2005

Adaptive edge-based prediction for lossless image compression

Rahul G. Parthe
West Virginia University

Follow this and additional works at: https://researchrepository.wvu.edu/etd

Recommended Citation
https://researchrepository.wvu.edu/etd/4183

This Thesis is brought to you for free and open access by The Research Repository @ WVU. It has been accepted for inclusion in Graduate Theses, Dissertations, and Problem Reports by an authorized administrator of The Research Repository @ WVU. For more information, please contact ian.harmon@mail.wvu.edu.
ADAPTIVE EDGE-BASED PREDICTION FOR LOSSLESS IMAGE COMPRESSION

by

Rahul G Parthe

Thesis submitted to the College of Engineering and Mineral Resources at West Virginia University in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Approved by

Dr. Donald Adjeroh, Committee Chairperson
Dr. Xin Li
Dr. Arun Ross

Lane Department of Computer Science and Electrical Engineering

Morgantown, West Virginia 2005

Keywords: Lossless Image Compression, Edge based predictor, Least-Square, Context based error encoding, BWT, F2BWT, alpha-path encoding.

Copyright 2005 Rahul G. Parthe
Abstract

ADAPTIVE EDGE-BASED PREDICTION FOR
LOSSLESS IMAGE COMPRESSION

by Rahul G. Parthe

The increasing use of images in a variety of applications has heightened the need for methods to compress an image. Image compression is needed in efficient management of the two main resources in computing systems; space and communication time. Computer based data compression has been practiced for decades and image compression has been studied since the 70’s. Thus compression algorithms are slowly approaching their theoretical limit. Image data modeling in particular, is very difficult due to the higher order correlations that exist in natural continuous images. Though the established compression results are hard to surpass, there are some aspects that can be considered to achieve better performance. The gain might appear small, but still significant, especially when considered from the viewpoint of increasing image size of available images.

In this work, we study two aspects of lossless compression where these improvements can be made. The research is focused on lossless image compression technique based on the traditional two-phase (prediction and encoding) method. Each module is studied separately and new techniques are suggested which lead to a better performance compared to other state-of-the-art technologies.

In the prediction module, we propose two different edge-based prediction schemes. Edge Based Predictor (EBP) and Least-Squares Edge Based Predictor (LS-EBP) place emphasis on edges in the image and make predictions accordingly. EBP is a gradient based nonlinear adaptive predictor. It switches between different prediction-rules based on few threshold parameters determined by a pre-analysis procedure. The pre-analysis procedure makes a first pass on the image and automatically determines a set of parameters that are further used by the prediction scheme to improve their decision-making. The LS-EBP predictor also works in conjunction with the pre-analysis method, but optimizes the prediction for each pre-analysis assigned edge location. Unlike in traditional LS based methods, LS-EBP applies the least-square approach only at the edge points in the image. The result is an adaptive scheme with improved compression and a significant reduction in the overall computation.

For the encoding module, a novel method is proposed for image residual error encoding. The Burrows Wheeler Transform (BWT) inspires this method. This method performs better than applying the BWT directly on the images. We also present a context based adaptive error modeling and encoding scheme. When coupled with the above-mentioned prediction schemes, the result is the best-known compression performance in the genre of compression schemes with same time and space complexity.
Dedication

I dedicate my thesis to my parents

Ganpat Parthe
Shakuntala Parthe
Acknowledgments

I would like to mention my gratitude and express my acknowledgements to all the people who have supported me. I would like to put this in words without missing a chance to mention how fortunate I am to have all these people around. I would like to thank all my professors, friends and family members without whom this difficult task wouldn't have been completed.

It was my honor to have Dr. Donald Adjeroh as my advisor and committee head. His knowledge in the field of data compression has always guided me in the right direction. His timely suggestions and observations have helped me to get in to the roots of the problem and find a systematic solution. His attention to detail, quest for excellence, love for perfection, ingenious ideas and understanding has always motivated me to give my best time and again. He has helped me develop and tune my academic skills including research, presentation and writing skills. I am highly obliged to have an advisor like him who has made my Masters program a very good experience.

I am very grateful that I had Dr. Xin Li, Dr. Arun Ross and Dr. Elaine Eschen’s support, in addition, which was also a very valuable contribution towards my thesis. Their expert guidance and suggestions have given me good ideas and solutions for my thesis.

I would also like to extend my gratitude to all my colleagues who have been with me all the time and made the environment exactly suitable for research endeavors. I would like to specially thank all Lane Department of Computer Science and Electrical Engineering professors for conducting quality research and course work. Contribution by the department secretaries and systems staff has also played a key role in the completion of my thesis.

Last but not the least; I would like to thank my family members, my in laws and my fiancé Ashwini Naik for their support, love and sacrifice. It was because of them I could survive hard times and be consistent on my work.
Table of Contents

Abstract.............................................................................................................................ii
Dedication........................................................................................................................iii
Acknowledgements .........................................................................................................iv
Table of contents ............................................................................................................. v
List of Figures .................................................................................................................. vi
List of Tables...................................................................................................................vii
Chapter 1: Introduction .................................................................................................... 1
  1.1 Introduction......................................................................................................... 1
  1.2 Problem and Motivation ..................................................................................... 4
  1.3 Contributions of the Thesis .............................................................................. 6
  1.4 Organization of the Thesis .............................................................................. 9
Chapter 2: Background ................................................................................................. 11
  2.1 Data Compression ........................................................................................... 11
  2.2 Image Compression ......................................................................................... 12
  2.3 BWT .................................................................................................................. 17
      2.3.1 BWT Functionality ................................................................................... 18
Chapter 3: The Edge Based Predictor .......................................................................... 24
  3.1 Introduction....................................................................................................... 24
  3.2 Image Characterization and Parameter Calculation ....................................... 27
  3.3 Prediction Based Gradients ............................................................................. 34
  3.4 Binary Mode Switching .................................................................................... 38
  3.5 Bias Cancellation using Context Modeling and Error Feedback ..................... 39
  3.6 Symbol Remapping .......................................................................................... 45
  3.7 Error encoding .................................................................................................. 46
Chapter 4: Edge Based Least Square Predictor .......................................................... 48
  4.1 Introduction....................................................................................................... 48
  4.2 Pre-analysis ...................................................................................................... 49
  4.3 Lease-Square Method for Prediction .............................................................. 50
  4.4 Prediction based on Switching ........................................................................ 53
Chapter 5: BWT and Image Encoding .......................................................................... 56
  5.1 F2BWT ............................................................................................................. 56
      5.1.1 F2BWT- Unique ID ................................................................................ 58
      5.1.2 F2BWT-Path Definition ......................................................................... 63
  5.2 α-paths ............................................................................................................. 64
  5.3 Image compression using α-paths ................................................................... 65
Chapter 6: Results ......................................................................................................... 69
  6.1 Pre-analysis Parameters ................................................................................. 69
  6.2 Predictor Coefficients ...................................................................................... 71
  6.3 Predictor Performance ...................................................................................... 72
  6.4 Results using Context Based Methods ........................................................... 77
  6.5 Encoding Methods ........................................................................................... 79
Chapter 7: Conclusion ................................................................................................... 81
References.....................................................................................................................84
# LIST OF FIGURES

Figure 2.1: N x N conceptual matrix $M'$ formed by N cyclic shifts on string $S^N$ .......... 19  
Figure 2.2: The lexicographical sorted matrix $M$ ......................................................... 20  
Figure 3.1: Basic block diagram of the Edge based predictor scheme ......................... 25  
Figure 3.2: The context window structure used for gradient calculation .................... 28  
Figure 3.3: Calculating $\xi$ (direction threshold) using pinch point $\omega$ .................... 31  
Figure 3.4: Calculating $C_t$ (threshold) parameter.......................................................... 33  
Figure 3.5: The quantization scheme for the error energy parameter ......................... 41  
Figure 4.1: Context window structure for least square neighbors ............................ 51  
Figure 4.2: Context window dimensions for least square context ............................. 51  
Figure 5.1: Tree construction and traversal................................................................. 62  
Figure 5.2: Zigzag scanning path used with $\alpha$-paths for images .......................... 67  
Figure 5.3: Context template for raster scan based $\alpha$-path error encoding............ 67  
Figure 6.1: Residue images for CAMERA image after different prediction schemes .. 75  
Figure 6.2: Residue images for PEPPERS image after different prediction schemes 76
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Results with $(\xi_1, \xi_2)$ determined automatically and $C_1$ varied</td>
<td>70</td>
</tr>
<tr>
<td>2</td>
<td>Results with $C_1$ determined automatically and $(\xi_1, \xi_2)$ varied</td>
<td>70</td>
</tr>
<tr>
<td>3</td>
<td>Coefficients $\alpha_1$ and $\alpha_2=\alpha_3$ set to constant values</td>
<td>72</td>
</tr>
<tr>
<td>4</td>
<td>$1^{st}$ order entropy for various state-of-the-art predictors</td>
<td>73</td>
</tr>
<tr>
<td>5</td>
<td>Higher order entropies for CAMERAMAN image</td>
<td>74</td>
</tr>
<tr>
<td>6</td>
<td>Higher order entropies for NOISESQUARE image</td>
<td>74</td>
</tr>
<tr>
<td>7</td>
<td>Higher order entropies for PEPPERS image</td>
<td>74</td>
</tr>
<tr>
<td>8</td>
<td>Comparative results for various techniques, including the proposed method</td>
<td>77</td>
</tr>
<tr>
<td>9</td>
<td>Comparing the proposed methods with EDP</td>
<td>78</td>
</tr>
<tr>
<td>10</td>
<td>Execution times for three predictors with MATLAB implementation</td>
<td>78</td>
</tr>
<tr>
<td>11</td>
<td>Prediction results for Integer wavelet based transforms</td>
<td>79</td>
</tr>
<tr>
<td>12</td>
<td>Results using different encoding schemes</td>
<td>80</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

1.1 Introduction

Data compression has been an important research challenge in the field of information theory for decades. Intense research in this field has been motivated by economic and logistics needs to conserve space and to save bandwidth. The main objective for development of compression techniques is to achieve high compaction performance and to maintain good quality of data. The quality measure of the recovered data after decompression leads to various types of compression techniques, the two major classifications being lossy and lossless data compression. Data recovery and searching on compressed data has also modulated the process of designing data compression algorithms. Even though the general Huffman’s algorithm [1] was introduced in 1952, increasing demands of real world data processing applications have made it necessary to invest computing resources in the design of data dependent adaptive compression techniques [2,3,4].

Since the invention of computers, digitized images have formed a major class of data source, which requires various kinds of processing including compression. By the nature of images, the picture elements (i.e. the pixels) in a local region are highly correlated with each other. In other words, the digitized image contains a lot of redundancy and therefore is inefficient for direct representation, storage, processing or communication. For this reason, image compression techniques are introduced to reduce the amount of data needed to represent the same information, either exactly or approximately. Hence compression can be defined as the process of changing the representation of data from one form to another with a higher information density. Compressors are either lossless (also known as noiseless, exact, invertible or information preserving) or lossy (approximate). The lossless compressors change their input into a form from which an exact replica of the original data can be reconstructed [5]. The lossy ones make some value judgment as to which
information is important for the intended application and discard the rest. Lossy compression can be used where the compression ratio rather than exact reconstruction of the data is the major issue. These types of techniques are more popular due to applications such as Internet video, mobile phones and entertainment archives. Lossless compression is employed in cases where the data needs to be further processed (example feature extraction), intensive editing or even repeated compression/decompression. Medical and judiciary records are usually compressed in a lossless manner.

The approaches for data compression can be distinctly classified into three major classes.

1) Transform coding methods, which result in energy packaging [6,7,8,9]. In this approach a reversible transform is applied on the image, which results in a set of coefficients that are further processed and compressed. Transforms like Discrete Cosine Transform, Discrete Wavelet Transform and Discrete Sine Transform have gained popularity and are frequently used. The truncation of the coefficients usually results in a lossy compression but the achieve loss less transform the error feedback is implemented. The truncation error is also included in the compression stream. These methods are fast but do not produce best compression results.

2) Vector quantization, which uses representative vector codebooks to provide compression [10,11,12]. These methods are used for both lossy as well loss less compression.

3) Finally prediction based image error encoding methods [13,14]. The focus of this thesis is on prediction-based techniques.

Prediction based image compression is generally a two-phase process: the first phase is the prediction process/method, and the second is the error modeling and encoding. Prediction is done based on the previously encoded data [15], which gives information about the local characteristics of the image area using some non-adaptive fixed rule set. The simplest technique used as a predictor is the Differential Pulse Code Modulation (DPCM) [16], which assumes the prediction for the current pixel to be the value of the previous pixel in raster scan. The error generated for encoding is thus calculated using the difference
between the current and previous pixel. Even though the technique is simple and fast it results in extremely low compression rate. The reason for this loss in efficiency is due to the simple prediction model used which does not exploit the intricate correlation in the image data. An improved version of adaptive DPCM was also introduced but it still lacked a good modeling ability [17,18]. A successful model for prediction and compression of natural images must efficiently and adaptively treat the various types of commonly occurring features, namely: edges, smooth regions, and textures. Edges represent abrupt changes in intensity that persists along straight or curved contours. Smooth regions are characterized by slowly varying intensities; textures contain a collection of localized intensity changes. Edges are of particular interest for compression since they are difficult to model and make predictions less reliable [19,20]. Due to the abrupt nature of the edges, simple non-adaptive predictor performance is poor since the edge model is not taken into account. If predictions are made across the edge the error is high (i.e. less predictable) and hence inefficient to compress. So the key to a better predictor is to design a technique that can estimate the orientation of the edge adaptively so that prediction can be made along the edge orientation and not across. Many predictors have been suggested till date, which analyze the local area of the image and approximate the direction of the edge. The Sunset algorithm [21] was the first to be standardized for lossless compression. It has seven different predictors and combines at least three neighboring pixel values to make a prediction, all of which are empirically designed and not adaptively tuned. The best predictor is selected and recorded upfront and used for the prediction of the whole image. This certainly treats all the image areas similarly, which is not efficient. The JPEG-LS [22,23,24] uses a median based predictor MED (median edge detector) (2.4) which makes prediction based on the median of the three surrounding pixels. This predictor is again one of the simplest and efficient, but it produces relatively lower compression rates compared to the complex predictors. Gradient Adjusted Predictor (GAP) used in the CALIC (context based, adaptive, lossless, image coding) compression algorithm [25] has set a benchmark in lossless image compression. The GAP predictor is a high complexity predictor, which analyzes the gradients around the pixel to be predicted and switches between previously, designed fixed predictor to make predictions. The predictors in this case are sets of linear
equations involving the pixels in the context and parameters, which are fixed and may not be suitable for all the images. The decisions based on the gradient analysis are also hard coded in the algorithm and hence leaves room for a more adaptive approach.

1.2 Problem and Motivation

State of the art lossless image coders typically perform prediction followed by entropy encoding of the prediction errors [14,23,24,26,27]. The prediction step is used to reduce the complexity of data modeling taking advantage of the data correlation to remove spatial redundancy. The major motivation for prediction is that the error residuals are typically lower in magnitude, making the error distribution highly skewed. These distributions are highly compressible compared to the distributions with longer tails [28]. A good model is the key to the lower entropy of the prediction error residue. Since each natural image behavior varies from scene to scene a universal model cannot be designed and used. For the prediction process to be efficient the image model has to be developed for each image adaptively or in other words the prediction process should be made image adaptive [29]. Images are usually non-stationary and can be reasonably modeled as smooth and textured areas separated by edges [30]. In this thesis, we consider an edge adaptive prediction scheme, which is designed to perform well around edges and on smooth areas of the images. Further, although often ignored by traditional methods for image compression we argue that a special consideration of the entropy coding stage (after predictor) will lead to a better compression for images. Below we motivate our approach by describing three important problem often overlooked by contemporary lossless image compression schemes.

Image Adaptivity: It is important to first identify and classify the image areas into edge and non-edge areas, so that each can be treated differently [31]. Once the pixels are designated and separated into two classes, a predictor suitable for the class and local image characteristics can be applied. An approximate model of the activity around the pixel to be predicted can be adaptively generated and used to better tune the predictor for the current
location. In case of predictors like GAP which is based on a similar concept the parameters are fixed and cannot be tuned depending on the image.

**Problem of Computation:** Several methods, which are highly image adaptive and have image dependent optimization with very good performance, have been suggested [32,33,34,35,86]. The cost for their better performance is the time taken for analysis and fine-tuning of the predictor done at every pixel location. The space and time complexity involved in these predictors make them impractical for most real-time applications. We suggest an edge-based predictor, which makes use of certain parameters that are dependent on the image characteristics. An image analysis technique is suggested that does a pre-analysis to determine the required parameters for the suggested prediction methods. The edge based predictor does gradient analysis and uses the previously calculated parameters to select the best predictor for edge areas and uses a simple linear predictor in case of smooth areas. Though the predictors are adaptive, the design process imposes a certain generality on the predictor coefficients. Thus the performance can be significantly improved by optimizing the predictor coefficients at each pixel location. This involves multiple calculations at each location and thus requires a higher execution time. Although methods based on least squares approach have been suggested, the time requirement for compression has always been a problem. The time problem can be reduced by intelligently applying the optimized predictor at the edge locations while using a simple pre-designed predictor in the smooth areas. We suggest the use of the classic method in conjunction with our pre-analysis method to deal with the problem of computation and achieve better performance in the class of the state-of-the-art predictors with same time complexity.

**Error Modeling and Encoding:** The second phase (i.e. error encoding) also plays an important role in the performance improvement of any compression technique. Error modeling is important to drive the encoders and is typically done based on the predictor used. If not properly done, error modeling and encoding can cause a huge overhead and degrade the performance even if the prediction results are good [36]. We have suggested different encoding methods inspired from one-dimensional and two-dimensional encoding methods. In two-dimensional methods a context template is used to calculate parameters
needed to generate conditional error distributions. For 1-D methods, a 1-D stream is processed in which no structure is used. Few of the previous elements encoded are used to generate the context. A good error modeling should capture the local characteristics of the image and behavior of the predictor so that the probability estimations formed are highly compressible. The final entropy of the error-residual values conditioned on different context (previously encoded pixels in a certain template structure) proves to be less than the first order entropy of the direct residual error [37,28]. The compression performance can be enhanced if the residual values are classified into various exponential distributions based on their variance calculated using the error values in the context [38]. One-dimensional method lacks the ability of capturing the spatial correlation in two dimensional image data and hence needs special scan paths and context treatment. To capture the higher order characteristics, the context size can be increased at the cost of higher complexity. Hence based on the encoding method selected there is a tradeoff between the compression ratio and time required to perform compression [39]. A compromise is needed.

1.3 Contributions of the Thesis

To achieve better performance, the prediction stage needs to be image adaptive and should treat the different image areas suitably. The predictor should be able to segregate the image areas into edge points and non-edge areas adaptively. This problem is similar to the problem of pattern recognition, where the absolute separation of the classes may not always be possible. There is an overlap between the two class members and an approximate separation point has to be determined that can separate the members with the least possibility of false detection. We have suggested an edge based pre-analysis technique for images to adaptively find a threshold that can efficiently separate the current position into edge or non-edge pixel using the information from the previously encoded pixels. The pre-analysis technique also calculates a few other parameters that are used to determine the direction of the edge at a particular location. The pre-analysis technique gives the predictor the ability to adaptively change functioning mode from edge to non-edge areas and vice-versa. Since the parameters are calculated for every image the predictor is automatically
adapted to the particular image. A virtual adaptive predictor tuning is done at different image areas using the parameters calculated during the pre-analysis process. This pre-analysis ability of the predictor has proved to be helpful in achieving better performances compared to other techniques, which use fixed decisions based on hard coded threshold parameters. The pre-analysis method makes the prediction technique adaptive to images and thus offers better performance on various classes of images.

For improved performance, predictors should treat the edge pixels depending on the edge orientation and magnitude to which it belongs. The edge based predictor (EBP) suggested in this thesis, analyzes the gradients in the context structure of the current pixel location and adaptively determines whether the current pixel should be treated as edge pixel or non-edge pixel. During the context analysis, other parameters are also calculated, which can determine the edge direction and its gradient magnitude. These parameters are then compared with the parameters calculated during the pre-analysis method to select a suitable predictor. This results in adaptively using an optimal or a sub-optimal predictor for predicting pixels depending on the edge parameters. Few other parameters are also used to modulate the prediction process, thus tuning it for every location makes highly adaptive. The EBP treats the smooth areas with a simple linear predictor modulated by the parameters determined during the pre-analysis process resulting in an efficient prediction in the smooth areas too. The predictor also employs binary mode switching that achieves better compression in areas where the pixel values are constant. The conditional probability distributions are formed based on simple relationships, which use the gradients and error values in the context of the current location. The conditioning based on gradient magnitudes lets the high variance error values in edge area to form distributions different from the error values with lower variances. Even though the prediction is highly image adaptive, it uses parameters a single set of threshold parameters for the whole image to switch between predictors. These predictors are slightly tuned for each pixel location based on these parameters since the rules for prediction are fixed and only a slight modulation takes place.
To optimize the predictor performance and make it adaptive, a new set of predictor coefficients should be found for every prediction location. This method of tuning the predictor to each pixel location removes the generality in decision-making and thus results in very low error values. This ability of adapting to each location is very useful in case of pixels on the edges, since the edge models can be very complex and varying for being preanalyzed and determining a universal parameter set as in EBP resulting in a sub-optimal solution. In case of smooth areas, simple linear predictors perform as well as optimized predictors, making it totally unnecessary to apply the complex predictors. Least-Squares (LS) based methods [40] can do the job of finding a separate set of coefficients that is designed for every prediction location [41,42,43,44,45]. This optimized coefficient vectors results in better predictions in case of edges at the cost a significant resource requirement, which is sometimes unnecessary. We suggest a method that uses the pre-analyzed parameters to classify the pixels into non-edge pixels and hence avoid using the LS predictor in such cases. In combination with the LS predictor, the binary mode switching is used which encodes pixels in areas with constant pixel values as a binary stream. In cases where the LS predictor cannot be used, the EBP is employed. The LS-EBP predictor has thus proved to be the best predictor and achieves better performance compared to all other predictors in the same complexity spectrum. Since both the prediction schemes change behavior adaptively when an edge pixel is encountered, the schemes are called adaptive edge-based prediction schemes.

The error encoding stage of the compression process was also given a special attention in this thesis because of its contribution to the overall performance of the compression algorithm. The functionality and performance of the Burrows Wheelers Transform (BWT) [46,47] has been studied in this thesis. Inspired by the BWT, a different way of reorganizing the data has been proposed. The reorganization process makes it easy to further compress the data [48]. The methods studied are different ways of looking towards reorganization and recovery of the data sets. A method inspired from using the second column of the BWT has been discussed and the schemes derived from the observations are presented. Comparisons are done on the basis of direct application of the original BWT on
the same data set. The encoding methods based on BWT are used in the cases of the above-mentioned predictor, which work on the concept of reorganizing the error data to make it more compressible.

The error values are scanned using a special scan path, which maintains the two dimensional relationship in the image forming a one-dimensional stream. In this thesis, we discuss a method, which scans the data and forms different streams (named $\alpha$-paths) based on the previously occurring context or a parameter calculated using the context. These $\alpha$-paths tend to separate error values into different classes with similar magnitudes. This results in different conditional distributions, conditioned on the context values that are highly skewed and thus give better compression. The $\alpha$-path error encoding method followed by an arithmetic encoder [49,50,51,52] proved to be the best encoding method and thus leading to the best compression results amongst methods in the same genre of lossless image compression.

### 1.4 Thesis Organization

Chapter 2 is a general discussion on the basic theory behind data compression and various aspects involved. The basic concept behind prediction-based compression and context based error encoding is also introduced. Finally BWT’s role in image compression and the problem of using it directly for image compression will be discussed in this chapter. The Burrows Wheelers Transform (BWT) along with a basic implementation methodology is presented and the reason for its better performance is also discussed.

Chapter 3 is devoted to the suggested predictor, which was the motivation for designing the edge adaptive prediction scheme. The edge-based predictor (EBP) is discussed in detail along with the pre-analysis method used to calculate the parameters needed for prediction. The algorithmic details are also provided which will make it easy to implement. The context based encoding used in the compression process is also presented in detail.
Chapter 4 describes the concept of Least-Square (LS) methods for image predictor design. Details of the proposed LS-EBP predictor are presented along with implementation aspects. The same pre-analysis method is used with the LS-EBP resulting in a better performance.

Chapter 5 discusses the BWT inspired methods that can be used to compress 1D data. The implementation issues and detailed understanding of each of the suggested algorithm are also presented. Finally the $\alpha$-path method that was successfully used for image compression will be presented.

Chapter 6 summarizes the results with a comparative study with other state-of-the-art methods. Results for various variations of the suggested scheme with varying parameters are also been included.

Chapter 7 concludes the thesis with suggestions for future research.
Chapter 2: Background

In this work we have studied various aspects of data compression. In this chapter we will consider the basic theory of data compression and treatment of data sources. The Burrows Wheeler Transform was considered for the possibility of encoding the image data, and hence a brief functionality of the BWT will be provided in a sub section of this chapter. Traditional image compression theory was studied and researched, from which an idea for a whole image compression technique was developed. Other relevant aspects of image compression and their counterparts in this thesis will also be discussed in this chapter.

2.1 Data Compression

The notion of redundancy is an important concept that comes natural in various fields of information theory. Two of the fields that evolved regarding the redundancy in data were compression and error correction. While data compression focuses on techniques to remove possible redundancy in the data, data protection via error detection and error correction consider methods for controlled introduction of redundancy in the data. Different mathematical principles were applied to come up with various algorithms to achieve efficient performance. Compression acquired more attention after the invention of computers.

Data in all these algorithms is treated as an information source. Redundancy in data is related to the seen as high frequency of data elements or predictability of the data occurrence. In the 1940’s and 1950’s Claude Shannon and many others provided a theoretical basis for the field of information theory [1,53,54]. Although in 1925, statistician R. A. Fisher introduced a technical definition of “information” [55], the Fisher's definition of information applies to statistical estimation and differs from Shannon's definition of information. Shannon formalized the definition of information source and also defined the fundamental limit of information content of a first order source. Probability theory is the
backbone of information theory. Statistical properties of the data sources help in compressing the data since a high occurrence symbol can be given a low cost (short) codeword. Representation of the raw data is done by considering it as the output of a uniform distribution source (equi-probable), which results in each symbol having the same codeword length. The actual measure of information in a data source is its entropy (2.1). Entropy of a data source is defined as follows [54,56];

$$E = - \sum_{m=1}^{\Sigma} p_m \log_2 p_m$$  \hspace{1cm} (2.1)

Here $p_m$ is the probability of the $m^{th}$ symbol in the source with symbol alphabets $\Sigma$. If all the symbols have equal probability, i.e. $p_m = \varphi$, $\forall m$, then the entropy is just $E = \log_2 (m)$ bits. To achieve compression, the probability distribution should be skewed. This results in shorter average code length, since the higher probability symbols are given shorter codes. Since these symbols occur with a higher probability they contribute less to the final code length because of their shorter codeword length. One simple but effective method of taking advantage of the skewed distribution is to use Huffman codes. Huffman’s algorithm [1,53,54] works by assigning integral shorter code lengths to high probability symbols. But the disadvantage of using Huffman is that the average code length cannot be less than 1 bit, that is, a code word less that 1 bit in length cannot be assigned to a symbol. Compressing blocks of data instead of one symbol can solve this problem. Hence arithmetic coding [49,51] is sometimes called the near optimal encoder. Though arithmetic coding produces better results it requires more computational and memory resources.

### 2.2 Image Compression

An image is basically a spatial representation of intensity data. The exact meaning of the intensity will depend on the particular application under consideration. The intensities are represented as pixels on a grid defined by the resolution of the image. The pixel intensities vary in magnitude depending on the bit depth resolution of the image. So if $B$ bits are
allocated per pixel then the pixel intensity can vary form 0 to \(2^B\). There are a wide variety of image types such as synthetic images, natural continuous tone images and texture images. Also different applications could produce different images based on model, for example sonar, DNA, microarray and x-ray images. Each image type has its own complex model. Compressing these images involves modeling them and using entropy encoding.

The entropy of the image using the original intensity values is usually high and using the entropy encoders directly does not give good results. There is a lot of redundancy and correlation in the image data that can be exploited to achieve better performance. The standard method of processing the image data is by scanning them in raster scan order.

Most of the state-of-the-art image compression schemes can be divided into two phases, such as modeling and encoding. Modeling is done for estimating the behavior of the image intensities based on the statistical information about the image pixels. These models are further used in making inductive inferences for predicting the current pixel. After scanning and encoding few pixels \(x^t = x_1 x_2 x_3 \ldots x_t\) at instance \(t\), they can be used in making an inference on \(x_{t+1}\) by assigning conditional probability distribution \(p(x_{t+1} / x^t)\). The code length for these predictions is \(-\log_2 \left( p(x_{t+1} / x^t) \right)\) bits, which is the entropy of the conditional probabilistic model [57,37].

The two major types of image compression are lossy and lossless compression. Independent of the type, two categories of processing sequential (simple pass) and two pass schemes can be applied. In sequential scheme, encoding of \(x_{t+1}\) is done adaptively using previous template \(x^t\) for modeling. On the other hand a two-pass scheme analyzes the image in the first pass and then sends the header information. In the second pass, the results of the analysis are used to encode the data.

The general algorithm for image compression can be summarized as follows:
1. Determine the prediction $\tilde{x}_{t+1}$ for the current pixel $x_{t+1}$ is using the $x^f$ causal template.
2. Compute the prediction error residual $e_{t+1} = x_{t+1} - \tilde{x}_{t+1}$ and perform modeling based on the context.
3. Define a context $C$ is based on the causal template, then update the conditional distribution of prediction error conditioned on context in which $x_{t+1}$ occurred.
4. Using the model, the prediction error is encoded conditioned on the context of $x_{t+1}$.

Prediction has been successfully used in decorrelating the data samples in case of continuously varying physical magnitude. The next sample can be closely predicted using a simple function (e.g. a linear combination) of previously encoded samples in a finite causal context. The error is then modeled using the context to drive the entropy encoders. The modeling process classifies the error values into different classes with different conditional probabilities. This helps in compressing the error residuals more efficiently compared to simple error encoding.

Low Complexity Lossless Compression of Images (LOCO-I) algorithm was suggested as the core for the latest JPEG-LS [22,23,24] compression standard. LOCO was developed to achieve competitive compression ratios compared to the state of the art compression techniques such as CALIC. LOCO was selected over CALIC [25] due to its optimal placement in a conceptual compression/complexity curve [58,59]. But from the compression performance aspect CALIC was the winner. JPEG-LS scheme uses a causal template for prediction and modeling. The predictor used in case of JPEG-LS is a non-linear predictor, which switches between three simple predictors ($P_W$, $P_N$, and $(P_N + P_W - P_{NW})$). The function suggested is nonlinear median edge detector (2.2) as it incorporates a rudimentary edge detection capability [60]. The predicted value is given by:
\( \hat{X}_{med} = \text{min}(P_W, P_N, P_{NW}) + \text{max}(P_W, P_N, P_{NW}) - P_{NW} \)

where

\(P_N, P_W\) and \(P_{NW}\) are the north, west and northwest pixels

Using a smaller context implied lower computations, and hence to a generally faster compression. However, the edge detection capability of the predictor is limited due to the smaller context window used and hence higher order edge behavior cannot be predicted. Moreover, the edge gradients are not taken into account, which results in further degradation of the performance. The adaptive component of the predictor is limited to an integer additive term, similar to an affine term in adaptive predictors. The context window of four also restricts the context for error modeling when the \(P_{NE}\) pixel is also included. There are 729 different contexts generated after quantizing the context pixel gradients

\(g_1 = P_{NE} - P_N, g_2 = P_N - P_{NW}\) and \(g_3 = P_{NW} - P_W\)

into 9 levels each. Due to the simplicity requirements many cuts are made on the prediction and error modeling complexity. In this case Golomb’s codes [61] are used instead of arithmetic coding. This resulted in relative lower compression results compared to CALIC, but a much faster compression speed.

CALIC (Context Adaptive Lossless Image Compression) was also designed to participate in the call for the latest JPEG-LS standard. CALIC had the best results and comparatively better performance compared to LOCO. But CALIC lost the race due to its higher complexity compared to LOCO. CALIC is highly tuned for the application of image compression. It has two modes of operation; binary and continuous mode. Binary mode is where the image intensity levels do not vary more than 3 levels. A ternary encoder is used to compress a separate ternary stream representing the data in particular region. This resulted in better performance in case of images like documents or synthetically generated pictures. In continuous mode, the scheme uses a context dependant prediction step which incorporates prior knowledge through a fixed component, termed GAP (Gradient Adjusted Predictor), which switches between combinations of neighboring pixel values based on gradient information in the context window of the current location. This makes the
prediction highly adaptive to image gradients at a particular location. CALIC also uses a context-dependent adaptive error correction is similar to the affine term in classical autoregressive (AR) models. GAP calculates two gradients $d_{90}(x,y)$ and $d_{0}(x,y)$ which represent the vertical and horizontal gradients at a particular location. Based on the gradient values the predictor estimates the orientation and magnitude of the edges and the weights assigned to neighboring pixels using a fixed set of rules. The rules have been experimentally tuned and are not image dependent. The threshold values used in the GAP predictor are fixed and can prove inefficient for different kinds of images that have different contrasts and thus edge gradients and magnitudes. All the weights assigned are also fixed and thus leave no room for image adaptive coefficient allocation.

The EBP (Edge Based Predictor) suggested in this thesis does a pre-analysis over the image to determine the parameters for the prediction step. The analysis of the edge is done based on four gradient values $[d_{0}(x,y), d_{90}(x,y), d_{45}(x,y), d_{135}(x,y)]$ and threshold values calculated during a pre-analysis procedure. The coefficient values assigned are also made adaptive based on the image gradients and other set of coefficients calculated during the pre-analysis. The predictor also corrects the prediction value by a procedure called bias cancellation, which is context dependent. Being image adaptive the prediction process is slightly slower compared to GAP but performs better on most of the images and images of all kinds (smooth, texture or synthetic).

A heavily adapted predictor based on least squares (LS) was suggested in [40,44,45], which tunes the predictor for each pixel location and finds a set of coefficients for each location. Since least-square optimization is used to find predictor coefficients for each pixel prediction, the entropy is relative low compared to other fixed predictors. The prediction results are better than GAP but at the cost of higher complexity and thus higher compression time. The LS approach needs more calculations and is not necessary in some cases. The adaptive selection between LS and APC was suggested in [40], which reduced the number of calculations, but the complexity is still comparatively high. The switching is decided on the basis of the singularity of the covariance matrix. The covariance matrix
calculation by itself needs higher time and space complexity. Though to achieve a better performance the author suggested a linear combination of both the prediction values the complexity was still high. A faster algorithm called EDP (Edge Directed Prediction) based on the LS approach was proposed in [62]. The algorithm employed intelligent switching between MED and LS predictor based on the magnitude of the error values. Once a coefficient set is found using LS optimization it is used till the error value is below a pre-defined threshold. Calculations for new coefficient set are done when the error exceeds this threshold. The rank of the covariance matrix is also checked to switch between MED and LS, since in smooth areas the solution is not unique. This technique significantly improved the speed of the algorithm, when compared to the pure LS applied at each location approach. The assumption made that the once calculated coefficient set is optimum for all the pixel locations until the error value exceeds a bound may not be efficient in all cases.

In this thesis, we describe the LS-EBP algorithm, a combination of the least-square approach with the proposed edge-based predictor (EBP). The key observation is that the time consuming LS algorithm only needs to be applied around the edges in the image. Hence, significant computation could be saved without affecting the overall prediction performance. The LS-EBP approach uses the pre-analysis done to decide where the LS approach is to be applied, which are most of the time, the edge areas. In case, where the LS approach is unnecessary, an EBP based prediction is done using the parameter calculated in the pre-analysis process. In case of smooth areas, the EBP uses a linear predictor that is simple in complexity and gives better results compared to fixed set of coefficients as used in EDP. The results are slightly better than EDP with similar execution times. The EBP and LS-EBP based compression techniques using context based and alpha-path error modeling will be discussed in the following chapters.

2.3 The BWT

Most of the data compression techniques based on statistical modeling, process the data sequentially. These methods have generally better performance but require more time.
Burrows and Wheeler [46,47] introduced an algorithm with almost the same performance as statistical methods, but relatively faster. Their algorithm processed data in blocks rather than sequentially. The algorithm has come to be known as the Burrows-Wheeler Transform, BWT [46]. The BWT applies a reversible transform on the data block that reorganizes the data making it easy for further processing and compression. After applying BWT a simple entropy encoder can be used for final compression.

BWT (2.3) is a reversible transform which reorganizes the data and the output has the same set of characters as the original set. The length of the output stream is also same as the original and requires a single integer to recover the data back. The BWT transform pair is defined as follows:

\[
BWT(S^N) = (L^N, Id) \quad \text{Forward Transform}
\]
\[
BWT^{-1}(L^N, Id) = S^N \quad \text{Inverse Transform}
\]  

(2.3)

Logically speaking, the BWT does not compress the data since the output string \(L^N\) has the same length \(N\) as input \(S^N\). The key advantage is that the output data organization/structure is more compressible compared to the original stream. BWT tends to cluster characters in such a way that there are higher chances of finding a character close to the instance of the same character. In other words, characters having the same prefix (or context) are accumulated together making it easy to apply fast locally adaptive algorithms, such as move to front (MTF) coding in combination with Huffman or arithmetic algorithm. BWT was designed primarily with text compression in mind. Text data has a lot of repeating contexts due to the structure of the English Language, which helps in compressing the data using BWT.

### 2.3.1 BWT Functionality

The BWT takes a string \(S^N\) of \(N\) characters as an input. The string \(S^N\) is then cyclically rotated \(N\) times forming \(N\) separate streams, each being the cyclic shift of the previous.
$N$ rotations are then sorted lexicographically, and the last character of each of the rotations is then extracted. The string formed from the last column (L) forms the output. The $i^{\text{th}}$ character of $L^N$ is the last character of the $i^{\text{th}}$ sorted rotation. The algorithm also computes an index $Id$ that represents the position of the original string $S^N$ in the sorted rotations. Given the string $L^N$ and the index $Id$, the original string $S^N$ can be recovered efficiently without any loss. The process of sorting tends to bring together rotations that have the same initial characters. Since the characters in $L^N$ are the prefixes for the initial characters of the rotations they also tend to cluster. Since $S^N$ contains a lot of similar repeating strings, they arrange together after sorting and the adjacent characters in L also collect similarly. Hence, if $S^N$ has many repeating contexts, there is a good chance of getting a good performance in compression using simple compression techniques.

A standard example used by Burrows and Wheeler in their report on BWT [46,47] is explained below. Let a string $S^N = "abraca"$ with length $N = 6$ be feed as an input to the BWT. The alphabet of the string $S^N$ is $\Sigma = \{a, b, c, r\}$, $|\Sigma| = 4$ with four unique characters. As mentioned in the algorithm $N$ cyclic rotations are done to form a conceptual $N \times N$ matrix $M'$ (Fig 2.1).

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>r</th>
<th>a</th>
<th>c</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>b</td>
<td>r</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>r</td>
<td>a</td>
</tr>
<tr>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>r</td>
</tr>
<tr>
<td>r</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>r</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
</tr>
</tbody>
</table>
```

Figure 2.1: $N \times N$ conceptual matrix $M'$ formed by $N$ cyclic shifts of the string $S^N = "abraca"$
This N x N matrix is then lexicographically sorted to form an ordered matrix M as follows.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>a</th>
<th>b</th>
<th>r</th>
<th>a</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>r</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>b</td>
<td>r</td>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>3</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>r</td>
</tr>
<tr>
<td>4</td>
<td>b</td>
<td>r</td>
<td>a</td>
<td>c</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>5</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td>r</td>
<td>a</td>
</tr>
<tr>
<td>6</td>
<td>r</td>
<td>a</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

Figure 2.2: The lexicographically sorted matrix M.

It can be seen that the original string is ranked 2 (Fig 2.2) in the sorted order. This rank is used as an index to recover the original string. The last column elements form the $L^N$ string, which is the output of the BWT. Hence the output of $BWT(S^N)$ is $\rightarrow (L^N, Id)$ (2.3) is the string $L^N =$ “caraab” and the index $Id=2$ in this case. Decoding is done using the index $Id$ and the $L^N$ vector. The $L^N$ vector after sorting becomes the first column of the matrix $M$. Now the decoder has the first and the last vector of matrix $M$ along with index $Id = 2$. Let the recovered string be $S'$. So first elements of recovered string $S'$ i.e. $S'(1) = F(2) \{a\}$. Now the decoder looks for the next instance of $F(2)$ in $L^N$. In this case $L(4) = F(2)$ so the second element $S'(3)=F(4) \{b\}$. Now $F(4)$ is found at $L(6)$ so the third elements $S'(3)=F(6) \{r\}$. This process continues until all $N$ elements are extracted. If the BWT is implemented as explained above the computational complexity is $O(n^2)$, which can be reduced by using suffix tree implementation. The output vector $L^N$ of BWT (Fig 2.2) is not compressed by itself i.e. the length of the output vector in same as the input with an extra integer. But the benefit of applying the BWT on such a block is that due to repeating contexts the characters having same context tend to collect together. This forms local accumulations in the output vector. The above example does not show much since the length of the vector is small and there are not enough context repetitions. In case of text data there is large accumulation of characters that are further passed through a scheme called move to front coding (MTF).
The MTF scheme is a typical representative of a List Update Algorithm (LUA) and is used as a Global Structure Transformation (GST) after the Burrows Wheelers Transform. Move to front coding is a locally adaptive algorithm, which maintains a list of all the symbols used in the output $L^N$ vector. The most recently used symbol is shifted to first position in the list and the others are shifted by one rank below. So as the clusters are encountered they are all ranked based on their recent occurrence. Most of the symbols independent of the magnitudes are mapped to a lower value since they tend to appear in clusters. For example MTF coding on a vector $V$={$10,10,10,2,2,2,40,40,40,10,10,10$} would result in an output vector $V'$={$1,0,0,1,0,0,2,0,0,2,0,0$}. The elements of the vector $V'$ thus represent how recently the elements were used. The decoder can mimic the same and recover the data. If carefully observed, it can be seen that the distribution of the output data has changed and has lower entropy. Next element in the pipeline of BWT is simple entropy encoder like Huffman or arithmetic encoder. Since the entropy of the data has reduced the entropy encoders give better compression compared to applying these encoders on the original data. Many other methods and modules have been suggested for processing the output of the BWT [63]. But there isn’t any significant change in the performance of the system.

Though BWT was designed with text compression in mind it works independent of the data source. The performance does vary on the type of data source that is feed to the BWT. If the data source is random and non-stationary and does not have exact context repetitions, the performance of the transform degrades. Hence BWT may not be an efficient algorithm for compressing data sources like natural continuous tone images. Either the data has to be pre-processed or the BWT has to be changed to provide improved performance.

Some methods have been suggested for compression of continuous tone images based on the BWT [64, 65, 48]. The BWT has been applied on the image pixel data or the error residual after prediction step. Others have applied the BWT on transform coefficients from linear transforms, such as Fourier Transform or Discrete Wavelet Transform [66,48]. BWT applied on image pixels does reorder them based on the context they are preceded by. The context formed by simple image pixels collected together does not capture the correlation
that exists in the image data. Pixel reordering can be done based on modified context information rather than pixel context themselves [67]. The method suggested in [67] “Lossless image compression using pixel reordering”, permutes the set of pixels into an order that reflects the assessed volatility of the conditioning context. This allows the set of pixels handled by one context flawlessly blend into the set of pixels associated with the next context. This makes it possible to use an encoder with one coding context, but with the changing probability estimates, which are captured by local adaptation in the coder. A list of tuples is formed using the modified context and the error calculated using a simple predictor. The tuples are formed as follows,

\[ T = (C(P_N, P_w), x, y, \Delta e) \]

where

- \( C \) is the context identifier
- \( P_N, P_w \) are the north and the west pixels
- \( x, y \) are the current location coordinates
- \( \Delta e \) is the error after prediction

The context identifier \( C(P_N, P_w) \) is determined as \( C = |P_N - P_w| \) whereas the error is calculated using the function,

\[ \Delta e = \frac{P_N + P_w}{2} - P(x, y). \]

The sorting used in the above method is done based on the first three columns and becomes more time consuming as the size of the image increases. The single locally adaptive entropy coder used does not adapt to the context flow fast enough, though it saves the separate context maintenance cost and context switching complexity. The method results in inefficient performance of the encoder and can be improved by using better context generation and prediction method.

The three keys based sorting of the tuples \( T \) (2.4) can be avoided by maintaining the exact order of the elements that follow a particular context. The information lost due to sorting can be maintained by having the same order of elements in the original stream. We will be
looking at a method inspired by the BWT, which uses the notion of \( \alpha \)-paths that are streams maintained for each context defined. The contexts defined are complex in this case and capture a better relationship to exploit the correlation in the image data. The contexts are further quantized and are few, making their handling easy. The prediction method used in this case is image-adaptive and edge-based which reduces the entropy of the source and thus improves the compression performance. Since the image is just scanned once to separate the error values belonging to each context the operation is linear in time. Further, since no sorting is needed the extra overhead is reduced. The entropy encoder for each context is different and no context switching is required. The separate encoders also result in less adaptation cost for each \( \alpha \)-path. The result is a better compression performance compared to single encoder that is locally adaptive. Sorting based on context makes the suggested method resemble the BWT.
Chapter 3: The Edge Based Predictor

3.1 Introduction

The proposed system in this thesis is an asymmetric compression technique, which implies that the time and space complexity of the encoder and the decoder are different. In comparison to the decoder, the encoder has higher complexity in both time and memory. The decoder decompresses the image based on a set of few parameters sent by the encoder. The system proposed compresses the image pixels in raster scan order and the decoder follows the same. The encoder is based on two passes of which the first one is for the parameter calculation and the second is to actually predict and compress the data. The decoder is a single pass technique that decompresses pixels in one pass based on the prediction scheme and parameters passed by the encoder.

The predictor proposed (Fig 3.1) uses context information for prediction and hence requires previous two rows to hold the context information for the current pixel. Since the context is formed of previously encoded pixel the system is causal. Similarly the decoder that is a mirror image of the encoder predictor has the same requirements. The predictor suggested is a non-linear predictor, which makes certain decisions, based on the locality of the pixel and the context of the pixel. The major two decisions made are to switch between binary and continuous mode of operation based on the gradient quantity sums (3.8). If the gradients in the context window have zero magnitude then the predictor automatically switches to binary mode, which encodes a separate binary stream along with the escape symbols. This method of binary encoding improves coding efficiency in cases of images having larger areas with same intensities like computer graphics images. If the context does not satisfy the binary mode criteria the predictor switches to continuous mode. In continuous mode the prediction is done based on the gradient of the image at the current location using the same context. Based on the magnitude and the direction of the gradients a linear equation is chosen from a set if equations to make prediction $P$ for the original
image pixel $I$. The use of gradients helps in making better predictions in cases of edges since most of the time the predictions are made along the edge and not across the edge. This is the major disadvantage of the DPCM [16,17] where it makes prediction independent of the edge orientation and magnitude. Gradient-based prediction is thus better than DPCM in many cases. As the predictions are made most expected error in a context is calculated by accumulating errors in each context. These accumulated errors are further used to improve the gradient-based prediction $P$ by implementing a feedback system. The error modeling is using the different types of finite contexts, which results into an adaptive and context based nonlinear system. Context based error modeling is used to refine the prediction $P$ to form $P'$. The new prediction value $P'$ is calculated by adding the expected error in a particular context which appears at that point in the image. Hence $P'$ can be defined as $P' = P + e_{mi}$, where $P$ is the gradient based prediction and $e_{mi}$ (3.18) is the conditional sample mean error of the $i^{th}$ context. The finite context definitions are not only used in the process of prediction but they are also used in entropy coding of the errors.

Figure 3.1: Basic Block Diagram of the Edge Based Prediction scheme.
The benefit of context based prediction and encoding is in modeling the error source as a collection of different sources with different statistical parameters and which are conditioned based on the context. The conditioning helps in separating each context error stream from the other error streams and hence reducing the conditional entropies. We know conditional entropy of a source is less than the first order entropy of original source itself. The only challenge is selecting proper contexts and assigning errors to the appropriate contexts. The prediction is also significantly improved using context based prediction improvement since the predictor now separates the current prediction and error value into a different sample space, which have more correlation than other samples in the image. It is observed that the sample error mean $e_{mi}$ is offset for each context which significantly improves the prediction $P'$ as compared to $P$. The context modeling of error also helps in driving the entropy encoder, which is an arithmetic encoder in this case.

There are many possible ways to define a context but most of them become impractical because of the large number possible variation in single context type. The simplest idea is to use the possible combinations of the pixels in a context, which would have $N^M$ possibilities if the length of the context was $M$ and there were $N$ levels of pixel value. This number of contexts is high and difficult to handle hence some sort of quantization is needed on a free variable that measures some parameter of the context. This parameter varies with different types of local image characteristics. The idea of quantizing the error energy estimate is adapted from the CALIC [25] method with a major difference in the error parameter calculation. In this case a combination of the gradients and previous errors is used to determine the context parameters that are then quantized to form a finite number of contexts. The quantization scheme proposed is based on empirical observations but could be designed using iterative functions that reduce a cost function, in this case the entropy of the error residual. Error remapping based on the prediction range and error sign prediction and flipping based on the context is also included in the suggested method. The overall algorithm performs better than most of the popular techniques and has a linear time and space complexity.
In the following discussions the different steps of the suggested compression technique will be explained from the theoretical and implementation point of view. Algorithms used will also be presented where required.

### 3.2 Image Characterization and Parameter Calculation

An image is a two-dimensional intensity function, \( I(x, y) \) where \( x \) and \( y \) are spatial coordinates and the magnitude of \( I \) represents the intensity of the image at that pair of coordinates [68,30]. Image pixels have correlation between them and this correlation can be exploited to make prediction \( P \) for pixel value \( I \). Analyzing the image beforehand gives an idea about the type of the image. The challenge lies in exploiting this information without paying too much of overhead cost. This information is used in making global decision or calculating parameters that help in the decision making for the prediction scheme. The suggested method divides the image into four parts (3.1) for analysis purpose. Division is done using midpoints one each axis.

\[
I = \bigcup_{i=1}^{4} I_i \\
I_1 = I(1 : \frac{x}{2}, 1 : \frac{y}{2}) \\
I_2 = I(\frac{x}{2} + 1 : x, 1 : \frac{y}{2}) \\
I_3 = I(1 : \frac{x}{2}, \frac{y}{2} + 1 : y) \\
I_4 = I(\frac{x}{2} + 1 : x, \frac{y}{2} + 1 : y) \tag{3.1}
\]

The first step of the analysis involves edge detection using the canny edge detector at a threshold of 5% of the maximum. The canny edge detector first applies smoothing on the image to eliminate any noise [69,70]. It then finds the image gradient to highlight regions with high spatial derivatives. The algorithm then tracks along these regions and suppresses any pixel that is not at the maximum (non-maximum suppression). The gradient array is now further reduced by hysteresis. Hysteresis is used to track along the remaining pixels.
that have not been suppressed. Canny edge detector is said to be an optimum edge detector and has higher edge detection accuracy. This proves efficient in finding the parameters for better performance. The pixels detected as edges, are used to calculate two parameters $\xi_1$ and $\xi_2$ (3.7). The gradient quantities $G_q(x,y) = \left[ d_0(x,y), d_{90}(x,y), d_{45}(x,y), d_{135}(x,y) \right]$ (3.2) are processed separately for pixels detected as true edges and non-edge pixels to determine a threshold $C_t$ (3.10) used during prediction process. To cut down the time and memory the analysis process is done on down sampled grid. The down sampling is done by a factor of four.

The image $I$ is processed in a raster scan with a height $H$ and width $W$. The rows of the images are represented as $y$ and ranges from $1 < y < H$ and the columns are represented as $x$ which range from $1 < x < W$. But for the edge analysis and the gradient calculations the pixel processed would be in the range from $3 < y < H$ and $3 < x < W-2$. The context of the current pixel $I(x,y)$ is as shown bellow.

<table>
<thead>
<tr>
<th>$I(x-2,y-2)$</th>
<th>$I(x-1,y-2)$</th>
<th>$I(x,y-2)$</th>
<th>$I(x+1,y-2)$</th>
<th>$I(x+2,y-2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I(x-2,y-1)$</td>
<td>$I(x-1,y-1)$</td>
<td>$I(x,y-1)$</td>
<td>$I(x+1,y-1)$</td>
<td>$I(x+2,y-1)$</td>
</tr>
<tr>
<td>$I(x-2,y)$</td>
<td>$I(x-1,y)$</td>
<td>$I(x,y)$</td>
<td>$I(x+1,y)$</td>
<td>$I(x+2,y)$</td>
</tr>
</tbody>
</table>

Figure 3.2: The Context Window Structure used for gradient calculation.

All twelve pixels represent the context of the current pixel and are used in the calculation of the gradients quantities $Gq(x,y)$ of the image at the current location. The gradient quantities can be formulated as follows.
As shown above the gradient quantities represented as $G_q(x, y) = [d_0(x, y), d_{90}(x, y), d_{45}(x, y), d_{135}(x, y)]$ are the image gradients at pixel location $(x, y)$ in four directions. The vertical direction gradient is represented by “$d_{90}$” and the horizontal by “$d_0$”. “$d_{45}$” represents the gradients in 45 degree direction and “$d_{135}$” in 135 degrees. These quantities are used in calculating the threshold $C_i$ (3.10) to separate pixels in two predefined classes’ designated edge and non-edge pixels. To determine the dominant edge direction four more parameters called the directional coefficients $c_i(x, y), i=1,2,3,4$ (3.3) are found which are calculated using the same gradient quantities.

$$c_i(x, y) = \begin{cases} 
    d_{135} / (d_{135} + d_{45} + \epsilon) & i = 1 \\
    d_{45} / (d_{135} + d_{45} + \epsilon) & i = 2 \\
    d_{90} / (d_0 + d_{90} + \epsilon) & i = 3 \\
    d_0 / (d_0 + d_{90} + \epsilon) & i = 4 
\end{cases}$$

Here $\epsilon$ is a small constant to avoid division by zero, in our case we have used $\epsilon = 0.001$. It can be seen above that the pair $c_1(x,y)$ and $c_2(x,y)$ represent a linear relation between the 45 and 135 degree direction gradients and $c_1(x,y) + c_2(x,y) = 1$. Similar $c_3(x,y)$ and $c_4(x,y)$ form a similar relation. After calculating the coefficients and the pair wise relationship a single representative is found for each pair.

$$R_1^p = \max \{c_1(p), c_2(p)\}$$
$$R_2^p = \max \{c_3(p), c_4(p)\} \quad p = (x, y)$$

(3.4)
$R_i^p$ represents the dominant direction in $d_{45}(x,y)$ and $d_{135}(x,y)$ where as $R_2^p$ represents the dominant direction in $d_0(x,y)$ and $d_{90}(x,y)$ \[ (3.4) \]. These representative coefficients are then accumulated into two separate vectors $\rho_1$ and $\rho_2$.

\[
\rho_k = \bigcup_{p=1:N_e} R_k^p \quad k = 1, 2 \tag{3.5}
\]

$N_e$ in this case is the number of edge pixels detected in the edge analysis process. Mean values of the two vectors $\rho_1$ and $\rho_2$ \[ (3.6) \] were first used as thresholds to determine the directional predictors to be used during the prediction process.

\[
\mu_l = \frac{1}{N_e} \sum_{p=1}^{N_e} \rho_p \quad l = 1, 2 \tag{3.6}
\]

But it was observed that the low contrast and images with properly defined edge orientations had lower mean $\mu_i$ and images with highly texture data have higher mean $\mu_i$. Thus, to avoid a frequent application of edge based predictions in smooth area images the parameters $\mu_1$ and $\mu_2$ must be increased. In case high contrast images and highly texture areas the thresholds must be reduced to get maximum application of the directional predictors. So by empirical observations a simple linear modification model for the means
The Edge Based Predictor

\[ \mu_1 \quad \text{and} \quad \mu_2 \quad (3.6) \quad \text{is suggested.} \]

\[ 0.7 \]

\[ 0.6 \]

\[ 0.5, 0.5 \]

\[ 1 \]

\[ 0.7 \]

\[ 1 \]

\[ \text{Original } \mu_i \]

\[ \text{Modified } \xi_i \]

\[ \text{Original } \mu_i \]

\[ \text{Modified } \xi_i \]

Figure 3.3: Calculating \( \xi \) (direction threshold) using pinch point \( \omega \)

Figure 3.3 shown above is the modification curve of the threshold parameters used to select the direction of gradient for prediction. The simplest model would assign assume a function \( f(x) = 1.5 - x \) i.e. just an inverse linear relation. But a new pinch point \( \omega \rightarrow (\omega_x, \omega_y) \) is used to generate the modification curve. The new modified directional threshold parameters \( \xi_1 \) and \( \xi_2 \) are calculated as follows.

\[
\xi_{i} = \begin{cases} 
\frac{(\mu_i - \omega_x)(\omega_y - 0.5)}{(\omega_x - 1)} + \omega_y & ; \mu_i \leq \tau \\
\frac{(\mu_i - 0.5)(1 - \omega_y)}{(0.5 - \omega_x)} + 1 & ; \mu_i > \tau
\end{cases} \quad (3.7)
\]

The pinch point \( \tau \), which is \( \omega_x \), is set to 0.7 in the suggested method. These parameters \( \xi_1 \) and \( \xi_2 \) (3.7) which are achieved modifying the means \( \mu_1 \) and \( \mu_2 \) (3.6) are finally used in the
prediction scheme to decide the direction of the gradient and hence the respective equations to be used.

To determine the threshold parameter $C_t$ (3.10) that helps in differentiating and designating edge and non-edge pixels during prediction is calculated using the gradient quantities $G_q(x,y)$ (3.2).

$$S_G(x,y) = \sum (G_q(x,y))$$

$$= \sum_{\gamma \in [0,90,45,135]} (d_\gamma(x,y))$$

(3.8)

In the case of the pixel where the edge is absent only sum of the gradient quantities $S_G$ (3.8) is calculated and accumulated in an array $G_e$. In case of pixels on edges the $S_G$ is calculated along with the coefficients $c_i(x,y)$ (3.3). $S_G$ for the non-edge pixels is accumulated in an array $G_{ne}$. So there are two arrays populated by the additions of gradient quantities $S_G$ in edge and non-edge areas. Let $H_e$ and $H_{ne}$ (3.9) be the histograms of the two arrays $G_e$ and $G_{ne}$. 

32
Define
\[ \delta(a, b) = \begin{cases} 1 & ; \ a = b \\ 0 & ; \ otherwise \end{cases} \]

Then, the histograms \( H_e \) and \( H_{ne} \) are given by:
\[
H_e(r_k) = n_k = \sum_{k=1}^{N_e} \delta(G_e(k), r_k) \\
H_{ne}(t_k) = m_k = \sum_{k=1}^{N_{ne}} \delta(G_{ne}(k), t_k)
\]

where
\[
r_k = 0, 1, \ldots, \max(G_e) \\
t_k = 0, 1, \ldots, \max(G_{ne})
\]
\[
N_e = \sum_{k=1}^{\max(S_g)} n_k
\]
\[
(H \ast W) - N_e = \sum_{k=1}^{\max(S_g)} m_k \quad (3.9)
\]

Here \( H \) and \( W \) are the height and width of the images and \( n_k \) is the count for \( r_k \) level in the vector array \( G_e \). Similarly \( m_k \) is the count for \( t_k \) level in the vector array \( G_{ne} \). \( N_e \) as discussed previously is the number of detected edge pixels.

![Diagram](image)

Figure 3.4: Calculating \( C_t \) (threshold) parameter.
This is done to find the crossing point of the histograms since edge areas have higher gradient sums $S_G$ (3.8) and the non-edge areas have lower values. The histograms (Fig 3.4) tend to have different maxima and non-overlapping shapes. Ideally there should be only one cross over point $C_t$ but in practice there could be many points due to the irregular shapes of the histograms. To avoid wrong crossover if the two histograms cross each other at different points $\lambda_i$ then the first few $\lambda_i$ are used to calculate the mean of the overlapping points, which are used as the threshold. Hence

$$C_t = \frac{1}{Y} \sum_{i=1}^{Y} \lambda_i$$

(3.10)

We have set $Y = 4$, which means first four points are averaged. $C_t$ is then used as threshold in the prediction process to decide whether the current pixel belongs to edge or a non-edge area.

### 3.3 Prediction Based On Gradients

We use the image gradients at the current prediction location to adapt a nonlinear predictor to those gradient direction and magnitude. Adapting to gradients helps in making better predictions than normal linear predictor or DPCM [71,72,57]. It performs better at the edges and because of its capability of separating pixels in edge area and non-edge area its overall performance is better.

Prediction is done in a raster scan order similar to the analysis process. For gradient quantities the absolute differences between pixels are calculated in each direction. These absolute differences result in lower magnitude gradient quantities in the direction of the edge and higher magnitudes across the edge. Intuitively, the pixels along the direction of the edge are considered for prediction i.e. the direction in which the gradient quantities are less. Some calculation done during prediction are redundant but can be avoided by re-using
the previous results. The speed of calculations can also be increased by parallelizing the process of calculating the \( G_q(x, y) = [d_{0}(x, y), d_{q_{0}}(x, y), d_{45}(x, y), d_{135}(x, y)] \) (3.2).

The \( G_q(x, y) \) calculated for a particular pixel location are then added to form a gradient sum \( S_{G} \) (3.8). This sum gives an idea about the local properties of the image around the pixel. If the current pixel lies on an edge or a textured area the sum tends to be high (see fig 3.4). In case of flat areas the sum is low enough. To alleviate the problem due to the relative terms low and high the previously calculated threshold \( C_{t} \) (3.10) is used. \( C_{t} \) being image adaptive results in better calcification of image areas. Fixing the parameter \( C_{t} \) to a constant value can help in cutting down the time and memory significantly with some loss in compression capacity. Based on the threshold \( C_{t} \) we classify the current position into edge area \( A_{e} \) and non-edge area, \( A_{ne} \) viz:

\[
\begin{align*}
\text{if} (S_{G}(x, y) > C_{t}) & \quad A_{e} \leftarrow A_{e} \cup (x, y) \\
\text{else} & \quad A_{ne} \leftarrow A_{ne} \cup (x, y)
\end{align*}
\]

In case of pixels that have their gradient quantity sums less than the threshold \( C_{t} \) (i.e. \( (x, y) \in A_{e} \)) linear prediction is used. The predictor selected in this case considers all the pixels in the neighboring area (i.e. context window) and predicts based on the weight vector formed by \( c_{1}(x,y) \), \( c_{2}(x,y) \), \( c_{3}(x,y) \) and \( c_{4}(x,y) \) (3.3). The two prediction vectors are defined as follows,

\[
W_{ne} = \begin{bmatrix} C_1(x,y) \\ C_2(x,y) \\ C_3(x,y) \\ C_4(x,y) \end{bmatrix}; \quad V_{ne} = \begin{bmatrix} \alpha_{i}I(x+1, y-1) + (1 - \alpha_{i})I(x, y-1) \\ \alpha_{i}I(x-1, y-1) + (1 - \alpha_{i})I(x, y-1) \\ \alpha_{i}I(x-1, y) + (1 - \alpha_{i})I(x-1, y-1) \\ \alpha_{i}I(x, y-1) + (1 - \alpha_{i})I(x-1, y-1) \end{bmatrix}
\]

(3.11)
In cases where the image area is smooth and the prediction scheme switches to binary mode, vector $W_{ne}$ will have some elements equal to 0. When the scheme switches to binary mode the prediction for the current location is assumed to be $I(x-1,y)$. Criteria used for switching is the sum of the vector $W_{ne}$ (3.11), which should be less than 1. Binary mode will be discussed in detail in the following sub sections. In case where all the $W_{ne}$ vector elements have non-zero elements the prediction value is calculated by vector multiplication of $W_{ne}$ transpose and $V_{ne}$. To normalize the prediction value, the result is divided by two.

$$P(x,y) = \begin{cases} I(x,y-1); & \text{if } \sum W_{ne} < 1 \text{ (binary mode)} \\ \frac{1}{2}W_{ne}^T V_{ne}; & \text{otherwise} \end{cases}$$

where, $X^T$ represents transpose of vector $X$.

A more precise or efficient vector can be found during the pre-analysis process, which will increase the performance of the predictor at the cost of significant extra time.

If the pixel belongs to an edge area where the sum of gradients is greater than the threshold $C_t$ (i.e. $(x,y) \in A_{ne}$), a different set of calculations is performed to make predictions. Here the $V$ vector is defined as follows,

$$V_e = \begin{bmatrix} \alpha_2 I(x+1,y-1)+(1-\alpha_2)I(x,y-1)+\beta_2(I(x+1,y-1)-I(x+2,y-2)) \\ \alpha_2 I(x-1,y-1)+(1-\alpha_2)I(x,y-1)+\beta_2(I(x-1,y-1)-I(x-2,y-2)) \\ \alpha_2 I(x,y-1)+(1-\alpha_2)I(x-2,y)+\beta_2(I(x-1,y)-I(x-2,y)) \\ \alpha_2 I(x,y-1)+(1-\alpha_2)I(x,y-2)+\beta_2(I(x,y-1)-I(x,y-2)) \end{bmatrix}$$

(3.12)

Along with the directional gradients $G_q(x,y)$ the four directional coefficients $c_1(x,y), c_2(x,y), c_3(x,y), c_4(x,y)$ are calculated. The $\xi_j$ and $\xi_2$ thresholds as give by (3.7) are used to check for the existence of at least one significant gradient. To determine the significant directional gradient the maximum of each pair $\max\{c_1(x,y), c_2(x,y)\}$ and $\max\{c_3(x,y), c_4(x,y)\}$ are compared with each other. The pair having the dominating maximum is used for prediction.
The Edge Based Predictor

whereas the other pair is ignored. The selected directional coefficient pair is further enhanced by raising the larger coefficient in the pair to the power of four. This procedure is described more precisely in the Algorithm 1 (3.13) below,

Algorithm 1:

\[ \Omega_1 = \max(c_1(x, y), c_2(x, y)) \]
\[ \Omega_2 = \max(c_3(x, y), c_4(x, y)) \]

if \((\Omega_1 > \xi_1 \mid \Omega_2 > \xi_2)\) //to check for at least one significant direction

if \((\Omega_1 > \Omega_2)\) //to determine if the diagonal gradients are dominant

if \((c_1(x,y) < c_2(x,y))\) //check if 45 degree gradient is dominant

\[ c_1(x,y) = c_1(x,y)^2; \quad c_2(x,y) = 1 - c_1(x,y) \] (3.13)

else // else 135 degree gradient is dominant

\[ c_2(x,y) = c_2(x,y)^2; \quad c_1(x,y) = 1 - c_2(x,y) \]

\[ c_3(x,y) = 0 \quad c_4(x,y) = 0 \]

else //else vertical-horizontal gradients are dominant

if \((c_3(x,y) < c_4(x,y))\) //check if 0 degree gradient is dominant

\[ c_3(x,y) = c_3(x,y)^2; \quad c_4(x,y) = 1 - c_3(x,y) \]

else //else 90 degree gradient is dominant

\[ c_4(x,y) = c_4(x,y)^2; \quad c_3(x,y) = 1 - c_4(x,y) \]

\[ c_1(x,y) = 0 \quad c_2(x,y) = 0 \]

Prediction in this case uses the same methodology as in the case of non-edge area.

The corresponding directional coefficient vector \(W_e\) is defined as follows,

\[ W_e = [c_1(x,y) \ c_2(x,y) \ c_3(x,y) \ c_4(x,y)] \]

The prediction is thus performed as before using the product of the two vectors,

\[ P(x, y) = W_e V_e \] (3.14)
Based on empirical observations the parameters $\alpha_1, \alpha_2, \alpha_3, \beta_2$ and $\beta_3$ (3.12) are set to 0.9, 0.75, 0.72, 1/8 and 1/2 respectively. The general structure of the $V$ vectors is based on the four directions of gradients. The first two elements of $V$ vector are for 45 and 135-degree predictions and the remaining two are for 0 and 90 degrees. The coefficients $\alpha_1$ and $\alpha_2$ give higher weight to the pixels in the respective direction along with some bias from neighboring pixel. In case of $V_{ne}$ the coefficient $\alpha_1$ is set to 0.9 since all other pixels are considered and so each pixel should contribute since it is a linear combination of all pixels. In $V_e$ the approximate gradients of the assumed edges directions are also added with $\beta$ as weighting coefficients. $\beta$ in case of diagonals has a smaller compared to vertical and horizontal directions due to the distance of the gradients from the current pixel location.

In both cases above (i.e. for edge and non-edge pixels) $P(x,y)$ is calculated except if the area belongs to a binary mode area. The binary mode switching is illustrated in the following subsection.

### 3.4 Binary Mode Switching

Many continuous tone images have areas where the pixel intensities don’t vary more than one level. This is especially the case for computer-generated images that have larger areas with the same intensity level. In such cases finding the error using the continuous tone image gradient-based technique is neither efficient nor necessary. It could be better to compress the pixel values in binary modes rather than compressing the error values which might have signs that need extra information. The alphabet size is so small that it can be efficiently compressed by representing them as binary streams. The proposed scheme switches between continuous mode and binary mode based on the context gradients of the current pixel being encoded.

To decide whether a pixel is in the continuous mode or binary mode area the gradients around the current pixel are checked. Sum of the coefficient $c_1(x,y), c_2(x,y), c_3(x,y)$ and $c_4(x,y)$ is checked. If the sum is less than 1 the binary mode is activated. It is easy to guess
that in cases of constant intensity areas the gradients will be zero and hence the coefficients sum will also be less than one. After switching to binary mode, the current pixel at location (x,y) is compared with the pixel $I(x,y-1)$. If $I(x,y)$ is equal to $I(x,y-1)$ then 0 is written in a separate binary stream and in the case where it fails a 1 is written as the escape sequence. In case of the escape symbol the prediction for the current pixel is set to $I(x-1,y)$ and error is encoded using the normal continuous mode method. Finally this binary stream is encoded separately using arithmetic encoder.

### 3.5 Bias Cancellation using Context Modeling and error feedback

The prediction $P(x,y)$ is computed using the gradient based non-linear predictor. In practice, the predictor suggested is not enough to model the complex correlation relation between image pixels. The lack of modeling power implies a bias on the predicted value for each predictor. To improve the prediction, we must cancel the bias on $P(x,y)$ by finding a correction term, which will be added to the predicted value after each prediction step. Let’s assume $\eta(x,y)$ is the correcting term. Hence the new improved prediction can be defined as

$$P'(x,y) = P(x,y) + \eta(x,y)$$

The residual error $e(x,y) = I(x,y) - P(x,y)$ is then modified to

$$\tilde{e}(x,y) = I(x,y) - P'(x,y)$$

The bias cancellation process can also be viewed as a form of error feedback system. The correcting term $\eta(x,y)$ has to be found by context modeling the error. Using this modeling a corrective term can be feedback after $e(x,y)$ is found to be in a particular context. The motivation behind context based error modeling is the fact that the error distribution of an image is in practice composed of various distributions. Context modeling facilitates in separating these errors into various classes. The simplest example would the errors in
smooth area will tend to remain low, while the errors in high gradient or textured area will tend to be higher. So each separate distribution will have different mean and variance. This will help in adjusting the prediction \( P(x,y) \) by adding the correcting term appropriate for that distribution. This results in a two-stage prediction method. In the first stage the prediction \( P(x,y) \) based on the adaptive gradients is made. Then based on the previously observed statistics \( P'(x,y) \) is generated and the statistics are modified for further reference. Selecting a proper context-modeling scheme is very important in enhancing the performance of the prediction system by error feedback and bias cancellation. Next we will consider the context modeling methods and the parameters used to find \( \eta(x,y) \), the correcting value for each context. In this thesis we have used sample mean error \( e_{mi} \) (3.18) as \( \eta \). And \( e_{mi} \) is the sample mean error of the \( i^{th} \) context, up to the point \( (x,y) \).

Prediction done by adapting the predictor to the gradients performs better than most of the linear predictors. Yet this does not fully exploit the complicated correlation between the image pixels. Other ways of modeling this correlation can be used to further improve the prediction \( P(x,y) \). For example, the variance of the errors in the causal context template can give us an idea about the smoothness of the image around the pixel \( I(x,y) \). The variance of the error can be directly used in modeling this correlation.

The gradients are also added as a measure of relative difference between the pixels that have the same error energy in the context but different error distribution due to different kinds of image areas. This kind of model was suggested in CALIC [25]. To model the variability, we use the error energy estimator defined as follows

\[
\psi(x,y) = [d_x(x,y) + d_y(x,y) + a \ast f_s(e(x,y))] \ast b
\]

In this case, the function \( f_s(e) \) is defined as a simple sum of error values in the neighborhood of \( I(x,y) \). Hence it can be defined as

\[
\hat{e}(x,y) = f_m(\hat{e}(x,y)) \quad \text{defined in Eq (26)}
\]

\[
f_s(e(x,y)) = (\hat{e}(x-1,y) + \hat{e}(x-1,y-1) + \hat{e}(x,y-1) + \hat{e}(x+1,y-1))
\]

The gradients selected are \( d_x(x,y) = d_{90}(x,y) \) and \( d_y = d_{0}(x,y) \) i.e. gradients in vertical and horizontal directions. The constants selected were \( a = 1/8 \) and \( b = 1/8 \). They were found by
the least square method for estimating the error energy. The parameter $\psi$ can be used to condition the probabilities of $\tilde{e}(x,y) = I(x,y) - P'(x,y)$. Thus different conditional probabilities can be formed classifying the error into different classes of variances. As we will show later $\psi$ is used in both error modeling and entropy coding. In both the cases, handling so many conditional probabilities $p(e/\psi)$ becomes difficult. So we quantize the parameter $\psi$ into more manageable number of levels. Designing the optimal quantization levels can be done using dynamic programming or other techniques [73,74]. The ultimate goal and hence the cost function is to reduce the total entropy of the error [25].

The random variable $\psi$ needs scalar quantization [75]. We have suggested 9 levels of quantization (3.16)(Fig 3.5), which were designed using tests on various images. The quantization function of $\psi$ can be defined as $Q(\psi)$ which will quantize $\psi$ into $|L|= 9$ levels.

$$Q(\psi(x,y)) = q_i \quad L(i) < \psi(x,y) < L(i+1)$$

$L = [0,2,3,4,5,8,14,19,37,50,\infty]$

(3.16)

![Figure 3.5: The Quantization Scheme for the error energy parameter.](image)
The quantized value thus take nine different values \([q_1, q_2, \ldots, q_8]\). The quantization vector \(L\) can be designed for each image this could be costly. The result would be certainly better than using an image independent \(Q(\psi)\). Now the conditional probability distribution \(p(e/ Q(\psi))\) will result in \(|L| = 9\) possible choices. The choice of \(|L|\) deserves a careful consideration. With a small value for \(L\), we would have the problem of context dilution, while a large value of \(L\) could result in problem of context scarcity. \(|L|\) can be changed based on the size of the image.

Error feedback based on error energy estimate \(Q(\psi)\) proves to enhance the performance of the prediction significantly. But still there are higher order structural relations, which cannot be resolved by error energy. In case of area like texture patterns sometimes the higher order statistics cannot be captured by \(Q(\psi)\) since it does not consider the structural pattern, as it is location independent. To alleviate this problem a context that considers location of events around the pixel has to be used. The context should maintain structural details.

Depending on the selection of the parameter the performance of the model varies. So for context based error modeling, the context in which the residual error \(\tilde{e}(x, y)\) occurred has to be determined. If the context were denoted as \(\theta\) then the probability \(p(e/ \theta)\) would be used in error coding and bias cancellation. The causal template of current pixel \(l(x,y)\) can also be defined as the context \(\theta\). The number of possible context number for \(\theta\) in this case would be large. Since in a 256 level image with the context length \(m\) the possible range of \(\theta\) would be \(m^{256}\). The context length \(m\) in this case is 12 so the total number of contexts of \(\theta\) will be \(|\theta| = 12^{256}\). Though this kind of modeling would be the most precise but practically impossible. To build a probabilistic model based on such a context will result into two main problems. One would be the insufficient statistics produced by such a huge number of contexts for small images. Second would be the space required to store the models. To tackle this problem the context \(\theta\) can be redefined and quantized to reduce the number of possible contexts. Quantizing the contexts in an efficient way not only reduces
space requirements, but proper selection of the contexts also leads to better compression performance.

The approach we use for location dependent context based error-modeling follows the technique described in [25]. So \( \theta = \{ u_0, u_1, \ldots, u_m \} \), where \( u_i \) are the events and the number of events have the range \( 0 < i < m-1 \). The context \( \theta \) is formed by placing all the surrounding pixel of \( I(x,y) \) in the vector. But in this case the last two elements are modified and the vector \( \theta \) can be represented as

\[
\theta(x,y) = \{ u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8 \} = \{ I(x-2,y), I(x-1,y), I(x-1,y-1), I(x,y-1),
I(x,y-2), I(x+1,y-2), 2 * I(x,y-1) - I(x,y-2), 2 * I(x-1,y) - I(x-2,y) \}
\]

Maintaining the structure as it is and using \( \theta \) in estimating \( p(e/\theta) \) will give the more accurate results, but will practically difficult to handle. Similar to \( \psi \) we can also quantize \( \theta \) for easy handling. But in case of \( \theta \) a vector quantization is done since maintaining the structure is important. To do this each element of the vector \( \theta \) is quantized into 2 values 0 or 1, using the prediction value \( P(x,y) \) as a threshold. This generates a binary number \( B = \{ b_7, b_6, \ldots, b_0 \} \) (3.17).

Applying vector quantization on \( \theta \) as described, we obtain

\[
Q(\theta) = B = \{ b_7, b_6, \ldots, b_0 \}
\]

where

\[
b_k = \begin{cases}  
0 & \text{if } u_k \leq P(x,y) \quad k=0,1,\ldots,7 \\
1 & \text{if } u_k > P(x,y)
\end{cases}
\]

The last 2 elements of \( \theta \) are \( 2I(x-1,y)-I(x-2,y) \) and \( 2I(x,y-1)-I(x,y-2) \). Comparing these values with the threshold helps in finding whether the pixel forms a concave or a convex waveform with its neighbors. Resolving this helps in classifying the pixels that have same gradient but with different gradient direction near the pixel. The binary number \( B \) formed
can take 256 values i.e. $2^8$. This is a small number to handle, but still captures the major structural formations. Now we can estimate conditional probabilities of the error conditioned on $B$, $p(e \mid B)$.

Since texture and error energy both are important in context modeling a combined context is formed using $0 < Q(\theta) < 9$ and $0 < Q(\psi) < 256$. The total possible combined contexts are $Q(\psi), Q(\theta)$. To find the correcting factor $\eta(x,y)$ the two context models are combined.

The combined context can be formed as follows,

$Q(\psi) = \psi_p, p \in \{1, 2, 3, \ldots, \mid p \mid\}$ \quad $\forall p, \psi_p = p$

$Q(\theta) = \theta_q, q \in \{1, 2, 3, \ldots, \mid q \mid\}$ \quad $\forall q, \theta_q = q$

The combined context is formed as follows

$\Phi_j = (\psi_p - 1)q + \theta_q, j \in \{1, 2, 3, \ldots, (pq)\}$

in our case $\mid p \mid = 9, \mid q \mid = 256$

It can be seen that $mn$ different conditional distributions of $\tilde{e}, p(\tilde{e} / \Phi)$ conditioned on the contexts are formed. This method of conditioning models is like product quantization of two image features, texture and error energy, which are dependent on each other. Merging some of the quantization can do further reduction in the number of compound context.

But even after doing so actual estimation of $p(\tilde{e} / \Phi)$ will be difficult and space consuming. Hence instead of estimating the probability distribution we estimate the expectation of the error value $e_{mi}$ for a given context defined as

$e_m = E\{\tilde{e} / \Phi\}$

For $i^{th}$ context $e_{mi}$ is formulated as follows

$e_{mi} = E\{\tilde{e} / \Phi_i\}$

$= \sum_{\tilde{e}_j = 0}^{256} \tilde{e}_j P(\tilde{e}_j / \Phi_i)$

$= \sum_{\tilde{e}_j = 0}^{256} \tilde{e}_j P(\tilde{e}_j / \psi_p) P(\psi_p / \theta_q) P(\theta_q)$
This value i.e. the mean error value $e_{mi}$ for a given compound context can be found by accumulating the error values and the count of error values occurred in that context. The mean error value is the estimation of the expected error for a given context and the offset for current prediction can be corrected using this value $e_{mi}$. Hence the correction factor $\eta(x,y)$ (3.18) is assigned the value $e_{mi}$. So the bias is removed by adding the mean error value. The improved prediction will be as follows.

$$P(x,y) = P(x,y) + \eta(x,y)$$

where

$$\eta(x,y) = e_m = E[\tilde{e}/\Phi_i]$$

It has to be noted that the new error term $\tilde{e}(x,y) = I(x,y) - P'(x,y)$ has to be encoded and further used for calculating the expectations.

### 3.6 Symbol Remapping

The residual error can take values from $-2^8$ to $2^8-1$. The error value can be treated, as they are i.e. the absolute error values and signs separately. But this leads to extra overhead in compression process. The performance is degraded by treating the signs separately because of the implicit assumption of uniform distribution for the signs. So it was suggested to remap the negative symbols into positive symbols. The simplest ways to do this is to shift the mean of the distribution to 256. Another way is applying the following scheme, which is a simple remapping of the error values in the range of 0 to 512. The function $f_m(e)$ can be defined as follows

$$f_m(\tilde{e}) = \begin{cases} 
2^*|\tilde{e}| & \text{if } \tilde{e} \leq 0 \\
2^*|\tilde{e}| - 1 & \text{if } \tilde{e} > 0
\end{cases}$$

The disadvantage of this system is as mentioned; the alphabet size is twice the alphabet size of the pixel values. The sign mapping technique suggested in this method can remap the error values in the same alphabet size as the pixel values. So the error values can be
mapped into the range 0 to $2^8-1$. This can be done using the constraint that the value $\tilde{e} = I(x, y) - P'(x, y)$ must fall into the interval $[-P(x, y), 255 - P'(x, y)]$. Since we know the prediction value $P'(x, y)$ and the error value $\tilde{e}$ we can remap it into the range [0, 255].

$$\hat{e}(x,y) = \begin{cases} 
2^* |\tilde{e}(x, y)| & \text{if } \tilde{e}(x, y) \leq 0 \\
2^* |\tilde{e}(x, y)| - 1 & \text{if } \tilde{e}(x, y) > 0 \\
|\tilde{e}(x, y)| + P'(x, y) & \text{if } \tilde{e}(x, y) > P'(x, y) \leq 128 \\
|\tilde{e}(x, y)| + (255 - P'(x, y)) & \text{if } (255 - |\tilde{e}(x, y)|) > P(x, y) > 128
\end{cases}$$

(3.19)

Since the decoder also has the same information available as the encoder before encoding a pixel, the decoder can follow the same procedure to get the original error value. The remapping process (3.19) also has another important advantage. Since the smaller error values are more probable than larger values the ordering of the error values from 0 outwards by the error magnitude gets them ordered in decreasing probability of occurrence. This ordering can be exploited in further encoding of the residual errors.

### 3.7 Error encoding

The ultimate goal of any compression scheme is to encode the error values. The adjusted predictions $P'(x, y)$ are used to find the error value $\hat{e}$. For entropy encoding the context-based error modeling is used.

Arithmetic coding used in the suggested method is driven by 9 different conditional distributions. Classifying the error based on their contexts forms these distributions. The context modeling of the error done for entropy coding is performed at the encoder based on a similar context definition used in the bias cancellation process. The context formed is based on the error energy factor and few gradients. The texture based context modeling is not used since it is not possible to maintain so many contexts for driving an entropy encoder. The context model described in (3.7) is used as it is with different values of the constants. The function $f_s(e)$ (3.7) and quantization function $Q(\psi)$ (3.8) also have the same definition except the quantization vector $L$. 
The quantized error energy is used in classifying the errors into nine different classes. The parameter thus formed is also biased by a certain amount by the previous parameter calculated for the previous pixel position. Biasing helps since the error energy tends to be correlated. $\psi$ is modified as follows,

$$\psi(x, y) = \alpha_3 \psi(x, y) + (1 - \alpha_3) \psi(x - 1, y)$$

The $\alpha_3$ parameter used for simulation was set to 0.1. The probability models for the different contexts are formed by accumulating the errors belonging to the same context. Arithmetic encoding is done separately on each of these streams.

The binary stream generated by the binary mode is also compressed using arithmetic coding. Since there is no context modeling involved in binary mode the binary stream is compressed by a single arithmetic coder.

The compressed data has minor overhead information about the image such as width, height and 5 parameters required for the predictor. The overhead also includes the two-byte counts of the binary stream symbols. The total overhead is less than 20 bytes, for an $N_1 \times N_2$ which can be practically ignored.
Chapter 4: Edge Based Least Square Predictor

4.1 Introduction

We have discussed the edge-based prediction in the previous section, which performs better than DPCM methods and linear prediction. Edge based prediction methods rely on the gradients of the image around the pixel to be predicted. The predictor’s ability to adapt to the edge areas according to the gradients of the image makes it possible to make predictions along the edge. With fewer gradient calculations the approximate direction of the edge can be estimated. This technique specializes in switching between set of predictors based on the analysis around the location of interest. Given that the set of predictors are empirically tuned, finding the parameters that will be optimal for all images is a difficult challenge. Further the predictor as explained assumes the edges to be broadly classified into four discrete directions: 0, 90, 45 and 135 degrees. But in natural continuous tone images, edges could have many different formations, and hence may not appear in perfect directions, as assumed by the predictor. The directions assumed lead to set of equations which consider pixels in fixed locations with different coefficients.

To alleviate this problem a better predictor is applied when edges are detected. Local optimization of the prediction scheme helps in increasing the performance of the predictor [76,77,78]. A set of optimum coefficients can be calculated for each pixel using auto regression [79]. Least square based adaptive prediction methods have been shown to provide in better prediction then switching based predictors. Lossless JPEG used local optimization in their pipeline for achieving better compression. Proposed scheme uses least square method for finding a set of coefficients for all the pixels in the context. The major drawback in using pixel-by-pixel LS based optimization is its high complexity in time and space. Many schemes have been suggested based on intelligently applying LS approach [62]. In this thesis, we propose a specialized switching method that will avoid extensive application of the least square method. In some cases it might be possible to use pre-
designed predictors instead of optimizing one. Thus a smart switching technique could be used to achieve a significant reduction in computation. For example, in case flat regions a simple linear predictor will produce the same results as the least-square based adaptive predictor and will require less time. In regions where the pixel intensities are constant applying LS-based approach will be wasting lots of resources and absolutely unnecessary. In such regions the compression can be further improved by using the binary mode switching technique. The performance after employing a least square predictor in combination with the edge based predictor is significantly better than the previously suggested method. We will be discussing the new scheme in detail in the following sections.

4.2 Pre-analysis

Analysis of the image is done similar to the edge based predictor scheme. The main idea is to classify the pixels into edge and non-edge area pixels. For this the image is first analyzed using edge detector. Here the Canny edge detector is employed with a threshold of 5% of the maximum. Details have been presented in section X on edge-based predictor. The results of the analysis include a threshold classifying the pixels into edge area and non-edge area. The threshold \( C_t \) which is found by overlapping the two histograms \( H_e \) and \( H_{ne} \) of the sum of the gradient quantities. Pixels in the edge area tend to have larger gradients sums, whereas the pixels in the flat area tend to have lower values.

To aid in identifying the orientation of the edge four more \([c_1(x,y), c_2(x,y), c_3(x,y), c_4(x,y)]\) parameters are found. These four parameters are two pairs of coefficients for the gradient orientations 90 degrees apart. Each coefficient measures the dominance of the gradient orientation in the pair of directions. So \( d_{45}(x,y) \) and \( d_{135}(x,y) \) gradients will have coefficients \( c_1(x,y) \) and \( c_2(x,y) \) associated with them. Similarly gradient quantities \( d_{90} \) and \( d_0 \) will have the pair \( c_3(x,y) \) and \( c_4(x,y) \). Remember that the sum of each pair of coefficients is one in case of non-flat area. The maximum coefficient in each pair shows the dominance of the
edge orientation in that pair. Similarly maximum of the two dominants can give us an idea about the actual edge orientation.

In the following method, a least-square based method is employed in making the pixel prediction. Where the pixel is classified as being in a flat area, a simple linear predictor is used. In case of the pixels that cannot be handled by least square predictor because of the case of multiple solutions for a single pixel an edge based prediction is applied. Pixels which lie outside the operational area of the least square method are also predicted using the edge based predictor.

4.3 Lease-Square Method for Prediction

As discussed earlier the prediction scheme performance depends on its ability to decorrelate the data. With better error decorrelation ability we will typically have a better performance in terms of error modeling. Pixels are processed in a raster scan order, so prediction of the current pixel can be done from the causal data available. We call this causal window as the context window. The image can be assumed to be a finite order Markov Source and hence prediction of the next pixel can be made from the previous pixels. The pixels selected for the prediction purpose are from the context window. An approximate assumption on the order of the image data source can be made and based on which the size of the window is determined. So selection of the proper size of the context has to be done in order to facilitate better modeling. It is also assumed that all the pixels in the context window belong to the same model. Markov sources have the property predictability of the next state based on the previous state. In case of images the current pixel can be predicted from its context pixels using a linear combination weighed by a set of coefficients that can be designed. Consider an order-n predictor. Let \( \tilde{X} = [x_1, x_2, \ldots, x_n] \) be the pixel vector in the context window. Let \( \tilde{a} = [a_1, a_2, \ldots, a_m]^{\top} \) be the corresponding vector of prediction coefficients. The predicted pixel value at the current location, denoted \( aX_{n+1} \) will be given by
$$P(X_{n+1}) = \bar{a} \cdot \bar{X}$$  \hspace{1cm} (4.1)

This set of coefficients can be solved by the Least-Square optimization method [62]. Least-Square (LS) based adaptive prediction scheme can achieve optimal prediction by approximating the nontrivial edge orientations. LS based prediction scheme locally optimizes the coefficients for the context pixels to achieve orientation adaptation. The context selected determines the size of coefficient vector to be found. For the proposed LS method, the context window (Fig 4.1) size is the same as that of the window selected for the edge based predictor. Most common selection of the context window is a double-rectangular window size determining the order of the source. The window is as shown bellow,

<table>
<thead>
<tr>
<th>X(n-11)</th>
<th>X(n-8)</th>
<th>X(n-6)</th>
<th>X(n-9)</th>
<th>X(n-12)</th>
</tr>
</thead>
<tbody>
<tr>
<td>X(n-7)</td>
<td>X(n-3)</td>
<td>X(n-2)</td>
<td>X(n-4)</td>
<td>X(n-10)</td>
</tr>
<tr>
<td>X(n-5)</td>
<td>X(n-1)</td>
<td>X(n)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4.1: Context window structure for least square prediction neighbors.

$X(n)$ is the pixel to be predicted and $X(n-i)$ (i=1,2,…N) are the causal neighbors of the pixel $X(n)$. The dimensions of the context window are generally of the form shown in Fig.2.2,

Figure 4.2: Context window dimensions for least square predictor context.
Using the above structure for the context window the total number of causal neighbors will be $M = 2T(T+1)$. The context window can be represented as an $M \times 1$ column vector $Y = [X(n-1), X(n-2), \ldots, X(n-M)]^T$. Another matrix $C$ (4.2) that is the prediction matrix of size $M \times N$ can be formed as follows,

$$
C = \begin{bmatrix}
X(n-1-1) & X(n-1-2) & \ldots & X(n-1-N) \\
X(n-2-1) & X(n-2-2) & \ldots & X(n-2-N) \\
\vdots & \vdots & \ddots & \vdots \\
X(n-M-1) & X(n-M-2) & \ldots & X(n-M-N)
\end{bmatrix}
$$

(4.2)

where each member of the matrix C i.e. $X(n-i-j)$ is the $j$th prediction neighbor of $X(n-i)$. With this formulation, finding the optimal The closed form solution of the LS-optimization as shown below can be used,

$$
A = (C^T C)^{-1} (C^T Y')
$$

(4.3)

The above coefficient vector $A$ is solved through LS-optimization for minimizing the sum of square prediction errors $\|Y-CA\|_2$. The vector $A = \tilde{a} = [a_1, a_2, \ldots, a_m]^T$ thus achieved is the optimized coefficient vector. We can see that the solution involves 3 matrix multiplications; one matrix inversion and two matrixes transpose operations (4.3). These operations are repeated at least $(W-4)\times(H-2)$ time if applied on every pixel where $H$ and $W$ are the dimensions of the image. So this is the major source of the computation problem. By carefully selecting the area to apply the LS-optimization, a significant cost reduction can be achieved.

The reason for the LS based adaptive predictor to work best on edges can be explained as follows. The pixels in the context window can be classified into two main classes such as edge and non-edge [62]. For non-edge area context the matrix $C$ is usually not a full ranked
matrix and the LS-optimization does not have a unique solution. In that case edge-based predictor can handle the pixels. For the edge area neighbors, the matrix is usually full-ranked and the LS optimization has a unique solution. Hence in case of edge areas the context window acts as training window for the LS optimization and an optimal coefficient vector can be found. The coefficient vector found is thus used to predict the current location pixel based on the assumption, that the current pixel is the part of the same local image model.

4.4 Prediction Based on Switching

From the forgiven, effective prediction can be achieved by combining both LS based adaptive predictor and edge based predictor using a switch mode. To further improve compression performance, enhancement on images that have large areas with constant pixel intensities, a binary mode is added. Intuitively, it can be said that LS based adaptive predictor should be applied only to the edge area pixels and not to other areas. The reason for such idea is to avoid the time required by the LS approach to find out whether the context processed belongs to a flat area. In the proposed LS-EBP method a threshold is used decides whether a pixel is to be processed by the EBP or LS based adaptive predictor.

If the sum $S_G$ falls below the threshold then the current pixel is believed to be a part of a low variation area. The flat area prediction section of the edge-based predictor then treats this pixel.

\[
\text{if } S_G > C_t \\
\quad (LS, EBP_{edge})
\]

\[
\text{else} \\
\quad (EBP_{flat})
\]

The sum of the directional coefficients $[c_1(x,y), c_2(x,y), c_3(x,y), c_4(x,y)]$ is checked against one. Most of the time the sum is below one since the image area having $S_G$ below the threshold $C_t$ has high probability belonging to flat region. In this case the pixels are processed using the binary mode.
In binary mode the pixel under consideration is compared with the pixel \( I(x, y-1) \). If it is same as its neighbor then a 0 is encoded in a separate stream. In case of a failure an escape symbol 1 is encoded in the same stream. The pixel is encoded using the normal flat area assumption of edge based predictor and assigned the same value as the \( I(x-1, y) \). If the sum of the directional coefficient itself is higher than one then simple linear prediction is done using all the pixels around the pixel to be predicted and directional coefficients as the prediction coefficients.

\[
\text{if } \sum (W_{ne}) < 1 \\
\text{Binary mode} \\
P(x, y) = I(x, y-1) \quad (4.4)
\]

\[
\text{else} \\
P(x, y) = \frac{W_{ne}V_{ne}}{2}
\]

Where \( V_{ne} \) is the surrounding pixel linear combination to make linear equation and the parameter \( \alpha \) in \( (4.3) \) is set to 0.95 in this case.

In case of the pixels where the \( S_G \) is greater than the threshold \( C_t \) combination of LS based optimization and edge based predictor is used as follows. The current pixel location is checked if it lies in the operational area of the LS based method. If the pixel location is outside the bound of square defined by two opposite vertices then it is processed by the edge-based method. \( \xi_1 \) and \( \xi_2 \) are found in the pre-analysis process as discussed earlier in Algorithm 1. The coefficients \( C_i(x,y), i=1,2,3,4 \) are computed using the Algorithm 1 (3.12) which has minor modifications. Here we append the following two lines at the end of Algorithm 1.

\[
\text{else} \\
P(x, y) = \text{median}(I(x-1,y), I(x,y-1), I(x-1,y) + I(x,y-1) - I(x-1,y-1))
\]

In this case \( \alpha_3 = 0.75 \) and \( \alpha_4 = 0.72 \), while \( \beta_4 \) and \( \beta_5 \) are set to 16 and 2 respectively.

The elements of the \( V_e \) vector are prediction equations for each of the four directions. Depending on the prediction direction the gradients that are taken into account and the pixels considered also differ. The directional coefficients are multiplied to these directional
predictions to give importance to the dominant gradient direction. If the condition \( if(\Omega_1 > \xi_1 | \Omega_2 > \xi_2) \) fails then simple median predictor \( P(x,y) = \text{median}(I(x-1,y),I(x,y-1),I(x-1,y)+I(x,y-1)-I(x-1,y+1)) \) is used.

In the area where the LS based adaptive predictor can be used the previously discussed LS approach is applied. The rank of the matrix \( C \) is checked with a predefine threshold \( T_r \) in this case depending on the order of the context. If the rank of the matrix \( C \) is less than \( T_r \) then the previously described median filter is used. The closed form solution of the LS based approach is used when the rank criteria is satisfied. The prediction \( P(x,y) \) is made using the coefficient vector \( A \) obtained from the LS approach and all the pixels in the context window.

\[
P(X_{n+1}) = \tilde{a}.\tilde{X}
\]

Note that \( A = \tilde{a} = [a_1, a_2, \ldots, a_m]^T \) is the coefficient vector found using the LS approach and \( \tilde{X} = [x_1, x_2, \ldots, x_n] \) is the pixel vector from the context window. It is seen that the prediction by LS approach is better in case of strong non-trivial edges. The advantage of switched prediction with LS approach is that it cuts on time and does better prediction in case of flat regions. The context modeling in case of LS-EBP is same as Edge based predictor scheme described earlier. Context modeling for error encoding is done using gradients around the current pixel.
Chapter 5: BWT and Image Encoding

As discussed earlier the Burrows Wheeler Transform (BWT) reorganizes the 1D data to form clusters that are easy to compress. The effective compression performance of BWT in text compression is due to the 1D correlation in text data and high context repetition. Encoding the error value is an important phase in image compression. But image information is two-dimensional and has to be linearized into a 1D data stream before the BWT can be applied.

Using the BWT as a black box does not produce good results on raw images [64]. This is due to the lack of exploitation of the two dimensional correlations and dearth of exact context repetition. In other words images don’t show the data structure similarities as text. Images have context repetition, but the contexts are usually similar and not exactly same.

Some other issues were also studied [80]. Therefore we found a need to modify the BWT to make it compatible for image compression and as a transform for other data sources too.

In this section, we will discuss methods inspired by the BWT that can be used on any data source. We will also present further methods specific to image compression. The different methods that were studied eventually did the same thing of reordering the data and then recovering it in a lossless form. The overhead added due to these methods has also played an important role in checking for the suitability of these methods. Some of the methods are directly inspired by the BWT and some are derived from a similar concept as discussed below.

5.1 F2BWT

The F2 in F2BWT stands for the second column of the BWT table (Fig 2.1). In regular BWT we rotate the stream \( S^N \) of length \( N \) and form \( N \) such rotated streams. Each stream from the \( N \) streams is a shifted version of the previously shifted stream. Then these \( N \) streams are lexicographically sorted and arranged in a matrix form. The last row of the
matrix i.e. \( L^N \) is used for compression. Due to the inherent property of the BWT the order required for recovery is maintained even after sorting. This helps in recovering the original stream \( S^N \) with the help of a single integer \( Id \) and the recovery vector generated using the last column \( L^N \). Due to repeating contexts, the lexicographical sorting clusters the similar elements together, this after applying move to front (MTF) coding gives good compression.

There is no mathematical proof for the improvement in sorting as we move from the last column to the first. But it is oblivious that the first column has to be the best-sorted column since it is just the sorted stream of the original stream \( S^N \). But after sorting, the vector required to recover the original stream from the first column could cost as much as the original stream. Hence it is not preferred to use the first column. In the F2BWT method a thought was given on using the second column of the reordered table of the BWT.

Let the stream to be compressed be \( S^N \), such that \( S_i \in \Sigma, i = 1, 2, ..., N \), where \( \Sigma \) is the symbol alphabet. The stream is rotated \( N \) times to form a \( N \times N \) matrix \( M' \). Each row in \( M' \) is a cyclic rotation of \( S^N \). Let \( w_1^N = S^N \) be the first row vector of the matrix \( M \).

The remaining rows in \( M' \) are obtained using a simple relation:

\[
w_{i+1}^N(p) = w_i^N((p + 1) \mod N) \quad p = 1, 2, ..., N \quad ; \quad i = 1, 2, ..., N
\]

Thus the BWT matrix formed is given by:

\[
M' = \begin{bmatrix}
w_1^N \\
w_2^N \\
\vdots \\
w_N^N
\end{bmatrix}
\]

The matrix \( M' \) is then lexicographically sorted to get a new matrix \( M \).

Denote \( A(k, \cdot) \) as the \( k \)-th row in matrix \( A \). Similarly let \( A(\cdot, k) \) be the \( k \)-th column. The first and second columns of the matrix \( M \) (denoted \( F1 \) and \( F2 \) respectively) are determined as follows,
\[ F_1 = M(\square_1) \]
\[ F_2 = M(\square_2) \]

The idea and motivation behind using F2, the 2\textsuperscript{nd} column of BWT is that, given the second column, the first column F1 can be easily generated by simply sorting F2. We observe that when we obtain the first two columns F1 and F2 we already have all the bigrams existing in the stream. The bigram vector can thus be formed as follows,

\[ B^2_i = [F^2(2i) * F^1(i)] \quad \forall i, \]

where the symbol "*" denotes concatenation.

The order of the bigrams is not maintained straightforward as in the original BWT. There are many possible streams that can be generated from the available bigrams by making possible combinations. Though the number of possible streams generate by combinations formed by the bigrams from the alphabet is large, the possible combinations generated by following the constraints of the available bigrams will result in a very less number of streams. The only problem with this constrained reconstruction of the stream from the bigrams is that the reconstruction procedure has to be guided at each step, or an identification number (ID) has to be provided to distinguish the right stream from all the possible generated streams. To address these problems, we suggest methods to correctly reconstruct the input stream.

5.1.1 \textit{F2BWT-Unique ID}

As discussed earlier the possible number of streams that can be generated by the using the bigrams is constrained due to the number of available bigrams. If we look at this problem from a tree construction point of view we can see that the leaves at depth N are the valid strings that can be formed using all the available set of bigrams.

Let us consider an example where a string \( S^N = \text{"abedbeacbe"} \) has to be compressed and transmitted. If we find the F2 i.e. second column of the BWT matrix for this string and then
sort it to get F1 (i.e. the first column), we can get the set of bigrams
\[ B_{1,2,...,N}^2 = \{aa, ab, ac, bc, be, ca, cb, cd, db\} \]. It has to remember that only F2 is
compressed since F1 can easily be recovered. The decompression module will have the
set \( B_{1,2,...,N}^2 \) generated from the F2 and F1. Each bigram in link has to be followed by an
overlapping bigram with one symbol overlap. In other words, an overlap constraint (5.1)
has to be satisfied, which is defined below by the bigram concatenation operator \( \otimes \),

Let
\( A^B \) be concatenation operator of A and B.
\( A \otimes B \) be overlapping concatenation operator of A and B.
That is
\[
A \otimes B = A[1,2,....n-1] * B[1,2,...,m]
\]
\[ |A| = n, |B| = m \]

Using the above overlapping concatenation operator, the different possible streams can be
found using the bigrams and the posed constraints on the construction process,
\[
SS_p^N = B_{r_p(1)}^2 \oplus B_{r_p(2)}^2 \oplus \ldots \oplus B_{r_p(N)}^2
\]
where
\[ p = 1, 2, \ldots, P_{\text{max}} \]
\[ P_{\text{max}} = \text{max number of valid strings i.e with strings with length } N \]

But the exact match of the original string is our interest. So let \( p = ID \) corresponding to the
vector \( r \) which results in the original string,
\[
S^N = B_{r_p(1)}^2 \oplus B_{r_p(2)}^2 \oplus \ldots \oplus B_{r_p(N)}^2
\]
The \( P \) vectors that result in a valid length string can be found by constructing a tree. For the
construction of the tree and to form a valid stream from the available bigrams a histogram
of unique pairs is formed,
Let \( \tilde{B}^2_{k,k=1,2,\ldots,m} \) be the \( m \) unique biagrams in \( B^2_{1,2,\ldots,N} \).

Define

\[
\delta(a,b) = \begin{cases} 
1 & ; \quad a = b \\
0 & ; \quad otherwise 
\end{cases}
\]

Then the histogram is given by:

\[
H_b(\tilde{B}^2_k) = n_k = \sum_{m=1}^{N} \delta(B^2_m, \tilde{B}^2_k)
\]

The histogram in case of our example will be \( H_b(\tilde{B}^2_{1,2,\ldots,8}) = [1,1,1,3,1,1,1,1] \). The tree construction process can then be formulated as follows.

The path from the root to a node \( ND \) can be defined as \( P^L \) of length \( L \). Let \( nd_i \) be the \( i^{th} \) node.

1. Let \( P^i = \tilde{B}^2_g = S[1,2] \) be the first bigram in \( S^N \). The number \( g \) has to be known by the decoder.

2. For next bigram search \( \tilde{B}^2_k \) \( \forall k \) which satisfies \( P^i \otimes \tilde{B}^2_k \) (i.e \( P^L(L) = \tilde{B}^2_k(1) \)). At \( i^{th} \) node \( nd_i \), form a vector of \( k \) values that satisfies the overlap concatenation constraint. \( nd_i = [k_1, k_2, \ldots, k_q] \).

3. If \( |nd_i| > 2 \)

\[
H^nd_b(\tilde{B}^2) = H_b(\tilde{B}^2)
\]

make current node \( ND = nd_i \)

3a. If (there is path available)

choose one available path

update \( P^i = P^L \otimes \tilde{B}^2_j \)

decrement histogram count: \( H_b(\tilde{B}^2_j) = H_b(\tilde{B}^2_j) - 1 \)

\textbf{goto} step 2

3b. If \( |nd_i| = 1 \) //no decision

update \( P^i = P^L \otimes \tilde{B}^2_j \)

decrement histogram count: \( H_b(\tilde{B}^2_j) = H_b(\tilde{B}^2_j) - 1 \)

\textbf{goto} step 2
3c. if $|nd| = 0$ //path exhausted
   if $L \neq N$ // invalid path
      goto step 3a
   elseif $L = N$ // valid path
      if $pathcount = ID$
         $S^N = P^L$, Stop
      else
         $pathcount = pathcount + 1$
      goto step 3a
   
In our example, 6 valid strings are formed, so $P_{\text{max}} = 6$, and $ID = 5$, which corresponds to original string is transmitted to the decoder. The procedure is illustrated in (Fig 3.1) for the running example.
<table>
<thead>
<tr>
<th></th>
<th>aa</th>
<th>ab</th>
<th>ac</th>
<th>bc</th>
<th>ca</th>
<th>cb</th>
<th>cd</th>
<th>db</th>
</tr>
</thead>
<tbody>
<tr>
<td>nd0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>nd1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>nd2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>nd3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nd4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>nd5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 5.1: Tree construction and traversal for $S^N = [a,b,c,d,b,c,a,a,c,b,c]$, $F_2 = [a,b,c,c,c,c,a,b,d,b]$, $g = 2$ and ID = 5.

For some of the nodes represented in the Fig 5.1a above the modified histograms are as shown in Fig 5.1b

In Fig 5.1 the histograms shown depict some of the nodes and the histograms modified till those nodes are reached. Due to constraints added, it is observed that the number of possible valid strings is much smaller compared to all the possible combinations. The overhead added due to this method is due to the ID size. For binary stream it is the same size as the length of the stream itself. So it provides no compression but for an alphabet of size $|\Sigma|$ it is certainly less than the length $N$. Thus the data needed by the decoder to correctly decode the original string is the integer ID, the second column F2 and g. F2 being highly sorted can be efficiently compressed. The final compression is better than the first order Markov compression. However, the space and time complexity makes this approach impractical.

A closer observation will reveal that the problem of generating the tree is more or less like arithmetic coding. The only difference is that arithmetic coding stops compressing a string after a certain threshold is reached. Arithmetic coding compresses a string by dividing a unit space into intervals of varying size based on the probability of each symbol. Then it goes on adding information as the symbols are encoded till a non-representable fraction is
reached. This fraction is then represented as a binary code and new symbol encoding is started afresh.

### 5.1.2 F2BWT-Path Definition

The method discussed earlier will prove to be more time consuming since the encoder and decoder will have to try all the possible constrained possibilities. Even though they are small in number compared to all the possibilities, the number is still significant enough to consume a lot of resources. The time complexity is exponential in \( N \), where \( N \) is the length of the sequence. To solve this problem, the process of tree construction can be guided and followed in the same manner by both the encoder and decoder. Since the encoder is aware of the information available it can cut down on time by not following the wrong branches while reconstructing the tree. The wrong branches in this context refer to the branches that end prematurely due to exhausted bigrams (i.e. failure of the constraint). The path definition process includes generating a code stream defining decisions at each point of multiple possibilities,

Let

\[
R^2 = S[1, 2]
\]

\( R^2 \) is the first bigrams of the original sequence \( S^N \). The new sequence \( R^i \) (i=2 at first assignment) is then extended by the overlapping concatenation operator \( \otimes \) (5.2) as follows,

\[
R^{i+1} = R^i \oplus \tilde{B}_k^2 \quad \quad i = 2, 3, \ldots, N
\]

where

\[
\tilde{B}_k^2 = S[i, i+1]
\]

\[
H_b(\tilde{B}_k^2) = H_b(\tilde{B}_k^2) - 1
\]

(5.2)

The histogram \( H_b(\tilde{B}_k^2) \) is modified as the bigrams are used in extending the \( R^i \). Using the modified histogram and based on the previous bigram, a conditional probability
\[ \mathcal{G} = p(H_2(\hat{B}_i^2) / R'(i - 1, i)) \] is formed. Using this condition probability \( \mathcal{G} \), a Huffman code is generated \( C^{\mathcal{G}} = -\mathcal{G} \log_2 \mathcal{G} \) and concatenated to the path definition code stream,

\[ C = C^{\mathcal{G}} \]

Thus the path definition \( C \) is then transmitted to the decoder with the first two elements \( R^2 \).

The compression ratio can then be given by,

\[ C.R. = \frac{|C|}{N \log_2 |\Sigma|} \]

Essentially this scheme chooses the correct path at each decision point and codes this with the codeword using the possible symbols at the node.

### 5.2 \( \alpha \)-paths

If the second column of the F2BWT scheme is used then it can be observed that the bigrams can be used to form a conditional probability distribution of second column elements conditioned on the first column elements. The second column represents the elements that are followed by a particular element in the first column as in the original stream. The F2BWT sorts the first column elements and does an internal sort among the elements that follow. Since the recovery of the sorted elements is difficult due to the missing order due to sorting, the order has to be maintained in an efficient manner.

The method suggested in this thesis maintains the order of the elements using the \( \alpha \)-paths. Here \( \alpha \) refers to the context that is either a combination of the few elements in the original stream or some parameter formed of the context elements. The elements that follow this \( \alpha \) parameter are accumulated in a separate stream for each \( \alpha \). We call this stream as an \( \alpha \)-path (5.3). The order of occurrence of the elements that follow the \( \alpha \) in the original stream is maintained as it is in the \( \alpha \)-path too. These \( \alpha \)-paths can be defined as follows,

\[ \alpha_i\text{-path} = \alpha_i\text{-path} \ast x_m \quad \text{s.t. } x_{m-1} = \alpha_i \quad (5.3) \]
In this method the bigrams are clustered according to the preceding elements. For the recovery process the starting alpha is provided, the first element of the alpha path is the element recovered. The recovered element then determines the next alpha formed and thus the alpha path to be looked for the next unreferenced element. Once an element has been used the reference pointer in the alpha path is shifted and the next element has to be used extracted the reference pointer is moved to the next element in the path for future recovery referrals. Similarly all other elements can be found since the order is maintained in the alpha paths. The overhead added due to this primary alpha is very less when the length of the original stream tends to be a larger number. The $\alpha$-paths are conditional distributions with the path elements conditioned on the alpha parameter.

The simplest $\alpha$-path formation would be considering alpha as a single character from the alphabet. The total number of alpha paths would be the size of the alphabet. The scheme would be single order system and not efficient for data sources like images because of the lack of enough statistics to be build good conditional distributions. A more sophisticated and complicated alpha definition can be used to exploit some structural correlation in the image data. The alpha formed in this case will be a higher order alpha, \[ \alpha_i = f(x_{i-1}, x_{i-2}, ..., x_{i-n}) \]
The alpha formed at a given instance $t$ uses $n$ previous elements which depends on the order of the scheme. The alpha formed will model the data source, so it should be designed in such a way that it captures appropriate features of the data source. A higher order $\alpha$-path is where more than one element is considered as the preceding elements. A different version would be where a dynamic length of the preceding element can be considered. Once the $\alpha$-paths are formed, processing can be done on them before source coding can be applied on them.

5.3 Image compression using $\alpha$-paths

In case of the images we have suggested an $\alpha$-path based method which uses an order-5 alpha. A parameter is formed using previous 5 elements on which the elements that follow
are separated and conditioned. The alpha path encoding scheme is applied on the residual error image generated by the prediction method. Although the α-path method is independent of the type of prediction used and hence can be applied on any type of data source, but with the suggested alpha parameter the alpha path would prove efficient only on residual error images since the alpha is an error energy parameter. The prediction methods will indirectly affect the final compression results based on their inherent efficiency and behavior in various image types. As mentioned earlier, image data has a two dimensional correlation, hence simple raster scan would not be efficient [81,75]. We suggest a zigzag scan, which will approximately capture the two-dimensional correlation. We have also used the α-path method using raster scan but without linearizing the data. The alpha formed in the suggested method is defined as follows,

$$\alpha_i = Q\left(\sqrt{\sum_{k=0}^{n} (x_{T-k})^2}\right)$$

where

$$Q(\alpha_i) = q_j \quad L(i) \leq \alpha_i < L(i + 1)$$

$$q_j = 1, 2, ...., 9$$

$$L = [0, 2, 3, 4, 5, 8, 14, 19, 37, 50, \infty]$$

Where \(n\) is the order of the scheme and \(n\) is set to 5 in our case. The quantization function \(Q(\alpha_i)\) quantizes the alpha parameter into 9 different levels. This makes the formation of the conditional probability distributions easy and manageable to drive the arithmetic encoder. The zigzag path used is as show below,
If we consider the previous five elements of the scan for location (a) they form the 180-degree context of the current pixel location. But the disadvantage of the zigzag scan (Fig 5.2) lies in the odd elements \([x_{i-1}, x_{i-3}, x_{i-5}]\), which do not get the context advantage as the even elements \([x_i, x_{i-2}, x_{i-4}]\) (see above). Previous five elements of the location (b) do not form the 180-degree context of the location. The results produced by using this simple raster scan for error encoding without linearizing the data has proved to be the best using the alpha parameter defined above. In this case for each location the context elements of the current location are used to form the alpha parameter as mentioned above. The order in this case would be \(n = 6\). The context template used is show below,

![Context Template](image)

**Figure 5.3: Context template for raster scan based \(\alpha\)-path error encoding**

The context in the above figure captures the 180-degree positions of the current location. The error values in the context template (Fig 5.3) provide important information about the type of local characteristics at a given location in the image. In case of highly unpredictable
areas like edges and texture, the error energy is high and tends to trail until the predictor is back to predictable areas. If the error values are conditioned \( p(e|q_j) \) on the error energy parameter the higher error values tend to cluster in higher alpha parameter paths and the lower error values in lower alpha values. This inherent classification makes the lower alpha value paths highly compressible because of the highly skewed distribution of the error values. The compression results are better than the first order entropy of the error residual. The results will be provided in the results section.
Chapter 6: Results

The main objective of the thesis was to design a scheme that would achieve comparable or better compression ratios compared to the state-of-the-art lossless image compression techniques. In case of lossy image compression many parameters such as visual quality, mean square error, signal to noise ratio etc can be considered for measuring the performance of the presented technique. The compression considered in this thesis being lossless does not involve any other measures other than the final compressed size of the image and the time needed. As seen previously each of the two phases (prediction and encoding) contribute to the final compression performance. In this chapter, we evaluate the performance of our proposed techniques for each module and the variation of the performance under various parameter settings since speed and performance are important [82,83]. All the tests were done on an AMD Athlon™ processor with 2.1 GHz clock speed, and 256MB RAM. Experiments were performed in a MATLAB environment (ver 6).

6.1 Pre-Analysis Parameters

The pre-analysis procedure discussed in Chapter 4 analyses the image to determine a set of parameters that are further used by the predictors. The procedure is adaptive and determines the parameters that result in the best (or close to the best) performance of the predictor under various possible parameter values. Here we consider the performance of the EBP (Edge Based Predictor) under various parameter settings, which are usually determined automatically by the adaptive pre-analysis procedure. The threshold parameters $C_t$ (3.10), $\xi_1$ and $\xi_2$ (3.7) are calculated by analyzing the edges in the image. These parameters are free parameters and can be set to any value within the corresponding range; $0 \leq C_t \leq \infty$, $0 \leq \xi_1 \leq 1$, $0 \leq \xi_2 \leq 1$. Fixing the parameter values to some constants will save the time required for the pre-analysis procedure and thus will increase the speed of compression. Improving the compression time performance will cost a slight loss in the
Results

 compression ratio performance. The results in Table 1 show the effect of setting the parameters to fixed values and not determining them automatically.

Table 1: Compression in bits per pixel (bpp). \((\xi_1, \xi_2)\) are determined automatically and \(C_t\) varied.

<table>
<thead>
<tr>
<th>Images</th>
<th>(C_t = 100)</th>
<th>(C_t = 300)</th>
<th>(C_t = 500)</th>
<th>(C_t = 1000)</th>
<th>(C_t = \text{automatic})</th>
</tr>
</thead>
<tbody>
<tr>
<td>BABOON</td>
<td>5.901</td>
<td>5.895</td>
<td>5.895</td>
<td>5.903</td>
<td>5.903</td>
</tr>
<tr>
<td>CAMERA</td>
<td>4.198</td>
<td>4.204</td>
<td>4.217</td>
<td>4.241</td>
<td>4.19</td>
</tr>
<tr>
<td>LENA</td>
<td>4.473</td>
<td>4.473</td>
<td>4.466</td>
<td>4.466</td>
<td>4.477</td>
</tr>
<tr>
<td>NOISESQUARE</td>
<td>5.407</td>
<td>5.394</td>
<td>5.395</td>
<td>5.403</td>
<td>5.404</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>4.381</td>
<td>4.377</td>
<td>4.381</td>
<td>4.384</td>
<td>4.384</td>
</tr>
<tr>
<td>COUPLE</td>
<td>3.664</td>
<td>3.697</td>
<td>3.71</td>
<td>3.712</td>
<td>3.634</td>
</tr>
<tr>
<td>Average(bpp)</td>
<td><strong>4.671</strong></td>
<td><strong>4.673</strong></td>
<td><strong>4.677</strong></td>
<td><strong>4.685</strong></td>
<td><strong>4.665</strong></td>
</tr>
</tbody>
</table>

In particular we can observe that different images could require different values of \(C_t\) for best performance.

Table 2: \(C_t\) is determined automatically and \((\xi_1, \xi_2)\) varied.

<table>
<thead>
<tr>
<th>Images</th>
<th>((0.5,0.5))</th>
<th>((0.75,0.5))</th>
<th>((1,0.5))</th>
<th>((0.5,0.75))</th>
<th>((0.5,1))</th>
<th>(\text{automatic})</th>
</tr>
</thead>
<tbody>
<tr>
<td>BABOON</td>
<td>5.938</td>
<td>5.938</td>
<td>5.938</td>
<td>5.938</td>
<td>5.939</td>
<td>5.903</td>
</tr>
<tr>
<td>CAMERA</td>
<td>4.21</td>
<td>4.209</td>
<td>4.207</td>
<td>4.21</td>
<td>4.21</td>
<td>4.19</td>
</tr>
<tr>
<td>LENA</td>
<td>4.501</td>
<td>4.5</td>
<td>4.501</td>
<td>4.5</td>
<td>4.5</td>
<td>4.477</td>
</tr>
<tr>
<td>NOISESQUARE</td>
<td>5.5</td>
<td>5.502</td>
<td>5.502</td>
<td>5.502</td>
<td>5.503</td>
<td>5.404</td>
</tr>
<tr>
<td>Average(bpp)</td>
<td><strong>4.706</strong></td>
<td><strong>4.706</strong></td>
<td><strong>4.706</strong></td>
<td><strong>4.707</strong></td>
<td><strong>4.708</strong></td>
<td><strong>4.665</strong></td>
</tr>
</tbody>
</table>

Note that the results in Table 2 correspond to the final compression of the images in bits per pixel (bpp) using the context based encoding method. Same encoding method is used for the previous results in Table 1.
In the above shown tables one of two parameter sets is kept constant while the other is determined automatically. In Table 1, the threshold parameter $C_t$ is set to 4 different values and in all the cases the parameter pair $\xi_1$ and $\xi_2$ are determined automatically. It can be seen that the performance of the scheme at each parameter setting is dependent on the image. The performance is not consistent for any fixed value. Recall from Chapter 4, that the $C_t$ parameter is a threshold value that is used to determine whether a pixel belongs to an edge or a non-edge value. The $C_t$ value has could be adapted depending on the contrast and overall brightness of the image. For example, for the image “couple” in Table 1, the brightness and contrast of the image is low and hence the scheme does well at a lower $C_t$ value, for the image “peppers” the contrast and brightness are high so a higher value of $C_t$ results in better performance. The last column of Table 1 shows that the automatic parameter estimation results in best performance on average compared to all other parameter settings.

In Table 2, parameters $\xi_1$ and $\xi_2$ are assigned some constants and $C_t$ is determined automatically. The parameters $\xi_1$ and $\xi_2$ represent the threshold values used to determine the edge orientation based on the directional coefficients (Chapter 4 - Algorithm 1). Similar to Table 1, it can be seen that none of the fixed parameter settings have consistent performance and need to be adaptively determined. The automatic estimation of $\xi_1$ and $\xi_2$ results in the best performance as tabulated in last column of Table 2. The parameter variation experiments show that automatic parameter estimation using the pre-analysis procedure improves the performance of the predictor, resulting in better compression ratios.

### 6.2 Predictor Coefficients

As discussed in Chapter 4 the edge based predictor has two modes of operation: binary and continuous mode. In the continuous mode the prediction method is decided based on whether the current location belongs to edge or a non-edge area. In case of non-edge area a simple linear predictor is used (4.10). The locations that belong to an edge area are predicted using the predictor show in chapter 4 (4.11). The predictor coefficients are
experimentally tuned and the performance of the predictor for various values is presented below. The coefficients $\alpha_1$ and $(\alpha_2=\alpha_3)$ in (4.11 and 4.10) are set to different values and the prediction scheme is applied on all the images from the image set. The performance of the edge-based predictor under these coefficient value variations is shown in Table 3.

Table 3: Coefficients $\alpha_1$ and $\alpha_2=\alpha_3$ set to constant values.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>$\alpha_1=0.95$</th>
<th>$\alpha_1=.75$</th>
<th>$\alpha_1=.55$</th>
<th>$\alpha_1=.75$</th>
<th>$\alpha_1=.55$</th>
<th>$\alpha_1=.75$</th>
<th>$\alpha_1=.55$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images\ bpp</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BABOON</td>
<td>5.904</td>
<td>5.903</td>
<td>5.906</td>
<td>5.919</td>
<td>5.917</td>
<td>5.92</td>
<td>5.935</td>
</tr>
<tr>
<td>BALLOON</td>
<td>2.841</td>
<td>2.83</td>
<td>2.827</td>
<td>2.846</td>
<td>2.837</td>
<td>2.832</td>
<td>2.855</td>
</tr>
<tr>
<td>LENA</td>
<td>4.481</td>
<td>4.477</td>
<td>4.476</td>
<td>4.489</td>
<td>4.485</td>
<td>4.484</td>
<td>4.5</td>
</tr>
<tr>
<td>NOISE SQUARE</td>
<td>5.41</td>
<td>5.404</td>
<td>5.405</td>
<td>5.413</td>
<td>5.413</td>
<td>5.413</td>
<td>5.422</td>
</tr>
<tr>
<td>Average(bpp)</td>
<td><strong>4.378</strong></td>
<td><strong>4.372</strong></td>
<td><strong>4.373</strong></td>
<td><strong>4.386</strong></td>
<td><strong>4.381</strong></td>
<td><strong>4.380</strong></td>
<td><strong>4.395</strong></td>
</tr>
</tbody>
</table>

It can be seen that the best results are achieved for the parameter setting of $\alpha_1=0.95$ and $\alpha_2=\alpha_3=0.75$ respectively. The $(C, \xi_1$ and $\xi_2)$ parameters in this case are found adaptively using the automatic process. The coefficient values used in the final compression scheme are approximately same as the ones that have the best results in Table 3.

### 6.3 Predictor Performance

To achieve better performance we use the adaptive switching between predictors for edge and non-edge area. The pre-analysis parameters that determine the performance of the predictor are chosen automatically in the final prediction scheme. The performance of the predictors before applying encoding techniques can be measured by the 1st order entropy of the error residue (see Table 4). The predictor performance forms the basis for better overall
Results

compression. In this subsection we report on different tests performed on the predictors to evaluate their comparative performance with other state-of-the-art predictors.

Table 4: 1\textsuperscript{st} order entropy for various state-of-the-art predictors

<table>
<thead>
<tr>
<th>Predictor</th>
<th>TNP</th>
<th>MED</th>
<th>GAP</th>
<th>EBP</th>
<th>EDP</th>
<th>LS-EBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAMERA</td>
<td>4.85</td>
<td>4.74</td>
<td>4.70</td>
<td>4.63</td>
<td>4.68</td>
<td>4.61</td>
</tr>
<tr>
<td>COUPLE</td>
<td>4.37</td>
<td>3.99</td>
<td>4.05</td>
<td>3.96</td>
<td>3.95</td>
<td>4.00</td>
</tr>
<tr>
<td>NOISESQUARE</td>
<td>5.59</td>
<td>5.73</td>
<td>5.60</td>
<td>5.38</td>
<td>5.43</td>
<td>5.42</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>4.85</td>
<td>4.94</td>
<td>4.75</td>
<td>4.51</td>
<td>4.45</td>
<td>4.45</td>
</tr>
<tr>
<td>LENA</td>
<td>4.91</td>
<td>4.89</td>
<td>4.73</td>
<td>4.64</td>
<td>4.52</td>
<td>4.52</td>
</tr>
<tr>
<td>BARB2</td>
<td>5.45</td>
<td>5.18</td>
<td>5.05</td>
<td>4.91</td>
<td>4.75</td>
<td>4.74</td>
</tr>
<tr>
<td>Average(bpp)</td>
<td>5.00</td>
<td>4.91</td>
<td>4.81</td>
<td>4.67</td>
<td>4.63</td>
<td>4.62</td>
</tr>
</tbody>
</table>

It can be seen that predictors like TNP (Two Nearest-point Prediction) [84] and MED (Median Edge Detector) [22] have relatively higher entropies compared to other predictors. But these predictors are very simple and require very few operations for making predictions. TNP is order-two predictor which uses its two nearest neighboring pixels whereas MED uses the nearest three pixels. GAP [25] that has higher complexity and is based on the gradient at the current pixel location produces better entropy compared to the simpler predictors. The suggested predictor EBP that uses a pre-analysis procedure to improve the adaptivity out performs GAP with a slightly higher complexity. EDP and LS-EBP that are based on least squares outperform all other predictors because of their ability to optimize the predictor coefficients at edge locations. LS-EBP, which uses the pre-analysis parameters for making the predictor adaptive, gains on performance compared to EDP with a slightly higher complexity. The complexity of the predictors increase from first row to the last row.

A higher order entropy test was also performed to check for the performance of the predictors when more than one symbol is considered at a time. The higher order implementations might be impractical, but they give an idea of the long-term performance of the prediction schemes. The test was performed on three images and the entropies up till 3\textsuperscript{rd} order are produced.
Table 5: Higher order entropies for cameraman image

<table>
<thead>
<tr>
<th></th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CAMERA</strong></td>
<td></td>
</tr>
<tr>
<td>TNP</td>
<td>4.82</td>
</tr>
<tr>
<td>MED</td>
<td>4.96</td>
</tr>
<tr>
<td>GAP</td>
<td>4.69</td>
</tr>
<tr>
<td>EBP</td>
<td>4.62</td>
</tr>
<tr>
<td>EDP*</td>
<td>4.65</td>
</tr>
<tr>
<td>LS-EBP</td>
<td>4.61</td>
</tr>
</tbody>
</table>

Table 6: Higher order entropies for noisesquare image

<table>
<thead>
<tr>
<th></th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOISESQUARE</strong></td>
<td></td>
</tr>
<tr>
<td>TNP</td>
<td>5.58</td>
</tr>
<tr>
<td>MED</td>
<td>5.71</td>
</tr>
<tr>
<td>GAP</td>
<td>5.6</td>
</tr>
<tr>
<td>EBP</td>
<td>5.37</td>
</tr>
<tr>
<td>EDP*</td>
<td>5.43</td>
</tr>
<tr>
<td>LS-EBP</td>
<td>5.42</td>
</tr>
</tbody>
</table>

Table 7: Higher order entropies for peppers image

<table>
<thead>
<tr>
<th></th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PEPPERS</strong></td>
<td></td>
</tr>
<tr>
<td>TNP</td>
<td>4.84</td>
</tr>
<tr>
<td>MED</td>
<td>4.93</td>
</tr>
<tr>
<td>GAP</td>
<td>4.74</td>
</tr>
<tr>
<td>EBP</td>
<td>4.51</td>
</tr>
<tr>
<td>EDP*</td>
<td>4.45</td>
</tr>
<tr>
<td>LS-EBP</td>
<td>4.45</td>
</tr>
</tbody>
</table>

*EDP higher order results were produced using the error residue generated using the program available on [www.csee.wvu.edu/~xin](http://www.csee.wvu.edu/~xin) as of (04/06/2005)*

It can be seen that EBP performs better at higher orders so if higher encoding schemes are applied the EBP predictor would produce best results in the above category of predictors.

Fig 6.1 & 6.2 show the residue images generated by various predictors with their 1st order entropies.
Figure 6.1: Residue images for CAMERAMAN image after different predictions (a) TNP (4.82 bpp), (b) MED (4.96 bpp), (c) GAP (4.69 bpp), (d) EBP (4.62 bpp), (e) EDP (4.65 bpp) (f) LS-EBP (4.611 bpp)
Figure 6.2: Residue images of PEPPERS image after different predictions (a) TNP (4.84 bpp), (b) MED (4.93 bpp), (c) GAP (4.74 bpp), (d) EBP (4.51 bpp), (e) EDP (4.45 bpp) (f) LS-EBP (4.45 bpp)
It can be seen in both cases the prediction improves at the edges as the predictor complexity increases. LS-EBP predictor tends to perform better at the edges compared to all other predictors.

### 6.4 Results using Context Based Encoding

We consider the final compression results using the predictors proposed in (Chapter 4 and Chapter 5). The algorithm used was based on EBP and LS-EBP in association with the automatic parameter estimation using pre-analysis and context based encoding method. This algorithm was applied on all the images from the set and the final compression results are represented in bits per pixel (bpp). The results are compared with the state-of-the-art compression schemes in the genre of compression algorithms with the same space and time requirements. Table 4 shows the comparative results.

Table 8: Comparative results for various techniques, including the proposed method

<table>
<thead>
<tr>
<th></th>
<th>CALIC-a</th>
<th>CALIC-h</th>
<th>JPEG-LS</th>
<th>JPEG-2K</th>
<th>LS-EBP</th>
<th>EBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BABOON</td>
<td>5.875</td>
<td>6.031</td>
<td>6.037</td>
<td>6.107</td>
<td>5.81</td>
<td>5.903</td>
</tr>
<tr>
<td>BALLOON</td>
<td>2.825</td>
<td>2.959</td>
<td>2.904</td>
<td>3.031</td>
<td>2.815</td>
<td>2.83</td>
</tr>
<tr>
<td>BARB</td>
<td>4.413</td>
<td>4.525</td>
<td>4.691</td>
<td>4.6</td>
<td>4.074</td>
<td>4.422</td>
</tr>
<tr>
<td>LENNAGREY</td>
<td>4.102</td>
<td>4.174</td>
<td>4.238</td>
<td>4.303</td>
<td>4.03</td>
<td>4.101</td>
</tr>
<tr>
<td>NOISSE SQUARE</td>
<td>5.443</td>
<td>5.788</td>
<td>5.683</td>
<td>5.634</td>
<td>5.464</td>
<td>5.404</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>4.421</td>
<td>4.55</td>
<td>4.513</td>
<td>4.629</td>
<td>4.35</td>
<td>4.384</td>
</tr>
</tbody>
</table>

The Table 4 shows that the results using the EBP are slightly better than those from CALIC-a, (the arithmetic coding version of CALIC [25]). The LS-EBP based method has the best compression performance compared to all other schemes, but as mentioned earlier, at the cost of higher compression time. The compression time requirement for LS-EBP methods is higher but relatively very less compared to high performance methods like...
Results

(MRP [84], TMW [34], GlicBawls [35]). The results were also compared to the available set of results produced by the EDP [62] method, see Table 5.

Table 9: Comparing the proposed methods with EDP.

<table>
<thead>
<tr>
<th></th>
<th>EDP</th>
<th>LS-EBP</th>
<th>EBP</th>
</tr>
</thead>
<tbody>
<tr>
<td>BABOON</td>
<td>5.81</td>
<td>5.81</td>
<td>5.9</td>
</tr>
<tr>
<td>BARB</td>
<td>4.11</td>
<td>4.07</td>
<td>4.42</td>
</tr>
<tr>
<td>LENA</td>
<td>4.4</td>
<td>4.4</td>
<td>4.47</td>
</tr>
<tr>
<td>LENNAGREY</td>
<td>4.02</td>
<td>4.03</td>
<td>4.1</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>4.35</td>
<td>4.35</td>
<td>4.38</td>
</tr>
<tr>
<td>GOLDHILL</td>
<td>4.39</td>
<td>4.39</td>
<td>4.42</td>
</tr>
<tr>
<td>BARB2</td>
<td>4.52</td>
<td>4.42</td>
<td>4.54</td>
</tr>
<tr>
<td>Average(bpp)</td>
<td>4.514</td>
<td>4.496</td>
<td>4.604</td>
</tr>
</tbody>
</table>

It can be seen that the proposed LS-EBP with context based encoding performs slightly better than EDP with almost same time and space complexity. EBP has a slightly inferior performance but has a lower complexity and thus higher compression speed and simpler implementation requirements.

Table 10: Execution times for three predictors with MATLAB implementation

<table>
<thead>
<tr>
<th></th>
<th>Execution time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size</td>
</tr>
<tr>
<td>CAMER A</td>
<td>256x256</td>
</tr>
<tr>
<td>COUPLE</td>
<td>256x256</td>
</tr>
<tr>
<td>NOIS E SQUARE</td>
<td>256x256</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>512x512</td>
</tr>
<tr>
<td>LENA</td>
<td>512x512</td>
</tr>
<tr>
<td>BARB2</td>
<td>576x720</td>
</tr>
</tbody>
</table>

To get a general idea about the time complexity we have compared the execution times of three predictors with implementations in MATLAB environment. It can be seen that EBP has very lower time complexity compared to LS based EDP and LS-EBP. The executions times can be made comparative to other algorithms by optimum implementation of the predictions schemes and in C environment.
Other methods that use transform based coding such as the “Lossless Image Compression With Projection-Based and Adaptive Reversible Integer Wavelet Transforms” produce relative less compression ratios are shown below to make a general comparison.

Table 11: Prediction results for Integer wavelet based transforms.

<table>
<thead>
<tr>
<th>Image</th>
<th>S+P (bpp)</th>
<th>S+Proj (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LENA</td>
<td>4.319</td>
<td>4.313</td>
</tr>
<tr>
<td>BALLOON</td>
<td>3.064</td>
<td>3.064</td>
</tr>
<tr>
<td>BARB2</td>
<td>5.007</td>
<td>5.007</td>
</tr>
</tbody>
</table>

The S transform is the integer version of the Haar Wavelet Transform. The P stands for the extra prediction used with Proj is used for projection-based prediction.

### 6.5 Encoding Methods

As discussed in Chapter 3, the possibility of BWT encoding for images and methods inspired from BWT were also studied in this thesis. We consider the BWT inspired $\alpha$-path encoding methods using different scan types on the LS-EBP prediction residuals. The scan paths used for compression were simple raster scan with context template (Raster-1) described in Chapter 3 (Fig 3.2), raster scan linearized stream (Raster-2) and finally the zigzag linearized scan mentioned in Chapter 3 (Fig 5.3). The advantage of the later two scans is that they don’t need context buffer memory as in template based raster scan. BWT as black box was also tested on the prediction residuals as well on the images directly. The results of the various experiments are show in the Table 6.
Table 12: Results using different encoding schemes

<table>
<thead>
<tr>
<th>Images</th>
<th>Raster-1</th>
<th>Raster-2</th>
<th>zigzag</th>
<th>BWT residual</th>
<th>BWT direct</th>
</tr>
</thead>
<tbody>
<tr>
<td>BABOON</td>
<td>5.844</td>
<td>5.914</td>
<td>5.83</td>
<td>5.954</td>
<td>6.29</td>
</tr>
<tr>
<td>BALLOON</td>
<td>2.867</td>
<td>2.869</td>
<td>2.866</td>
<td>2.861</td>
<td>3.867</td>
</tr>
<tr>
<td>BARB</td>
<td>4.169</td>
<td>4.135</td>
<td>4.165</td>
<td>4.155</td>
<td>5.827</td>
</tr>
<tr>
<td>CAMERA</td>
<td>4.212</td>
<td>4.351</td>
<td>4.261</td>
<td>4.381</td>
<td>5.213</td>
</tr>
<tr>
<td>LENA</td>
<td>4.4</td>
<td>4.447</td>
<td>4.408</td>
<td>4.475</td>
<td>5.37</td>
</tr>
<tr>
<td>LENNAGREY</td>
<td>4.028</td>
<td>4.08</td>
<td>4.034</td>
<td>4.112</td>
<td>5.094</td>
</tr>
<tr>
<td>NOISESQUARE</td>
<td>5.43</td>
<td>5.476</td>
<td>5.453</td>
<td>5.453</td>
<td>5.183</td>
</tr>
<tr>
<td>PEPPERS</td>
<td>4.333</td>
<td>4.414</td>
<td>4.37</td>
<td>4.422</td>
<td>5.122</td>
</tr>
<tr>
<td>COUPLE</td>
<td>3.691</td>
<td>3.837</td>
<td>3.753</td>
<td>3.818</td>
<td>4.791</td>
</tr>
<tr>
<td>Avgbpp</td>
<td><strong>4.330</strong></td>
<td><strong>4.391</strong></td>
<td><strong>4.349</strong></td>
<td><strong>4.403</strong></td>
<td><strong>5.195</strong></td>
</tr>
</tbody>
</table>

BWT when applied directly on the image does not produce appreciable results since the modeling done by BWT lacks the advantages of the dedicated methods for image compression. The scheme used in this case was linear BWT followed by MTF and an arithmetic encoder. In case of BWT residual, the BWT was applied on the prediction residuals of LS-EBP scheme (without MTF). The results are better than BWT direct since the prediction removes a significant amount of correlation of the image pixels. The results are further improved by modifying the scan paths and the error modeling. It can be seen that the Raster-1 scan with error modeling methods gives the best results, and is far better than using the BWT directly.
Chapter 7: Conclusion

Image compression is one the most studied areas in compression theory. A large number of real world applications need image processing and thus efficient image data handling viz storage and communication. Lossless image compression is slowly attracting attention due to the applications that require higher quality requirements. Image compression by itself is new compared to text data compression. Though introduced in the late 70’s, the image compression algorithms are slowly reaching the limits of compression. The first order entropy has been surpassed long back using data decorrelation and image modeling. A wide spectrum of compression algorithms have been suggested varying from those with high time and space complexity to those with near optimal compression performance. Some methods provide a good balance between both the aspects and have thus gained more popularity. Algorithm selection for compression standards is usually based on their position on the time and complexity curves.

This thesis focused on a lossless compression technique that offers a good tradeoff between compression time and compression ratio. The research started with the investigation of whether BWT, a popular text compression method could be used for image compression. Then looking at the shortcomings of the text compression scheme for image modeling methods inspired by the BWT, but designed keeping image compression in mind were suggested and studied for performance. The two major methods suggested were F2BWT and $\alpha$-path based encoding method. Both the methods work independent of the prediction method used and can be treated as separate modules. The performance of F2BWT won’t change if the data feed is not prediction residuals. In case of the suggested higher order $\alpha$-path based methods, the performance would change if the data feed is not prediction residual since it is based on the energy parameter which is good only in case of image residuals. Due to implementation issues, the $\alpha$-path methods were given more attention and their performance evaluation is done in detail. The performance of the compression technique depends heavily on the prediction technique used.
The key concept behind the success of a good predictor is its ability to detect edges and make predictions accordingly. Various predictors with a wide range of complexity have been proposed some of which require explicit edge analysis whereas others have implicit edge analyzers. Edge detection makes it possible for the predictors to make predictions along the edge orientation where there is less variation and avoid going across edges, which has unpredictable variation. The above-mentioned encoding methods were tested on predictor residuals produced by MED and DPCM. The performance of the predictors motivated further research into designing a new prediction method. The Edge Based Predictor (EBP) proposed in this thesis is designed to make predictions based on the gradient analysis and the class of the current pixel location (edge or non-edge). To determine the class of the current location, a pre-analysis stage is proposed which results in a highly adaptive predictor. A set of parameters is determined at the pre-analysis stage that is further used by the predictor to decide the type of processing done and the prediction mode. The binary mode switching capability makes this method suitable for compressing synthetic and document images unlike most of the continuous tone image compression methods. The context-based error modeling and encoding further enhances the compression performance. The compression results produced by the EBP with context based error encoding are superior to results with other methods in the same complexity genre (CALIC-a, CALIC-h, JPEG-LS, and JPEG-2000).

Performance evaluation of the EBP based method showed that the compression ratios could be further improved by designing a pixel-by-pixel optimized predictor. This lead to the investigation of a least-squares based approach. Pure LS based methods are highly complex and require a higher time and space complexity. To achieve a better tradeoff between time and compaction ability, an intelligent way of applying LS has to be designed. The LS-EBP predictor suggested in this thesis in association with the pre-analysis procedure not only reduces the time of execution, also significantly improved the compression performance.

Thus the overall objective of the thesis research to design a good lossless compression scheme was successful. The compression results are appreciable and competitive with other state-of-the-art compression schemes. The BWT based methods have opened a new avenue
for different and efficient reorganization of the input data. Further studies may be required
to adequately exploit this new possibility for efficient compression of images.
References:


