Noncausal predictive lattice modeling for image compression.

Raymond Chi Ho. Chan

University of Windsor
INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1348 USA
800-521-0800

UMI
Noncausal Predictive Lattice Modeling
for Image Compression

by

Raymond C. H. Chan

A Thesis
Submitted to the College of Graduate Studies & Research through the
Faculty of Engineering - Electrical Engineering in Partial Fulfillment
of the Requirements for the Degree of
Master of Applied Science
at the
University of Windsor

Windsor, Ontario, Canada
December, 1997
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author’s permission.

L’auteur conserve la propriété du droit d’auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-52451-5
Dedicated with love
to my parents,
Abstract

The objective of this thesis is to investigate the lattice modeling technique on the application of image compression. Auto-Regressive (AR) modeling is a classical model for linear prediction. To obtain the model parameters based on the Least Square Error (LSE) criterion, lattice algorithms provide an efficient and robust approach. With prediction coding as the framework, an accurate model or prediction can yield a better reduction in entropy or redundancy. As a result, coding on the less well correlated prediction residue will result in a compression. The characteristics of the prediction residue have also benefited the quantization process. For this purpose, Vector Quantization and various alternative approaches have been studied. Different from time domain, a 2-D image has already contained all the information when we intend to process it, and therefore, it allows a noncausal support plane. To avoid the synthesis problem that encountered for noncausal support, this thesis presents two alternative methods: Binary Decomposition and Matrix Method. Various aforementioned methods have been implemented and tested using Matlab and C++. Comparisons of results show the competitiveness and effectiveness of the noncausal lattice modeling for image compression.
**Acknowledgments**

I would like to express my sincere gratitude to my thesis advisor Dr. H. K. Kwan for his invaluable guidance, discussions and comments, and constant encouragement throughout the progress of this thesis. His insightful suggestion on the use of noncausal 2-D lattice modeling for image compression has been proved to be fruitful in the thesis. I would also like to thank him for going through the first draft of the thesis in short notice and for making various suggestions to improve the presentation of the thesis.

I would also like to thank my department reader, Prof. P. H. Alexander, and my outside department reader, Dr. Y. G. Park for their suggestions and comments. In addition, I would like to thank Prof. Alexander for advice to enhance the thesis presentation.

In addition, I would like to sincerely thank all my friends who have encouraged and supported me in the course of my graduate studies, especially, Tracy Li, Steve and Stella Wong, Sam Tam and Tim Demonte, Alex Chan, Bryan Au and Eddy Cheung. Thanks are also going to my dear fellows, Walter Jin and Wayne Chiang, for their encouragement and friendship in this journey. They all have provided me lots of inspirations and motivations.

Last but far from least, my very special thanks must be given to my family and particularly my parents for their patience and wholehearted support, and for standing by me unconditionally. Without them, none of this could be possible.
# Table Of Contents

1. **INTRODUCTION** ................................................................................................................. 1  
   1.1 *INTRODUCTION* .............................................................................................................. 1  
   1.2 *THE MOTIVATION* ......................................................................................................... 1  
   1.3 *THESIS OBJECTIVES* .................................................................................................... 4  
   1.4 *THESIS ORGANIZATION* .............................................................................................. 4  

2. **SURVEY ON SIGNAL MODELING** .................................................................................... 6  
   2.1 *INTRODUCTION* .............................................................................................................. 6  
   2.2 *1-D MODELING* .............................................................................................................. 6  
      2.2.1 *A linear prediction.* .................................................................................................. 6  
      2.2.2 *Least square error* ................................................................................................... 9  
      2.2.3 *Lattice structure* ...................................................................................................... 10  
      2.2.4 *Lattice structure on AR and ARMA modeling* ......................................................... 12  
      2.2.5 *Advantages of 1-D lattice method* ......................................................................... 15  
   2.3 *2-D MODELING : LATTICE APPROACHES* ................................................................... 15  
   2.4 *2-D AR NONCAUSAL MODELING* ................................................................................ 20  

3. **LATTICE MODELING ALGORITHMS** .............................................................................. 21  
   3.1 *BACKGROUND* .............................................................................................................. 21  
      3.1.1 *Modeling* ................................................................................................................ 21  
      3.1.2 *Lattice Algorithm* .................................................................................................. 27  
      3.1.3 *Entropy coding* ....................................................................................................... 30  
   3.2 *2-D LATTICE ALGORITHMS : A SIMPLE 2-D LATTICE STRUCTURE* ................. 35  
      3.2.1 *Introduction* ............................................................................................................ 35  
      3.2.2 *A simple 2-D lattice structure* ............................................................................... 36  
      3.2.3 *The reflection coefficients* ..................................................................................... 40
**List of Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-D</td>
<td>One-dimensional</td>
</tr>
<tr>
<td>2-D</td>
<td>Two-dimensional</td>
</tr>
<tr>
<td>AR</td>
<td>Autoregressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>Autoregressive-moving-average</td>
</tr>
<tr>
<td>bpp</td>
<td>Bits per pixel</td>
</tr>
<tr>
<td>GLA, GL algorithm</td>
<td>General Lloyd Algorithm</td>
</tr>
<tr>
<td>MMSE</td>
<td>Minimum mean square error</td>
</tr>
<tr>
<td>LSE</td>
<td>Least square error</td>
</tr>
<tr>
<td>LVQ</td>
<td>Learning vector quantization</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak signal to noise ratio</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to noise ratio</td>
</tr>
<tr>
<td>SOFM</td>
<td>Self-Organizing Feature Map</td>
</tr>
<tr>
<td>VQ</td>
<td>Vector quantization</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1: A basic lattice block. 12
Figure 2.2: Parker and Kayran AR lattice model. 17
Figure 2.3: Orthogonal 2-D AR lattice structure with 2x2 support plane. 19
Figure 3.1: Picture of modeling. 21
Figure 3.2: Basic lattice block. 29
Figure 3.3: Example of a Huffman tree. 30
Figure 3.4: Arithmetic coding example. 33
Figure 3.5: A prediction on y(0,0). 35
Figure 3.6: A lattice section in signal flow diagram. 38
Figure 3.7: n-stage lattice modeling filter's structure. 42
Figure 3.8: Synthesis lattice structure. 44
Figure 3.9: Support plane mask on the data field. 46
Figure 3.10: 2-D orthogonal support plane. 47
Figure 3.11: 2-D orthogonal lattice structure. 50
Figure 3.12: A noncausal support plane. 54
Figure 4.1: a) Typical distribution of a residue (left), b) The original image (right). 59
Figure 4.2: Distribution of a) residue of "Lena" (left), b) original "Lena" (256x256x8). 59
Figure 4.3: Distribution of a) residue of "Goldhill" (left), b) original "Goldhill" (256x256x8). 59
Figure 4.4: Levels of the Non-uniform Quantization. 60
Figure 4.5: Quantization mapping function. 61
Figure 4.6: Distribution of the quantized result. 61
Figure 4.7: General picture of a Vector Quantizer. 63
Figure 4.8: Elements in a VQ. 65
Figure 4.9: General Lloyd Algorithm. 68
Figure 4.10: Original Lena Picture. 69
Figure 4.11: Vector Quantized Result. 70
List of Tables

Table 1.1: Comparison of quarter-plane, half-plane and noncausal full-plane residues. 2
Table 3.1: Huffman code for the Huffman tree. 31
Table 3.2: Element of eq (3.34). 41
Table 3.3: Comparison of Simple Lattice and Orthogonal Lattice. 52
Table 4.1: Comparison between VQ and SOFM. 76
Table 6.1: Lossless compression result of 8bpp images. 107
Table 6.2: Lossy lattice method comparison. 108
Table 6.3: Comparison with JPEG. 110
Chapter 1

1. Introduction

1.1 Introduction

Noncausal predictive lattice modeling on image compression can be divided into three parts. It consists of a linear prediction, a 2-D AutoRegressive (AR) modeling and an efficient lattice structure. The linear prediction and AR modeling are inherently embedded with each other. They have a long history on audio compression. As a result of the motivation on the success in 1-D compression, application of 2-D noncausal predictive lattice modeling on image compression is being attempted here.

1.2 The Motivation

In order to convey the reasons behind the choice of noncausal predictive lattice modeling, several questions are posed.

Why predictive coding?

Predictive coding for data compression has a long history. Famous examples are, for example, DPCM and JPEG lossless mode [1]. Predictive coding has been very successful
in speech compression, recognition and synthesis applications. It is one of the popular methods to approach the problems of compression. Predictive coding has many varieties. Differences between these methods are: the support plane constructs, the algorithms of model parameter estimation, etc. The principle is based on the assumption that the values of a signal are closely related to each other (correlated) in a local vicinity, and therefore, redundancy exists.

**Why use a noncausal support plane rather than quarter-plane or half-plane?**

The decision on using full-plane support lattice model on image compression can be treated in two stages. Firstly is the question of whether it should be quarter-plane, half-plane or full-plane support. Secondly, why using lattice structure. About the comparison between quarter-plane, half-plane or full-plane support, let us examine the following result from compression estimation.

<table>
<thead>
<tr>
<th>picture</th>
<th>size</th>
<th>compression (bpp)</th>
<th>prediction PSNR</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Quarter</td>
<td>Half</td>
<td>Full</td>
<td>Quarter</td>
</tr>
<tr>
<td>Boat</td>
<td>512x512</td>
<td>4.76</td>
<td>4.75</td>
<td>4.08</td>
</tr>
<tr>
<td>GoldHill</td>
<td>256x256</td>
<td>5.57</td>
<td>5.56</td>
<td>5.20</td>
</tr>
<tr>
<td>GoldHill</td>
<td>512x512</td>
<td>4.97</td>
<td>4.96</td>
<td>4.49</td>
</tr>
<tr>
<td>Lena</td>
<td>256x256</td>
<td>4.92</td>
<td>4.90</td>
<td>4.29</td>
</tr>
<tr>
<td>Mandrill</td>
<td>512x512</td>
<td>6.24</td>
<td>6.22</td>
<td>5.84</td>
</tr>
<tr>
<td>Mountain</td>
<td>640x480</td>
<td>6.83</td>
<td>6.81</td>
<td>6.70</td>
</tr>
<tr>
<td>Peppers</td>
<td>512x512</td>
<td>4.87</td>
<td>4.76</td>
<td>4.53</td>
</tr>
</tbody>
</table>

The experiment is conducted using a lossless compression estimation. All images used were 256 gray levels images (8 bits per pixel). The images were first being predicted by models with different support planes. The residues (prediction errors) were obtained and were rounded up to the nearest integer. This quantization loss was the only loss. The residues were then entropy encoded. The result was put in bits per pixel (bpp). To indicate the predictive power, the residues were treated as the prediction noise and corresponding PSNR were measured. Since it is a least squares prediction, the variances can indicate the square errors of the predictions.
It can be seen from the result that the full-plane prediction is much superior to the others. In terms of square of error, full-plane variance is the least which show much better minimization of the prediction error. In the event of PSNR values, these indicate how well the predictions can be. Even the values are varying from picture to picture; the full-plane support model shows consistent better performance. It may be due to the fact that quarter-plane and half-plane models are dealing with the extrapolation problem while the noncausal support plane model is an interpolation problem. Basically, the entropy compression performance varies as the distribution of the data; the lower square error would not impose any benefit to the entropy coding. On the other hand, when lower square error is obtained, it also indicates a better prediction. The error will be more concentrated on the center zero. As a result, it can be seen that the entropy compression gets a better result as the prediction is more accurate (lower variance or PSNR in the table).

**Why lattice method?**

Using the lattice method on 2-D modeling is considered because of the success in 1-D lattice structure applications. In 1-D lattice, robustness to finite word length criterion, efficient computation, recursive formulae, simple stability criteria, efficient hardware implementation, etc. are the motivation to extend its use from 1-D to 2-D lattice modeling.

The full-plane support model is superior; however, the synthesis is a problem. Since the full-plane support implies a noncausal model, the synthesis filter will become a non-recursively computable filter. Moreover, the stability problem is also a limitation on modeling & synthesis pair because both modeling and inverse filters need to be stable.
1.3 Thesis Objectives

In summary, the work presented in this thesis bears two main objectives. One is to conduct a research study on the available least square lattice modeling methods for two-dimensional data. Another one is to apply the modeling method into the application of image compression. Certain technical problems have been encountered; two alternative approaches have been suggested.

1.4 Thesis Organization

In Chapter 2, a comprehensive survey is conducted on lattice modeling. It includes the development from the early stage up to the recent research work in the area.

In Chapter 3, various theoretical backgrounds are presented. It includes the modeling basic principles. In the course of the background introduction, several essential concepts are brought out: orthogonality, Yule-Walker normal equation, lattice structure derivation, etc. Moreover, the Huffman coder and arithmetic coder are compared. Besides, two 2-D lattice structures are depicted. Comparisons of these two 2-D lattice modeling algorithms are presented. Algorithms details are presented in this chapter.

In Chapter 4, the quantization issue is being discussed. In this chapter, the main focus is on a popular research subject, Vector Quantization (VQ). It is mainly because of its high compression ratio. Neural network approaches to VQ are discussed in the course.
In Chapter 5, codec (coder-decoder) frameworks for the noncausal modeling are introduced. They are alternatives to the synthesis problem. The two methods include a binary pyramid decomposition method and a matrix approach.

In Chapter 6, a complete detailed structure for the codec was presented again for the sake of integrity. Result comparison is included in this section. In Chapter 7, a conclusion and some future directions on the subject are given.
Chapter 2

2. Survey on signal modeling

2.1 Introduction

In this section, a comprehensive survey on the area of signal modeling is presented. It includes the approaches from the early stage for one-dimensional data up to the current research work for two-dimensional data in this area. Lattice algorithms are focused in this survey because of their attractive characteristics.

2.2 1-D modeling

2.2.1 A linear prediction.

Linear prediction in the analysis of discrete signals is to model the given signal as a linear combination of its past values and present and past values of a hypothetical (e.g. a white noise) input to a system whose output is the given signal. Applying Z-transform, it can be shown this linear combination is equivalent to a transfer function in the Z-domain, which could be further transformed to the frequency domain with a zero-pole spectrum. In other
words, we program the locations of zero-poles to tailor the approximated spectrum to the signal spectrum.

The mathematical analysis of the behavior of general dynamic systems has been an area of concern since the beginning of this century. The analysis of the outputs of dynamic systems is the most part of the concern, which is developed mainly within the fields of statistics, econometrics, and communications. Most of the previous work was actually conducted by statisticians. Statistical terms like Autoregressive (AR) and Autoregressive-moving-average (ARMA) [2] are common in this area of research while "all-pole" filter, "zero-pole" filter are more familiar in the signal processing field. This modeling technique has also been applied in the field of control theory based on state-space concepts and time domain analysis. In the early stage of the applications, they also included: Neurophysics [3], [4] where brain activity was recorded as electroencephalograms (or EEG signals). It was of interest to describe the spectrum in a simple mathematical manner which could characterize the different rhythms for the analysis of the brain activity; Geophysics [5], [6] where seismographs (seismic traces) were used to determine the sedimentary rock layer structure; Speech Communication [7] - [10] where the speech signal type determination and the sound identification are of interest. This information could be of benefit to a speech recognition system or a speech compression system.

Recently, predictive coding has been widely used in data compression either for speech or for images [11]. For instance, in telephony, the DPCM has been used in the source coding which reduces the number of bits required for similar quantization quality. DPCM could
be thought as a 1st order prediction which used last history to predict the present value. Instead of storing the original value, the prediction error (residue) is stored. Prediction coding yields the compression based on the assumption that the signal values are closely related to each other (correlated) in local vicinity, and therefore, redundancy exists. Having part of the signal already received or known, the idea of the prediction coding is to avoid storing the excessive information with the present signal value.

Linear prediction is commonly used for signal modeling because it is more mathematically tractable. Signal modeling is a process to estimate parameters of characterizing the signal. It assumes the signal is generated with a white noise source passing through a system. The system is approximated with a transfer function. The estimate of the parameters in the function is the modeling. For AR modeling processes, the functions are in all-poles form, i.e. no numerator parameters are present in the transfer function except a constant. In other words, the AR modeling is to predict its present output value with the history of the output and only the present value of the input. Associated with that is the modeling filter, which will whiten the input signal, i.e. given the signal to be modeled, the modeling filter will output a white noise. Therefore, the modeling filter is also called the whitening filter and it is an all-zero filter for an AR model.
2.2.2 Least square error

The fundamental prediction model is based on an all-pole model, i.e. an AR model. The minimization criteria are mainly based on the error $\varepsilon_n$ measurement between the predicted signal by a linear combination of the past samples, and the desired signal.

$$\varepsilon_n = y(n) + \sum_{i=1}^{N} a_i y(n-i)$$

To determine the parameters used in the obtained model, the least square error (LSE) measure, due to Gauss who first reported on it in the early 1800's, is commonly used. It is used because of its mathematical tractability. This criterion puts greater emphasis on the large errors and less emphasis on the small errors. Moreover, since it is defined purely on time domain, from a spectral modeling point of view, it is hard to say whether such an error measure is an appropriate measure or not. Fortunately, for the linear prediction problem, this error measure could be written in the frequency domain such that we could measure the fitness between the given signal spectrum and the approximated model spectrum [12], [13].

There exist many methods in spectral estimation [14]: Fourier transform; periodogram; maximum likelihood [15], [16]; Burg's maximum entropy method [17], etc. The last two methods are closely related and are used in the linear prediction model. Among different methods in solving the model parameters, a class of lattice methods [18] is popular in this area. To obtain the linear prediction model parameters, these lattice methods involve solving the Yule Walker normal equations.
\[
\begin{bmatrix}
R_0 & R_1 & \cdots & R_N \\
R_{-1} & R_0 & \cdots & R_1 \\
\vdots & \vdots & \ddots & \vdots \\
R_{-N} & \cdots & R_{-1} & R_0
\end{bmatrix}
\begin{bmatrix}
1 \\
\alpha_1 \\
\vdots \\
\alpha_N
\end{bmatrix}
= E(\epsilon_N^2)
\]

With the statistics point of view, these normal equations can be arranged into a matrix form - the leftmost square matrix is the famous auto-correlation matrix [14], [19], [20]. Nevertheless, in practical cases, signals obtained are of finite samples. Auto-correlation is available by an estimate or an approximation only. This leads to the different estimate methods associated with it: auto-correlation method, covariance method, pre-windowing and post-windowing [14], [19]. The latter three methods attempt to approximate a correlation matrix. With the number of samples approaching infinity, they are equivalent to the correlation matrix. Besides the stochastic approach, a deterministic approach has been used which aims exactly on sample data. The direct result from the deterministic approach will be the covariance method. Therefore, as the number of available samples increases, the covariance method will approach the auto-correlation method.

2.2.3 Lattice structure

The beauty of the correlation matrix is its properties of symmetry, positive definiteness, and Toeplitz structure. The former two properties lead to Cholesky decomposition as a possible method solving for parameters. With \( p \) being the number of unknowns in the system of equations to be solved, Cholesky decomposition requires \( \frac{N^3}{6} + O(N^2) \) multiplication and \( \frac{N^3}{2} \) storage location while the general Gauss elimination method
requires double these computation \((N^3/3+O(N^2))\) and storage locations \((N^2/2)\) [21]. With all the three properties, the Levinson-Durbin [22] algorithm can be applied, which only requires \(O(N^2)\) computation and \(O(N)\) storage locations [20]. This algorithm also leads to the lattice structure on the implementation. Figure 2.1 shows a basic lattice block which represents one order increment in a filter. They introduce the reflection coefficients or PARtial CORrelation (PARCOR) instead of the canonical parameters. In [18], it reveals a stable and efficient lattice structure with guaranteeing the stability of the all-pole filter. Nonetheless, it should be noted that, as mentioned in [19], many of the lattice methods are actually approximations to the stationary least squares solution. Nonetheless, one of the main attractions of lattice filters is the fact that lattice filters tend to be more robust with respect to quantization and round-off noise. Another advantage is that, the \(N\)th order lattice filter has contained all the lattice filters of order \(i<N\). It is beneficial when the correct filter order is not known \textit{a priori} since all the previous overhead on the obtained \(N-I\)-th order lattice filter are useful for the \(N\)-th order lattice filter. This property is not shared with canonical realizations where one additional degree of freedom in the model formula will result in a complete recomputation of all the parameters subject to the least square error (LSE) criterion.
2.2.4 Lattice structure on AR and ARMA modeling

Auto-Regressive and Moving Average (ARMA) models are recognized as superb filters in speech analysis [23] because they can represent both the concentration and the dispersion of power in a spectrum while AR models represent the concentration only. Although theoretically AR models can describe any spectrum with infinite order [14], [19], its inherent lack of programmable zeros makes it practically inferior to ARMA model in modeling performance. Most of the work, however, has been focused on the development of the computationally efficient least squares algorithm for AR model. One of the difficulties with the ARMA lattices is that their non-linear nature is difficult to tackle. Many approaches are extended from AR modeling tools. Different approximations or different manipulations will make different algorithms. As a result, no general ARMA lattice filter has been defined as was done for the AR lattice.
In the AR lattice, from the LSE criterion, it can be shown that the innovation (prediction error) sequence processes orthogonal property to the past of the given signal. This also asserts the orthogonal principle [2], [14]. With the orthogonal principle, the square error minimization process can be interpreted as a projection problem in some inner product space. The AR lattice can be shown as undergoing a Gram-Schmidt orthogonalization process [19], [24], [25] where each stage of the lattice orthogonalizes the innovation sequence with respect to all the previous stage outputs.

ARMA modeling, on the other hand, did not developed as thoroughly as the AR model. It is because the problem of least square filtering and modeling of the ARMA process is intrinsically nonlinear and becomes more complex in nature. Many of the algorithms used in the ARMA model utilize the AR model facility, i.e. extending the algorithm in an AR model to be an ARMA modeling algorithm. Lee et al. [26] have proposed a series of ladder realizations for ARMA processes. The idea is based on a joint innovations representation of the input and output of a system. In other words, it vectorizes the given system input and output to form a 2-channel AR process. This ARMA model is embedded into a 2-channel AR model. In this ARMA lattice filter, the AR order will equal the MA order. Benveniste and Chaure [25] have developed a more general ARMA lattice structure based on a stochastic formulation of the ARMA lattice modeling problem. But in both cases, the ideas are also to vectorize the ARMA model into a AR model. From [27], Lim and Parker have shown an alternate two-channel AR lattice approach for ARMA modeling which eliminates the necessity of separate computing of a term in the vectorized AR systems. As a result from using AR modeling tools, the lattice structures are not as
efficient as they could be [24], [28], [29]. And it is noticed that this is equivalent to the extended least-squares approximation [30]. Furthermore, a AR model characteristic which will carry all the optimal filters of lower orders has not been carried over to these forms of ARMA lattice structure.

Karlsson and Hayes [24] have developed a new ARMA lattice which can have different AR and MA orders. Moreover, it is fully consistent with the geometric characteristics of the optimal AR and MA lattice filter structures of lower orders. In other words, it evaluates all optimal ARMA lattice filters of lower order. The algorithm is based on the concept of projection error sample averaging. It begins by formulating the modeling problem as a projection problem in Hilbert space. Similarly with the projection error concept and orthogonal principle, Prasad and Joshi [28], [29], have also developed algorithms for ARMA model. They has defined a projector space representation for their development. Lately, Monin and Salut [30] have proposed an exact ARMA lattice predictor. The term "exact" meant it was not based on the extended least-square approach. On the other hand, its principle was also based on the Gram-Schmidt orthogonalization in some inner product space. All methods mentioned above have derived different structures in their lattices. Nevertheless, no general ARMA model is being defined and commonly recognized.
2.2.5 Advantages of 1-D lattice method.

- Efficient; Levinson-Durbin recursion only requires $O(N^3)$ computation while using matrix inversion needs $(N^3/3+O(N^2))$.
- Robust under finite word length condition.
- Guaranteed stable with simple condition applied.
- Identical block structure which is easier for hardware implementation.

2.3 2-D modeling: lattice approaches

A fundamental approach to the use of reflection coefficient concept in the modeling of 2-D fields has been proposed by Marzetta [31]. He has developed a class of 2-D minimum-phase prediction error filters (PEF) using a sequence of reflection coefficients. Compared to 1-D lattice modeling, an order update (order increment) requires one new point (support) in 1-D case while this 2-D minimum-phase class order update requires $O(p)$ new points, where $p$ is the model order. The technique used is to rearrange the 2-D fields into one dimension. This approach retains several desired properties in 1-D lattice such as correlation matching and minimum phase. However, this requires the half-plane support filter to have infinite support in one of the two dimensions. It leads to a filter with long delays. Nonetheless, the fundamental advantage of working in this reflection coefficient domain is that the minimum-phase condition is automatically guaranteed merely by constraining the magnitudes to be less than one. On the other hand, if the PEF has
discontinuous support, then it is not generally guaranteed to be minimum phase. The advantage of being minimum phase has a significant implication in the synthesis process. When the PEF is minimum-phase, the inverse filter for the synthesis will be stable [32], and vice versa, whereas, owing to its continuous support requirement, it is more attractive in literature than practical application.

In a different approach, Parker and Kayran [33] have extended the theory of one-dimensional lattice AR modeling in a natural way to two-dimensional fields. They have presented the quarter-plane support AR model as well as the asymmetric half-plane AR model. For the quarter-plane model, the developed lattice structure has three parameters per stage. Instead of having only two-input-two-output ports as the one-dimensional lattice block, this 2-D prediction error lattice block has four-input and four-output error fields. An order update is an additional cascade of a new lattice block at the end of the filter. See Figure 2.2
They also presented the relationships between the lattice parameters and the quarter-plane two-dimensional transfer function (canonical) coefficients. Besides the modeling, the development of the synthesis filter was shown for regenerating two-dimensional fields from a random field signal input. This lattice filter minimization, which could be programmable for a particular error among four or sum of all errors, is performed locally stage by stage. Another characteristic of the filter structure is that each order update, i.e. one additional stage, will introduce many points of support but with only three programmable parameters (degrees of freedom). One of the attractive traits of this proposed filter structure is the simplicity in computation, including both parameters estimation and actual predictor error filtering. Moreover, the identical lattice block structure in every stage makes it easier in filter implementation for both hardware or software.
Simultaneously introducing many points into the support while only three parameters being available during an order update, has been criticized by Bose [34] that it lacks sufficient parameters to represent all classes of 2-D AR quarter-plane filters. It can be seen from the degrees of freedom available for the minimization. The number of points in the support plane means the same number of canonical parameters should be presented in the AR model. The support plane size is incremented as 2x2-1, 3x3-1, 4x4-1, etc. as the order increases in steps of one. Whereas, the corresponding available parameters will be 3, 6, 9, 12, ..., and so on. It could be seen that only the 1st order PEF will have the same available number of parameters, i.e. 3. It follows by 8 to 6, 15 to 9, 24 to 12, etc. As a result, as the filter order increases, the ability for modeling the same support plane will decrease dramatically in the point of view of degree of freedom. In addition to this deficiency, it has also been reported the lack of orthogonality. Hence, the cascading of stages may not lead to an optimum filter. However, its simplicity is really attractive. Moreover, good results in image compression have been reported using this approach [35]-[37].

With a different approach than the aforementioned model, Kayran [38] has developed another two-dimensional orthogonal lattice for AR modeling. The method used is similar to Lenk and Parker [39] who have shown a method to extend the 1-D Levinson algorithm into the 2-D case. They are similar in the way that they rearranged a two-dimensional field into a one-dimensional vector followed by obtaining a correlation matrix as the 1-D case did. Burg's matrix formulation is used to derive the Levinson-type recursion on this correlation matrix. Kayran has also established a stability condition for this 2-D model. The lattice structure contains the lattice blocks which are connected in a triangular shape
instead of a pipeline structure as 1-D lattice (See Figure 2.3). For one additional parameter (point of support), increment of the number of lattice block will be a triangular number, i.e. if there is a 3-3-3 triangle, the next order update will become a 4-4-4 triangle, four additional lattice blocks on the diagonal line are needed. Despite the complexity of the algorithm, this algorithm gives the solution of the normal equations by those recursive formulae. Therefore, it could really represent the canonical two-dimensional AR model. The total number of lattices used is $M^*(M+1)/2$, where $M$ is the number of points in the support plane, i.e. $M=N_1*N_2-1$. It is the simplest lattice structure thus far. However, it will require $O(M^2)$ parameters for a AR process of $O(M)$ parameters.

![Figure 2.3: Orthogonal 2-D AR lattice structure with 2x2 support plane.](image)

About the ARMA model, Kayran has proposed using the two-channel AR lattice approach [40]. The idea is similar to those in the one-dimensional lattice literature [25], [26]. Moreover, he has introduced the method in [27] to this two-dimensional model such that
the algorithm eliminates a separate calculation of a term. In addition, Kayran has tailor-made a new generator matrix for a 2-D Schur algorithm in his AR model.

2.4 2-D AR noncausal modeling

For images, the region of support can be causal, semi-causal or even noncausal. Unlike the time domain 1-D case, images are defined in a spatial domain and therefore, causality is not a rigid limitation for the modeling process. A noncausal model has merit in the image modeling. For a causal or semi-causal model (quarter-plane, half-plane), we cannot utilize all the adjoining pixels for the estimation. On the other hand, correlational information is extracted from as many surrounding pixels as possible. Thus more correlational information will be reflected in the model parameters. Naturally speaking, a noncausal model will be expected to have better description power over the causal model in image applications [41], [42].

Zhao and Yu [42] have shown a method to deal with the noncausality. The main idea is to split the full-plane support into two asymmetric half-planes, namely nonsymmetric half-plane (NSHP) causal and NSHP anti-causal support. The minimization is on the total sum of squares of these prediction errors. Hasan and Azimi-Sadjadi [41] have introduced a descriptor approach for the noncausal models.
3. Lattice modeling algorithms

3.1 Background

This section gives the brief introduction to the theories and concepts of modeling, lattice algorithms and entropy coding.

3.1.1 Modeling

3.1.1.1 Deterministic Approach

We first use a deterministic approach. It can be shown that this yields the covariance and correlation methods in statistical approach.

Consider the causal AR filter model,

$$H(z) = \frac{1}{A(z)}$$ ................................................................. (3.1)

where

$$A(z) = a_0 + a_1 z^{-1} + \cdots + a_N z^{-N}$$ ................................................................. (3.2)

Then $H(z) A(z) = 1$, or, in time domain,
\[ h(n) * a_n = \delta(n) \] .......................................................... (3.3)

The FIR filter \( A(z) \) is thus the inverse filter for \( H(z) \), which whitens \( h(n) \) to produce \( \delta(n) \), as depicted in Figure 3.1. Since \( A(z) \) has order \( N \), we call it a support with \( N \) order. Using \( 2N + 1 \) consecutive values for \( h(n) \) beginning at \( n = -N \), (3.3) provides \( N + 1 \) linear equations from which to solve for \( a_n \), i.e., given \( h(0) \) through \( h(N) \) with the fact that \( h(n) = 0 \), for \( n < 0 \), the convolution may be written in matrix form to produce the following \( N + 1 \) equations for \( a_n \):

\[
\begin{bmatrix}
  h_0 & 0 & \cdots & 0 \\
  h_1 & h_0 & \ddots & \vdots \\
  \vdots & \ddots & \ddots & 0 \\
  h_N & \cdots & h_1 & h_0
\end{bmatrix}
\begin{bmatrix}
  a_0 \\
  a_1 \\
  \vdots \\
  a_N
\end{bmatrix}
=
\begin{bmatrix}
  1 \\
  0 \\
  \vdots \\
  0
\end{bmatrix}
\] .......................................................... (3.4)

Nevertheless, given a desired or measured impulse response \( h_d(n) \) that is not exactly AR and/or of greater than \( N \)th order, we can only approximate it. Therefore, (3.3) becomes

\[ h_d(n) * a_n = \delta(n) + e(n) \] .......................................................... (3.5)

where \( e(n) \) is the approximation error, and \( A(z) \) is only an approximate inverse filter as illustrated in Figure 3.1. Similar to (3.4), we write (3.5) into matrix form, we have

\[ H_d \mathbf{a} = \delta + e \] .......................................................... (3.6)

where

\[ H_d = \begin{bmatrix}
  h_{d0} & 0 & \cdots & 0 \\
  h_{d1} & h_{d0} & \ddots & \vdots \\
  \vdots & \ddots & \ddots & \vdots \\
  h_{dN} & \cdots & h_{d,N-1} & h_{d0} \\
  \vdots & \ddots & \ddots & \vdots \\
  h_{dl} & \cdots & \cdots & h_{d,L-N}
\end{bmatrix}, \]
\[
a = [a_0, a_1, \ldots, a_N]^T, \ \delta = [1, 0, \ldots, 0]^T \text{ and } e = [e_0, e_1, \ldots, e_N]^T.
\]

Note that we have assumed more data samples \( h_d(n) \), for \( n=0,1,\ldots,N \), than the minimum required (i.e., \( L>N \)) in order to approximate \( h_d(n) \) closely over more than the minimum interval. Or, in other words, we can assume the available data for the approximation is up to \( N \). Applying the least-square error criterion, we are about to minimize

\[
E_N = e^T \cdot e = \sum_{n=0}^{L} e_n^2 \tag{3.7}
\]

This is the standard problem in least-squares estimation with over-determined equations.

From (3.6), (3.7) becomes

\[
E_N = (H_d a - \delta)^T(H_d a - \delta)
\]

\[
= (a^T H_d^T - \delta^T)(H_d a - \delta)
\]

\[
= a^T H_d^T H_d a - 2 a^T H_d^T \delta + \delta^T \delta
\]

To minimize \( E_N \), we put the corresponding partial derivatives to zero, i.e. in vector form,

\[
\frac{\partial E_N}{\partial a} = 0
\]

which gives

\[
2 H_d^T H_d a - 2 H_d^T \delta = 0
\]

or simply

\[
H_d^T H_d a = H_d^T \delta \tag{3.8}
\]

Equation (3.8) is the desired normal equation for the LSE solution \( a \). Note that if we pre-multiply both sides of (3.6) by \( H_d^T \), we have

\[
H_d^T H_d a = H_d^T \delta + H_d^T e \tag{3.9}
\]

Comparing (3.8) and (3.9), it implies
\[ H_d^T e = 0 \]  \hspace{1cm} (3.10)

### 3.1.1.2 Concept of orthogonality

Equation (3.9) is an example of the famous orthogonality principle of least squares estimation, which states that the error vector \( e \) corresponding to the minimizing solution \( a \) must be orthogonal to each of the columns of \( H_d \) and thus to the subspace spanned by the columns of \( H_d \). Now, if we assume the signal that we want to model be \( y(n) \), the columns of \( H_d \) are corresponding to each delayed version of \( y \), i.e. \( y(n-1), y(n-2), \ldots, y(n-N) \). In other words, in terms of predictive modeling, they stated the prediction error (residue) is orthogonal to \( y \) within the support plane. It has been shown that this orthogonality is a sufficient condition for least squares measure. Therefore, with the criterion of orthogonality fulfilled, the result is a least squares estimation.

### 3.1.1.3 Statistical form

#### 3.1.1.3.1 Covariance method.

We denote

\[ \Phi_N = H_d^T H_d \]  \hspace{1cm} (3.11)

and note also that

\[ H_d^T \delta = h_{d0} \delta \]  \hspace{1cm} (3.12)

Equation (3.8) becomes

\[ \Phi_N a = h_{d0} \delta \]  \hspace{1cm} (3.13)
where $\Phi_N$ is the $(N+1) \times (N+1)$ symmetric covariance matrix. The parameters $a$ can be obtained by (3.13),

$$a = h_{d0} \Phi_N^{-1} \delta.$$ ..........................(3.14)

3.1.1.3.2 Correlation method

As $L \to \infty$, the elements $\phi_{ij}$ of $\Phi_N$ approach the auto-correlation values $R(i-j)$ where

$$R(m) = \sum_{n=0}^{\infty} h_d(n) \cdot h_d(n + m).$$ ..........................(3.15)

Since $h_d(n)$ is real, $R(-m) = R(m)$. Therefore, $\Phi_N$ becomes the symmetric Toeplitz auto-correlation matrix:

$$R_N = \begin{bmatrix}
R_0 & R_1 & \cdots & R_N \\
R_{-1} & R_0 & \ddots & \vdots \\
\vdots & \ddots & \ddots & R_1 \\
R_{-N} & \cdots & R_{-1} & R_0
\end{bmatrix}$$ ..........................(3.16)

Equation (3.13) becomes the famous Yule-Walker equation [19].

$$R_N a = h_{d0} \delta.$$ ..........................(3.17)

3.1.1.3.3 Spectrum matching

The modeling is based on the time domain least squares measurement. On the other hand, it can be shown that as $N \to \infty$, the least square modeling is equivalent to frequency spectra matching. Start by transforming (3.5) to the frequency domain,

$$E(z) = H_d(z) A(z) - 1$$

By Parseval's Theorem, the energy conservation principle, the error norm becomes
\[ E_N = \sum_{n=0}^{\infty} e_n^2 = \frac{1}{2\pi} \int_{0}^{2\pi} |H_d(\omega)A(\omega) - 1|^2 d\omega \] .......................... (3.18)

This shows \( A(\omega) \) whiten \( H_d(\omega) \) in the frequency domain because, from equation (3.6), \( E_N \to 0 \) as the order \( N \to \infty \). When \( N \) is increasing, the parameters dimension can approach the desired system response dimension. The matrix equation will become determined instead of over-determined case. The adjustment error vector \( e \) is not needed anymore.
3.1.2 Lattice Algorithm

Recalling the Yule-Walker equation (3.17), with \( a_0 = 1 \), and rearranging into a row-wise notation,

\[
[1 \ a_1 \ a_2 \ \ldots \ a_N] \ R_N = [R^e \ 0 \ 0 \ \ldots \ 0] \tag{3.19}
\]

The filter equation in (3.1)-(3.3) can be converted into the following notation,

\[
e_N(t) = y(t) + \sum_{i=1}^{N} a_i y(t-i) \tag{3.20}
\]

We call it a forward error \( e_N \). Now we define the backward error \( r_N \) as follow,

\[
r_N(t) = y(t) + \sum_{i=1}^{N} b_{N+1-i} y(t-i) \tag{3.21}
\]

By the same approach as obtaining (3.19), we have

\[
[b_N \ b_{N-1} \ \ldots \ b_1 \ 1] \ R_N = [0 \ \ldots \ 0 \ R^r] \tag{3.22}
\]

where \( R^e = \mathbb{E}(e_N(t) e_N(t)') \) and \( R^r = \mathbb{E}(r_N(t-I) r_N(t-I)') \).

To develop the recursive relationship for the lattice filter, we introduce an additional subscript to the \( a, b, R^e \) and \( R^r \) for noting the order that the canonical coefficients belongs to, e.g. \( a_{N,1} \) denote the parameter \( a_1 \) in a \( N^{th} \) order \( A(z) \).

By grouping (3.19) and (3.22) together and we try the following \((N+1)^{th}\) order solution,

\[
\begin{bmatrix}
I & a_{N,1} & \cdots & a_{N,N} & 0 \\
0 & b_{N,N} & \cdots & b_{N,1} & I
\end{bmatrix} \ R_{N+1} = \begin{bmatrix}
R^e_N & 0 & \cdots & 0 & \Delta_{N+1}^e \\
\Delta_{N+1} & 0 & \cdots & 0 & R^r_N
\end{bmatrix} \tag{3.23}
\]

where
\[
\Delta_{N+1}^r = R_{N+1} + \sum_{i=1}^{N} a_{Ni} R_{N-i+1}
\]
\[
\Delta_{N+1}^r = \sum_{i=1}^{N} b_{Ni} R_{N-i+1} + R_{N-1}
\]

Converting (3.23) into a form of (3.19) & (3.22), it becomes
\[
\begin{bmatrix}
I & -\Delta_{N+1}^r (R_{N}^r)^{-1} \begin{bmatrix} I & a_{N,1} & \cdots & a_{N,N} & 0 \end{bmatrix} \\
-\Delta_{N+1}^r (R_{N}^r)^{-1} & I \end{bmatrix} \begin{bmatrix} \begin{bmatrix} R_{N+1}^r & 0 & \cdots & 0 \end{bmatrix} \\
0 & 0 & \cdots & 0 & R_{N+1}^r \end{bmatrix}
\]

Therefore, the new canonical model parameters of order N+1 is given by:
\[
\begin{bmatrix}
I & a_{N+1,1} & \cdots & a_{N+1,N} & a_{N+1,N+1} \\
b_{N+1,N+1} & b_{N+1,N} & \cdots & b_{N+1,1} & I \\
0 & b_{N,N} & \cdots & b_{N,1} & I
\end{bmatrix}
\]

From (3.24), if we define
\[
K_{N+1}^r = \Delta_{N+1}^r (R_{N}^r)^{-1}
\]
\[
K_{N+1}^e = \Delta_{N+1}^e (R_{N}^e)^{-1}
\]

then (3.24) becomes
\[
\begin{bmatrix}
I & a_{N+1,1} & \cdots & a_{N+1,N} & a_{N+1,N+1} \\
b_{N+1,N+1} & b_{N+1,N} & \cdots & b_{N+1,1} & I
\end{bmatrix} = \begin{bmatrix}
I & -K_{N+1}^r & \begin{bmatrix} I & a_{N,1} & \cdots & a_{N,N} & 0 \end{bmatrix} \\
K_{N+1}^e & I & 0 & b_{N,N} & \cdots & b_{N,1} & I
\end{bmatrix}
\]

(3.25) gives the reflection coefficients of lattice structure. Look closely into the quantity \(\Delta_{N+1}\), it can be interpreted as the cross correlation of the forward and backward prediction errors at one unit of delay. Refer to [19] for a complete review and coverage of the lattice structure. However, (3.26) have revealed the lattice structure already. Let us
take the Z-transform on (3.26), and expand the recursive function. The \((N+1)^{th}\) order transfer function is given by,
\[
\begin{bmatrix}
A_{N+1}(z) \\
B_{N+1}(z)
\end{bmatrix} = \begin{bmatrix}
I & -K_{N+1}' \\
K_{N+1}' & I
\end{bmatrix} \begin{bmatrix}
A_N(z) \\
z^{-1}B_N(z)
\end{bmatrix}
\]
\[
= \begin{bmatrix}
I & -z^{-1}K_{N+1}' \\
K_{N+1}' & z^{-1}
\end{bmatrix} \cdots \begin{bmatrix}
I & -z^{-1}K_1' \\
K_1' & z^{-1}
\end{bmatrix} A_0(z)
\]
\[
\begin{bmatrix}
A_0(z) \\
B_0(z)
\end{bmatrix}
\]

If we treat it as a signal flow system, let \(A_0(z)\) and \(B_0(z)\) as \(Y(z)\), each matrix stage containing \(K\) will form a lattice block. The \(N^{th}\) stage forward error is the \(N^{th}\) order prediction filter residue.

![Figure 3.2: Basic lattice block.](image)

Another attractive feature of lattice structure is the convenience of getting a synthesis filter, i.e. the inverse filter of \(A(z)\). The synthesis filter is just a direction reversion on the forward error signal path.
3.1.3 Entropy coding.

3.1.3.1 Huffman coding

The idea behind Huffman coding is based on the entropy of a given symbol. In other words, it simply uses shorter bit patterns for more common characters, and longer bit patterns for less common characters. The approach is summarized based on the descriptions of references [43] and [44], the latter of which gives a concise and clear explanation.

Huffman code has the desirable property of instantaneous decodability by making use of the concept of a coding tree, an automaton or state machine. The coding procedure may best be explained by an example.

Imagine a set of five symbols, $M = 5$, which are A, E, I, O, U, occurring with the respective probabilities $P_n$, 0.12, 0.42, 0.09, 0.30, 0.07. The construction of a Huffman code is accomplished in the following manner:

3.1.3.1.1 Huffman-tree building procedure

1. The $M$ symbols of a source define $M$ active nodes containing their respective probabilities.
2. Find the two active nodes with the smallest probability, sum them up to make a new active node, and drop the two old nodes from the list of active nodes.

3. Continue the procedure of 2) until there is only one active node left with respective probability of one.

4. To see the code, redraw all the nodes in the form of a tree. Each node either has no branch or two branches, depending on whether it is a terminal node or not. Label the left branches as 0 and right branches as 1 (or vice-versa). The code for a character is the sequence of zeros or ones that lead to it, starting at the node with probability one.

The construction of the tree with the root nodes being the initial M symbols and with the other nodes being their combination according to Huffman's rules is shown in Figure 3.3.

In general, it is possible that the two smallest probabilities might both be original nodes, or both be composite nodes. In the example shown, it happens that the two nodes with smallest probabilities are always an original node and a composite one. The Huffman code for the five symbols is given in Table 3.1.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.12</td>
<td>100</td>
</tr>
<tr>
<td>E</td>
<td>0.42</td>
<td>0</td>
</tr>
<tr>
<td>I</td>
<td>0.09</td>
<td>1011</td>
</tr>
<tr>
<td>O</td>
<td>0.30</td>
<td>11</td>
</tr>
<tr>
<td>U</td>
<td>0.07</td>
<td>1010</td>
</tr>
</tbody>
</table>

Table 3.1: Huffman code for the Huffman tree.

Any string of zeros and ones can now be decoded into an alphabetic sequence. Consider for example the string 1011111010. Starting at the top of the tree we descend through 1011 to I, the first character. Since we have reached a terminal node, we reset back to the
top of the tree, next descending through 11 to O. Finally, 1010 gives U. The string is
decoded to IOU.

Based on the entropy theory, the optimal bit rate is given by:

$$H_{opt} = -\sum P_i \cdot \log_2 P_i$$

In this example,

$$H_{opt} = 1.995 \text{ bits per symbol}$$

$$H_{huff} = (3 \times 0.12 + 1 \times 0.42 + 4 \times 0.09 + 2 \times 0.30 + 4 \times 0.07) = 2.02 \text{ bits per symbol.}$$

The compression by Huffman is inferior to the theoretical case.

The Huffman coder is inferior to the ideal case because it would have been necessary to
use code words with non-integer length, for example the letter I required $-\log_2(0.09) =
3.47$ bits, whereas the Huffman code assigns 4 bits. Nonetheless, it is possible to assign
non-integer length code words to symbols, which is achieved by arithmetic coding.

### 3.1.3.2 Arithmetic coding

Arithmetic coding was invented by Rissanen [45]. While Huffman coding gives a way of
rounding the code words to close integer values and constructs a code with those lengths,
arithmetic coding actually manages to encode symbols using non-integer numbers of bits.
The method is summarized based on reference [44]. In arithmetic coding, an input
message of any length is represented as a real number $R$ in the range $[0,1]$. The longer the
message, the more precision is required for $R$. The code is constructed as follows:
3.1.3.2.1 Arithmetic coding procedure

1. Divide the interval [0,1] into segments corresponding to the M symbols; the segment of each symbol has a length proportional to its probability.
2. Choose the segment of the first symbol in the string message.
3. Divide the segment of this symbol again into M new segments with length proportional to the symbols probabilities.
4. From these new segments, choose the one corresponding to the next symbol in the message.
5. Continue steps 3) and 4) until the whole message is coded.
6. Represent the segment's value by a binary fraction.

![Arithmetic Coding Example](image)

Figure 3.4: Arithmetic coding example.
The code is again illustrated by the same example as in the previous section. The coding of the stream IOUA is shown in Figure 3.4. The interval [0, 1] is divided into five segments; the $i^{th}$ symbol length of a segment is corresponding to $P_i$. We see that the first message character, I, narrows the range of $R$ to $0.3763 < R \leq 0.4033$. This interval is now subdivided into five subintervals, again with lengths proportional to the $P_i$. The second message character, O, narrows the range of $R$ to $0.37630 < R \leq 0.37819$. The U character further narrows the range to $0.3779632 < R \leq 0.37819$ and finally, the A character narrows it to $R=0.37819$. 
3.2 2-D lattice algorithms: A simple 2-D lattice structure

3.2.1 Introduction

In 2-D AR modeling, the quarter-plane support means the current output depends on the linear combination of the past values of both dimensions.

![Figure 3.5: A prediction on y(0,0).](image)

The quarter-plane support is using the values on the crosses in the above figure to predict the value of y(0,0). In mathematical term,

\[
\hat{y}(i, j) = \sum_{m=0}^{M} \sum_{n=0}^{N} a(m, n) \cdot y(i - m, j - n) \quad \text{except the term when both } m \& n = 0.
\]

where \( \hat{y}(i, j) \) is the predicted (or estimated) value of \( y(i, j) \).
\( a(m, n) \) is the parameter of the corresponding history of \( y(i, j) \).

Let \( e(i, j) \) be the error of the actual value compared to the predicted value, therefore,

\[
e(i, j) = y(i, j) - \hat{y}(i, j)
\]

\[
e(i, j) = \sum_{m=0}^{M} \sum_{n=0}^{N} a(m, n) \cdot y(i - m, j - n)
\].............................................(3.27)

with \( a(0,0) = 1 \)
To obtain the autoregressive form, or all-pole form, we could apply the 2-D Z-transform and rearrange the terms. Let $E$ and $Y$ be the Z-transform of $e(i,j)$ and $y(i,j)$ respectively. Therefore,

$$E = \sum_{m=0}^{M} \sum_{n=0}^{N} a(m,n) \cdot z_1^{-m} \cdot z_2^{-n} \cdot Y \tag{3.28}$$

rearranging,

$$Y = \frac{1}{\sum_{m=0}^{M} \sum_{n=0}^{N} a(m,n) \cdot z_1^{-m} \cdot z_2^{-n}} \cdot E \tag{3.29}$$

Equation (3.29) is an all-pole transfer function. If the input $E$ is the Gaussian white noise, then it is called a AR process. For modeling, the canonical parameters $a(m,n)$ are obtained by the minimum mean square error (MMSE) criterion. As long as the process being modeled is an AR process, or the modeling has enough order, the prediction error $E$ will become independent to his past history and will become white noise. A modeling filter performance is therefore measured by how well this whitening process can be done. It can be seen that the 2-D AR quarter-plane model can be treated as past history prediction filter. For the case of a noncausal support plane, the parameters $m$ and $n$ will start from negative values instead of starting from zero for the quarter-plane model.

### 3.2.2 A simple 2-D lattice structure

This model uses a lattice structure to model the canonical transfer function. A new set of parameters, reflection coefficients are introduced. The MMSE criterion is applied stage by stage to obtain the reflection coefficients. Even though the canonical parameters and the reflection coefficients have different degrees of freedom for the same order of the filter,
they could be related each other. This algorithm does not produce a true 2-D model as for
an AR model of order N, the number of canonical coefficients is \((N+1)^2-1\), while the
number of reflection coefficients available is \(3N\).

3.2.2.1 The lattice model

The lattice model can be described by the following matrix equation.

\[
\begin{bmatrix}
    e_{00}^{n+1}(i,j) \\
    e_{10}^{n+1}(i,j) \\
    e_{11}^{n+1}(i,j) \\
    e_{01}^{n+1}(i,j)
\end{bmatrix} =
\begin{bmatrix}
    1 & -k_{10}^{n+1} & -k_{11}^{n+1} & -k_{01}^{n+1} \\
    -k_{10}^{n+1} & 1 & -k_{11}^{n+1} & -k_{10}^{n+1} \\
    -k_{11}^{n+1} & -k_{01}^{n+1} & 1 & -k_{10}^{n+1} \\
    -k_{01}^{n+1} & -k_{11}^{n+1} & -k_{10}^{n+1} & 1
\end{bmatrix} \begin{bmatrix}
    e_{00}^{n}(i,j) \\
    e_{10}^{n}(i-1,j) \\
    e_{11}^{n}(i-1,j-1) \\
    e_{01}^{n}(i,j-1)
\end{bmatrix}
\]

(3.30)

where \(e_{00}^{n+1}(i,j)\), \(e_{10}^{n+1}(i,j)\), \(e_{11}^{n+1}(i,j)\), \(e_{01}^{n+1}(i,j)\) are defined as the first, second, third
and fourth error fields for the \(n\)th lattice stage respectively. At 1st stage, the errors are
initialized as

\[
e_0^0(i,j) = e_{10}^0(i,j) = e_{11}^0(i,j) = e_{01}^0(i,j) = y(i,j)
\]

and the reflection coefficients \(k_{10}^{n+1}\), \(k_{11}^{n+1}\), \(k_{10}^{n+1}\) are calculated by minimizing the mean-
squared prediction error which is defined either as the sum of all mean-squared values of
all four errors of the \((n+1)\)st stage or the mean-squared value of \(e_{00}^{n+1}(i,j)\) alone.

Equation 3.30 is shown in a signal flow diagram in Figure 3.6.
Figure 3.6: A lattice section in signal flow diagram.

3.2.2.2 Relationship with transfer function

Taking the Z-transform of (3.30) yields:

\[
\begin{bmatrix}
E_{00}^{n+1}(z_1, z_2) \\
E_{10}^{n+1}(z_1, z_2) \\
E_{11}^{n+1}(z_1, z_2) \\
E_{01}^{n+1}(z_1, z_2)
\end{bmatrix} =
\begin{bmatrix}
1 & -k_{10}^{n+1} & -k_{11}^{n+1} & -k_{01}^{n+1} \\
-k_{10}^{n+1} & 1 & -k_{01}^{n+1} & -k_{11}^{n+1} \\
-k_{11}^{n+1} & -k_{01}^{n+1} & 1 & -k_{10}^{n+1} \\
-k_{01}^{n+1} & -k_{11}^{n+1} & -k_{10}^{n+1} & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & z_1^{-1} & 0 & 0 \\
0 & 0 & z_1^{-1}z_2^{-1} & 0 \\
0 & 0 & 0 & z_2^{-1}
\end{bmatrix}
\begin{bmatrix}
E_{00}^{n}(z_1, z_2) \\
E_{10}^{n}(z_1, z_2) \\
E_{11}^{n}(z_1, z_2) \\
E_{01}^{n}(z_1, z_2)
\end{bmatrix}
\]

Simplifying,
\[
\begin{bmatrix}
E_{00}^{n+1}(z_1, z_2) \\
E_{10}^{n+1}(z_1, z_2) \\
E_{11}^{n+1}(z_1, z_2) \\
E_{01}^{n+1}(z_1, z_2)
\end{bmatrix} =
\begin{bmatrix}
1 & -k_{10}^{n+1}z_1 & -k_{11}^{n+1}z_1^{-1}z_2^{-1} & -k_{01}^{n+1}z_1^{-1}z_2^{-1} \\
-k_{10}^{n+1} & z_1^{-1} & -k_{01}^{n+1}z_1^{-1}z_2^{-1} & -k_{11}^{n+1}z_1^{-1}z_2^{-1} \\
-k_{11}^{n+1} & k_{01}^{n+1}z_1^{-1} & z_1^{-1}z_2^{-1} & -k_{10}^{n+1}z_2^{-1} \\
-k_{01}^{n+1} & -k_{11}^{n+1}z_1^{-1} & -k_{10}^{n+1}z_1^{-1} & z_2^{-1}
\end{bmatrix}
\begin{bmatrix}
E_{00}^n(z_1, z_2) \\
E_{10}^n(z_1, z_2) \\
E_{11}^n(z_1, z_2) \\
E_{01}^n(z_1, z_2)
\end{bmatrix}.
\]

It is a recursive relationship.

Therefore, the \((n+1)^{th}\) order prediction errors can be related to the input \(Y\) easily, with the initial condition \(E_{00}^0(z_1, z_2) = E_{10}^0(z_1, z_2) = E_{11}^0(z_1, z_2) = E_{01}^0(z_1, z_2) = Y(z_1, z_2)\).

Moreover, the transfer function is readily seen from this relation, even though it is quite complicated when derived in detail. With the recursive substitutions, the overall lattice equation related to the input is given by:

\[
\begin{bmatrix}
E_{00}^{n+1}(z_1, z_2) \\
E_{10}^{n+1}(z_1, z_2) \\
E_{11}^{n+1}(z_1, z_2) \\
E_{01}^{n+1}(z_1, z_2)
\end{bmatrix} =
\begin{bmatrix}
1 & -k_{10}^{n+1}z_1 & -k_{11}^{n+1}z_1^{-1}z_2^{-1} & -k_{01}^{n+1}z_1^{-1}z_2^{-1} \\
-k_{10}^{n+1} & z_1^{-1} & -k_{01}^{n+1}z_1^{-1}z_2^{-1} & -k_{11}^{n+1}z_1^{-1}z_2^{-1} \\
-k_{11}^{n+1} & -k_{01}^{n+1}z_1^{-1} & z_1^{-1}z_2^{-1} & -k_{10}^{n+1}z_2^{-1} \\
-k_{01}^{n+1} & -k_{11}^{n+1}z_1^{-1} & -k_{10}^{n+1}z_1^{-1} & z_2^{-1}
\end{bmatrix}
\begin{bmatrix}
1 & -k_{10}^{n}z_1 & -k_{11}^{n}z_1^{-1}z_2^{-1} & -k_{01}^{n}z_1^{-1}z_2^{-1} \\
-k_{10}^{n} & z_1^{-1} & -k_{01}^{n}z_1^{-1}z_2^{-1} & -k_{11}^{n}z_1^{-1}z_2^{-1} \\
-k_{11}^{n} & -k_{01}^{n}z_1^{-1} & z_1^{-1}z_2^{-1} & -k_{10}^{n}z_2^{-1} \\
-k_{01}^{n} & -k_{11}^{n}z_1^{-1} & -k_{10}^{n}z_1^{-1} & z_2^{-1}
\end{bmatrix}...
\]


\[
\begin{bmatrix}
E_{00}^{n+1}(z_1, z_2) \\
E_{10}^{n+1}(z_1, z_2) \\
E_{11}^{n+1}(z_1, z_2) \\
E_{01}^{n+1}(z_1, z_2)
\end{bmatrix} =
\begin{bmatrix}
1 & -k_{10}^{n}z_1 & -k_{11}^{n}z_1^{-1}z_2^{-1} & -k_{01}^{n}z_1^{-1}z_2^{-1} \\
-k_{10}^{n} & z_1^{-1} & -k_{01}^{n}z_1^{-1}z_2^{-1} & -k_{11}^{n}z_1^{-1}z_2^{-1} \\
-k_{11}^{n} & -k_{01}^{n}z_1^{-1} & z_1^{-1}z_2^{-1} & -k_{10}^{n}z_2^{-1} \\
-k_{01}^{n} & -k_{11}^{n}z_1^{-1} & -k_{10}^{n}z_1^{-1} & z_2^{-1}
\end{bmatrix}
\begin{bmatrix}
E_{00}^{n}(z_1, z_2) \\
E_{10}^{n}(z_1, z_2) \\
E_{11}^{n}(z_1, z_2) \\
E_{01}^{n}(z_1, z_2)
\end{bmatrix}.
\]

Now, with focus to \(n^{th}\) order instead of \((n+1)^{th}\), (3.32) becomes:
\[
\begin{bmatrix}
E_{00}(z_1, z_2) \\
E_{10}(z_1, z_2) \\
E_{01}(z_1, z_2) \\
E_{11}(z_1, z_2)
\end{bmatrix} = \begin{bmatrix}
A_{11}(z_1, z_2) & A_{12}(z_1, z_2) & A_{13}(z_1, z_2) & A_{14}(z_1, z_2) \\
A_{21}(z_1, z_2) & A_{22}(z_1, z_2) & A_{23}(z_1, z_2) & A_{24}(z_1, z_2) \\
A_{31}(z_1, z_2) & A_{32}(z_1, z_2) & A_{33}(z_1, z_2) & A_{34}(z_1, z_2) \\
A_{41}(z_1, z_2) & A_{42}(z_1, z_2) & A_{43}(z_1, z_2) & A_{44}(z_1, z_2)
\end{bmatrix} \begin{bmatrix}
Y(z_1, z_2) \\
Y(z_1, z_2) \\
Y(z_1, z_2) \\
Y(z_1, z_2)
\end{bmatrix}
\]

where \(A_i(z_1, z_2)\) are polynomials of \(z_1\) and \(z_2\).

Defining the following prediction error transfer functions for the corresponding errors,

\[
B^a_{00}(z_1, z_2) = \frac{E^a_{00}(z_1, z_2)}{Y(z_1, z_2)}
\]

then

\[
\begin{align*}
B^a_{00}(z_1, z_2) &= A_{11}(z_1, z_2) + A_{12}(z_1, z_2) + A_{13}(z_1, z_2) + A_{14}(z_1, z_2) \\
B^a_{10}(z_1, z_2) &= A_{21}(z_1, z_2) + A_{22}(z_1, z_2) + A_{23}(z_1, z_2) + A_{24}(z_1, z_2) \\
B^a_{01}(z_1, z_2) &= A_{31}(z_1, z_2) + A_{32}(z_1, z_2) + A_{33}(z_1, z_2) + A_{34}(z_1, z_2) \\
B^a_{01}(z_1, z_2) &= A_{41}(z_1, z_2) + A_{42}(z_1, z_2) + A_{43}(z_1, z_2) + A_{44}(z_1, z_2)
\end{align*}
\]

### 3.2.3 The reflection coefficients

The reflection coefficients are obtained stage by stage by minimizing the mean square error (either one or summation of all four errors) with respect to each of them. These minimization results can be summarized as in the following form. From this system of equations, the reflection coefficients can be solved. Since it is just a 3-by-3 matrix, Cramer's rule, Gauss elimination or even direct substitution can be used as long as the covariance matrices \(R\) and \(r\) have been calculated.

\[
R^{(n)} \cdot k^{(n+1)} = r^{(n)}
\]
where \( k^{(n)} = \begin{bmatrix} k_{10}^{(n)} \\ k_{11}^{(n)} \\ k_{01}^{(n)} \end{bmatrix}, R^{(n)} = \begin{bmatrix} R_{11}^{(n)} & R_{12}^{(n)} & R_{13}^{(n)} \\ R_{12}^{(n)} & R_{22}^{(n)} & R_{23}^{(n)} \\ R_{13}^{(n)} & R_{23}^{(n)} & R_{33}^{(n)} \end{bmatrix} \) and \( r^{(n)} = \begin{bmatrix} r_{1}^{(n)} \\ r_{2}^{(n)} \\ r_{3}^{(n)} \end{bmatrix} \)

When different subject are being minimized, the contents of \( R_{ij} \) and \( r_i \) are different and they are summarized in the following Table 3.2. One can easily obtain the result, for a particular error as the minimizing subject, by taking the expectation of the self-multiplied error and differentiate it with respect to \( k_{10}, k_{11} \) and \( k_{01} \). Three equations result. After arranging accordingly, the above system of equation can be formed.

<table>
<thead>
<tr>
<th>( Min. E_{00} )</th>
<th>( Min. E_{10} )</th>
<th>( Min. E_{11} )</th>
<th>( Min. E_{01} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_{11} )</td>
<td>( \Phi_{10}(0,0) )</td>
<td>( \Phi_{00} e_{00}(0,0) )</td>
<td>( \Phi_{01} e_{00}(0,0) )</td>
</tr>
<tr>
<td>( R_{12} )</td>
<td>( \Phi_{10}(1,0) )</td>
<td>( \Phi_{00} e_{10}(0,0) )</td>
<td>( \Phi_{01} e_{10}(0,0) )</td>
</tr>
<tr>
<td>( R_{13} )</td>
<td>( \Phi_{10}(1,-1) )</td>
<td>( \Phi_{00} e_{10}(-1,0) )</td>
<td>( \Phi_{01} e_{10}(-1,0) )</td>
</tr>
<tr>
<td>( R_{22} )</td>
<td>( \Phi_{11}(0,0) )</td>
<td>( \Phi_{01} e_{11}(0,0) )</td>
<td>( \Phi_{01} e_{11}(0,0) )</td>
</tr>
<tr>
<td>( R_{23} )</td>
<td>( \Phi_{10}(1,0) )</td>
<td>( \Phi_{01} e_{10}(0,0) )</td>
<td>( \Phi_{01} e_{10}(0,0) )</td>
</tr>
<tr>
<td>( R_{33} )</td>
<td>( \Phi_{11}(0,0) )</td>
<td>( \Phi_{11}(0,0) )</td>
<td>( \Phi_{01} e_{11}(0,0) )</td>
</tr>
<tr>
<td>( r_1 )</td>
<td>( \Phi_{00} e_{10}(-1,0) )</td>
<td>( \Phi_{01} e_{10}(-1,0) )</td>
<td>( \Phi_{01} e_{10}(0,0) )</td>
</tr>
<tr>
<td>( r_2 )</td>
<td>( \Phi_{00} e_{10}(-1,-1) )</td>
<td>( \Phi_{01} e_{10}(-1,-1) )</td>
<td>( \Phi_{01} e_{10}(-1,-1) )</td>
</tr>
<tr>
<td>( r_3 )</td>
<td>( \Phi_{00} e_{10}(0,-1) )</td>
<td>( \Phi_{01} e_{10}(0,-1) )</td>
<td>( \Phi_{01} e_{10}(0,-1) )</td>
</tr>
</tbody>
</table>

Table 3.2: Element of eqt (3.34).

where \( \Phi_{x,y}(i,j) = E[e_x(k_1,k_2) e_y(k_1+i, k_2+j)] \) is the correlation function.

Since it is applying on a limited data field, only estimation of this expectation values can be obtained. And the estimation formula used is

\[
E[f(k_1, k_2)] = \frac{1}{(K_1-n_1-1)(K_2-n_2-1)} \sum_{k_{10}, k_{11}} \sum_{k_{01}} f(k_1, k_2)
\]  

(3.35)
i.e. taking the average value of the overlapping area between the two limited data fields.

The data outside the overlapping zone has no contribution to the expectation estimation.

In other words, it is the same as taking the covariance of the two functions.

Besides these individual minimizations, another criterion can be used. It is the minimization of the sum of the four square errors. This criterion will correspond to Burg's maximum entropy autoregressive model. And the values for $R_{ij}$ and $r_i$ are the summation of the corresponding $R_{ij}$ and $r_i$ in each of the four error's cases, i.e. $R_{11} = \Phi e_{10}e_{10}(0,0)+\Phi e_{00}e_{00}(0,0)+\Phi e_{01}e_{01}(0,0)+\Phi e_{11}e_{11}(0,0)$, sum of the row in the Table 3.2.

---

**Figure 3.7:** n-stage lattice modeling filter's structure.
3.2.4 Synthesis model arrangement

The purpose of modeling is to find an estimate to the system parameters. With the estimated parameters, we can synthesize the signal by a white noise, i.e. giving \( E_{n0} \) to yield \( Y \). For the canonical transfer function, the synthesis of an AR model is an inverse of the polynomial, an inverse filter. This will change the all-zero filter (in modeling) into the all-pole filter (in synthesis). However, for lattice structure, there is another easier method to implement the synthesis. Take a closer look at the lattice matrix below, and treat it as a system of equations instead of a signal flow structure. From (3.30), if we separate the first row from the rest and rearrange \( e_{oo}^{n}(i,j) \) as the subject instead of \( e_{oo}^{n+1}(i,j) \). It will become two sets of equations:

\[
e_{00}^{n}(i,j) = \begin{bmatrix} 1 & k_{10}^{n+1} & k_{11}^{n+1} & k_{01}^{n+1} \end{bmatrix} \begin{bmatrix} e_{00}^{n+1}(i,j) \\ e_{10}^{n}(i-1,j) \\ e_{11}^{n}(i-1,j-1) \\ e_{01}^{n}(i,j-1) \end{bmatrix}
\]

\[
\begin{bmatrix} e_{10}^{n+1}(i,j) \\ e_{11}^{n+1}(i,j) \\ e_{01}^{n+1}(i,j) \end{bmatrix} = \begin{bmatrix} -k_{10}^{n+1} & 1 & -k_{01}^{n+1} & -k_{11}^{n+1} \\ -k_{10}^{n+1} & -k_{01}^{n+1} & 1 & -k_{11}^{n+1} \\ -k_{01}^{n+1} & -k_{11}^{n+1} & -k_{10}^{n+1} & 1 \end{bmatrix} \begin{bmatrix} e_{00}^{n}(i,j) \\ e_{10}^{n}(i-1,j) \\ e_{11}^{n}(i-1,j-1) \\ e_{01}^{n}(i,j-1) \end{bmatrix}
\]

The inputs are \( e_{oo}^{n+1}(i,j) \), \( e_{10}^{n}(i,j) \), \( e_{11}^{n}(i,j) \), \( e_{01}^{n}(i,j) \), with delay correspondingly, while \( e_{00}^{n}(i,j) \), \( e_{10}^{n+1}(i,j) \), \( e_{11}^{n+1}(i,j) \), \( e_{01}^{n+1}(i,j) \) are the outputs. The significance here is in the role of \( e_{00}^{n}(i,j) \). Since for \( n=0 \), it is equal to \( Y \). It represents the overall input and output (\( Y \) and the prediction error \( E_{n0} \)). Now, the signal flow direction for \( e_{00}^{n}(i,j) \) is changed, i.e. given \( e_{00}^{n}(i,j) \), we can find \( e_{00}^{n-1}(i,j) \) (See Figure 3.8). In other words, we can input
the prediction error $E_{\infty}$ and output the desired signal $Y$ with the original lattice parameters. It is the process of the synthesis.

The result is evidenced mathematically. As long as the same mathematical relation in the system of equation holds, we can obtain the values of $Y$ since these values still satisfy the equations. The stability becomes a concern when it is implemented in a signal flow system. Since each of these lattice structures now has one path in reversed direction, they all form a different feedback path that will potentially contribute to an unstable system. When a

Figure 3.8: Synthesis lattice structure. Note the change of direction and also the reflection coefficients change sign on the top path.
loop gain is greater than one, then the system will become unstable. Refer [33] for stability conditions.

3.2.5 Discussion

To see how the reflection coefficients related to the transfer function, here we exploit the equation for the first two orders.

From (3.32), for \( n = 0 \),

\[
E_{00}^1(x_1, x_2) = \left[ 1 - k_{10}^{-1} - k_{11}^{-1} - k_{01}^{-1} \right] \cdot Y(x_1, x_2)
\]  

(3.37)

The error function (3.37) is the prediction of \( Y \) with three other histories. They form a quarter-plane support with each lattice parameter (reflection coefficient) corresponds to one canonical parameter. This transfer function is able to use the MMSE criterion since it has enough parameters (degree of freedom) to adapt the canonical system characteristics.

For \( n = 1 \),

\[
E_{00}^2(x_1, x_2) = \left[ 1 + \left( k_{10}^1 k_{10}^{-1} - k_{10}^{-1} \right)x_1^{-1} - k_{10}^2 x_1^{-2} 
+ \left( k_{01}^2 k_{01}^{-1} - k_{01}^{-1} \right)x_2^{-1} + \left( k_{11}^2 k_{11}^{-1} - k_{11}^{-1} + k_{01}^2 k_{11}^1 + k_{10}^2 k_{11}^1 \right)x_1^{-1} x_2^{-1} 
+ \left( k_{11}^2 k_{11}^{-1} + k_{01}^2 k_{01}^1 \right)x_1^{-2} x_2^{-1} 
- k_{01}^2 x_2^{-2} + \left( k_{11}^2 k_{10}^1 + k_{01}^2 k_{10}^1 \right)x_1^{-1} x_2^{-2} - k_{11}^2 x_1^{-2} x_2^{-2} \right] \cdot Y(x_1, x_2)
\]

(3.38)
The error function (3.38) this time is also a prediction of \( Y \) with 8 neighbors on a quarter-plane. On the contrary, there are only 6 reflection coefficients for a general canonical transfer function of 8 parameters. The main problem is the lack of the number of degrees of freedom. Therefore, even the error function can be minimized for this special lattice structure, the error may not be a true minimum for a general AR transfer function with the same support plane.

This deficiency is more obvious when the order is increased. For \( N=2 \), lattice parameters are 9 while the allowable degree of freedom of the same support in a canonical transfer function is 15. In general, the lattice parameters are \( 3N \) while the canonical parameters are \( (N+1)^2-1 \). The higher the order, the larger the difference is.
3.3 2-D lattice algorithms: 2-D orthogonal lattice algorithm

Due to the deficiency of the modeling power on the previous model, Kayran had introduced another lattice structure [38]. His new model is called orthogonal lattice model. Orthogonality indicates his new approach yields a true least squares estimation.

The algorithm starts with arranging the 2-D support plane into a one-dimensional parameter vector $a_0^{(m)}$. See Figure 3.10. A method similar to the Levinson algorithm is employed to resolve the auto-correlation matrix recursively. Unlike the Levinson algorithm on 1-D yielding a pipeline structure lattice filter, this algorithm results in a triangular lattice filter. The forward prediction error associated with the prediction of the zero\textsuperscript{th} sample from the previous m samples within the prediction region can be defined as

$$e_0^{(m)}(k_1, k_2) = a_0^{(m)T} y_{0,m}(k_1, k_2)$$  \hspace{1cm} \hspace{1cm} (3.39)$$

where $a_0^{(m)} = [1 \ a_0^{(m)}(1) \ \cdots \ a_0^{(m)}(m)]^T$

$$y_{p,q}(k_1, k_2) = [y((k_1, k_2) - p) \ y((k_1, k_2) - p - 1) \ \cdots \ y((k_1, k_2) - q)]^T$$

where the notation $y((k_1, k_2) - i)$ denotes the $i^{th}$ element behind $y(k_1, k_2)$.

Similar to (3.39), a backward prediction error is defined by

$$v_0^{(m)}(k_1, k_2) = b_0^{(m)T} y_{0,m}(k_1, k_2)$$  \hspace{1cm} \hspace{1cm} (3.40)$$

where $b_0^{(m)} = [b_0^{(m)}(m) \ b_0^{(m)}(m-1) \ \cdots \ b_0^{(m)}(1)]^T$
Using the Yule-Walker normal equation, we get

\[
\mathbf{R}_{0,m} \mathbf{a}_m = \begin{bmatrix} \mathbf{E}_{e_0}^{(m)} \\ \mathbf{0}_m \end{bmatrix} \quad \text{and} \quad \mathbf{R}_{0,m} \mathbf{b}_m = \begin{bmatrix} \mathbf{0}_m \\ \mathbf{E}_{v_0}^{(m)} \end{bmatrix}
\]  

(3.41)

where

\[
\mathbf{R}_{0,m} = \begin{bmatrix} \mathbf{R}_{0,m-1} & \mathbf{r}_{0,m}^R \\ \mathbf{r}_{0,m}^T & r(0,0) \end{bmatrix} = \begin{bmatrix} r(0,0) & r_{0,m}^T \\ r_{0,m} & \mathbf{R}_{1,m} \end{bmatrix}
\]

and

\[
\mathbf{R}_{\rho,q} = \mathbb{E}[y_{\rho,q}(k_1,k_2) y_{\rho,q}^*(k_1,k_2)]
\]  

(3.42)

\[
r(0,0) = \mathbb{E}[y^2((k_1,k_2) - p)]
\]

\[
r_{i,m} = \mathbb{E}[y((k_1,k_2) - i)y_{i-1,m}(k_1,k_2)]
\]

\[
r_{i,m}^R = \mathbb{E}[y((k_1,k_2) - i)y_{i-1,m}^*(k_1,k_2)]
\]

where superscript B denotes the backward arrangement.

With (3.41), we would like to find out a recursive relationship such that

\[
\mathbf{a}_0^{(m)} = \begin{bmatrix} \mathbf{a}_0^{(m-1)} \\ 0 \end{bmatrix} + \Gamma_{e_0}^{(m)} \begin{bmatrix} \mathbf{0} \\ \mathbf{b}_m^{(m-1)} \end{bmatrix}
\]  

(3.43)

\[
\mathbf{b}_m^{(m)} = \begin{bmatrix} 0 \\ \mathbf{b}_m^{(m-1)} \end{bmatrix} + \Gamma_{e_0}^{(m)} \begin{bmatrix} \mathbf{a}_0^{(m-1)} \\ 0 \end{bmatrix}
\]

Pre-multiplying \( \mathbf{R}_{0,m} \) into (3.43), together with (3.41) & the 1st equation in (3.42),

\[
\begin{bmatrix} \mathbf{E}_{e_0}^{(m)} \\ \mathbf{0}_m \end{bmatrix} = \begin{bmatrix} \mathbf{E}_{e_0}^{(m-1)} \\ \mathbf{0}_{m-1} \end{bmatrix} + \Gamma_{e_0}^{(m)} \begin{bmatrix} \mathbf{0}_{m-1} \\ \mathbf{E}_{v_0}^{(m-1)} \end{bmatrix}
\]  

(3.44)
where \( \Delta_{e_0}^{(m-1)} = r_0^{\text{T}} a_0^{(m-1)} \) and
\( \Delta_{v_0}^{(m-1)} = r_0^{\text{T}} b_0^{(m-1)} \)
and note that \( R_{0,m-1} a_0^{(m-1)} = \begin{bmatrix} E_{e_0}^{(m-1)} \\ 0_{m-1} \end{bmatrix} \) and \( R_{0,m-1} b_0^{(m-1)} = \begin{bmatrix} 0_{m-1} \\ E_{v_0}^{(m-1)} \end{bmatrix} \).

(3.44) allows the calculation of the forward and backward reflection factors and the minimum-mean-square errors,

\[
\begin{align*}
\Gamma_{e_0}^{(m)} &= -\Delta_{e_0}^{(m-1)}/E_{e_0}^{(m-1)} \\
\Gamma_{v_0}^{(m)} &= -\Delta_{v_0}^{(m-1)}/E_{v_0}^{(m-1)} \\
E_{e_0}^{(m)} &= E_{e_0}^{(m-1)} \left(1 - \Gamma_{e_0}^{(m)} \Gamma_{v_0}^{(m)}\right) \\
E_{v_0}^{(m)} &= E_{v_0}^{(m-1)} \left(1 - \Gamma_{e_0}^{(m)} \Gamma_{v_0}^{(m)}\right)
\end{align*}
\] (3.45)

It can be shown that \( \Delta_{e_0}^{(m-1)} = \Delta_{v_0}^{(m-1)} \) and they can be interpreted as a cross-correlation between the forward and backward prediction errors.

i.e. \( \Delta_{e_0}^{(m-1)} = \Delta_{v_0}^{(m-1)} = E[e_0^{(m-1)}(k_1,k_2)\nu_m^{(m-1)}(k_1,k_2)] \)

Using (3.39) & (3.40), (3.43) will form a single structure,

\[
\begin{bmatrix}
e_0^{(m)}(k_1,k_2) \\
\nu_m^{(m)}(k_1,k_2)
\end{bmatrix} =
\begin{bmatrix}
1 & \Gamma_{v_0}^{(m)} \\
\Gamma_{e_0}^{(m)} & 1
\end{bmatrix}
\begin{bmatrix}
e_0^{(m-1)}(k_1,k_2) \\
\nu_m^{(m-1)}(k_1,k_2)
\end{bmatrix}
\] (3.46)

The partition of \( R_{0,m} \) into \( R_{0,m-1} \) and \( R_{1,m} \) gives (3.43). Similarly, we further partition \( R_{0,m-1} \) and \( R_{1,m} \) into \( R_{0,m-2} \) \( R_{1,m-1} \) and \( R_{2,m} \), we would get the following results.

\[
\begin{align*}
a_0^{(m-1)} &= \begin{bmatrix} a_0^{(m-2)} \\ 0 \end{bmatrix} + \Gamma_{e_0}^{(m-1)} \begin{bmatrix} 0 \\ b_0^{(m-1)} \end{bmatrix} \\
b_m^{(m-1)} &= \begin{bmatrix} 0 \\ b_m^{(m-2)} \end{bmatrix} + \Gamma_{v_0}^{(m-1)} \begin{bmatrix} a_1^{(m-2)} \\ 0 \end{bmatrix}
\end{align*}
\] (3.47)
and

\[
a_i^{(m-1)} = \begin{bmatrix} a_1^{(m-2)} \\ 0 \end{bmatrix} + \Gamma_{a}^{(m-1)} \begin{bmatrix} 0 \\ b_m^{(m-2)} \end{bmatrix} \\
b_i^{(m-1)} = \begin{bmatrix} 0 \\ b_i^{(m-2)} \end{bmatrix} + \Gamma_{b}^{(m-1)} \begin{bmatrix} a_0^{(m-2)} \\ 0 \end{bmatrix}
\]

Again, together with (4.39) & (4.40), (3.47) & (3.48) form two single structures. Continuing with the pattern, the orthogonal lattice structure will result. The above formula implies a triangular lattice structure as depicted in Figure 3.11.

![2-D orthogonal lattice structure](image)

**Figure 3.11: 2-D orthogonal lattice structure.**

### 3.3.1 Discussion

The algorithm depicted here started from the Yule-Walker normal equation. As a result, the structure has the orthogonal property, i.e. it estimates the least-mean-square error. This lattice structure contains lattice blocks which are connected in a triangular shape instead of a pipeline structure as in the 1-D lattice. It has a disadvantage in the algorithm. For one additional parameter (point of support), increment of lattice blocks will be in an...
AP, i.e. number of lattice blocks = N(N+1)/2. Besides the complexity of the algorithm, this algorithm gives the solution of the normal equation by those recursive formulae. Therefore, it could really represent the canonical two-dimensional AR model. The total number of lattice elements used is N(N+1)/2, where N is the number of points in the support plane, i.e. N=N₁*N₂-1. It is the simplest noncausal orthogonal lattice structure thus far. Furthermore, it will require \( O(N^2) \) parameters for an AR process of \( O(N) \) parameters.

In addition, one superior advantage is that the algorithm is extendible to noncausal support. From the basic principles of the algorithm, revealed from the correlation matrix in (3.42), all required is that the correlation matrix be symmetric. However, the auto-correlation function is an even function. This makes the correlation matrix symmetric in nature. Therefore, the algorithm can apply to noncausal support modeling.
3.4 Contrast between two 2-D lattice methods

<table>
<thead>
<tr>
<th>Simple Lattice</th>
<th>Orthogonal Lattice</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td>AutoRegressive.</td>
</tr>
<tr>
<td><strong>Order update</strong></td>
<td>Introduce many supports with each order update.</td>
</tr>
<tr>
<td><strong>No. of lattice blocks for each order update</strong></td>
<td>One additional stage cascade to the lower order filter.</td>
</tr>
<tr>
<td><strong>True least square measure?</strong></td>
<td>Optimized within the degrees of freedom that the filter has.</td>
</tr>
<tr>
<td><strong>Computation burden</strong></td>
<td>Pipeline structure. The increase in computation is a linear function of the order.</td>
</tr>
<tr>
<td><strong>Storage</strong></td>
<td>Each stage requires an implementation of 2-D delays. That requires a minimum storage of one row, if the maximum vertical delay required is one.</td>
</tr>
<tr>
<td><strong>Ability to model high order</strong></td>
<td>No. The 3 parameters per stage limit the describing power of this simple lattice structure.</td>
</tr>
<tr>
<td><strong>Ability to apply on noncausal</strong></td>
<td>It is a tailor made algorithm for quarter-plane. Extension is not flexible.</td>
</tr>
</tbody>
</table>

Table 3.3: Comparison of Simple Lattice and Orthogonal Lattice.
In conclusion, the simple lattice is good for 2nd order (2x2 support plane) application. It is simple and fast compared with the orthogonal lattice. On the other hand, this 2-D orthogonal lattice structure provides flexibility and capability on modeling; however, the price is the computation burden. The orthogonal lattice method is chosen primarily because it can be applied to a noncausal support plane. Furthermore, its capability is far more superior to the aforementioned simple lattice method. This method has potential for further achievements.
3.5 Using 2-D orthogonal lattice method into noncausal support

Although only causal and semi-causal cases are described in [38], the 2-D orthogonal lattice method can be applied to noncausal modeling. As emphasized in section 3.3, the requirement of using this lattice structure is that the correlation matrix \( \mathbf{R}_{a,m} \) has to be a symmetric matrix, i.e. \( \mathbf{R}_{a,m} = \mathbf{R}_{a,m}^T \). As long as \( \mathbf{R}_{a,m} \) follows (3.42), the recursive resolving algorithm depicted in section 3.3 applies.

In order to utilize this 2-D orthogonal lattice method, a sufficient condition is to construct a noncausal support plane such that the corresponding correlation matrix \( \mathbf{R}_{a,m} \) has a symmetry property. Consider the ordering of the support plane as shown in Figure 3.12. We are going to prove that this noncausal support plane has a symmetric correlation matrix.

![Support plane domain](image)

Figure 3.12: A noncausal support plane.

The parameter vector \( \mathbf{a} \) will be arranged as \( \mathbf{a} = [1 \quad a(1) \quad \cdots \quad a(8)]^T \) and the corresponding \( y(i,j) \)'s during the modeling filter can be written

\[
\begin{bmatrix}
y(k_1, k_2) \\
y(k_1, k_2 - 1) \\
\vdots \\
y(k_1 - 1, k_2 - 1)
\end{bmatrix}
\]

as, \( y(k_1, k_2) = \mathbf{a}^T \cdot y(k_1, k_2) \) ............................................. (3.49)
Applying the result from section 3.3.1, we have Yule-Walker normal equation. It should be noted that in the parameter vector $\mathbf{a}$ in (3.17), the first parameter is not normalized. We can normalize (3.17)'s $\mathbf{a}$ into a suitable form for this section. Let $a_0$ be the first parameter of $\mathbf{a}$ in (3.17). Dividing both sides by $a_0$, (3.17) becomes,

$$
\mathbf{R} \cdot \mathbf{a} = \left( \frac{h_{d0}}{a_0} \right) \delta \tag{3.50}
$$

But $a_0$ is an unknown. Nevertheless, from the error definition (3.49), with auto-correlation definition, we can obtain the another relation for (3.50). Now it becomes,

$$
\mathbf{R} \cdot \mathbf{a} = \begin{bmatrix}
E(\mathbf{e}^T \cdot \mathbf{e}) \\
0 \\
\vdots \\
0
\end{bmatrix} \tag{3.51}
$$

where $\mathbf{e}$ is defined as in (3.6) and $E(\cdot)$ is the expectation operator that defined in correlation function estimator. That is, whatever the correlation estimator used, it applies to both side of (3.51).

To reveal the symmetry properties, we go back one step to show what is in $\mathbf{R}$. In the course of deriving this normal equation, (3.51) will appear as,

$$
E\{\mathbf{y}(k_1, k_2) \cdot \mathbf{y}(k_1, k_2)^T\} \cdot \mathbf{a} = \begin{bmatrix}
E(\mathbf{e}^T \cdot \mathbf{e}) \\
0 \\
\vdots \\
0
\end{bmatrix} \tag{3.52}
$$

It is obvious that $\mathbf{R}$ is symmetric if $\mathbf{y}(k_1, k_2)$ is real. By $(\mathbf{a} \cdot \mathbf{b})^T = (\mathbf{b}^T \cdot \mathbf{a}^T)$,
\[
R^T = \{E(y(k_1, k_2) \cdot y(k_1, k_2)^T)\}^T
= E\{(y(k_1, k_2)^T) \cdot (y(k_1, k_2))\}^T
= R
\] (3.53)

It is an important result from (3.53) that the exact ordering is not being involved in the proof. That implies as long as \(y(k_1, k_2)\) is real, no matter how the parameters sequence is ordered and where the support points are, the auto-correlation matrix is symmetric in nature. In fact, if in 1-D cases that all support points are in adjacent sequence, the auto-correlation matrix is not just symmetric but also Toeplitz.

In conclusion, because of the symmetric property of the correlation function, the recursive resolving algorithm on the 2-D orthogonal lattice structure can be applied to any noncausal support. The viability of application to a noncausal support has been shown.
Chapter 4

4. Quantization

Quantization is a critical but embedded process in image compression. In JPEG [1] lossy mode, part of the main compression is due to an appropriate quantization of discrete cosine transform (DCT) coefficients. The quantization step size will affect the image very much. A coarse quantization will result in worse PSNR performance. However, the resource (no. of bits) required for a coarse quantization will be less, and therefore, better compression will result. It is a dilemma. Therefore, a good compression scheme should avoid the worse performance and at the same time achieve a low bit rate. In modeling, residues have certain characteristics that can allow better compression than a direct application of methods on the original pixels. Various techniques can be applied to utilize this merit. This chapter, first of all, will briefly introduce a non-uniform quantization which illustrates how residues will be benefited from the quantization process. The purpose of that section is to illustrate the concepts which are extended to the next section: vector quantization. Vector quantization (VQ) is claimed as the ultimate quantization technique [46]. An example will illustrate its pros and cons. The application of VQ on lattice
modeling will be used in Chapter 6. Finally, an neural network approach will be compared with the conventional VQ algorithms.

4.1 Non-uniform quantizer

In general, quantization is in uniform step size for adapting the varieties of situations. For example, the 8 bits per pixel gray level in image representation is a linear step size quantization. Since the intensity will vary unpredictably from 0 to 255 dynamic range values, the quantization resource has to be shared equally. Nonetheless, with certain characteristics in the signal to be quantized, some optimization can be achieve.

Figure 4.1 shows a typical residue from a lattice modeling prediction filter. One of the approaches to reduce the overall average distortion, or to increase the compression with the same distortion level, is to allocate more resources on the more frequently occurring values and less for the less frequent ones. Thus the majority will be reduced in distortions while just a minority will increase in distortions. The overall average distortion will then decrease. Residues, in general, have most of its occurrences concentrated around zero. In this case, even they spread over a wide range, about -60 to 60; the dynamic range is still narrower than the original image range. A non-uniform quantization will definitely improve the overall resolution. On the other hand, for image compression, we would like to reduce the bits per pixel used instead of improving the resolution.
Chapter 4: Quantization

Figure 4.1: a) Typical distribution of a residue (left). It is from "Karen" with quarter-plane support lattice modeling. b) The original image (right).

Figure 4.2: Distribution of a) residue of "Lena" (left), b) original "Lena" (256x256x8).

Figure 4.3: Distribution of a) residue of "Goldhill" (left), b) original "Goldhill" (256x256x8).
As depicted in the above Figure 4.1—Figure 4.3, the distribution of ordinary image's pixel intensities vary. Moreover, they will be of different shapes of distribution. Non-uniform level quantization seems not able to achieve good result. But with the special distribution shape of residues, we can optimize the compression by using this technique. The following quantization example used 15 levels to quantize the above image Karen's residue. The original image "Karen" was a 256x256 8 bits gray level image. Its intensity histogram was shown in Figure 4.1. The residues were obtained by using a two-stage quarter-plane lattice modeling filter. The Lloyd Algorithm was used to search for the optimal quantization levels. Result is shown in Figure 4.4—Figure 4.6. The input-output quantization mapping function is shown in Figure 4.5. Figure 4.6 shows the distribution of the quantized intensity value. Only the index number is shown for simplicity. The distribution should be similar to the original distribution.

![Non-uniform level Quantization](image)

Figure 4.4: Levels of the Non-uniform Quantization.
Analysis is focused on the quantization error in terms of signal to noise ratio. With this non-uniform quantization, the signal-to-noise ratio (SNR) is 15.34 dB. Comparing with a uniform-step quantizer which has a SNR of 10.26 dB, this optimized quantizer has improved significantly the performance. Since the residue and the original image pixels are of different dynamic range, direct comparison of the SNR is unfair and cannot justify the
performance. Therefore, the synthesized image of the quarter-plane model obtained from
the 15-level quantized residue is used for comparison. The synthesized image compared
with the source image is of PSNR of 32.54 dB. With 16-level uniform quantization
(corresponding to 4 bpp) on the source image, the quantization error made a PSNR of
30.10 dB. In addition, the improvement in PSNR is not the only story. After the
quantization on the residue, it can achieve ~2.6 bpp with entropy coding.

4.2 Vector Quantization

4.2.1 Introduction

Vector quantization (VQ) is a generalization of scalar quantization. The word "vector"
actually means an ordered set of real numbers. It is a jump from one-dimension to multi-
dimensions. VQ is usually for data compression from digital to digital representation. Even
though it appears as "vector", "multi-dimensions", the application of "vector" is usually
used as an ordered set rather than a multi-dimensional spatial representation. This ordered
set representation is able to describe almost any type of patterns. Therefore, VQ can be
treated as a form of pattern recognition where the input pattern is approximated by a
predetermined pattern (a code vector). Instead of storing, either the original pattern or the
approximated one, an index to a set of predetermined patterns (codebook) will be the
representation of the pattern. Figure 4.7 shows a general picture of the process of vector
quantization. With input $X$, $I$ will be the index to be stored.
In fact, VQ is not merely a generalization of scalar quantization. It is the “ultimate” solution to the quantization of a signal vector. It is claimed no other coding technique exists that can do better than VQ. The word “better” means with the same compression ratio, its signal-to-noise ratio will outperform the others. It seems an exaggerate statement, but in fact, by a general term vector quantization, it is just a framework which includes all the possibilities. In other words, it is a super set of vector coding technique (obviously true for the scalar case), or, any of a vector coding technique is a subset of VQ.

With an \textit{ad hoc} or heuristically designed coding technique that codes a set of \(k\) signal samples with \(N\) distinct patterns, a VQ can be at least the same as this design with \(N\) of \(k\)-dimensional code vectors in the codebook. There is a price for the superior compression performance - the computation burden. Both the codebook design and encoder algorithm are computational consuming.

\textbf{4.2.2 Basic definitions}

A vector quantizer \(Q\) of dimension \(k\) and size \(N\) is a mapping from a vector in \(k\)-dimensional Euclidean space, \(\mathbb{R}^k\), into a finite set \(C\).
$Q : \mathbb{R}^k \rightarrow C$

where $Q$ is the vector quantizer and $R$ is the Euclidean space, and

$C$ is called codebook and is defined by $C = (y_1, y_2, \cdots, y_M)$, $y_i \in \mathbb{R}^k$, where a $y_i$ is called a code vector.

Associated with every code vector is a partition (cell) in set of $\mathbb{R}^k$. The $i^{th}$ cell which is associated with $y_i$ is defined by

$$R_i = \{x \in \mathbb{R}^k : Q(x) = y_i \},$$

for which to be a cell, it naturally follows that

$$\bigcup_i R_i = \mathbb{R}^k \text{ and } R_i \cap R_j = 0 \text{ for } i \neq j,$$

so that the cells form a partition of $\mathbb{R}^k$. A cell can be bounded or unbounded.

The above basic definitions have described the components in a vector quantizer. Figure 4.8 shows a 2-D VQ, with $N=6$ code vectors in the codebook. The input $X$ is in $3^{th}$ cell, and therefore, will be quantized to vector $y_3$ with index $I=3$. i.e. the quantized $X$, named $\hat{X}$ in Figure 4.7., is equal to $y_3$. 
4.2.3 VQ design

In terms of design procedure or operation algorithm, a vector quantization can mainly be split into two parts: a codebook design and an encoder design. The decoder operation is exactly a table-look-up procedure. However, the computation burden is mainly on the encoder and the codebook design. In general, the nearest neighbor encoder and the General Lloyd Algorithm for the codebook design are the common choices. The codebook structure and the encoder algorithm can be independently or closely related. Some tree-structured VQ, classified VQ, etc., will require a close cooperation between the codec and the codebook structure.
4.2.3.1 Nearest Neighbor Quantizer

As depicted in Figure 4.8, the encoder is responsible for indexing procedure. The nearest neighbor criterion is the simplest choice. There are some advantages in Nearest Neighbor Quantizer (NNQ). Partition is determined by codebook with distortion measurement. Distortion measure is defined by the Euclidean distance between two vectors:

\[ d(x, y) = \|x - y\|^2 = \sum_{i=1}^{k} (x_i - y_i)^2 \]

Partition cells are given by:

\[ R_i = \{ x : d(x, y_j) \leq d(x, y_j) \forall j \in \mathbb{N} \} \]

Once the codebook is designed, the partitions are virtually known. There will be less overhead for maintaining the partitions information. Furthermore, the code vector matching criterion is simple.

Nearest Neighbor Encoding Rule

1. Set \( d = d_0, j = 1, \) and \( i = 1. \)
2. Compute \( D_j = d(x, y_j). \)
3. If \( D_j < d, \) set \( D_j = d \) and set \( i = j. \)
4. If \( j < N, \) set \( j = j + 1 \) and go to step 2, else go to step 5.
5. Stop. \( i \) is the result nearest neighbor encoding index.

From the above encoding procedure, it reveals the drawback on this method. The minimum Euclidean distance search is an exhaustive search algorithm which walks through all the code vectors in the codebook. Each walk requires an Euclidean measurement which
is an intensive computation as the vector dimension increases. These considerations motivate serious study of more efficient algorithms that yield the nearest neighbor code vector without an exhaustive search through the codebook.

4.2.3.2 General Lloyd Algorithm (GLA)

Lloyd algorithm is commonly used in scalar quantization for obtaining optimal quantizer. It is an iterative codebook improvement algorithm based on the necessary conditions for optimality. The idea can extend to multi-dimension. Figure 4.9 shows the flowchart of GL Algorithm.

General Lloyd Algorithm (1977)

1. Begin with initial codebook $C_1$.

2. Given $C_m$, perform Lloyd Iteration to generate the improved codebook $C_{m+1}$.

3. Compute average distortion for $C_{m+1}$. If its change is smaller than the predefined amount, stop. Otherwise, $m+1 \rightarrow m$ and go back to step 2.

Lloyd Iteration

1. Given a codebook, $C_m=\{y_i\}$, partition the training set $T$ into cluster sets $R_i$ using the Nearest Neighbor Condition.

$$R_i = \{x \in T: \quad d(x, y_i) \leq d(x, y_j); \forall j \neq i\}$$
2. Compute the centroids for the cluster sets just found for the new codebook, 
\[ C_{m+1} = \{ \text{centroid}(R_i) \} \]. If an empty cell was generated in 1), an alternate code 
vector assignment is made for that cell, e.g. split the most crowded cell.

![Diagram of General Lloyd Algorithm]

Figure 4.9: General Lloyd Algorithm.

The GLA needs an initial codebook to start. This initial codebook can be randomly 
generated or randomly selected in the training set. It may affect the iterations required to 
achieve the converged distortion level. There is a lemma proved that the steps in Lloyd 
Iteration reduced or left unchanged the average distortion. With this monotonic 
decreasing average distortion sequence, these average distortion sequence convergency is 
guaranteed in a finite number of iterations for a finite training set. On the contrary, it is a 
computational intensive algorithm. The centroids calculation as well as the distortion 
measure in each iteration involves Euclidean distance calculation. Furthermore, the same 
as many iteration algorithm, local minima may occur which is a kind of convergency 
problem. Thus the initial codebook may affect the optimality in the final result.
4.2.3.3 An example from simulation result

A 128x128x8 gray color "Lena", Figure 4.10, was used in the experiment. The codebook was obtained by using the General Lloyd Algorithm. The simple nearest neighbor encoder was used to code the image. Indexes of the code vectors were used for storage and decoding. The decoded result was shown in Figure 4.11. The codebook was designed as consisting of 256 distinct vectors. Each of them was 4x4 in dimension. This was equivalent to a 0.5 bit per pixel (bpp) if the codebook size was ignored.

![Original Lena Picture](image-url)
Figure 4.11: Vector Quantized Result. Compression rate at 0.5 bpp.

Figure 4.12: Codebook for this example. It consists of 256 code vectors each with 4x4 dimension.
Figure 4.12 shows the 256 code vectors in the codebook. For the sake of simplicity, all code vectors are packed adjacent to each other. In the figure, each 4x4 small block represents a code vector.

The result shows the VQ’s superior compression power. The above decoded image has a PSNR of 30.67 dB with only 0.5 bpp. In general, VQ can easily achieve a good compression ratio (smaller than 10 to 1), however, there is some main weakness as in VQ. It has a blocky and sawtooth effect. Human perception is very visually sensitive to this kind of artifacts. Current research works on area of VQ are focused on codebook design algorithms, efficient encoders and the reduction of the blockiness artifacts. For the latter, many interpolation techniques have been introduced to improve the performance. In conclusion, VQ is still very attractive in data compression despite its intensive computation.

4.3 Neural Network approach

Owing to the computation burden on codebook design and encoding process of VQ, many methods have been introduced in VQ. One of them is using an Artificial Neural Network (ANN) which has drawn lots of attention in research studies. It is because of its mystery learning power and its wide applications. The discussion on the area of ANN is out of our scope here. Nevertheless, a special kind of ANN has been introduced in the area of VQ. It is the Learning Vector Quantization (LVQ). LVQ is a variant of the Self Organization Feature Map (SOFM). They both were introduced by Kohonen.
LVQ is a pattern classification method in which each output unit represents a particular class or category. The weight vector for an output unit is often referred to as a reference vector (code vector) for the class that the unit represents. Figure 4.13 show the LVQ structure. For the aforementioned case in the previous section, a codebook with 256 code vectors of 4x4 in dimensions, number of output neurons (m) will be equal to 256 and that of input neurons (n) will be equal to 16. The set of \( \{w_{ij}: i = 1, 2, \ldots, 16\} \) will be the \( j^{th} \) code vector.

4.3.1 Learning algorithm: codebook design.

In brief, the procedure is first to find out the winner output neuron. The winner is defined by the one whose weight vector closest matches with the input vector \( x \). Euclidean measure is used for the distance measurement in the matching. If the winner’s class is the same as the input training vector \( x \)’s class, the learning will reinforce the winner’s weight.
vectors. i.e. pulling the weight vector toward the direction to $x$. If they are of different class, the algorithm will push the weight vector away from $x$.

The LVQ learning Algorithm

\[ x \] training vector $(x_1, \ldots, x_i, \ldots, x_n)$.
\[ T \] correct category or class for the training vector.
\[ w_j \] weight vector for $j^{th}$ output unit $(w_{ij}, \ldots, w_{ij}, \ldots, w_{nj})$.
\[ C_j \] category or class represented by $j^{th}$ output unit.

1. Initialization: initial weight and the learning rate parameters.
2. While stopping condition is false do
   
   \{
   
   For each input vector $x$,
   
   \{
   
   For each $j$, compute $D(j) = \sum (w_{ij} - x_i)^2$.

   Find index $J$ such that $D(J)$ is a minimum.

   If $T = C_j$, then
   
   \[ w_j \text{(new)} = w_j \text{(old)} + \alpha[x - w_j \text{(old)}]; \]
   
   else
   
   \[ w_j \text{(new)} = w_j \text{(old)} - \alpha[x - w_j \text{(old)}]; \]
   
   \}

   Update learning rate.

\}

LVQ is a supervised learning process, i.e. we have to know the sample's class before we start the learning process. One must assign the appropriate class to it before the training. The learning will definitely be better for supervised learning. Therefore, it is unfair to compare LVQ with the GLA directly since the latter is an unsupervised learning process. The SOFM, which is similar to GLA, is an unsupervised learning process. In terms of the
algorithm and computation, it is similar to LVQ. The structure of SOFM is the same as the LVQ shown in Figure 4.13. Besides, the algorithm of SOFM is close to the LVQ too.

Self Organized Feature Map algorithm (SOFM)

1. Initialization: initial weight, topological neighborhood parameters and the learning rate parameters.
2. While stopping condition is false do
   
   \{ 
   
   For each input vector \( x \),
   
   \{ 
   
   For each \( j \), compute \( D(j) = \sum_{i} (w_{ij} - x_i)^2 \).
   
   Find index \( J \) such that \( D(J) \) is a minimum.
   
   For all units \( j \) within a specified neighborhood of \( J \),
   and for all \( i \): \( w_{ij} (new) = w_{ij} (old) + \alpha[x_i - w_{ij} (old)] \)
   
   \}

   Update learning rate.
   
   Reduce radius of topological neighborhood at specified times.

\}

If we look closely between LVQ and SOFM, it is obvious that they are very similar. The only differences in structure are the class decision and the neighborhood parameter. Therefore, the SOFM is an alternative. The learning rate \( \alpha \) is decreasing. Thus the impact by those further iterations will be getting smaller and smaller. The phenomenon is similar to simulated annealing which is a stochastic relaxation technique. The reason is to reduce the chance of falling into local minima. This stochastic relaxation technique has been introduced to some improved methods of VQ. However, it is already embedded in ANN approaches.
4.3.2 NN approach vs. GL algorithm

An experiment on SOFM was conducted for a comparison with GLA. With the binary decomposition introduced on the next section, ¼ of the residue was used for the codebook training while ¾ of the image size was quantized. A codebook of size 64 with code vector of 4x4 dimension was used. This corresponds to 64 one-dimensional cluster arrays (output neuron). The initialization took 64 samples which were randomly selected as initial codebook (64 initial clusters). Learning rate was taken as 0.6. To keep as similar as LVQ, radius = 1 was set.

The decoded result is

<table>
<thead>
<tr>
<th>PSNR</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>using SOFM</td>
<td>31.15 dB</td>
</tr>
<tr>
<td>using VQ</td>
<td>33.62 dB</td>
</tr>
</tbody>
</table>

The result cannot conclude which method is the winner. It is because the neural network has several parameters to play with which will affect the learning rate as well as the final result. Nonetheless, one point can be concluded is that they are of similar quantization performance. It is not surprising since they both use a similar distortion measurement (Euclidean distance). On the other hand, NN approaches offer more benefits than VQ.

4.3.2.1 Benefits of using neural network approach

- More control on the learning process. e.g. learning parameter and the number of iterations.
• Even though the structure of the result SOFM and a traditional VQ are similar, the neural network approach got lots of research support for efficient hardware implementations.

• Parallel processing is possible if hardware allows.

• If hardware is available, the SOFM is already an encoder, which is different from traditional VQ encoder.

4.3.2.2 Comparison between VQ and SOFM

<table>
<thead>
<tr>
<th>VQ</th>
<th>SOFM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Computation consuming centroids</td>
<td>Simple update formula</td>
</tr>
<tr>
<td>calculation</td>
<td></td>
</tr>
<tr>
<td>2 No convergency control parameter. It</td>
<td>Learning rate parameters, neighborhood</td>
</tr>
<tr>
<td>is based on Lloyd Iteration.</td>
<td>radius.</td>
</tr>
<tr>
<td>3 Codebook design and encoder design</td>
<td>Encoder is already embedded alone with</td>
</tr>
<tr>
<td>may be separate.</td>
<td>codebook design</td>
</tr>
<tr>
<td>4 May fall in local minima</td>
<td>Embedded relaxation process which reduces</td>
</tr>
<tr>
<td></td>
<td>the chance of locking in local minima</td>
</tr>
</tbody>
</table>

Table 4.1: Comparison between VQ and SOFM.

4.4 Summary

In this chapter, we introduce the concept of how quantization contributes to data compression. It starts illustrations from a scalar quantizer. With the generalization, the VQ is introduced which is believed containing superior compression power. Its pros and cons have been discussed. An example has illustrated the general picture of the VQ. Finally, the ANN approach has been discussed and compared with the VQ. Even though there are many advantages to ANN approaches, conventional VQ was used for the experimental work in this thesis. Without hardware support, the encoding process of the SOFM will be
the same as the conventional Nearest Neighbor Encoder. Moreover, attempts have to be tried on the SOFM's learning parameters to yield a good convergence. Eventually, it makes the conventional VQ a simple choice.
5. Solutions to noncausal support: The codec framework

5.1 Introduction

In lattice modeling, it is evident that noncausal support is superior to causal support [41][42]. For the usage in image compression, a lattice modeling technique requires for both a modeling filter and a synthesis filter. That involves a stability requirement for both the modeling filter and the synthesis filter. In the field of digital filter design, most of the research efforts have been concentrated on the synthesis and realization of causal filters. Noncausal digital filters receive far less attention.

For real time processing in 1-D time signals, there is almost no solution for a noncausal digital synthesis filter. The current output depends on the past outputs as well as the future outputs. A recursive computability property is not shared, whereas, with the flexibility of digital computers, the signal can be stored in the computer and block processing can be implemented. The problem is tackled by decomposing the noncausal filter into a causal
subfilter and a purely noncausal subfilter. With the time domain (or spatial domain in 2-D) reversal, the noncausal subfilter can be replaced by the causal equivalent filter [47].

The main problem in the noncausal support is in the synthesis whose transfer function is the reciprocal of the modeling filter transfer function. Unless all the zeros in the modeling filter are inside the unit circle, the reciprocal of the transfer function implies an unstable filter since the zeros in the modeling filter correspond to the poles in the synthesis filter. The method in [47] suggests the splitting of the noncausal filter into purely causal and purely noncausal part, in cascade or in parallel. We then allocate all the problem poles (correspond to being outside the unit circle zeros in the modeling filter) into the purely noncausal subfilter and the rest into the causal subfilter. The purely noncausal subfilter will become a purely causal filter with the time (or spatial) reversal technique.

The aforementioned technique seems only applicable to one-dimensional signals. For two-dimensional systems, the zeros-poles analysis is difficult. This is mainly due to the nature of two-variable polynomials that appear in 2-D transfer functions. The poles and zeros are inseparable in general. In one dimension, poles of $H(z)$ are points in the 2-D complex $z$ plane. In 2-D, the poles of $H(z_1,z_2)$ are surfaces in the 4D complex $z_1,z_2$ space [14].

If the compressed image using prediction modeling cannot be retrieved, it is meaningless to use the noncausal support modeling in image compression applications. The following sections provide two alternative methods that could avoid the synthesis problems encountered in traditional approaches; however, at the same time, they keep the merits of
noncausal support modeling. The first one is using matrix notation and operations. The second one is a kind of multi-resolution decomposition, the binary pyramid decomposition.

5.2 Matrix method

Instead of using the traditional approach in the digital signal processing field, the relationship between the input image and the prediction residue can be obtained by other means. One of the methods is using a transformation matrix which connects the relationship between them. The idea was introduced to a noncausal prediction codec by Balram and Moura [48]. This approach first reorganizes the 2-D image into a column vector. A transformation matrix can be derived from the model parameters. The input image and the output prediction residue will form a matrix equation, $Ax = \varepsilon$, or in other words, a system of $(M.N)$ equations, where $MxN$ is the image size.

5.2.1 The algorithm

For the sake of simplicity, we assume a noncausal support plane with only two parameters to illustrate the idea of using a matrix method. Figure 5.1 shows the picture of the noncausal prediction modeling. We use the four surrounding neighbors to predict the center one.

5.2.1.1 Matrix representation

Let the image be defined on a $NxM$ square lattice $L$ and named as $X$. 
\[ X = \{ x_{ij} : (i, j) \in L \} \]

where \( x_{ij} \) is the value of the image attribute (intensity level) in pixel \((i, j)\).

We stack the pixels values in row \(i\)th in a column vector \(x^i\) and then stack these in a \(N.M\)-dimensional vector \(x\), i.e.

\[
x^i = \begin{bmatrix} x_{i,1} & x_{i,2} & \cdots & x_{i,M} \end{bmatrix}^T \text{ and } x = \begin{bmatrix} x^1 & x^2 & \cdots & x^N \end{bmatrix}^T
\]

![Image](image.png)

Figure 5.1: Support plane.

From the prediction relation, we can write,

\[
x_{i,j} + \beta_v (x_{i-1,j} + x_{i+1,j}) + \beta_h (x_{i,j-1} + x_{i,j+1}) = e_{i,j}
\]

With boundary condition of

\[
x_{ij} = 0, \quad \forall (i, j) \not\in L
\]

Therefore, if we run the indexes \(i\) & \(j\) through all available values, (5.1) becomes a system of equations.
\[ Ax = \varepsilon \]  \hspace{3cm} (5.2)

where \( \varepsilon = [ (\varepsilon^1)^T \ \cdots \ (\varepsilon^N)^T ]^T \) is the NM-dimensional vector of random noise samples \( \varepsilon_{i,j} \) and \( A \) is NMxNM square matrix.

\[
A = A^T = \begin{bmatrix}
B & C & 0 & \cdots & 0 \\
C^T & B & C & 0 & \cdots \\
0 & C^T & B & C & 0 \\
\vdots & \vdots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & C^T & B \\
\end{bmatrix}
\]  \hspace{3cm} (5.3)

where

\[
B = \begin{bmatrix}
1 & \beta_h & 0 & \cdots & 0 \\
\beta_h & 1 & \beta_h & 0 & \cdots \\
0 & \cdots & \ddots & \ddots & \ddots \\
\cdots & \beta_h & 1 & \beta_h & 0 \\
0 & 0 & \beta_h & 1 & 1 \\
\end{bmatrix}
\]  \hspace{3cm} (5.4)

\[
C = \beta_h I
\]  \hspace{3cm} (5.5)

Matrix \( A \) in (5.3) is a symmetric matrix. With this special support plane, i.e. 4 neighbors on the cross positions, and together with the boundary condition, it is a Toeplitz structure. In general, \( A \) is a tri-diagonal block Toeplitz matrix.

With the matrix equation in (5.2), we have already avoided the synthesis filter problem by putting the image and its residue into a revertible relation as long as the matrix \( A \) is nonsingular and its inverse exists. For synthesis, the question is to obtain \( x \) from \( \varepsilon \). Theoretically, use of matrix inversion can obtain \( x \). Moreover, the matrix \( A \) is a huge but sparse matrix (a blockwise diagonal matrix); simpler and more efficient procedures are available.
5.2.1.2 Noncausal to causal transformation

Instead of Gauss elimination, if the matrix is positive definite, Cholesky decomposition can be applied [21]. With Cholesky decomposition, (5.2) becomes,

\[ Ux = w \]  \hspace{1cm} (5.6)

with

\[
U = \begin{bmatrix}
U_1 & \theta_1 & \cdots & 0 \\
0 & U_2 & \theta_2 & \vdots \\
\vdots & 0 & U_{N-1} & \theta_{N-1} \\
0 & \cdots & 0 & U_N \\
\end{bmatrix}
\]  \hspace{1cm} (5.7)

where
\[ U_i^T U_i = S_i \quad \text{and} \quad U_i^T \theta_i = C, \]
\[ S_1 = B, \quad S_i = B - C^T S_{i-1} C \quad 2 \leq i \leq N. \]

There is some significance in the above representation. Compared with (5.2), this upper triangular matrix equation is a recursive or unilateral equivalent representation while (5.2) is a noncausal representation. The transformation into upper triangular form therefore transforms the noncausal representation into causal. In other words,

\[ Ax = e \quad \leftrightarrow \quad Ux = w \]
where \( U \) is upper triangular matrix.

Expanding them into blockwise system of equations,

\[ Ax = e \quad \rightarrow \quad C^T x^{i-1} + Bx^i + Cx^{i+1} = e^i \]  \hspace{1cm} (5.8)
\[ Ux = w \quad \rightarrow \quad U_i x^i + \theta_i x^{i+1} = w^i \]  \hspace{1cm} (5.9)

In (5.8), current row \( x^i \) is in terms of the past row \( x^{i-1} \) as well as the future row \( x^{i+1} \). Therefore, it is a noncausal (or bilateral) regressor. On the other hand, (5.9) relates the current row \( x^i \) with the future row \( x^{i+1} \). This equation represents a causal (or unilateral) regressor.
5.2.1.3 Synthesis: Backward substitution

For synthesis, the question is to obtain $\mathbf{x}$ from $\varepsilon$. $U$ is an upper triangular matrix. To solve equation (5.6), backward substitution can be applied. This eliminates the need for evaluating the matrix inverse of $A$. The procedure starts with the last row $\mathbf{x}^N$.

$$\mathbf{x}^N = G_N^b \mathbf{w}^N \tag{5.10a}$$

For the rest of the row $\mathbf{x}^i$, they follow the following recursive relations,

$$\mathbf{x}^i = F_i^b \mathbf{x}^{i+1} + G_i^b \mathbf{w}^i, \quad 1 \leq i \leq N - 1 \tag{5.10b}$$

where

$$G_i^b = U_i^{-1}$$

$$F_i^b = -U_i^{-1} \theta_i = -G_i^b (G_i^b)^T C.$$ 

These formulae can be obtained by extending the ordinary backward substitution procedure with blockwise vector notation.

5.2.1.4 Modifications for full-plane noncausal support

The above method can extend to more support plane parameters. For instance, an 8 neighbors noncausal support is shown as Figure 5.2. Equation (5.1) becomes (5.11)

$$x_{i,j} + a_{x_{i-1,j-1}} + a_{x_{i-1,j}} + a_{x_{i-1,j+1}} + a_{x_{i,j-1}} + a_{x_{i,j+1}} + a_{x_{i+1,j-1}} + a_{x_{i+1,j}} + a_{x_{i+1,j+1}} = \varepsilon_{i,j} \tag{5.11}$$
The corresponding change in (5.3), (5.4) & (5.5) will be given by:

\[
A = \begin{bmatrix}
B & C & 0 & \cdots & 0 \\
D & B & C & 0 & \cdots \\
0 & D & B & C & 0 \\
\vdots & \cdots & \cdots & \cdots & \cdots \\
0 & \cdots & 0 & D & B \\
\end{bmatrix}
\]  

(5.12a)

\[
B = \begin{bmatrix}
1 & a_5 & 0 & \cdots & 0 \\
a_4 & 1 & a_5 & 0 & \cdots \\
0 & \cdots & \cdots & \cdots & \cdots \\
\vdots & a_4 & 1 & a_5 & \cdots \\
0 & 0 & 0 & a_4 & 1 \\
\end{bmatrix}
\]  

(5.12b)
The matrix A now is still a blockwise tri-diagonal Toeplitz matrix. As the pattern of the algorithm derivations shows, if we increase the support plane vertically (more rows in the support plane windows), the matrix A will introduce more diagonal blocks. As in this 8 neighbors model, the support plane has a upper row support, \((a_1, a_2, a_3)\), and a lower row support \((a_6, a_7, a_8)\). They are corresponding to the diagonal blocks \(D\) and \(C\) respectively.

The more the number of rows in the support plane, the more the number of blockwise diagonals, matrix \(C, D, \ldots\), etc. Similarly, the more the number of columns in the support plane, the more the number of parameters there which must be filled in the corresponding diagonal block matrix, say \(C\) and \(D\) in this case.

The noncausal to causal transformation is changed with a different support plane. The form of matrix \(U\) in (5.7) will be unaltered. Nevertheless, the recursive formulae are slightly different and are shown as follow.
\[
U = \begin{bmatrix}
U_1 & \theta_1 & \cdots & 0 \\
0 & U_2 & \theta_2 & \vdots \\
\vdots & 0 & U_{N-1} & \theta_{N-1} \\
0 & \cdots & 0 & U_N
\end{bmatrix}
\]  

where 
\[L_i U_i = S_i \quad \text{and} \quad L_i \theta_i = C,\]
\[S_1 = B, \quad S_i = B - D S_{i-1}^{-1} C \quad 2 \leq i \leq N.\]

They can be obtained by first using elementary row operations to eliminate \(D\). Then the more general LU decomposition is used to truly force the matrix into upper triangular form.

The Cholesky method requires the matrix to be positive definite. It might not be the case when all model parameters are freely programmed by the model estimation. Therefore, the LU decomposition should be used instead.

### 5.2.2 Matrix method codec

The codec procedure can be outlined as follows.

#### 5.2.2.1 Coding

2. Rearranging pixel data into a \(NM\)-dimensional vector \(x\).
3. Transforming \(A\) into a causal representation \(U\).
4. Generating the transformed residue \(w\).
5. $w$ can be quantized and entropy compressed. Both the parameters and $w$ are for the transmission or storage.

5.2.2.2 Decoding

1. Reconstructing $w$ from entropy coding.

2. Obtaining parametric matrix $A$ from model parameters

3. Transforming parametric matrix into causal representation, $U$.

4. Reconstruction of $x$ from backward substitution.

5.2.2.3 Discussion

This codec structure consists of a huge matrix to be manipulated. Even though using the simplified procedure to calculate the modeling or synthesis, this method is still a computation consuming algorithm. It is mainly due to the size of the matrix. It requires $M$ number of LU (or Cholesky) Decompositions of order $N \times N$. However, if matrix inversion is used instead, it is of order $M \times N \times M \times N$. For a 256x256 image, it is a matrix inversion of 65536x65536 in dimensions.

There is another concern. The Backward Substitution means a recursive relationship which will introduce accumulative error. Coarse quantization may result in a disaster. An experiment using “Karen” 200x200 of dynamic range from 0-1 is shown on Figure 5.3. A noncausal 8-neighbor support plane was used. The residue shown was in the same domain as $e$. A threshold value of 0.01 was set such that any data with absolute values less than the threshold will be set to zero. The modified residue then underwent the matrix
conversion to obtain the recovered image. The original and the “synthesis” images are shown on Figure 5.4. It is obvious that the synthesized image was totally distorted and even could not be recognized. It should be pointed out that it was a really coarse quantization (truncation) which led to about 30% of the data being altered.

![Figure 5.3: a) modeling residue of Karen and b) the distribution.](image)

![Figure 5.4: a) Original Karen 200x200 (left). b) Synthesis result with threshold of 0.01 (right).](image)
For a fair comparison, another experiment was made. In Figure 5.5, it shows the original "Lena" of 128x128 8bits image and its 8-neighbor support modeling residue $e$. The noncausal residue $e$ was transformed into a causal representation, $w$, Figure 5.6. Direct recover of the "Lena" was achieved, Figure 5.7. Despite some rounding error, it is the same as the original "Lena". To illustrate the effect of the quantization on the residue, the transformed error $w$ was quantized with uniform step size of 2. i.e. the allowable levels were ...-4,-2,0,2,4, etc. Figure 5.8 shows the final result. Note that there was some distortion on the recovered image. However, it is a natural consequence of quantization.

Figure 5.5: a) Original Lena 128x128x8 (left). b) Full-plane support residue(right).

Figure 5.6: Transformed causal representation residue, i.e. $w$. 

Figure 5.7: Synthesis result without quantization.

Figure 5.8: Synthesis result with uniform quantization step of size 2.

5.2.3 Summary

A matrix method for the synthesis has been introduced. The method is a rearrangement of a 2-D image into a 1-D column vector. A transformation matrix is used to relate the image to the prediction residue. The complete procedure of the application to noncausal modeling has been covered. Discussion on the method has been included along with some
simulation results in the codec section. It requires M number of LU (or Cholesky) Decompositions of order NxN. The backward substitution recursive formula could introduce accumulative error which would result in a disaster if coarse quantization is applied. However, it is a feasible method for a noncausal codec framework.
5.3 Binary Pyramid Decomposition

Binary pyramid decomposition is a kind of multi-resolution representations. It is similar to the quad-pyramid which is widely used in wavelet subband coders. Image pyramids represent the picture information as a set of bands of increasing size (number of sample points) and bandwidth. The use of binary decomposition was proposed by Robinson [49].

The binary pyramid uses prediction filtering followed by downsampling to yield two channels, $H$ and $L$, each with half the number of points of the original. The prediction filtered $H$ band contains the residue. For $L$ band, the downsampling is done in a special way by removing every even pel on every odd-numbered row and every odd pel on every even-numbered row. This downsampling equivalently rotates the sampling lattice by $45^\circ$.

Figure 5.9: A picture of a two levels Binary Pyramid Decomposition.
Moreover, we can treat it as a lowpass process. The reversed decoding process can treat it as an upsampling followed by interpolation process This operations repeat for further decomposition levels; the lowpass channel is further filtered and downsampled.

Binary decomposition is employed to avoid the synthesis in noncausal modeling. This binary decomposition representation provides a framework for a predictive codec. A lattice modeling prediction can be applied in this framework as the prediction method. On the other hand, the support plane has to have some restrictions so as to allow the decoding of the noncausal residue. Figure 5.10 shows a general available support plane. Non-supported points must intervene between two supported points in the support plane. It is the consequence of the binary decomposition. The allowable support points are the points in the \( L \) band. Since the \( L \) band is available for the prediction, together with model parameters corresponding to the positions in the support plane, the residue and the predicted value can yield the original value without the process of signal synthesis. Therefore, instead of obtaining all the residue at a time, half of the image information is converted into the residue domain in each decomposition. We can continue the decomposition until the \( L \) band's correlated redundancy has all been extracted.
5.3.1 Decomposition procedure

1. Set the image as $L_0$ band, $i = 0$.

2. Obtain subband $L_{i+1}$ by taking every even number pixel in even rows and every odd number pixels in odd rows.

3. Obtain subband $H_{i+1}$ by getting the prediction residue based on the support plane, whose characteristics is shown in Figure 5.10., over the $L_{i+1}$ band. $H_{i+1}$ is to be stored or transmitted instead of the original pixels values.

4. Virtually rotate the whole result lattice grid by $45^\circ$ such that the orientation is back to the direction of the original band grid.

5. Repeat 2-4 until the number of decomposition level is reached.

![](image.png)

Figure 5.11: Pixels block of an image. Each label represents a pixel location.

Figure 5.11 shows a block of an image with every pixel named as L0, the $L_0$ band before decomposition. L1 will be every odd number pixel in odd rows and every even number pixel in even rows. H1 will be the prediction error based on the $L_1$ band, with the appropriate support plane model. The result is a square lattice oriented at $45^\circ$ to the original pixel grid. Or we can pack the downsampled $L_1$ into a 2:1 rectangular array for convenient storage.
Since the binary pyramid decomposition split into two subbands and each subband contains half the image pixels, eventually, the total storage required is the same as for the original image. With \( H \) bands occupying the throw-away original pels during Step 3, the storage can be kept unchanged. Each residue value is responsible for one prediction error recovery. After four decompositions, Figure 5.12 shows a membership of each pel in the original image.

5.3.2 An example

![Image](image.png)

Figure 5.13: An example of two level decomposition. From Lena 128x128.

A 128x128 Lena is used as an example. Two decompositions are shown and the resulting residue, \( H_1 \) band, is put in a 2:1 lattice array for the sake of storage. The right hand side block is the \( H_1 \) band. The \( H_2 \) band is put on the lower left corner in Figure 5.13. Only the
$L_2$ band contains most of the information. Other residue bands will concentrated around zero. They are suitable for quantization and entropy coding.

5.3.3 Summary

A brief overview of the structure of binary pyramid decomposition has been covered. The procedure of decomposition has been reported in details. Unlike the quad-pyramid in a wavelet subband codec which uses complicated quadrature mirror filters, respectively for horizontal highpass, vertical highpass and diagonal highpass, the $H$ band is simply the prediction error instead of real highpass filter. This method's procedure is a structural reorganizing algorithm rather than a mathematical solution to a synthesis problem. The binary pyramid has similar properties as the quad-pyramid. They are multi-resolution presentations that can perform a progressive refining resolution during transmission.

About the compression aspect, the quantization on the lower level $H$-band (larger in size) can be coarser than the other. It is because the encoding decomposition is recursive; it is the same for decoding. The lower the level, the less recursive relations it has. Thus quantization error in higher level $H$-band influence more than the lower level $H$-band. As a result, the lower level $H$-band, e.g. $H_1$, allows coarser quantization than the other.
Chapter 6

6. The codec.

In this chapter, methods mentioned in the previous chapters are integrated together to achieve a good compression result in the application of lattice modeling. It was shown in previous chapters that noncausal lattice modeling is better than causal, vector quantization yields greater compression ratios, and the binary pyramid decomposition allows noncausal modeling. Finally, residue can be benefited from entropy coding. These methods are combined together in this chapter. The details are described and simulation results are reported and compared.

6.1 The Codec Structure

![Figure 6.1: Lossy encoder functional block diagram.](image-url)
Figure 6.1 shows the lossy codec employed for the noncausal lattice modeling. The image is input into the lattice modeling-binary decomposition stage. The binary pyramid decomposition and noncausal lattice modeling are interacting since each band needs one model estimation. Part of the residue has undergone the vector quantization. Finally, the Huffman coding stage further compresses the data.

![Diagram of decoder functional block diagram.](image)

**Figure 6.2: Lossy decoder functional block diagram.**

The decoder is merely the reversion of the encoder. The decoding stages are more simple than the encoding stages. For vector quantization, the decoding is only a table-look-up procedure while the encoding involves code vector matching. In the binary pyramid decomposition, the decoding only requires the model parameters and therefore there is no lattice modeling stage in the decoder. The decoder is shown in Figure 6.2. For the lossless codec, all the functional blocks and stage sequences are similar to those of the lossy codec. Only the vector quantization stage is skipped. Figure 6.3 shows the functional blocks of both the encoder and decoder.
Figure 6.3: Lossless codec structure.

6.2 The implementation and procedure of the codec

The detailed procedure in each functional block has been depicted in previous chapters. In this section, the procedural outline of the encoding and the decoding are given. The simulations are done with the assistance from both C/C++ and Matlab. Matlab gives the short development time for algorithms with many mathematical operations, especially those which can utilize matrix manipulations. C/C++ provides flexibility in data structure and fast computational speed. It is beneficial to introduce merits from both tools. However, the price is the burden in the interface between using these two environments. Unless your Matlab has included the feature of Matlab to C Compiler which can bend Matlab code into a C program, you have to interface your result from one to another.
the following, besides the outlines, footnotes are included with some MATLAB/C++ interfacing techniques that have been used.

6.2.1 Encoder

1. Input image

Images used are assumed to be 256 gray levels images 8-bit Window Bitmap format with each dimension divisible by 16. Each image is fed into Matlab using BMPREAD from image tool box from Matlab. The indexed image is converted into a gray-scale image with range from 0 to 255. Now the image is a matrix with each element representing a pixel.

2. Lattice modeling - Binary pyramid decomposition

On each $L$, obtain the lattice model parameters and the corresponding residue for $H$ band. Pixels of residue in each $H$ band are quantized into the nearest integer presentation. Depending on the presentation of the data, the support plane is different for different odd and even levels of $L$ band. Starting from $L_0$ for decomposition of $L_1$ and $H_1$, assuming $L_0$ is of 1:1 in dimension, a rectangular storage is employed such that the $L_1$ and $H_1$ are squeezed into two ratio of 2:1 rectangles. Nevertheless, the overall size of $L_1+H_1$ is equal to $L_0$. Support plane for modeling $L_0$ is on cross position, i.e. 2,4,6 and 8 used to predict pixel at position 5. When the decomposition continues, the support plane for $L_1$ modeling is of 1,2,7 and 8 since $L_1$ has been squeezed into a 2:1 storage. This time, the modeling is for the $H_2$ band. After downsampling from $L_1$, $L_2$ is covered into 1:1. In general, the modeling on the even $L$ band is on the cross position while that on the odd $L$ band is on 1,2,7 and 8. Refer to Section 5.3 for the decomposition procedure. Eight levels of decomposition are used. The orientations of all the $H$ bands and a $L$ band are shown on Figure 5.13.
3. **Vector quantization**

Depend on lossy or lossless codec, VQ appears only on the lossy compression. Besides, because of the recursive relationship between decomposed bands, coarse quantization cannot be applied to the last few bands. Therefore, here we only vector quantize the two largest bands, $H_1$ & $H_2$. Before the encoding process, some samples are selected to obtain a generic codebook for the compression. GLA is used. The window size for each “vector” is 4x4. It is represented as a column vector in Matlab. i.e. a 4x4 window is arranged into a 16x1 vector. For encoding an input image matrix, firstly, it converts each 4x4 block in the 2-D image into a 16x1 vector. All these column vectors are united accordingly to form a new matrix 16xW for the input of Matlab function, where $W=MxN/16$ if the input image is MxN.

4. **Run-length coding**

A binary tree structure is employed to organize the binary pyramids such that the zeros appears in all those $H$ bands can be closely organized. The same concept of run-length code can then apply to code the end of the tree node. Therefore, instead of a block of zeros, a end leaf code is assigned to note it is the end of a branch. To turn the binary pyramid into a tree, each parent (a pixel) in level $H_n$ is associated with pairs of children in level $H_{n-1}$. The following statements refer to the presentation shown as Figure 5.12.

- When $n$ is even, each pixel in level $H_n$ has one child on $n/2$ rows above in the same column, and one child $n/2$ columns to the left in the same row.

- When $n$ is odd,
  - if a pixel’s parent is to its bottom, then its children are $(n-1)/2$ rows below it and $(n-1)/2$ columns to its left and right.
5. Huffman coding

Because of the restriction of the Matlab data structure, the implementation of Huffman coding is almost impossible. C/C++ is used to implement the simulation. To cope with the interface of the run-length coding and Huffman coding with the preceding stages, the residue, which with presentation shown as Figure 5.13, in Matlab environment is written into an ASCII format file. This file is then input into a C program. A class 2DDataField were written for this text input. An object of this class can be instantiated by giving a file name. Data are read into a two-dimensional dynamic-allocated array. For building a tree from the data, a class BTree was built. It inherited from class 2DDataField so as to handle the 2-D data. For run-length coding, end-leaf codes are added when all the children are zeros in a branch. A recursive member function in class BTree is responsible for walking through the residue binary tree. It walks to another branch when it is the end of the branch or a leaf-code is encountered. The Huffman coding is a two pass process, one for building up the code table and the other for coding the symbol accordingly. Therefore, there are three walks for the binary tree, one for marking the leaf-code, one for building up the code table and one for the actual encoding. Besides, for lossy coding, the VQ index data can be Huffman coded too. The VQ index is stored as a row vector. A separate Huffman codec is used for coding this data.

6.2.2 Decoder

1. Huffman decoding

The process sequence is exactly the reverse of the Huffman coding except only one pass is enough. The decoding process first retrieves the overhead information from
the file stream, which includes the table of the Huffman coding. Since it is a variable code length codec, the decoding is not just a table-look-up procedure. Instead, an automaton can be built according to the code table. This automaton is in a binary tree structure.

2. Run-length decoding

Zeros are added back to the appropriate positions during Huffman decoding. The decoded sequences are synchronized with the binary tree walk. Zeros are inserted for each leaf-code during the walk.

3. Decoding of Vector quantization

The decoding of the VQ indexes are just a table-look-up procedure. After the Huffman & run-length decoding stages, the output is put in an ASCII format file. To read into Matlab, command load with parameter -ascii is used.

4. Binary pyramid composition

After constituting back all the bands from the preceding stages, the reverse process of the binary decomposition can be proceeded. It starts from the highest \( L \) band. Here, it is \( L_8 \). Firstly, \( L_8 \) is up-sampling. The pixels are positioned back to the original place. The intervening pixels are filled with the predicted values from using the neighbors and the corresponding model parameters. It is equivalent to a 2-D FIR filter. \( H_8 \), which is the prediction error, is used to compensate the predicted value toward the original true value. The result band is \( L_7 \). The process repeats until the final \( L_0 \) band is obtained. \( L_0 \) is the output decoded image.

6.3 Results comparison

The simulations are divided into lossless compression and lossy compression. Some other codecs are introduced for comparison.
6.3.1 Lossless compression

Six other coders were used for comparison. There are two categories: four image coders and two general-purpose data coders. In the image coder categories, there are CALIC, GIF, TIFF and Deluxe Paint's format. The first one are chosen as the representative of the state of the art in lossless photographic image compression while the later three are the commercial available image coder. CALIC [50] was proposed for the new ISO lossless image coding standard and performed the best in the first round of evaluation. In the general data compression coder, PKZIP and ALDC from IBM were chosen because of the popularity and availability. Figure 6.4 shows a typical relative performance of the coders on Lena 512x512 8 bits per pixel image.

![Lossless Compression of Lena](image)

*Figure 6.4: Lossless Compression of Lena.*

*Remark: Some formats have more than 8 bpp is because of their format overhead.*

To have a more comprehensive comparison, nine images were used. The results are tabulated as below, Table 6.1., and was plotted on Figure 6.5 as well. It could be seen from the chart that different images have different bits per pixel. The fact was due to
different images had different information contents. With more redundancy in an image, lower the bpp value could be achieved.

Figure 6.5: Lossless compression result.
### Table 6.1: Lossless compression result of 8bpp images.

<table>
<thead>
<tr>
<th></th>
<th>size</th>
<th>CALIC</th>
<th>Lattice</th>
<th>TIFF</th>
<th>GIF</th>
<th>Deluxes Paint</th>
<th>PKZIP</th>
<th>IBM (ALDC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARB</td>
<td>512x512</td>
<td>6.57</td>
<td>5.39</td>
<td>8.85</td>
<td>8.78</td>
<td>7.63</td>
<td>6.94</td>
<td>8.20</td>
</tr>
<tr>
<td>BIRD</td>
<td>256x256</td>
<td>3.49</td>
<td>4.35</td>
<td>6.22</td>
<td>5.80</td>
<td>6.74</td>
<td>4.99</td>
<td>5.94</td>
</tr>
<tr>
<td>BKMIST</td>
<td>256x256</td>
<td>3.56</td>
<td>3.88</td>
<td>4.86</td>
<td>4.32</td>
<td>6.59</td>
<td>3.99</td>
<td>4.86</td>
</tr>
<tr>
<td>BOAT</td>
<td>512x512</td>
<td>4.23</td>
<td>5.11</td>
<td>7.34</td>
<td>7.22</td>
<td>7.02</td>
<td>6.06</td>
<td>7.18</td>
</tr>
<tr>
<td>GOLDFILL</td>
<td>256x256</td>
<td>5.25</td>
<td>5.90</td>
<td>8.54</td>
<td>8.36</td>
<td>7.72</td>
<td>6.90</td>
<td>7.94</td>
</tr>
<tr>
<td>LENA256</td>
<td>256x256</td>
<td>4.50</td>
<td>5.32</td>
<td>8.63</td>
<td>8.43</td>
<td>8.00</td>
<td>6.96</td>
<td>8.03</td>
</tr>
<tr>
<td>LENA512</td>
<td>512x512</td>
<td>4.18</td>
<td>4.88</td>
<td>8.21</td>
<td>8.08</td>
<td>7.54</td>
<td>6.58</td>
<td>7.98</td>
</tr>
<tr>
<td>MANDRILL</td>
<td>512x512</td>
<td>5.98</td>
<td>6.38</td>
<td>9.40</td>
<td>9.36</td>
<td>8.02</td>
<td>7.20</td>
<td>8.54</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>512x512</td>
<td>4.45</td>
<td>5.02</td>
<td>8.24</td>
<td>8.11</td>
<td>7.37</td>
<td>6.76</td>
<td>7.96</td>
</tr>
</tbody>
</table>

In the table, "Lattice" refer to the noncausal lattice modeling method. When it was applied to a background style picture BKMIST, i.e. a relatively more homogeneous pattern, the modeling method got a very close performance to CALIC coder. It was depicted in Figure 6.6. Also shown in the same figure are the comparison of Lena 256x256 to Lena 512x512. The smaller Lena picture required more bits per pixel to code. In other words, the bigger the image, the less the number of bits were required. It is because bigger pictures

![Figure 6.6: Lossless result; Lena512, Lena256, BKMIST.](image-url)
contained more redundancy. The picture of Lena256x256 was obtained by downsampling the 512x512 picture. Thus the pixels correlation has already been reduced. It was the natural consequence that the coding required more bpp.

6.3.2 Lossy compression

6.3.2.1 Different combinations in lattice method

In lossy compression, vector quantization was introduced to achieve further compression after the prediction modeling. Moreover, besides $H_1$ & $H_2$ being vector quantized, the $L_8$ band and the indexes from VQ were being entropy coded. The following data, Table 6.2&

<table>
<thead>
<tr>
<th></th>
<th>$L_8$ band entropy coded &amp; VQ were stored directly</th>
<th>$L_8$ band &amp; VQ were stored directly</th>
<th>$L_8$ band stored directly &amp; VQ were entropy coded</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARB</td>
<td>1.22</td>
<td>1.17</td>
<td>1.02</td>
</tr>
<tr>
<td>BIRD</td>
<td>1.03</td>
<td>0.99</td>
<td>0.81</td>
</tr>
<tr>
<td>BKMIST</td>
<td>0.96</td>
<td>0.93</td>
<td>0.72</td>
</tr>
<tr>
<td>BOAT</td>
<td>1.19</td>
<td>1.15</td>
<td>1.01</td>
</tr>
<tr>
<td>GOLDHILL</td>
<td>1.27</td>
<td>1.22</td>
<td>1.15</td>
</tr>
<tr>
<td>LENA256</td>
<td>1.25</td>
<td>1.20</td>
<td>1.07</td>
</tr>
<tr>
<td>LENA512</td>
<td>1.12</td>
<td>1.08</td>
<td>0.91</td>
</tr>
<tr>
<td>MANDRILL</td>
<td>1.43</td>
<td>1.38</td>
<td>1.32</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>1.09</td>
<td>1.06</td>
<td>0.91</td>
</tr>
</tbody>
</table>

Table 6.2: Lossy lattice method comparison.

Figure 6.7, show how the further compression being achieved. The combinations were: 1) whether the $L_8$ band was entropy coded with the rest of $H$ bands, or stored directly. 2) whether the indexes of the VQ were entropy coded or stored directly. The result revealed that the VQ indexes also had a distribution that benefited the entropy coding. This implied many of the VQ indexes fell into certain symbols. That further implied many parts of the residues were looked alike. This further entropy coding gained about 0.1 bpp margin.
6.3.2.2 Comparison with JPEG

For lossy compression, JPEG was chosen as the benchmark. It is a transform coding method. JPEG employs DCT to transform a 8x8 window into frequency domain. DCT is a one-to-one mapping. Without quantization in frequency coefficients, no data loss in the spatial domain, whereas, no compression was the consequence. Compression was gained from the quantization on the coefficients. The quantized coefficients were then run-length coded and eventually Huffman coded. Results of the comparison were put in the following Table 6.3 & Figure 6.8.
<table>
<thead>
<tr>
<th></th>
<th>Lattice</th>
<th>JPEG</th>
<th>PSNR (dB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BARB</td>
<td>1.02</td>
<td>0.95</td>
<td>28.40</td>
</tr>
<tr>
<td>BIRD</td>
<td>0.81</td>
<td>0.55</td>
<td>31.81</td>
</tr>
<tr>
<td>BKMIST</td>
<td>0.72</td>
<td>0.45</td>
<td>33.46</td>
</tr>
<tr>
<td>BOAT</td>
<td>1.01</td>
<td>0.78</td>
<td>31.73</td>
</tr>
<tr>
<td>GOLDHILL</td>
<td>1.15</td>
<td>1.13</td>
<td>29.28</td>
</tr>
<tr>
<td>LENA256</td>
<td>1.07</td>
<td>0.91</td>
<td>31.53</td>
</tr>
<tr>
<td>LENA512</td>
<td>0.91</td>
<td>0.67</td>
<td>32.60</td>
</tr>
<tr>
<td>MANDRILL</td>
<td>1.32</td>
<td>1.20</td>
<td>26.53</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>0.91</td>
<td>0.70</td>
<td>31.31</td>
</tr>
</tbody>
</table>

Table 6.3: Comparison with JPEG.

The PSNR were measured from the lattice method decoded image. The JPEG coder bits-per-pixel value was then adjusted until the PSNR were similar. In this experiment, the VQ codebook used was a generic one. It was only trained from residues of three images. Moreover, only 64 code vectors might also be the cause of the resulting low PSNR.
6.3.3 Summary

The lossless coding of the noncausal lattice modeling got comparable result with the state of the art coder. In contrast with some general-purpose coders, this noncausal lattice modeling with binary decomposition showed superior performance. Taking "Peppers" as an example, they are 5.02, 6.76 and 7.96 bpp from lattice method, PKZIP and ALDC respectively. A bigger picture usually contains more redundancy. As a result, smaller bit-per-pixel value can be achieved. For an example, "Lena" 256x256 required 5.32 bpp while 512x512 required only 4.88 bpp. It can be seen from the information contents point of view. For the lossy compression, several combinations for optimization have been tried. However, the result from the lattice method did not beat the JPEG coder although they are of similar order, for instance, 512x512 "Lena" image, lattice method required 0.91bpp while JPEG required 0.67 bpp. In general, JPEG bpp rate is ranging from half to two third of the lattice method's. Noncausal lattice modeling provided just a framework; the compression required further schemes or methods. Similar to JPEG using DCT as the backbone of the method, the compression was achieved by utilizing quantization, run-length code and Huffman code within an optimized scheme. Prediction coding is primarily strong on lossless coding. This attempt is to demonstrate the availability of noncausal predictive coding on lossy image compression.
Chapter 7

7. Conclusion

7.1 Conclusion

Building on the success of 1-D modeling techniques on speech and audio compression, this thesis explores the potential of modeling techniques on 2-D signal - images. The objective of this thesis is to investigate the lattice modeling technique and its application to image compression. A noncausal support AR orthogonal lattice model is used because of its superior performance. Causal AR is a classical model for linear prediction, whereas, research shows a noncausal support is a better choice. However, noncausal support modeling suffers from the inherent difficulties of noncausal filters.

To avoid the synthesis problem that is encountered for noncausal support, this thesis presents two alternative methods: Binary Pyramid Decomposition and Matrix method. Matrix method is a computation intensive algorithm. It requires $M$ number of LU decompositions with each of order $N \times N$. This does not include the matrix block multiplication. Moreover, the backward substitution recursive formula could introduce
cumulative error which would result in a disaster if coarse quantization is applied. Another alternative, Binary Pyramid Decomposition, is a more preferable choice. This method provides a framework for a noncausal predictive codec. Its algorithm employs downsampling followed by prediction residue calculation. Depending on the support plane, the main calculation burden is similar to a FIR modeling filter. On the contrary, one drawback is that the support plane has to be in a special form – which includes intervening unsupported pixels. This drawback significantly reduces the power of noncausal support modeling. Similar to the DCT as the backbone of JPEG compression, noncausal lattice modeling provides a prediction coding framework. Further schemes need to be applied to achieve compression.

For lossless compression, a run-length coding and Huffman coding were used. Huffman coding is one of the entropy coders. Compared with Arithmetic coding, Huffman coding can be analogous a discrete code length implementation of the entropy theory while Arithmetic coding is a continuous code length implementation. Even though Arithmetic coding can achieve better compression ratios, its implementation burden limits its popularity [1]. In fact, it merely achieves 5-10% better compression than that of the Huffman coder [49].

For lossy compression, VQ has been employed. VQ is attractive for its high compression ratio but it has inherent blockiness artifacts. Application of VQ on residue domain has shown reduction on the blockiness effect and a better PSNR. The reason behind this characteristic is owing to the averaging effect on the decoding FIR filter. The reduced blockiness artifacts might be the consequence of the binary pyramid decomposition. Since the pixels are shuffled from band to band during decomposition, the blockwise boundary
edges are shuffled away when recovered from the decomposed bands. This is an additional
merit of using binary pyramid decomposition.

While the lossy performance is not superior to the main stream bench marks, the lossless
compression shows comparable results to the state of the art lossless coder. The good
performance in lossless compression reveals that the 2-D noncausal lattice modeling can
effectively reduce the redundancy in images. It is a good backbone for image compression.
About lossy compression, the performance measure depends not only on redundancy
extraction power, but also on the visual perception of the discarded information. An
appropriate scheme is the one that can discard information which is not sensitive to human
vision.

In conclusion, noncausal predictive lattice modeling is another backbone for image codec.
Its availability has been shown. The effectiveness and the competitiveness of the codec
have been justified.

7.2 Future work

As can be seen from the compression performance, this method still needs to be improved
for higher compression. There are some limitations from getting better results.

One of the successes in JPEG is that the zero coefficients are well organized. A zigzag
scanning scheme can group most of the zeros together. A run-length code is efficient in
JPEG compression. For residue, the zeros are spread all over the whole image. A scheme
which can arrange all the zeros together will definitely improve the compression ratio.
The suggested two alternatives to avoid synthesis problem have their pros and cons. They both put restrictions on the codec. The ultimate goal is to be able to perform a stable synthesis process. By then, the support plane can have all the surrounding pixels and all the residue can be coded directly instead of just $H$ bands in binary pyramid decomposition.

From the theory point of view, the LSE measure is the principle behind the whole algorithm. It is a macroscopic measure. LSE on modeling is initially introduced from statistician. For images, homogeneous regions are scattered from place to place. It is hard to find any image with one homogeneous region. Future work in this direction can be exploring the effect of separate modeling on some local homogeneous regions. For this, region identification, segmentation technique, organization scheme, etc., will be involved.
8. References


VITA AUCTORIS

NAME: Raymond, Chi Ho, CHAN

PLACE OF BIRTH: Hong Kong

DATE OF BIRTH: April, 1970.

EDUCATION: B.Eng.
Department of Electronic Engineering
Hong Kong Polytechnic
Hong Kong, China,

M.A.Sc.
Department of Electrical Engineering
University of Windsor
Windsor, Ontario, Canada,
1995-1997