-Hall, pp. 31-42, 1996.

[3] B. Carpentieri, M. J. Weinberger, and G. Seroussi, "Lossless compression of continuous-tone images," *Proc. IEEE*, Vol. 88, No. 11, pp. 1797-1809, Nov. 2000.

[4] N. Memon and X. Wu, "Recent developments in context-based predictive techniques for lossless image compression," *Comput. J.*, Vol. 40, No. 2/3, pp. 127-136, 1997.

[5] Xiaolin Wu, "An algorithmic study on lossless image compression," *Proc. Data Compression Conf.*, Snowbird, UT, USA, pp. 150-159, 31 March 3 April 1996.

[6] X. Wu and T. Qiu, 'Wavelet Coding of Volumetric Medical Images for High Throughput and Operability," *IEEE Trans. Medical Imaging*, Vol. 24, No. 6, pp. 719-727, Jun. 2005.

[7] A. T. Deever and S. S. Hemami, 'Lossless Image Compression With Projection-Based and Adaptive Reversible Integer Wavelet Transforms," *IEEE Trans. Image Process.*, Vol. 12, No. 5, pp. 489-499, May 2003.

[8] M. Grangetto, E. Magli, M. Martina, and G. Olmo, 'Optimization and Implementation of the Integer Wavelet Transform for Image Coding," *IEEE Trans. Image Process.*, Vol. 11, No. 6, pp. 596-604, Jun. 2002.

[9] N. V. Boulgouris, D. Tzovaras, and M. G. Strintzis, 'Lossless image compression based on optimal prediction, adaptive lifting, and conditional arithmetic coding," *IEEE Trans. Image Process.*, Vol. 10, No. 1, pp. 1-14, Jan. 2001.

[10] J. Reichel, G. Menegaz, M. J. Nadenau, and M. Kunt, 'Integer Wavelet Transform for Embedded Lossy to Lossless Image Compression," *IEEE Trans. Image Process.*, Vol. 10, No. 3, pp. 383-392, Mar. 2001.

[11] N. Memon, X. Kong, and J. Cinkler, "Context-based lossless and near-lossless compression of EEG signals," *IEEE Trans. Information Technology in Biomedicine*, Vol. 3, pp. 231-238, Sept. 1999.

[12] K. H. Yang and A. F. Faryar, "A contex-based predictive coder for lossless and near-lossless compression of video," in *Proc. IEEE Int. Conf. Image Process.*, Vol. 1, pp. 144-147, Sept. 10-13, 2000.

[13] X. Y. Lin, L.-K. Shark, M. R. Varley, and B. J. Matuszewski, "A hybrid lossless compression scheme using region-based predictive coding and integer wavelet transform," in *Proc. IEEE Int. Conf. Image Process.*, Vol. 3, pp. 486-489, Oct. 7-10, 2001.

[14] X. Wu and N. Memon, "Context-based, adaptive, lossless image coding," *IEEE Trans. Communications*, Vol. 45, No. 4, pp. 437-444, April 1997.

[15] M. J. Weinberger, G. Seroussi, and G. Sapiro, "The LOCO-I lossless image compression algorithm: principles and standardization into JPEG-LS," *IEEE Trans. Image Process.*, Vol. 9, No. 8, pp. 1309-1324, Aug. 2000. [Online].

Available: http://www.hpl.hp.com/loco/locodownold.htm

[16] G. Motta, J. A. Storer, and B. Carpentieri, "Lossless image coding via adaptive linear prediction and classification," *Proc. IEEE*, Vol. 88, No. 11, pp. 1790-1796, Nov. 2000.

[17] I, Avcibas, N. Memon, B. Sankur, and K. Sayood, 'A Successively Refinable Lossless Image-Coding Algorithm," *IEEE Trans. Communications*, Vol. 53, No. 3, pp. 445-452, Mar. 2005

[18] M. K. Das, C. C. Li, and S. R. Burgett, "A new multiresolution predictive scheme for lossless compression of medical images," *Proc. IEEE Int. Symp. Computer-Based Medical Systems*, pp. 34-39, June 13-16, 1993.

[19] X. Wu, "Lossless compression of continuous-tone images via context selection, quantization, and modeling," *IEEE Trans. Image Process.*, Vol. 6, No. 5, pp. 656-664, May 1997.

[20] B. Meyer and P. E. Tischer, "TMW-a new method for lossless image compression," in *Proc. Int. Picture Coding Symp.*, Berlin, Germany, Oct. 1997.

[21] B. Aiazzi, L. Alparone, and S. Baronti, "Fuzzy logic-based matching pursuits for lossless predictive coding of still images," *IEEE Trans. Fuzzy Systems*, Vol. 10, No. 4, pp. 473-483, Aug. 2002.

[22] R. D. Dony, and S. Haykin, "Neural network approaches to image compression," *Proc. IEEE*, Vol. 83, No. 2, pp. 288-303, Feb. 1995.

[23] L.-J. Kau, Y.-P. Lin and C.-T. Lin, "Lossless image coding using adaptive, switching algorithm with automatic fuzzy context modeling," *IEE Proc. Vision, Image and Signal Processing*, Vol. 153, No. 5, pp. 684-694, Oct. 2006.

[24] C. N. Manikopoulos, "Neural network approach to DPCM system design for image coding," *Proc. IEE Communications, Speech and Vision*, Vol. 139, No. 5, pp. 501-507, 1992.

[25] N. Kuroki, T. Nomura, M. Tomita and K. Hirano, "Lossless image compression by two-dimensional linear prediction with variable coefficients," *IEICE Trans. Fundamentals*, Vol. E75-A, No. 7, pp. 882-889, July 1992.

[26] X. Li and M. T. Orchard, "Edge-directed prediction for lossless compression of natural images," *IEEE Trans. Image Process.*, Vol. 10, No. 6, pp. 813-817, June 2001.

[27] L.-J. Kau and Y.-P. Lin, "Adaptive lossless image coding using least squares optimization with edge-look-ahead," *IEEE Trans. Circuits and Systems II*, Vol. 52, No. 11, pp. 751-755, Nov. 2005.

[28] L.-J. Kau and Y.-P. Lin, "Least Squares-Based Switching Structure for Lossless Image Coding," *IEEE Trans. Circuits and Systems I*, Vol. 54, No. 7, pp. 1529-1541, July 2007.

[29] H. Ye, G. Deng and J. C. Devlin, "A weighted least squares method for adaptive prediction in lossless image compression," in *Proc. Picture Coding Symp.*, pp. 489-493, Saint-Malo, France, 2003.

[30] B. Meyer and P. E. Tischer, "Glicbawls - grey level image compression by adaptive weighted least squares," in *Proc. Data Compression Conf.*, pp. 503, Snowbird, Utah, USA 2001.

[31] H. Ye, G. Deng, and J. C. Devlin, 'Adaptive linear prediction for lossless coding of greyscale images," in *Proc. IEEE Image Process.*, Vol. 1, pp. 128-131, Sep. 2000.

[32] E. D. Sontag, "Feedback stabilization using two-hidden-layer nets," *IEEE Trans. Neural Networks*, Vol. 3, No. 6, pp. 981-990, Nov. 1992.

[33] Chin-Teng Lin and C. S. George Lee, *A Neural-Fuzzy Synergism to Intelligent Systems*, Prentice Hall, pp. 235-250, 1996.

[34] I. H. Witten, R. M. Neal, and J. G. Cleary, "Arithmetic coding for data compression," *Communications of the ACM*, Vol. 30, No. 6, pp. 520-540, June 1987.

[35] S. J. Leon, *Linear Algebra with Applications,* New Jersey, Prentice Hall, 2002, pp. 482-488.

[36] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C,* Cambridge University Press, 2002, pp. 96-98. [Online]. Available: http://www.library.cornell.edu/nr/cbookcpdf.html

[37] M. Sugeno and G. T. Kang, "Structure identification of fuzzy model," *Fuzzy Sets and Systems*, Vol. 28, pp. 15-33, 1988.

[38] T. Takagi and M. Sugeno, "Fuzzy identification of systems and its applications to modeling and control," *IEEE Trans. Syst. Man. Cyber.*, Vol. 15, pp. 116-132, 1985.

[39] K. Tanaka and Hua O. Wang, *Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach*, John Wiley & Sons, Inc. 2001.

[40] C. F. Juang and C. T. Lin, "An online self-constructing neural fuzzy inference network and its applications," *IEEE Trans. Fuzzy Systems*, Vol. 6, pp. 12-32, 1998.

[41] Y. C. Chen and C. C. Teng, "A model reference control structure using a fuzzy neural network," *Fuzzy Sets and Systems*, Vol. 73, pp. 291-312, 1995.

[42] C. H. Lee and C. C. Teng, "Identification and control of dynamic systems using recurrent fuzzy neural networks," *IEEE Trans. Fuzzy Systems*, Vol. 8, No. 4, pp. 349-366, 2000.

[43] C. H. Lee and C. C. Teng, "Fine tuning of membership functions for fuzzy neural systems," *Asian Journal of Control*, Vol. 3, No. 3, pp. 18-25, 2001.

[44] C. H. Lee and M. H. Chiu, "Adaptive Nonlinear Control Using TSK-Type Recurrent Fuzzy Neural Network System," *Lecture Notes in Computer Science*, Vol. 4491, pp. 38-44, 2007.

[45] C. H. Lee, W. Y. Lai, and Y. C. Lin, "A TSK-type fuzzy neural network systems for dynamic systems identification," in *Proc.* 42*nd IEEE Int. Conf. Decision and Control*, pp. 4002-4007, Hawaii, U.S.A, 2004.

Appendix

Appendix A

Pseudo Code of the Proposed Edge-look-ahead Approach

{Pseudo-C implementation of the proposed edge-look-ahead approach in the 1-D scenario.}

/* Symbols definition */

// σ^2: variance

// σ_h^2: variance_h

// σ_l^2: variance_l

// γ_l: gamma_l

// γ_2: gamma_2

// x_n: x[n]

// e_n: e[n]

// θ: theta

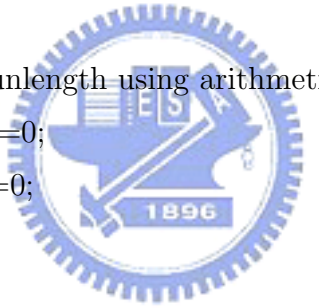
// $x_n(k)$: x[n-k]

// x_p: xp

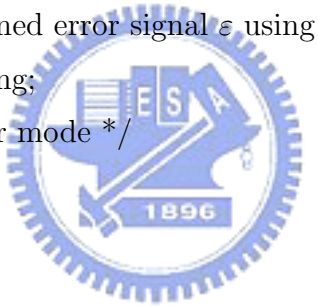
// $a(k)$: a[k]

```
runmode=0;
while (scan unfinished)
{
    if ( runmode or (variance==0) )
    /* the case that run mode is initiated */
       {
          if ( x[n]==x[n-1] )
             {
                runmode=1;
                runlength++;
             }
          else
             {
                Encode runlength using arithmetic coding.
                runlength=0;
                runmode=0;
             }
       }
    else
          runmode=0;

    if (not runmode)
    /* the case that regular mode is used */
       {
          edge_detected=0;
          if ( variance >= gamma_1 )
             {
```



```
            Calculate variance_h and variance_l;
            if ( variance >= gamma_2*(variance_h + variance_l) )
            /* the case that an edge exists */
               edge_detected=1;
         }
      if ( edge_detected or (e[n-1] >= theta) )
         Adapt predictor;
      /* Perform prediction */
      xp=0;
      for (k=1; k<=N; k++)
         xp=xp+a[k]*x[n-k];
      Perform error compensation;
      Encode the refined error signal ε using conditional
      arithmetic coding;
   } /* end of regular mode */
} /* end of scan */
```

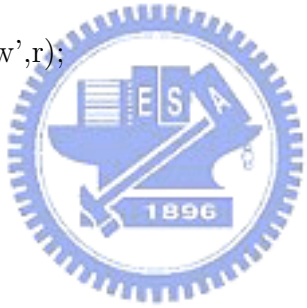


Appendix B

Matlab Code for Sobel operator

{Matlab code for the Sobel operator.}

```
%Main Program
fid=fopen('Lennagrey.raw',r);
f=fread(fid,[512,512]);
f=f';
EdgeImage=Sobel(f,50);
imagesc(EdgeImage);
axis square;
colormap(gray);
```



```
%Sobel function
function y=Sobel(image, th)
[Nn Nm]=size(image);
h=[-1 -2 -1;0 0 0;1 2 1];
Gx=filter2(h,image);
Gy=filter2(h',image);
F=abs(Gx)+abs(Gy);
```

```
for i=1:Nn
    for j=1:Nm
        if F(i,j) < th
          y(i,j)=255;
        else
           y(i,j)=0;
        end
    end
end
```

Appendix C

Pseudo Code of the P-Controller Compensation Approach

{Pseudo code of the P-controller compensation approach.}

/* Symbols definition */

// x_n: xn

// x_p: xp

// z_i: z[i]

// u_i: u[i]

// μ_{ij}: mu[i][j]

// m_{ij}: m[i][j]

// σ_{ij}: sigma[i][j]

// α_k: alpha[k]

// w_k: w[k]

// a_k^r: a[k][r]

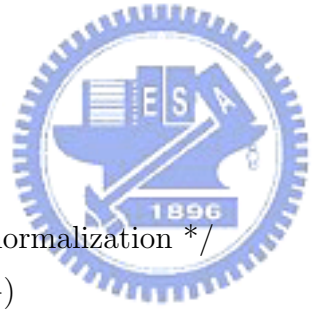
// b_k^s: b[k][s]

// $\delta^{(4)}$: delta[4]

```
// Δa_k^r: Delta_a[k][r]
// Δb_k^s: Delta_b[k][s]
// Δm_ij: Delta_m[i][j]
// Δσ_ij: Delta_sigma[i][j]
// η_A: eta_A
// η_B: eta_B
// η_m: eta_m
// η_σ: eta_sigma
// τ: tau

NodeinLayer1=2;
NodeinLayer2=3;
NodeinLayer3=9;
Av=255;
while (scan unfinished)
{
    /* Texture context normalization */
    for (t=1; t<=6; t++)
       x[t]=x[t]/Av;
    xf=xn/Av;
    z[1]=x[1]-x[3];
    z[2]=x[2]-x[3];
    u[1]=z[1];
    u[2]=z[2];

     /* Compute the output of Layer2 */
    for (i=1; i <= NodeinLayer1; i++)
```



```
    for (j=1; j <= NodeinLayer2; j++)
        mu[i][j]=exp(-sqr(z[i]-m[i][j])/sqr(sigma[i][j]));

/* Compute the output of Layer3 */
k=1;
for (i=1; i <= NodeinLayer2; i++)
    for (j=1; j <= NodeinLayer2; j++)
        {
            alpha[k]=mu[1][i]*mu[2][j];
            k++;
        }

/* Layer4 */
y=0;
for (k=1; k <= NodeinLayer3; k++)
{
    w[k]=0;
    for (r=1; r <= 6; r++)
        w[k]=w[k]+a[k][r]*x[r];
    for (s=1; s <= 2; s++)
        w[k]=w[k]+b[k][s]*u[s];
    yy[k]=alpha[k]*w[k];
    y=y+yy[k];
}
/* Compute the actual output */
xp=Round(y*Av);
if xp > 255 then xp=255
```

else if xp < 0 then xp=0;
e=xn-xp;

Perform error compensation to get refined error ε;
Encode the refined error ε using conditional arithmetic coding.

/* Update a_k^r and b_k^s in the connection between Layer3 to Layer4*/
delta[4]=xf-y;
for (k=1; k <= NodeinLayer3; k++)
{
  for (r=1; r <= 6; r++)
  {
    Delta_a[k][r]=eta_A*delta[4]*alpha[k]*x[r] + tau*Delta_a[k][r];
    a[k][r]=a[k][r] + Delta_a[k][r];
  }
  for (s=1; s <= 2; s++)
  {
    Delta_b[k][s]=eta_B*delta[4]*alpha[k]*u[s] + tau*Delta_b[k][s];
    b[k][s]=b[k][s] + Delta_b[k][s];
  }
}

/* Compute Δm_{ij} */
Delta_m[1][1]=eta_m*2*(z[1]-m[1][1])*delta[4]*(yy[1]+yy[2]+yy[3])/sqr(sigma[1][1])
+ tau*Delta_m[1][1];
Delta_m[1][2]=eta_m*2*(z[1]-m[1][2])*delta[4]*(yy[4]+yy[5]+yy[6])/sqr(sigma[1][2])

```
+ tau*Delta_m[1][2];
Delta_m[1][3]=eta_m*2*(z[1]-m[1][3])*delta[4]*(yy[7]+yy[8]+yy[9])/sqr(sigma[1][3])
+ tau*Delta_m[1][3];
Delta_m[2][1]=eta_m*2*(z[2]-m[2][1])*delta[4]*(yy[1]+yy[4]+yy[7])/sqr(sigma[2][1])
+ tau*Delta_m[2][1];
Delta_m[2][2]=eta_m*2*(z[2]-m[2][2])*delta[4]*(yy[2]+yy[5]+yy[8])/sqr(sigma[2][2])
+ tau*Delta_m[2][2];
Delta_m[2][3]=eta_m*2*(z[2]-m[2][3])*delta[4]*(yy[3]+yy[6]+yy[9])/sqr(sigma[2][3])
+ tau*Delta_m[2][3];

/* Compute Δσ_ij */
Delta_sigma[1][1]=eta_sigma*2*sqr(z[1]-m[1][1])*delta[4]*(yy[1]+yy[2]+yy[3])
/power(sigma[1][1],3) + tau*Delta_sigma[1][1];
Delta_sigma[1][2]=eta_sigma*2*sqr(z[1]-m[1][2])*delta[4]*(yy[4]+yy[5]+yy[6])
/power(sigma[1][2],3) + tau*Delta_sigma[1][2];
Delta_sigma[1][3]=eta_sigma*2*sqr(z[1]-m[1][3])*delta[4]*(yy[7]+yy[8]+yy[9])
/power(sigma[1][3],3) + tau*Delta_sigma[1][3];
Delta_sigma[2][1]=eta_sigma*2*sqr(z[2]-m[2][1])*delta[4]*(yy[1]+yy[4]+yy[7])
/power(sigma[2][1],3) + tau*Delta_sigma[2][1];
Delta_sigma[2][2]=eta_sigma*2*sqr(z[2]-m[2][2])*delta[4]*(yy[2]+yy[5]+yy[8])
/power(sigma[2][2],3) + tau*Delta_sigma[2][2];
Delta_sigma[2][3]=eta_sigma*2*sqr(z[2]-m[2][3])*delta[4]*(yy[3]+yy[6]+yy[9])
/power(sigma[2][3],3) + tau*Delta_sigma[2][3];

/* Update m_ij and σ_ij in Layer2 */
for (i=1; i <= NodeinLayer1; i++)
    for (j=1; j <= NodeinLayer2; j++)
```

```
        {
            m[i][j]=m[i][j]+Delta_m[i][j];
            sigma[i][j]=sigma[i][j]+Delta_sigma[i][j];
        }
} /* end of scan */
```

高立人 簡歷

基 本 資 料

中文姓名：高立人

出生日期：58 年 12 月 2 日

現在住址：花蓮市民勤里 11 鄰永安街 105 號

聯絡電話：（03）8234152、0933-996072

現 職

大漢技術學院電腦與通訊工程系助理教授 兼任 圖書暨資訊中心主任

經 歷

2006.02 ~	大漢技術學院電腦與通訊工程系助理教授
1998.08 ~ 2006.02	大漢技術學院電腦與通訊工程系講師
1998.03 ~ 1998.06	中央研究院物理研究所（以部分時間人員身份協助物理所陳啟東教授實驗室開發單電子電晶體之電壓/電流放大器；for the design of Single Electron Current / Voltage Preamplifier）
1996.06 ~ 1998.01	交通部台灣北區電信管理局/中華電信北分公司高級技術員
1995.11 ~ 1996.06	工研院電通所（碩士班就學期間；擔任定期契約人員；負責萬用遙控/URC 發展系統之開發）
1993.08 ~ 1995.07	國立交通大學控制工程學系專任助教

教 育 程 度

2000.9 ~ 2000.09	國立交通大學電機與控制工程學系博士班
1995.9 ~ 1997.12	國立交通大學電機與控制工程學系碩士班
1987.9 ~ 1991.06	國立交通大學控制工程學系

考 試、證 照

1. 八十四年特種考試交通事業電信人員考試高員級及格。
2. 高級電信工程人員資格證。
3. BS7799 / ISO 27001 LA（Lead Auditor；ISO 27001 資訊安全認證主導稽核員）

校 內 服 務

1. 兼任圖書暨資訊中心主任（97 學年度起）
2. 兼任電子計算機中心主任兩年六月（94 學年度第 2 學期起 至 96 學年度）
3. 兼代系/科主任三年（88、92、93 學年度）

校 外 服 務

1. 花蓮縣政府『2007 資訊網路行銷委託專業服務案』評審委員（**Jun. 26, 2007**）
2. 花蓮縣政府『2007 公務文化創意發表會』評審委員（**Apr. 27, 2007**）
3. 擔任『2006 全國觀光休閒盃網頁設計大賽』評審委員（**Nov. 24 ~ Dec. 16, 2006**）
4. 花蓮縣環境保護局『移動污染源資訊網站維護計畫』評審委員（**1999, 2000**）

學 術 服 務

1. Reviewer, IEEE Trans. Circuits and Systems I（Regular Paper）.
2. Reviewer, IEEE Trans. Circuits and Systems II（Express Brief）.
3. Reviewer, IEEE Signal Processing Letters.
4. Reviewer, IEEE Trans. Systems, Man and Cybernetics, Part B.
5. Reviewer, EURASIP Journal on Advances in Signal Processing
6. Reviewer, TANET2006（2006 網際網路研討會）
7. Session chair and preparatory committee, TANET2006（2006 網際網路研討會）

榮　譽

全國性競賽：

1. 指導學生參加教育部『八十九學年度通訊專題製作競賽』晉級複賽，並獲得教育部三萬元獎助。該論文收錄於『第四屆教育部通訊專題競賽入選論文集』。
2. 所指導學生之專題『校園電子消費系統』獲得『八十九學年度全國大專專題製作競賽』佳作。
3. 指導學生參加教育部『八十八學年度通訊專題製作競賽』晉級複賽，並獲得教育部三萬元獎助。該論文收錄於『第三屆教育部通訊專題競賽入選論文集』。

學術類：

1. 榮獲九十二學年度（**92.07.14**）國立交通大學電機資訊學院博士班『朱順一合勤獎學金』。
2. 八十九學年度以第一名成績進入交通大學電機與控制工程學系博士班就讀。
3. 國立交通大學八十四學年度第一學期書卷獎。

教學服務類：

1. 當選「大漢技術學院 95 學年度第 2 學期教學優良教師」（Oct. 29, 2007）
2. 當選「96 學年度花蓮縣新城鄉資深優良教師」（Sep. 27, 2007）
3. 當選「大漢技術學院 95 學年度第 1 學期教學優良教師」（Jun. 4, 2007）
4. 當選「大漢技術學院 94 學年度第 2 學期教學優良教師」（Sep. 14, 2006）

他項榮譽：

1. 榮獲『花蓮縣洄瀾願景 2010 徵文競賽優等獎』（花蓮縣政府；Nov. 5, 2006）。
2. 榮獲列名未來領袖名人錄『**Marquis Who's Who of Emerging Leaders, 2007**』， **the 1st Edition**。
3. 榮獲列名 2007 年亞洲名人錄『**Marquis Who's Who in Asia, 2007**』，**the 1st Edition**。
4. 榮獲列名 2006-2007 年科學與工程名人錄『**Marquis Who's Who in Science and Engineering, 2006-2007**』，**the 9th Edition**。
5. 榮獲當選『花蓮縣第五屆 e 世代未來領袖暨優秀青年代表』（花蓮縣科技青年會，2006 年）。
6. 榮獲列名 2005 年世界名人錄『**Marquis Who's Who in the World, 2005**』，**the 22nd Edition**。
7. 榮獲列名 2005-2006 年科學與工程名人錄『**Marquis Who's Who in Science and Engineering, 2005-2006**』，**the 8th Edition**。

（資 料 備 查）

List of Publications（著作表列）

姓名（中文）：高立人
姓名（英文）：Lih-Jen Kau

[A] Journal Papers（期刊論文集）

[1] **Lih-Jen Kau** and Yuan-Pei Lin, "Least Squares-Based Switching Structure for Lossless Image Coding," *IEEE Trans. Circuits and Systems I*, pp. 1529-1541, Vol. 54, No. 7, July 2007.

[2] **Lih-Jen Kau**, Yuan-Pei Lin and Chin-Teng Lin, "Lossless Image Coding using Adaptive, Switching Algorithm with Automatic Fuzzy Context Modeling," *IEE Proc. Vision, Image & Signal Processing*, pp. 684-694, Vol. 153, No. 5, Oct. 2006.

[3] **Lih-Jen Kau** and Yuan-Pei Lin, "Adaptive Lossless Image Coding using Least Squares Optimization with Edge-look-ahead," *IEEE Trans. Circuits and Systems II*, pp. 751-755, Vol. 52, No. 11, Nov. 2005.

[4] Ching-Hung Lee, **Lih-Jen Kau**, and Yuan-Pei Lin, "Enhancing the Predictive Coding Efficiency with Control Technologies for Lossless Compression of Images," submitted to *IEEE Trans. Circuits and Systems I*.

[B] Conference Papers（會議論文集）

[1] **Lih-Jen Kau** and Yuan-Pei Lin, "Least Squares-Adapted Edge-look-ahead Prediction with Run-Length Encodings for Lossless Compression of Images," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP 2008)*, pp.1185-1188, Las Vegas, Nevada, U.S.A., Mar. 30 - Apr. 4, 2008.

[2] **Lih-Jen Kau** and Yuan-Pei Lin, "Least Squares-Based Lossless Image Coding with Edge-look-ahead," in *Proc. IEEE International Symposium on Circuits and Systems (IEEE ISCAS 2006)*, pp.4931-4934, Island of Kos, Greece, May 21- 24, 2006.

[3] **Lih-Jen Kau** and Yuan-Pei Lin, "Lossless Image Coding using a Switching Predictor with Run-Length Encodings," in *Proc. IEEE International Conference*

on Multimedia and Expo (IEEE ICME 2004), pp.1155-1158, Taipei, Taiwan, R.O.C., June 27–30, 2004.

[4] **Lih-Jen Kau**, "Lossless Image Coding using Adaptive Predictor with Automatic Context Modeling," in *Proc. IEEE International Conference on Electronics, Circuits and Systems (IEEE ICECS 2003)*, pp. 116-119, Sharjah, United Arab Emirates, Dec. 14-17, 2003.

[5] **Lih-Jen Kau** and Yuan-Pei Lin, "A Switching Predictor for Lossless Image Coding," in *Proc. IEEE International Conference on Systems, Man and Cybernetics (IEEE SMC 2003)*, pp. 228-233, Washington, D.C., USA. Oct. 5–8, 2003.

[6] **Lih-Jen Kau**, "Adaptive Predictor with Dynamic Fuzzy K-Means Clustering for Lossless Image Coding," in *Proc. IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2003)*, pp. 944-949, St. Louis, MO, USA, May 25-28, 2003.

[7] **Lih-Jen Kau**, "A Look Up Table Based Real Time Color Classification System," in *Proc. International Symposium on Nonlinear Theory and Its Applications (NOLTA 2002)*, pp. 759-762, Xi'an, PRC, Oct. 7-11, 2002.

[C] Technical Reports（技術報告及其他）

[1] **高立人**, 董恩家, 黃懷慶, 楊馥華, 楊曜誠, 陳志銘, 陳威樵, "無線視訊遙控車," 教育部八十九學年度通訊專題製作競賽入選論文集, pp. 67-84, May 2001

[2] **高立人**, 謝金寸, 莊正吉, 陳宏杰, 張坤隆, "校園電子消費系統之實做," 教育部八十八學年度通訊專題製作競賽入選論文集, pp. 393-398, May 2000