AUTOMATIC COMPRESSION FOR IMAGE SETS USING A GRAPH THEORETICAL FRAMEWORK

BARRY GERGEL
Bachelor of Science, University of Lethbridge, 2005

A Thesis
Submitted to the School of Graduate Studies
of the University of Lethbridge
in Partial Fulfillment of the
Requirements for the Degree

MASTER OF SCIENCE

Department of Mathematics and Computer Science
University of Lethbridge
LETHBRIDGE, ALBERTA, CANADA

AUTOMATIC COMPRESSION FOR IMAGE SETS USING A GRAPH THEORETICAL FRAMEWORK

BARRY GERGEL

Approved:

Signature *Date*

Supervisor: Dr. Howard Cheng

Committee Member: Dr. Stephen Wismath

Committee Member: Dr. Craig Coburn

External Examiner: Dr. Herb Yang

Chair, Thesis Examination Committee:
Dr. Scott Allen

I dedicate this thesis to my sons Zakery, Kaleb, and Noah.

Abstract

A new automatic compression scheme that adapts to any image set is presented in this thesis. The proposed scheme requires no *a priori* knowledge on the properties of the image set. This scheme is obtained using a unified graph-theoretical framework that allows for compression strategies to be compared both theoretically and experimentally. This strategy achieves optimal lossless compression by computing a minimum spanning tree of a graph constructed from the image set. For lossy compression, this scheme is near-optimal and a performance guarantee relative to the optimal one is provided. Experimental results demonstrate that this compression strategy compares favorably to the previously proposed strategies, with improvements up to 7% in the case of lossless compression and 72% in the case of lossy compression. This thesis also shows that the choice of underlying compression algorithm is important for compressing image sets using the proposed scheme.

Acknowledgments

I wish to thank my supervisor Dr. Howard Cheng. It would not have been possible to complete this work without his encouragement, patience, suggestions, generosity, support, and friendship. I would like to thank my committee members Dr. Stephen Wismath and Dr. Craig Coburn for their advice and willingness to help during the past two years. From the University of Alberta, I wish to thank Dr. Xiaobo Li for his suggestions and support, and Dr. Herb Yang for serving as the external examiner.

I would like to acknowledge the faculty and staff of the Department of Mathematics and Computer Science at the University of Lethbridge. Their encouragement, advice, and kindness have made my stay at the university a great experience. Furthermore, I would like to express thanks to my colleagues and friends Ethan Kim, Zac Friggstad, David Lenz, Ben Burnett, Dr. Jackie Rice, Sebastian Hanlon, Neil Anderson, Terence Kaplan, Mike Wiebe, Dallas Thomas, Sean Legge, Nicole Wilson, and Trent Takeyasu for their friendship and encouragement, and for helping me keep my sanity as I completed this thesis.

Without the support and generosity of my family, returning to school to complete my education would have been extremely difficult. I would like to thank my parents, my grandparents, and my in-laws. Most importantly, I would like to thank my wife, Lorelei. It would not have been possible to complete this work without her patience, love, support, and faith. She is my steadying stone and I would be nowhere without her. I also want to thank my sons Zakery, Kaleb, and Noah for their smiles, laughter, and hugs.

This work was supported by the Hope and Keith Ferguson Memorial Scholarship and I would like to thank the University of Lethbridge for making such scholarships available. I gratefully acknowledge Dr. Alan Tong of Lacombe Research Centre, Agriculture and Agri-Food Canada for making the Ultrasound images and data available to this study.

Contents

Approval/Signature Page ii

Dedication iii

Abstract iv

Acknowledgments v

Table of Contents vi

List of Tables viii

List of Figures ix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Automatic Image Set Compression . . . 2
- 1.3 Thesis Outline . . . 3

2 Background 4
- 2.1 Digital Images . . . 4
  - 2.1.1 Difference Image . . . 5
  - 2.1.2 Image Similarity . . . 7
- 2.2 Graph Theory . . . 7
- 2.3 Image Compression . . . 8
  - 2.3.1 Entropy . . . 10
  - 2.3.2 Wavelet Compression . . . 11
  - 2.3.3 Wavelet Packet Compression . . . 14
- 2.4 Summary . . . 15

3 Analysis of Related Works 16
- 3.1 The Centroid Methods . . . 16
- 3.2 The Template Method . . . 17
- 3.3 The Min-Max Methods . . . 18
- 3.4 The Minimum Spanning Tree Methods . . . 19
- 3.5 The Quadtree Method . . . 19
- 3.6 The 3D Image Set Method . . . 22
- 3.7 The Wavelet Method . . . 23
- 3.8 The Principal Components Method . . . 23
- 3.9 Summary . . . 24

4 Graph Theoretical Framework for Image Set Compression **26**
4.1 Graph Construction . . . 26
4.2 Compression and Decompression . . . 28
4.2.1 Lossless Case . . . 28
4.2.2 Errors Introduced by Lossy Compression . . . 29
4.3 The Cost of a Compression Strategy . . . 31
4.4 Optimality of Performance . . . 32
4.5 Compression Strategies . . . 36
4.5.1 Traditional Strategy . . . 36
4.5.2 Centroid Strategy . . . 36
4.5.3 Minimum Spanning Tree Strategy (MST) . . . 38
4.5.4 MST with Average Image Strategy (MST_a) . . . 38
4.6 Summary . . . 39

5 Experimental Results **40**
5.1 Test Image Sets . . . 40
5.1.1 Galway . . . 40
5.1.2 Pig . . . 42
5.1.3 Joe . . . 42
5.1.4 GOES . . . 46
5.1.5 Combination . . . 46
5.2 Implementation Details . . . 46
5.3 Lossless Compression Results . . . 47
5.3.1 Galway . . . 49
5.3.2 Pig . . . 49
5.3.3 Joe . . . 49
5.3.4 GOES . . . 50
5.3.5 Combination . . . 50
5.4 Estimating Lossless Performance with Entropy . . . 51
5.5 Lossy Compression Results . . . 54
5.5.1 JPEG2000 Results . . . 58
5.5.2 Wavelet Packets . . . 58
5.5.3 9-bit Lossy Compression . . . 65
5.6 Summary . . . 68

6 Conclusion **70**
6.1 Limitations . . . 71
6.2 Future Directions . . . 71

Bibliography **74**

List of Tables

5.1 Lossless compression results (in bytes) using JPEG2000. 48
5.2 Lossless compression results (in bytes) using JPEG-LS. 48
5.3 Types of tree edges chosen by MST_a using JPEG2000 for edge weights. . 48
5.4 Galway entropy results. 51
5.5 Pig entropy results. 51
5.6 Joe entropy results. 52
5.7 GOES entropy results. 52
5.8 Combination entropy results. 53
5.9 Types of tree edges chosen by the MST_a strategy using entropy as edge weights. 53
5.10 Bitrate achieved by the traditional and MST_a strategies using JPEG2000 compression at various average RMSE values. 54
5.11 Bitrate achieved by the traditional and MST strategies using JPEG2000 compression at various average RMSE values. 55
5.12 Bitrate achieved by the traditional and MST_a strategies using wavelet packet compression at various average RMSE values. 61

List of Figures

2.1 A comparison between a pair of original images and a computed difference image from a collection of ultrasound images. 6
2.2 Source encoder and decoder for image compression. 9
2.3 Visual representation of the decomposition of a one-dimensional input source using a wavelet transformation using three passes. 12
2.4 Two-dimensional wavelet transformation decomposition structures. . . . 14

3.1 A quadtree example for a small binary image. 20
3.2 A second binary image and its quadtree. 21

4.1 Decompressing image I_7. 28
4.2 A path representing how two images from a set may be compressed. . . . 30
4.3 Lossy compression error propagation in a spanning tree. 31
4.4 Using an edge in T_G as an edge of a spanning tree in G'. 33
4.5 Traditional strategy. 36
4.6 Centroid strategy with a single cluster. 37
4.7 Centroid strategy with multiple clusters. 37
4.8 Example of local edge choice in the MST_a strategy. 39

5.1 Typical images from the Galway set. 41
5.2 Typical images from the Pig set. 43
5.3 Typical images from the Joe set. 44
5.4 Typical images from the GOES set. 45
5.5 A portion of the computed MST for the Joe image set. 50
5.6 Lossy compression performance using JPEG2000 on the Galway set. . . . 55
5.7 Lossy compression performance using JPEG2000 on the Pig set. 56
5.8 Lossy compression performance using JPEG2000 on the Joe set. 56
5.9 Lossy compression performance using JPEG2000 on the GOES set. . . . 57
5.10 Lossy compression performance using JPEG2000 on the Combination set. 57
5.11 Histograms of a regular image and a difference image from the Pig set. . . 59
5.12 The best basis chosen for a difference image from the GOES set. 60
5.13 Lossy compression performance using wavelet packets on the Galway set. 61
5.14 Lossy compression performance using wavelet packets on the Pig set. . . 62
5.15 Lossy compression performance using wavelet packets on the Joe set. . . 62
5.16 Lossy compression performance using wavelet packets on the GOES set. . 63
5.17 Lossy compression performance using wavelet packets on the Combination set. 63
5.18 Lossy compression performance comparing JPEG2000 and wavelet packets on the Galway set. 65
5.19 Lossy compression performance comparing JPEG2000 and wavelet packets on the Pig set. 66

5.20 Lossy compression performance comparing JPEG2000 and wavelet packets on the Joe set. 66
5.21 Lossy compression performance comparing JPEG2000 and wavelet packets on the GOES set. 67
5.22 Lossy compression performance comparing JPEG2000 and wavelet packets on the combination set. 67

Chapter 1

Introduction

The efficient storage of digital images is vital to meeting the demands of modern computing. The use and volume of digital images has seen explosive growth in the past two decades as digital imaging equipment continues to improve and replace traditional imaging methods. For example, Nagata and Tanaka in 2003 predicted that Japan alone would generate three petabytes[1] of digital medical images annually [36]. Unless these images can be stored efficiently, it is impractical to retain such large volumes of image data.

There are many applications that involve the storage of a large number of images. A personal photo-album is an example of a typical image database. Another example would be large database systems containing medical or satellite images. Videos and webcam images can be interpreted as collections of images whose individual frames are time indexed. Depending on the application, the images in a collection may or may not be similar to each other, and the relationship among images may or may not be known. Algorithms for one type of images may not work well for others.

1.1 Motivation

In this thesis, the problem of compressing sets of images is examined. The goal of data compression is to represent data succinctly by removing redundancies found in data. Traditionally, image compression has focused on compressing images individually by taking advantage of coding, inter-pixel, and psychovisual redundancies *within* the image [20]. In the related area of video compression, algorithms take advantage of redundancies existing among consecutive frames and those contained with each frame. In a large collection of

[1] A petabyte is 10^{15} bytes.

images, there may be other similarities among the images in the collection. It is unclear what types of redundancies are useful for compression, how to detect them, and how to remove them. A compression algorithm that takes advantage of these inter-image similarities can improve the overall compression performance of the image set. Strategies to automatically discover these redundancies and improve image set compression have received little attention.

A number of image compression strategies have been presented to take advantage of different types of inter-image redundancies [2, 4, 5, 26, 27, 28, 32, 35, 38, 39, 53, 54, 55]. While these attempts perform well on image sets with certain types of inter-image relationships, it is not clear which strategy is best for a particular image set *a priori*. In particular, many of these strategies work well only when images in the image set are very similar to one another. In other words, there is a high amount of redundancies among images. Furthermore, many of these methods have only been studied experimentally. There is no method to analyze if there exists an optimal compression strategy for image set compression. It is also not understood how close the performance of existing algorithms are to that of the optimal strategy, if it exists. This thesis proposes an automatic compression scheme that adapts to any image set, and analyzes its performance relative to the optimal strategy both theoretically and experimentally.

1.2 Automatic Image Set Compression

The proposed scheme is based on a unified framework that allows the comparison of all compression strategies that deal with inter-image redundancies between pairs of images. Regardless of the properties of the image set, the scheme automatically selects the optimal strategy for lossless compression or a near-optimal strategy for lossy compression. The framework uses a graph to represent inter-image relationships within an image set. The

minimum spanning tree of this graph represents the optimal compression method in the lossless case. In the lossy case, the errors introduced by lossy image compression perturb the graph. Therefore, the compression strategy obtained by the framework may not be optimal. However, theoretical analysis shows that the performance of the proposed scheme is relatively close to the optimal scheme.

The performance of the scheme is analyzed theoretically and demonstrated through experiments on several image sets. In the lossless case, the scheme shows an improvement up to 7% better than coding images individually with JPEG2000 [1, 9]. For the lossy case, a 72% improvement is obtained. Part of this thesis work has been published in [17, 18]. The thesis also demonstrates that the underlying image compression algorithm for the coding of difference images has a significant effect. Wavelet packets [8, 33] are more suitable for these types of compression strategies than JPEG2000. The framework is also effective for analyzing and evaluating other set compression strategies as it allows for quantitative comparison of a strategy to the optimal and near optimal scheme.

1.3 Thesis Outline

The thesis is organized as follows. In Chapter 2, the background concepts and terminology common to image compression are defined. Then, Chapter 3 contains a summary of related works by other researchers. The proposed automatic compression scheme for image set compression is presented in Chapter 4. The experimental results for lossless and lossy compression are covered in Chapter 5. Finally, the concluding remarks follow in Chapter 6.

Chapter 2

Background

This chapter contains a brief introduction to the concepts and terminology used throughout this thesis. The basics of digital images are reviewed with attention given to the computation of difference images and a method to measure the similarity between a pair of images. A summary of the graph theory concepts utilized in this thesis follows. An overview of image compression including wavelet and wavelet packet compression concludes the chapter. More detailed information can be found in [10, 12, 20, 24, 51].

2.1 Digital Images

An image can be mathematically defined as a two-dimensional function $f(x,y)$ where x and y are spatial coordinates. Given an image of dimensions $M \times N$, each pair of coordinates (x,y) defines an element in the image called a *pixel*. The amplitude of f for any pixel is the *intensity* of the image at that point. If the coordinate and amplitude values are all finite and discrete, then the image is called a *digital image*. The term *image* will be used to represent a digital image for the remainder of this thesis.

A common method to represent an image is to use an $M \times N$ matrix of the form

$$f = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{2.1}$$

where each element is a pixel. The intensity values for an image are in the range of

$[0, L-1]$ where L is a positive integer. Typically, $L = 2^k$ for some $k \geq 0$ for *gray scale* (or *monotone*) images and the intensity values are referred to as the *gray levels* of the image. Such an image is called a k-bit image. Therefore, the number of bits required to store an uncompressed k-bit image is

$$b \;=\; M \times N \times k \tag{2.2}$$

where M is the number of rows and N is the number of columns. For example, an uncompressed 640×480 pixel 8-bit gray scale image would require 2,457,600 bits of storage space.

2.1.1 Difference Image

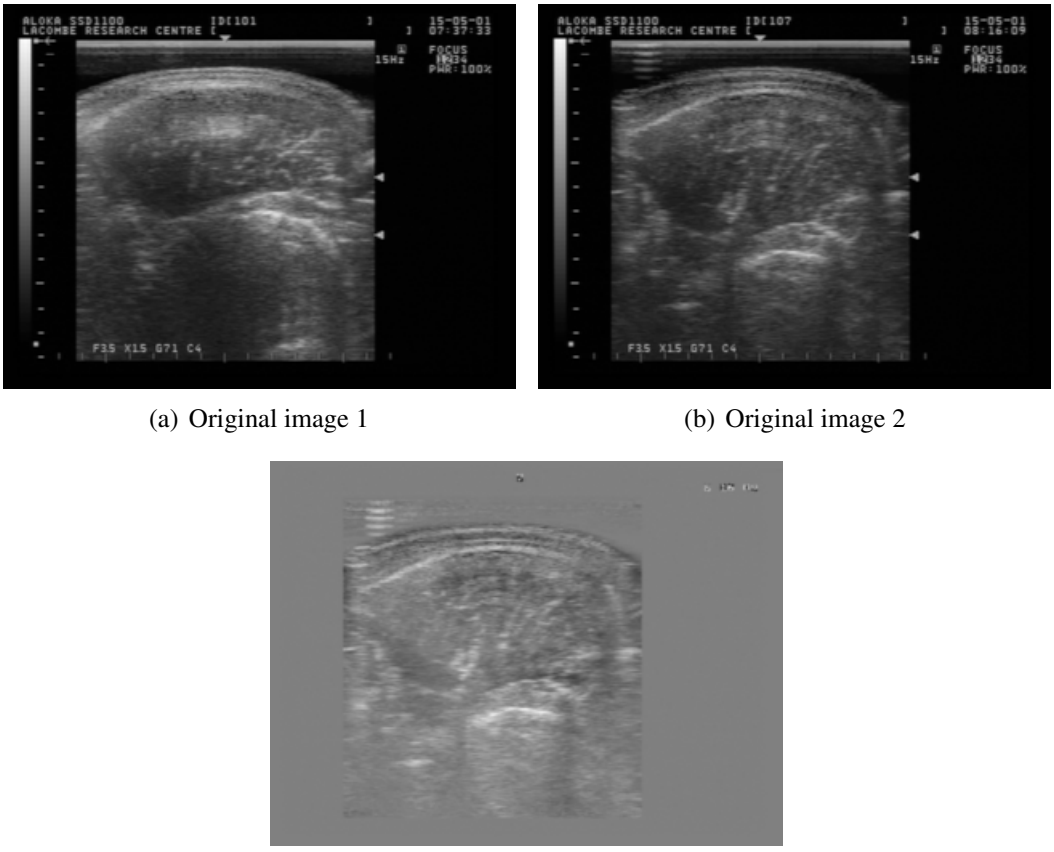
Given two k-bit gray scale images $f_1(x,y)$ and $f_2(x,y)$, the difference between the two images can be expressed as

$$d(x,y) \;=\; f_1(x,y) \;-\; f_2(x,y). \tag{2.3}$$

The image d contains the difference between all pairs of corresponding pixels in f_1 and f_2. The intensities of d have a range of $[-(2^k-1), (2^k-1)]$ so that the uncompressed difference image requires $k+1$ bits for each pixel. Figure 2.1 displays an original image and a difference image computed between two images from a collection of 8-bit gray scale ultrasound images.[1]

A common method to measure the difference between pairs of images is the root-

[1] The values in the difference image are shifted by 256 which results in a range of $[0, 511]$ for the gray levels because pixel values are stored as unsigned integers. Note that a value of zero is the color black and a value of 511 is the color white.



(a) Original image 1

(b) Original image 2

(c) Difference image

Figure 2.1: A comparison between a pair of original images and a computed difference image from a collection of ultrasound images.

mean-squared-error (RMSE) [20]

$$RMSE(f_1, f_2) = \left[\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} [f_1(x,y) - f_2(x,y)]^2 \right]^{\frac{1}{2}}. \tag{2.4}$$

A small RMSE indicates that the pair of images are very similar, while a large RMSE indicates the images are significantly different from one another.

2.1.2 Image Similarity

The definition of *similarity* for images is very subjective because its meaning may change depending on the requirements of an application. In this thesis, similarity between a pair of images is defined by examining their difference image. Specifically, three methods are used to examine difference images. They are RMSE, entropy (Section 2.3.1), and the number of bits required to store the difference image using a particular compression algorithm. These measures are chosen because they are related to the performance of compression algorithms. Furthermore, the chosen measures do not depend on the order in which the difference images is computed. That is, the measures are the same whether the difference $f_1 - f_2$ or $f_2 - f_1$ is used. Intuitively, a small value for the difference measure implies that the two images are very similar. Conversely, a large value suggests that the images are quite different.

2.2 Graph Theory

A *graph* G is a pair of sets (V, E) such that V is a finite set and $E \subseteq V \times V$ [12]. The elements of V represent the vertex set, and the elements of E represent the edge set. Two vertices $u, v \in V$ are considered connected if the edge $(u, v) \in E$. A graph is *undirected*

when $(u,v) \in E$ if and only if $(v,u) \in E$. The edges in a *weighted* graph have a weight $w(u,v)$ assigned to each edge $(u,v) \in E$. If all pairs of distinct vertices are connected with edges, the graph is called a *complete* graph.

Let $V' \subseteq V$ be a subset of vertices and $E' \subseteq E$ be a subset of edges. Then, the graph $G' = (V', E')$ is a *subgraph* of G such that $(u,v) \in E'$ only if $u, v \in V'$. A *path* from u to v is a sequence of edges $p_1, p_2, \ldots, p_k$ (for some $k > 0$) such that $p_i = (u_i, u_{i+1}) \in E$ (for $1 \leq i < k$) with $u_1 = u$ and $u_k = v$. A *cycle* is a path where $k \geq 3$ and $u_k = u$. A *spanning tree* is a subgraph (V, E') of G with the following properties:

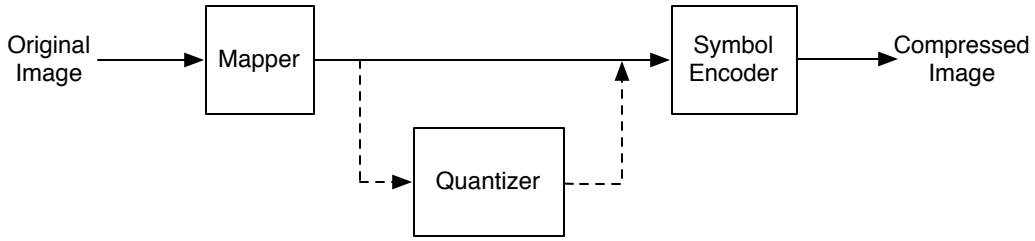
1. there is a path between any vertices $u, v \in V$ consisting only of edges in E';
2. no cycle can be formed from edges in E'.

Given a weighted graph G, the cost of connecting all the vertices can be computed by summing the weights of the edges in a spanning tree of G. A spanning tree of a weighted graph with the smallest possible total weight is called a *minimum spanning tree* (MST). Algorithms to find a minimum spanning tree of a graph have been extensively studied, and the algorithms of Kruskal and Prim [10] are commonly used to compute an MST. This thesis does not explore MST algorithms, as their computational complexity do not significantly add to the complexity of the proposed framework (Section 4.3). For example, the complexity of Kruskal's algorithm is $O(|E| \log^* |V|)$. [2]

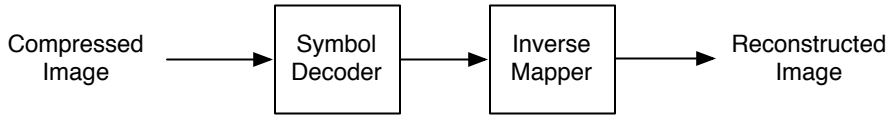
2.3 Image Compression

Image *compression* is the process of encoding an image to reduce the number of bytes required to store or transmit the image [20]. *Decompression* is the process of reconstructing an image from the encoded representation. Both processes are summarized in Figure 2.2.

[2] $\log^* n$ refers to an iterated logarithm function as defined in [10] as: $\log^* n = \min\{i \geq 0 : \log^{(i)} n \leq 1\}$. This is a very slow growing function.



(a) Encoding



(b) Decoding

Figure 2.2: Source encoder and decoder for image compression.

Compression is achieved by reducing redundancies contained within the image data. The three basic types of redundancies that occur within an image are *coding*, *inter-pixel*, and *psychovisual* redundancies. Coding redundancies occur when the number of coding symbols required to represent the image data is not the minimum. That is, the number of coding symbols used to code image data that has a high frequency is similar or equal to the number of symbols used for data that has a low frequency. This type of redundancy is reduced during the symbol encoder stage of compression using algorithms such as Huffman coding or arithmetic coding [20] by assigning few coding symbols (or bits) to represent data that has a high frequency. The coding symbols used to store an image may not take into account the correlation that exists between pixels. This results in inter-pixel redundancies between adjacent pixels. These redundancies can be minimized during the mapper stage of compression where the data is transformed into a more efficient representation. Lossless compression removes coding and inter-image redundancies, and the reconstructed image is identical to the original image—there is no information loss.

Psychovisual redundancies occur in an image when there may be information in the

image that cannot be *perceived* by human vision. For example, a 16-bit gray scale image could be quantized to use only 8 bits for each pixel. Such information can be removed from an image without significantly impacting the perceived image quality. This type of redundancy is reduced during the quantization stage and results in lossy compression. To improve compression ratios, lossy algorithms utilize quantization to remove psychovisual redundancies. Because this process is irreversible, an image compressed using a lossy algorithm is not exactly the same as the original image.

The file sizes generated by lossy algorithms are generally smaller than those produced by lossless algorithms. Lossy algorithms allow for a trade off between the image quality and the amount of compression achieved. By reducing image quality, a much higher compression ratio can be achieved by lossy algorithms. A common method to compare between multiple lossy compression algorithms is to examine the RMSE between the original image and the reconstructed image at the same compression ratio.

Although these three stages minimize redundancies within an image, others may exist. For example, video compression algorithms reduce *temporal* redundancies. Another type of redundancy that occurs in image sets is *inter-image* redundancy. A group of images, such as a set of medical images or a video stream, may share similar properties that can be utilized in image set compression. In video, the temporal relationship implies an inter-image relationship between successive image frames. A set of medical images, such as a group of ultrasound images of a body part, would form a group of similar images that contain inter-image redundancy.

2.3.1 Entropy

Entropy represents the average amount of information per symbol [43], and can be used as a measure of the compressibility for an image. Let $z = \{P(a_1), \ldots, P(a_J)\}$ where $P(a_j)$

is the probability of a gray level a_j in an image. The first-order entropy is defined as

$$H(z) = -\sum_{j=1}^{J} P(a_j)\log_2 P(a_j) \tag{2.5}$$

and it is measured in bits per pixel. The first-order entropy shows how much coding redundancy may be removed, but does not address inter-pixel redundancy. The higher-order entropies consider blocks of symbols instead of individual symbols $a_1, \ldots, a_J$, and give an estimate to inter-pixel redundancy. The second-order entropy uses overlapping horizontal pixel pairs as the symbols in the entropy calculation, while the fourth-order uses overlapping 2×2 blocks of pixels. In this thesis, the second-order and fourth-order entropy are also considered.

2.3.2 Wavelet Compression

Wavelets provide a powerful tool to improve image compression, and they are a key component of modern compression algorithms such as JPEG2000 [9]. The wavelet transform is a change of basis linear transformation. The application of a wavelet transform changes the standard basis functions of an input source to different basis functions corresponding to the wavelets. These wavelet basis functions vary in both frequency and spatial support. More specifically, each wavelet function has a different frequency range and support. The differing support allows different spatial information to be captured by each wavelet function. The goal is to obtain the coefficients corresponding to the wavelet basis functions because they capture both the frequency and the spatial information of the image. The wavelet transform is applied in the mapping stage (Figure 2.2) of image compression to separate the images into its spatial and frequency components.

Consider a one-dimensional input source $f(x)$. The wavelet transform is the applica-

tion of a pair of filters—a lowpass filter that yields the general trend of f, and a highpass filter that gives the details of f. Application of the transform results in two output images with different images with different spatial frequencies represented. The filtered output can be subsampled by a factor of two because the frequency range of each of the output streams is half of the input sequence. Recursively, the process is applied to the output stream of the lowpass filter and the process stops after a set number of passes. This results in coefficients corresponding to frequency and spatial information at different scales.

Figure 2.3 demonstrates the basics of this process. Note that the height of each box in the figure represents the length of the stream it contains. Using the basis functions resulting from the wavelet transform, an input stream can be exactly reconstructed because the wavelet transform is an invertible change of basis linear transformation. Therefore, the filters are invertible and using the inverse filters the process can be applied to the output sequences in the reverse order to reconstruct the input stream [20].

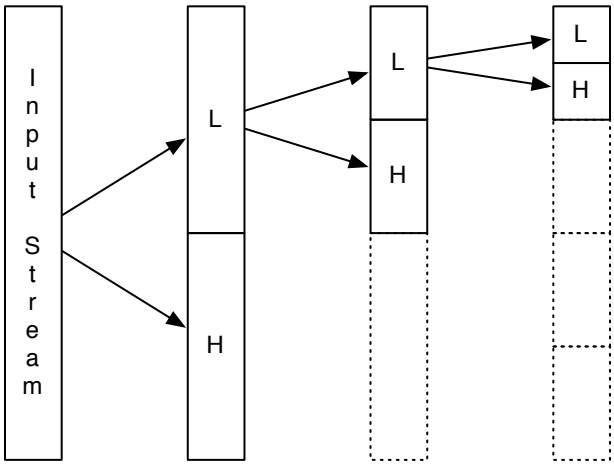


Figure 2.3: Visual representation of the decomposition of a one-dimensional input source using a wavelet transformation using three passes.

Many different functions can be used in the generalized wavelet method. For example, the Haar wavelet works on pairs of values from the input stream which are neighbors. The

highpass filter computes a scaled difference between two values and it is defined as

$$H(i) = \frac{f(2i) - f(2i+1)}{2} \tag{2.6}$$

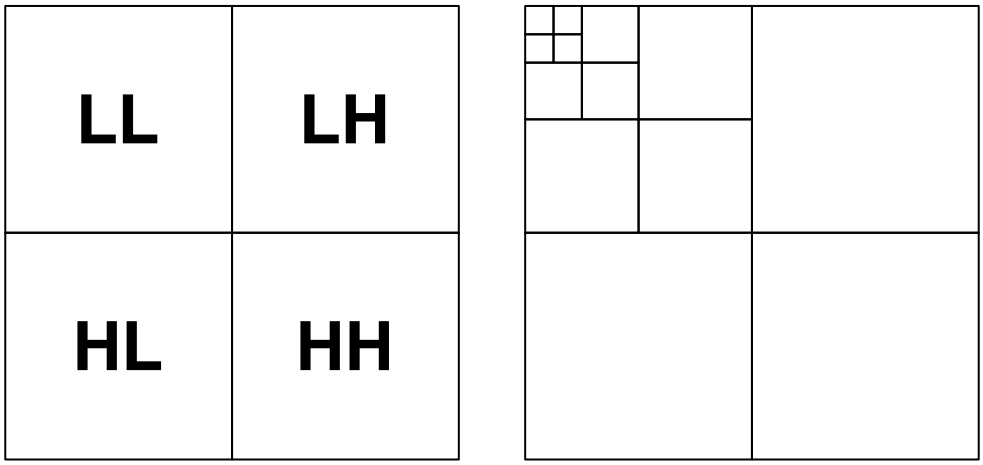
where $0 \leq i < N/2$. The lowpass filter computes an average between two values and it is expressed as

$$L(i) = \frac{f(2i) + f(2i+1)}{2}. \tag{2.7}$$

The reconstruction filters can be derived from the forward transformation. Further examples of the wavelet techniques include the Morlet wavelet [21], the Daubechies wavelet [11], and the Mexican hat wavelet [34]. JPEG2000 uses the Daubechies 9-tap/7-tap wavelet for lossy compression and the LeGall 5/3 wavelet for lossless compression [46].

A two-dimensional wavelet transform is an extension of the one-dimensional transformation described above. Given a two-dimensional input source, a one-dimensional wavelet transform is first applied to the columns of the input source followed by its application to the rows of the output. This results in four output bands (see Figure 2.4(a)) consisting of the combinations of highpass and lowpass filters. Each letter signifies the frequency range, with the first letter identifying the results of the transform on the columns followed by the rows. For example, the letters LH signify the low frequencies of the column, or vertical, application of the transform and the H signifies the high frequencies were chosen in the horizontal direction. The recursive application of this method results in what is commonly known as the *pyramid decomposition* as shown in Figure 2.4(b). The number of passes through the transformation is shown by the number of levels in the pyramid, which for the example shown is four passes. The majority of the information in most regular photographic images is contained within the low frequencies representing a low resolution representation of the image. Therefore, a large quantity of the high

frequency basis coefficients can be quantized without having a detrimental impact on the visual quality of the image. This results in lossy compression of the image.



(a) Output bands (b) Pyramid decomposition

Figure 2.4: Two-dimensional wavelet transformation decomposition structures.

2.3.3 Wavelet Packet Compression

It may be the case that important information within the wavelet coefficient domain is not located in the low frequencies in certain types of images such as fingerprint images [33]. In the pyramid decomposition, each application of the wavelet transform to a sub-band is a change of basis linear transformation. If decomposition is applied to other sub-bands, the result is a different set of basis functions. The wavelet packet transform considers all possible decompositions and the corresponding basis functions. Then, the set of basis functions that best represents the input stream is chosen. Intuitively, the best basis compresses the information in an image into a small number of coefficients that have little correlation among each other. The wavelet packet transform is completely specified by the wavelet transform and decomposition scheme. Computing all the decompositions of

an input stream increases the computational complexity for the wavelet packet transform when compared to the wavelet transform. The wavelet transform has a computational complexity of $O(M^2)$ while the wavelet packet transform is computed in $O(M^2 \log M)$ time for $M \times M$ input images [20].

2.4 Summary

A summary of the concepts and terminology used for this thesis were covered in this chapter. The definitions of digital images and difference images were given, and methods to measure the similarity between a pair of images were discussed. The basic concepts of graph theory used in this thesis were then presented. Finally, an overview of image compression was given.

Chapter 3

Analysis of Related Works

In this Chapter, previous works are briefly described. Their advantages and disadvantages are summarized.

3.1 The Centroid Methods

Karadimitriou and Tyler investigated a lossless strategy, called the Centroid method, of compressing a set of medical images around an average image [27, 28]. In this strategy, an average image of the set is computed. Then, a difference image for each image in the set is computed using the difference between the original image and the average image. Only the average and the difference images are encoded. The performance of this strategy is reliant on the set of images forming a tight cluster of very similar images. Otherwise, the difference images may not be easy to compress.

Karadimitriou and Tyler also showed that a clustering algorithm can be applied to partition a set of images into clusters of similar images, and the Centroid method can be applied to each cluster independently. This idea was examined experimentally by Nielsen *et al.* [39], who used the global average approach and proposed a lossy compression algorithm based on different clustering criteria. This work was mainly focused on parameter selection and trade-off analysis. Their experiments showed up to 25% performance improvement over JPEG2000 on individual images. Note that an average image is introduced for each cluster, so the computational overhead increases with the number of clusters. As a result, this strategy is suitable when the image set contains few tight clusters. Furthermore, redundancy among the clusters is not exploited as they are encoded independently.

El-Sonbaty *et al.* proposed an extension to the Centroid strategy [13]. Given a set of

images, a median image is computed. Using this median image, a set of difference images is then computed. This is essentially the Centroid method defined above where the only difference is how the central image is computed. Next, using the set of difference images, a new median image is computed. A new set of difference images is computed using the second median image and the initial set of difference images. Recursively, this process can be completed to a predetermined number of steps. The objective is to minimize the differences contained in the difference image set to bring them closer to zero, and therefore improve compression performance. Their results showed that this strategy was only effective to two steps and provided only a slight improvement over the single average image Centroid strategy.

3.2 The Template Method

For some applications, a set of images may have known similarities. For example, a set of form document images may contain an underlying form that is common to all the images. Wang and Yan proposed a lossless compression strategy to take advantage of this inter-image redundancy [55]. They used a predefined template image, such as an image of a blank form, instead of a computed average image. As in the Centroid strategy, the template is subtracted from each image. Then the template image and the difference images are compressed. If a common part exists, the Centroid method should also discover the underlying image as the average image. Using multiple template images, their algorithm partitioned a set of images into multiple clusters by matching each image to a template image, similar to the Centroid method with multiple clusters. This strategy is effective for predefined image sets, but it is not feasible for image sets whose properties are unknown prior to compression.

3.3 The Min-Max Methods

Karadimitriou and Tyler proposed a compression strategy to reduce inter-image redundancy by using a maximum value image and a minimum value image [27, 32]. Ait-Aoudia and Gabis proposed an improvement for the strategy [2] by using the LOCO-I algorithm [56] to compress the error images. The *max* and *min* images are computed by comparing all images in the set. The max image contains the maximum value for each pixel location and the min image contains the smallest pixel value for each location. Then, an error image is computed for each image in the set. The pixels of an error image contain the smallest difference between the original value and either the max or min value. The goal is to reduce the range of pixel values contained in a difference image which should improve compression.

Compressing an image using this method requires that the change between using the min and the max images to be synchronized between the encoder and the decoder. To achieve this, the encoder uses one of the images, either the min or max, until

$$e(i,j) \; > \; \frac{m(i,j)+M(i,j)}{2} \tag{3.1}$$

where (i,j) specifies the current pixel location, m is the min image, and M is the max image. To swap between the min and max images, the encoder records the current difference value and the next pixel evaluated will use the opposite image. The decoder follows the same strategy to correctly reconstruct the image. Karadimitriou showed improved compression using this method, but his results do not differentiate the contribution from removing inter-pixel and inter-image redundancies separately. In image sets that do not form tight clusters of similar images, this strategy performs poorly due to outlier images.

3.4 The Minimum Spanning Tree Methods

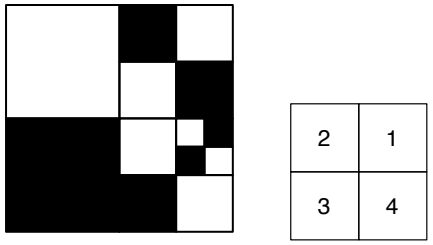
Another way to exploit inter-image redundancy was proposed by Chen *et al.* [5] and by Nielsen and Li [38]. In this strategy, an MST is computed from a complete graph. The vertices of the graph represent the images, and the weight of each edge is a measure of the cost of encoding one image given the other image of the edge. Chen *et al.* [5] used this strategy to link different views of 3D objects to represent the prediction relationship among the views. The edge weight used was based on motion estimation and compensation. On the other hand, Nielsen and Li [38] used the root-mean-squared error between the two images on the edge as the edge weight. Both of these methods were proposed for lossy compression and only experimental results were presented. Furthermore, there was no analysis of the optimality of this strategy.

An MST method is effective for compressing image sets that contain outlier images as an MST algorithm will find a spanning tree of minimum total weight. These methods requires a complete graph to be computed, which was computationally expensive ($\frac{n(n-1)}{2} = O(n^2)$ edge weight computations for n images). This method is not the optimal strategy for lossy compression because of the errors introduced to the images during compression. The proposed graph theoretical framework presented in Section 4.1 is based on the MST strategy of Nielsen and Li.

3.5 The Quadtree Method

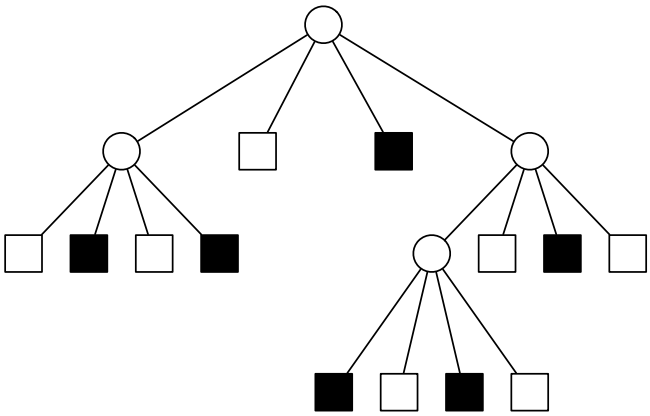
A quadtree is a common hierarchical structure used to represent an image [14, 42]. There are a number of image compression algorithms based on quadtrees [7, 31, 45, 47]. The following outlines the basic process to generate a lossless quadtree representation of an individual image. First, a root node is assigned to the image. If the image is homoge-

neous, the algorithm stops as this node is made a leaf node that fully represents the image. Otherwise, the image is partitioned into four equal quadrants and four corresponding children nodes are added to the root node. Each quadrant is considered a sub-image and each child node the root of a subtree. Now, the above process is recursively applied to each quadrant until the image is fully defined in the tree by leaf nodes. The resulting tree is the quadtree representation for the given image. Figure 3.1 demonstrates this process using a small binary image. Note that the leaf nodes in the resulting quadtree decomposition are the squares which are either white or black to represent the homogenous color in the leaf node's quadrant.



(a) Binary image

(b) Quadrant ordering

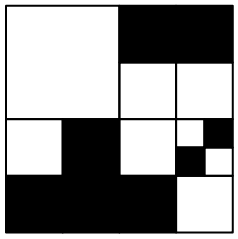


(c) Quadtree decomposition

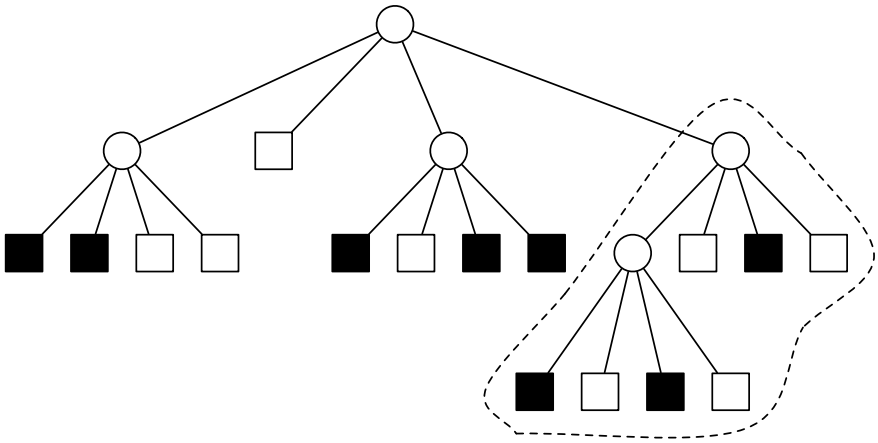
Figure 3.1: A quadtree example for a small binary image.

The quadtree structure lends itself well to image set compression. Inter-image redun-

dancies can be identified by comparing the subtree structures among the quadtrees for an image set. For example, the comparison of the binary image and its corresponding quadtree decomposition in Figure 3.1 and the image and quadtree in Figure 3.2 shows a common subtree existing between the quadtrees of the two images. It is highlighted in Figure 3.2(b). These common parts represent inter-image redundancies that can be reduced to improve set compression.



(a) Second binary image



(b) Second quadtree decomposition

Figure 3.2: A second binary image and its quadtree.

A number of algorithms have been proposed to improve image set compression using quadtrees [4, 26, 53, 54]. Vassilakopoulos *et al.* proposed a quadtree set compression strategy using an overlapping structure [54]. The images are stored in a sequential order where image i is considered to overlap the previous image $i-1$ if parts of their quadtrees

are the same. The quadtree for image i is modified to reference the commonly shared parts in image $i-1$. The results in [54] showed an improvement of up to 90% for similar binary images. Unfortunately, the performance of quadtree image compression is known to be inferior to algorithms such as JPEG2000 for many types of non-binary images, such as photographic images [6], because quadtree methods often require a higher bitrate to achieve a particular distortion on these types of images.

3.6 The 3D Image Set Method

Advanced medical imaging tools, such as a magnetic resonance imaging (MRI) scanner and a computed tomography (CT) scanner, generate large volumes of images annually [36]. A common application of these tools is to generate multiple images at different cross sections of an object. The medical imaging of a body part is a common example. These two-dimensional (2D) images, or *slices*, are taken at varying intervals and they can be used to construct a three-dimensional (3D) representation of the imaged object. Qi *et al.* have proposed an algorithm to compress such a group of images [40, 41]. The authors refer to a group of these images as a 3D image data set. Intuitively, video compression algorithms appear to be a good approach to compress such a set, but the imaging process introduces too much noise for this type of compression strategy to work effectively. The authors proposed a prediction algorithm which utilizes the previous image and the current image to compute a prediction image. The resulting image is compressed using a wavelet compression algorithm. Although their method is similar to video compression algorithms, it does not use any time-based variables for the prediction. This is a specialized compression algorithm which assumes that the input will be a 3D image data set of a single object. Therefore, this algorithm may not be good for compressing image sets whose properties are very different from 3D image sets.

3.7 The Wavelet Method

Tashakkori proposed an alternative approach to compressing sets of images which focused on reducing inter-image redundancies by working within the wavelet coefficient domain [49]. First, the method applies the wavelet transform to each image in the set. Then, linear regression is used to predict one image from another, and the resulting error image is compressed. Tashakkori concluded that linear regression was much more effective in the wavelet domain compared to the linear regression technique on the original images, because there is a higher correlation among images in the wavelet coefficient domain. However, prediction is a linear process. Therefore, it is equivalent to apply the same prediction in the image domain and then apply the wavelet transform to the resulting error.

3.8 The Principal Components Method

Principal components analysis (PCA) is a linear transformation that provides the theoretical optimal representation for a data set [23]. This transform is also commonly known as the Hotelling transform or the Karhunen-Loeve transform. Given a set of images, the transform computes a set of basis functions that decorrelates the information contained within the set. These basis functions are ranked in descending order of importance. When an image is written as a linear combination of these basis functions, the coefficients corresponding to the least important basis elements can be quantized because they contain little information needed to represent the image. Therefore, the number of basis functions required to approximate the image set is significantly reduced.

More formally, PCA can be described as follows. Consider a set of n images. Normally, an image is represented as a two-dimensional array (Section 2.1), but it can also be considered as a column vector of dimension MN. Now, let Φ_i represent the i^{th} image

with the average removed. The covariance matrix of the image set is $C = AA^T$ where $A = [\Phi_1, \Phi_2, \ldots, \Phi_n]$. The eigenvalues and eigenvectors of C are computed. The eigenvectors can be ranked by the magnitude of the associated eigenvalues. An eigenvector corresponding to a larger eigenvalue is more important for representing the image set. On the other hand, the coefficients of eigenvectors corresponding to small eigenvalues can be quantized more coarsely without losing much information.

Although PCA has been used successfully for the facial recognition problem in large sets of similar images [30, 52], it is not suitable for compression. Musatenko and Kurashov proposed a lossy compression strategy for image sets using PCA [35]. Their algorithm utilizes Gram-Schmidt orthogonalization to compute the basis functions for an image set. Then, each basis function is encoded using the embedded zerotree wavelet (EZW) coding algorithm [44]. The compression ratio for each basis function is set based upon how much it contributes to the representation of the image set. PCA is computationally expensive [27], and the cost of their algorithm has a complexity of $O(n^3MN + n^2)$ where n was the number of images in the set and the dimensions of the images are $M \times N$. Musatenko and Kurashov concluded that their algorithm was only effective for sets with up to 10 images. Furthermore, this technique can only work for lossy compression on a set of images that are very similar so that the set can be approximated by a subspace of small dimensions.

3.9 Summary

A number of previously proposed strategies for image set compression were examined. There is a group of strategies, such as the Centroid method, that are effective on image sets with certain qualities, but are not effective on all image sets. Another group of strategies, such as the PCA method, are theoretically elegant but are too computationally

expensive for practical use. Furthermore, many of the previous strategies have only been studied experimentally. Tashakkori's wavelet method for predicting images is similar to the method proposed in this thesis, but it does not examine which pairs of images should be used in the prediction.

Chapter 4

Graph Theoretical Framework for Image Set Compression

Graphs play an important role in computer science as they are often used to model relationships among objects. An image set can be modeled as a graph where each vertex represents an image in the set and each edge represents a measure of the difference between a pair of images. Such a structure can be useful for analyzing the inter-image relationships and useful for compressing image sets. Traditionally, image compression has focused on compressing each image individually and ignores any similarity that may be found among other images in a set. A compression algorithm could examine the graph to remove inter-image redundancies by compressing the difference images corresponding to the edges. Intuitively, the smaller the difference between a pair of images, the easier the difference image will be to compress. Thus, choosing the edges with the smallest weights will lower the cost to compress an image set. Using such a graph structure is central to the proposed framework and it will be investigated in this chapter.

4.1 Graph Construction

Let $S_n = \{I_1, I_2, \ldots, I_n\}$ be a set of n images of the same dimensions. In order for the proposed framework to model the existing strategies [5, 27, 28, 38, 39], two additional images need to be defined:

- a zero image $I_{n+1} = I_z$ with $I_z(i, j) = 0$ for all (i, j);

- an average image $I_{n+2} = I_a$ where

$$I_a(i,j) = \frac{1}{n}\sum_{k=1}^{n} I_k(i,j). \tag{4.1}$$

These images are used to define two slightly larger image sets:

- $S_z = S_{n+1} = S_n \cup \{I_{n+1}\}$;
- $S_a = S_{n+2} = S_{n+1} \cup \{I_{n+2}\}$.

Given an image set $S_* \in \{S_z, S_a\}$, a complete undirected, weighted graph (Section 2.2) is defined. The vertices of G are $V = \{I_i \mid I_i \in S_*\}$, and $E = \{(I_i, I_j) \mid I_i, I_j \in V\}$ defines the graph edges. The weight for each edge (I_i, I_j) is defined by the function $w(I_i, I_j)$, where $w : S_* \times S_* \to \mathbb{R}_{\geq 0}$ is a function that measures the cost to reconstruct I_j assuming I_i is known. In this thesis, w is assumed to be symmetric and that the weight of an edge is the cost of encoding the difference image $I_i - I_j$. For example, the function w can be an entropy measure of the difference image, which represents the potential of compressing the difference image. Another choice of w is the actual size of the compressed difference image using compression algorithms such as JPEG-LS [56].

The proposed framework differs from that of Chen *et al.* [5] in two ways. By introducing an average image, all the previous strategies can be included in this framework. Also, this framework assumes that the edge weights are symmetric which is necessary for obtaining performance guarantees of the proposed lossy compression scheme.

4.2 Compression and Decompression

4.2.1 Lossless Case

Decompressing an image in a set corresponds to following a path from some starting image to the image itself. Thus, the first step in compressing an image set is to choose the starting images that provide entry points into the underlying graph. Then, a set of edges is chosen such that there is a path to any image in the set from a starting image. The starting images and the difference images corresponding to the chosen edges are encoded using an image compression algorithm.

In order to decompress an image in the set, a starting image and a path to the image are required. Beginning with the starting image, each difference image in the path is reconstructed incrementally using the previous image and the connecting difference image in the path. Therefore, each image in the set is either a starting image or it is connected to another image that can be reconstructed. The storage cost for S_n is the cost of storing the starting images and the difference images corresponding to the chosen edges, as well as a negligible amount to record which edges are chosen.

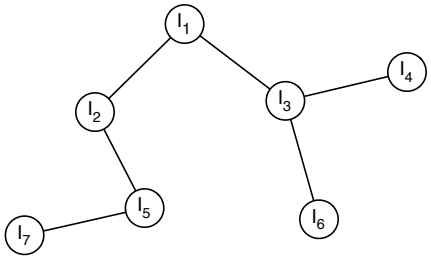


Figure 4.1: Decompressing image I_7.

For an example of decompressing an image, consider the set of images encoded by the edges shown in Figure 4.1, with I_1 being the starting image. In order to decompress image I_7, first I_1 must be decompressed. Next, the difference image from the edge (I_1, I_2) is de-

compressed. With I_1 and the difference image, I_2 is reconstructed by adding the difference image to I_1. To reconstruct the next image I_5, the difference image corresponding to the edge (I_2, I_5) is decompressed and added to I_2. Finally, the difference image for the edge (I_5, I_7) is decompressed and combined with I_5 to reconstruct I_7.

The zero image I_z has a key role within the proposed framework. Including the "zero edges" (I_i, I_z) in the graph allows the images to be represented as difference images corresponding to the edges $I_i - I_z$. The importance of keeping the graph connected is achieved through I_z, and the chosen edges now form a spanning tree (Section 2.2) of G. Because I_z is known to the decoder, the need for storing any starting images is removed. The cost of storing an original image in the set moves from a vertex to an edge in the spanning tree, and this becomes important when working with image sets that have clusters of similar images. In this case, coding the difference between clusters may be more expensive than coding the original images. Choosing a "zero edge" to some image in each cluster may reduce the encoding cost.

4.2.2 Errors Introduced by Lossy Compression

The errors introduced by lossy compression will affect the performance of the proposed framework. Using the method described in Section 4.2.1, errors resulting from the lossy compression process may propagate down the paths in the spanning tree because the reconstructed difference images are added together. As each image along a path is reconstructed, the errors from all the previous images accumulate and result in the reconstructed image becoming unrecognizable. Care must be taken to prevent such errors from propagating in the spanning tree.

To understand this process informally, consider the spanning tree shown in Figure 4.2. Let $e(I, I')$ be the error between an original image I and its reconstructed image I' in

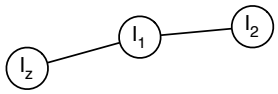


Figure 4.2: A path representing how two images from a set may be compressed.

the framework, including both the error introduced by the underlying lossy compression algorithm and the error introduced by the reconstruction process for the MST strategies. It is assumed that $e(d,d') = \Delta$ if d is a difference image directly encoded by the lossy compression algorithm. To compress the image I_1, the difference image for the edge (I_1, I_z) is encoded. Moving down the path, the image I_2 is compressed by encoding the difference image for the edge (I_2, I_1). To decompress the set, the image I_1 is reconstructed followed by I_2. Thus, decompressing the difference image for the edge (I_1, I_z) results in the image $I_1' = (I_1 - I_z)'$ which has an error of $e(I_1', I_1) = \Delta$. Next, reconstructing image I_2 using the image I_1' and the decompressed difference image for the edge (I_2, I_1) results in

$$I_2' = (I_2 - I_1)' + I_1'$$

and has an error of

$$\begin{aligned} e(I_2', I_2) &= e((I_2 - I_1)' + I_1', (I_2 - I_1) + I_1) && (4.2) \\ &\approx e((I_2 - I_1)', (I_2 - I_1)) + e(I_1', I_1) && (4.3) \\ &= 2\Delta. && (4.4) \end{aligned}$$

The approximation in Equation (4.3) is valid if the errors introduced by the lossy compression on the two images do not cancel. This scenario is shown in Figure 4.3. The reconstruction error for an image is proportional to its distance from I_z in the spanning tree.

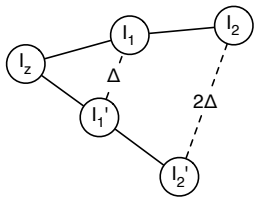


Figure 4.3: Lossy compression error propagation in a spanning tree.

To reduce these errors when compressing the difference images, the difference between the original images is not taken. Since the "parent" of the edge (the vertex closer to I_z) has already been compressed, the encoder compresses the difference image between the child and the decompressed parent image. The decoder can perform the same process since the parent can be decompressed. This ensures that errors in the lossy compression process do not propagate. The process is similar to the method used in video compression in which a frame is predicted based on the decompressed previous frame [20]. A more detailed analysis of the errors introduced will be discussed in Section 4.4.

4.3 The Cost of a Compression Strategy

Any compression strategy that takes advantage of inter-image redundancy between pairs of images can be represented as a spanning tree within the proposed framework. A compression strategy is represented by a subset of edges $E' \subseteq E$. Intuitively, the chosen edges correspond to the difference images that are coded. For each compression strategy, the total weight

$$w(E') = \sum_{(I_i,I_j)\in E'} w(I_i,I_j) \tag{4.5}$$

of the spanning tree represents the storage cost for the strategy. Note that the spanning tree for each strategy has to be encoded either implicitly in the algorithm or explicitly, but the cost is negligible.

Given the graph G, a minimum spanning tree is an acyclic subgraph that connects all the vertices and that has a total weight (in Equation (4.5)) that is the minimum. Intuitively, the motivation for using the minimum spanning tree in this framework is to find the smallest cost to compress an image set. Since the cost of a compression strategy is the sum of the edge weights in a spanning tree, the minimum spanning tree gives the optimal lossless compression strategy for an image set. Kruskal's algorithm (Section 2.2) was used for the experiments in this thesis. It has a computational complexity of $O(|V|^2 \log^* |V|)$ because $|E| = O(|V|^2)$.

4.4 Optimality of Performance

Since the MST computation is performed on the graph constructed from the original images, errors introduced by lossy compression may change the actual graph. As a result, it is not necessarily true that the MST computed is the optimal strategy for lossy compression. Furthermore, it is difficult to compute the MST of the "optimal perturbed graph"—how each image is perturbed in the final graph depends on its path from I_z in the spanning tree, and the path in turn depends on the tree edges chosen from the original graph.

This section gives a bound on the difference between the weight of the optimal MST compared to that of the computed MST. The bound depends on the maximum distortion introduced by the lossy compression process, and this result is valid for an MST of any graph that results from perturbing each vertex in the original graph by at most the maximum distortion. Therefore, the bound gives a performance guarantee of the MST strategy based on the original graph compared to the optimal MST strategy, without explicitly computing the optimally perturbed graph. The optimally perturbed graph is difficult to compute because of the cyclic dependency between the choice of tree edges and the perturbation of edge weights between vertices.

The following outlines how the performance bound is obtained. The derivation of these results assumes that the edge weight function is a metric [3]. The root-mean-square-error (RMSE), which is a commonly used metric to measure the difference between images, can be used for the edge weights in these results. First, the original MST is computed. Then, a spanning tree of the perturbed graph is computed using the edges chosen in the original MST. The goal is to understand the difference in weight between the original MST and the spanning tree of the perturbed graph.

In the following, let N be the number of vertices in the graph G. Let I'_i $(1 \leq i \leq N)$ be an image such that the RMSE between I_i and I'_i is bounded by Δ, and G' be the graph constructed from $\{I'_i\}$. Δ is thought of as a "quality" parameter in the underlying lossy compression algorithm, and I'_i as the reconstructed images. T_G and $T_{G'}$ are used to denote an MST of G and G', respectively. In addition, $w(T)$ is the cost (the sum of edge weights) for any spanning tree T.

Suppose e_{ij} is the edge between I_i and I_j in G and it is included in T_G. Let e'_{ij} be the edge connecting I'_i and I'_j in G' (Figure 4.4). The edge e'_{ij} is the result of decompressing the difference image $I_i - I_j$ when taking into account the errors introduced by lossy compression.

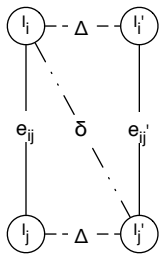


Figure 4.4: Using an edge in T_G as an edge of a spanning tree in G'.

Since the edge weight function is a metric, the triangle inequality implies that

$$\delta - \Delta \leq w(e'_{ij}) \leq \delta + \Delta,$$
$$w(e_{ij}) - \Delta \leq \delta \leq w(e_{ij}) + \Delta,$$

so that

$$w(e_{ij}) - 2\Delta \leq w(e'_{ij}) \leq w(e_{ij}) + 2\Delta. \tag{4.6}$$

Now, if $I_i = I_z$, there is no reconstruction error so $I_z = I'_z$. In that case, the bound is refined to

$$w(e_{ij}) - \Delta \leq w(e'_{ij}) \leq w(e_{ij}) + \Delta. \tag{4.7}$$

Since there are $N-1$ edges in a spanning tree, the following performance guarantee is obtained on using the MST computed from G in G'.

Theorem 1 *Let T_G be an MST of G, and D be the degree of I_z in T_G. If T is the spanning tree of G' obtained by using the same edges as T_G, then*

$$|w(T) - w(T_G)| \leq (2N - 2 - D)\Delta.$$

This gives a bound on the performance of the compression strategy compared to that predicted by the MST algorithm on the original graph G. However, it does not relate to the optimal strategy given by an MST of G'.

Now, since T in Theorem 1 is a spanning tree of G', it follows that

$$w(T_{G'}) \leq w(T) \leq w(T_G) + (2N - 2 - D)\Delta. \tag{4.8}$$

In the derivation of Theorem 1, the only assumptions on G and G' are that $I_z = I'_z$ and the RMSE between I_i and I'_i is bounded by Δ. It does not matter whether G or G' is the graph constructed from the original images. Therefore, the roles of G and G' can be interchanged to also obtain

$$w(T_G) \leq w(T) \leq w(T_{G'}) + (2N - 2 - D')\Delta. \tag{4.9}$$

Since $D, D' \geq 1$, it follows from (4.8) and (4.9) that

$$|w(T_G) - w(T_{G'})| \leq (2N - 3)\Delta. \tag{4.10}$$

Combining with Theorem 1 gives the following performance guarantee of the MST compression strategy relative to the optimal strategy.

Theorem 2 *Let T be a spanning tree of G' obtained by using the same edges as T_G, and D as the degree of I_z in T_G. Then*

$$|w(T) - w(T_{G'})| \leq (4N - 5 - D)\Delta.$$

It is important to note that there is no assumption on the actual perturbations made on the images for the graph G', so that the performance bound above applies to any graph with the same quality bound Δ. Thus, the performance bound indeed gives a relationship between the quality of compression Δ and the coding performance relative to the optimal strategy. As Δ becomes smaller, the quality of compression increases and the difference between G and G' decreases. Thus, $T_{G'}$ is closer to the optimal MST. Although the analysis was done for the proposed framework, it also applies to [5, 38]. Thus, it also gives a performance guarantee for [5] and [38] assuming edge weights are symmetric.

4.5 Compression Strategies

This section examines four compression strategies and models them within the proposed framework. Although only four strategies are mentioned, any compression strategy that utilizes inter-image redundancy between two images can be represented within the framework.

4.5.1 Traditional Strategy

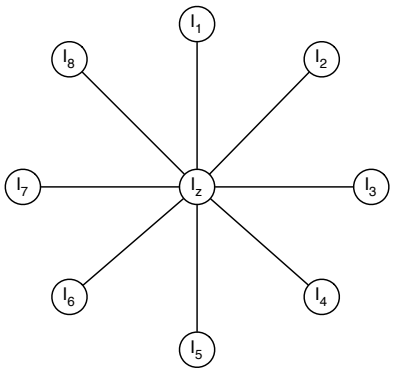


Figure 4.5: Traditional strategy.

The traditional scheme results in a star graph created from the set S_z with I_z as the center as shown in Figure 4.5. The graph represents encoding each image individually as the only edges in the graph are (I_i, I_z). This strategy requires $n = O(n)$ calls to the compression algorithm to encode the image set.

4.5.2 Centroid Strategy

In the proposed framework, the Centroid strategy with a single cluster is represented by a star graph created from S_a. The center for this spanning tree is the average image I_a,

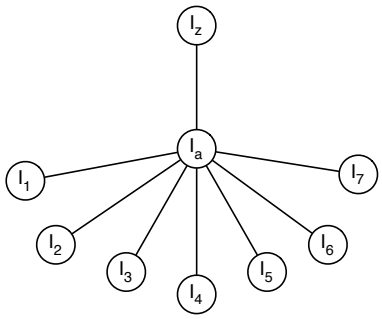


Figure 4.6: Centroid strategy with a single cluster.

and each edge (I_i, I_a) is the difference of the image I_i from the average I_a (Figure 4.6). Similarly, the Centroid strategy with multiple clusters can be represented by a spanning tree with an extra vertex for each cluster average. Figure 4.7 shows an example where images I_1 to I_4 form a cluster with the average image I_{a1}, and images I_5 to I_7 form another cluster with the average image I_{a2}. In addition, template extraction strategies [55] can also be represented by using the template image in place of the average image in the spanning tree. As in the traditional strategy, the Centroid strategy with a single cluster requires $n = O(n)$ calls to the compression algorithm to encode the image set.

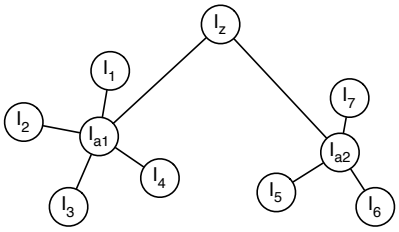


Figure 4.7: Centroid strategy with multiple clusters.

4.5.3 Minimum Spanning Tree Strategy (MST)

Since all compression schemes considered in the proposed framework must be a spanning tree, the optimal compression scheme is provided by the minimum spanning tree. The MST strategies by Chen *et al.* [5] and Nielsen and Li [38] can also be adapted for lossless compression and represented in the framework. This strategy requires a complete graph of S_z to be constructed, and this requires $\frac{n(n-1)}{2} = O(n^2)$ calls to the compression algorithm to construct the graph. Then a minimum spanning tree algorithm is applied to the graph, and the resulting MST is guaranteed to be the optimal compression strategy for S_z (Section 4.1) for lossless compression.

4.5.4 MST with Average Image Strategy (MST_a)

There are situations when the Centroid strategy is better than the MST strategy, and vice versa. The MST_a method automatically chooses the best strategy locally for a subset of the images in the complete graph constructed from S_a (Section 4.1). For example, if S_n contains a group of similar images and a few outliers, it may be more efficient to encode the outliers using inter-image differences instead of the differences to the average. Figure 4.8 demonstrates how this may occur in the graph. The length of an edge is used to represent its weight in the figure, where a longer edge implies a higher edge weight. From the figure, it is clear that $w(I_3,I_2) < \min(w(I_3,I_z),w(I_3,I_a),w(I_3,I_1))$, and in this case an MST would contain the edge (I_3,I_2). Yet for image I_7, it is clear that $w(I_7,I_a) < \min(w(I_7,I_z),w(I_7,I_6))$, and the edge connecting with the average image would be selected.

The graph is similar to the one used in the MST strategy above except that S_a is used. As in the Centroid strategy, additional average images can be introduced if there are mul-

tiple clusters. Notice that the MST does not have to be computed from scratch, and can be computed incrementally from the MST computed in the previous strategy [22] as additional average images are added to the graph. The resulting MST is guaranteed to be the optimal lossless compression strategy for S_a, and in some cases is better than the MST strategy without an average image.

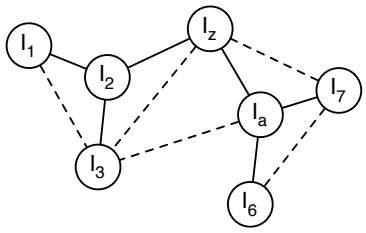


Figure 4.8: Example of local edge choice in the MST$_a$ strategy.

4.6 Summary

The inter-image relationships between pairs of images can be modeled within the proposed graph theoretical framework. This framework allows for the comparison of all strategies that work to reduce inter-image redundancies between pairs of images. For lossless compression, the optimal strategy can be obtained using an MST algorithm on the complete graph. On the other hand, the errors introduced by lossy compression perturb the graph and make the computation of the optimal strategy difficult. In this case, a performance guarantee is given for the a strategy that is near optimal.

Chapter 5

Experimental Results

The experimental results are contained in this chapter. The experiments were conducted to test the proposed framework using both lossless and lossy image compression algorithms.

5.1 Test Image Sets

Five sets of images were used to test the proposed framework. All the images were 8-bit gray scale images, and typical images from each set are shown in Figures 5.1 through 5.4. The fifth image set was composed of images from the Galway and Pig image sets. Many fields, such as medicine and satellite imaging, accumulate large collections of images, and the test sets represent a sampling of these diverse areas. Although currently there are no standardized sets of images for testing image set compression, these experiments used the same sets as the ones used by Nielsen *et al.* [38, 39] for their MST and clustering results.

For each image set, a spanning tree was generated for each of the four compression strategies presented in Section 4.5. The resulting spanning tree was then used to compress the image set. Since clustering algorithms are not a focus in this thesis, the results for the Centroid strategy only used a single cluster. However, the Centroid strategy with multiple clusters can be modeled with the proposed framework (Figure 4.5.2).

5.1.1 Galway

The first image set contained 28 images from a webcam [16] from Galway City, Ireland. Focused on High Street, the images from Galway.Net Camera 1 are 800×600 pixels in size. The images were taken intermittently between 7 P.M. and 11 P.M. on November

Figure 5.1: Typical images from the Galway set.

7, 2004. Although taken sequentially, the images were shot sporadically with up to 15 minutes between images. While the intensity values and most of the scene remained static throughout the set, the images contain moving people (Figure 5.1).

5.1.2 Pig

The second test set consisted of 304 Ultrasound images of pig rib cages provided by Dr. Alan tong of Agriculture Canada. The 800×600 pixel Ultrasound images were recorded with an Aloka Flexus Model SSD-110 equipped with a 3.5 Mhz/127mm transducer Ultrasound system, from between the 3^{rd} and 4^{th} ribs from the last rib and 7 cm off the mid-line of hog carcass. For every image in the set, only the diagnostic region and some text changed while the remaining areas were unchanged. Thus, most of the images in the set were very similar (Figure 5.2).

5.1.3 Joe

The third image set contained 162 webcam images from Joe Tourist Weather [25]. The webcam was located in Victoria, British Columbia, Canada. The images are 320×240 pixels in size and were taken at intervals throughout the day. The set contained images taken between November 23, 2004 and December 28, 2004. Changing weather and the location of the sun resulted in a wide range of intensities in the images within the set. This resulted in inter-pixel redundancy in the difference images as large portions of each image change in a similar manner due to the lighting conditions (Figure 5.3).

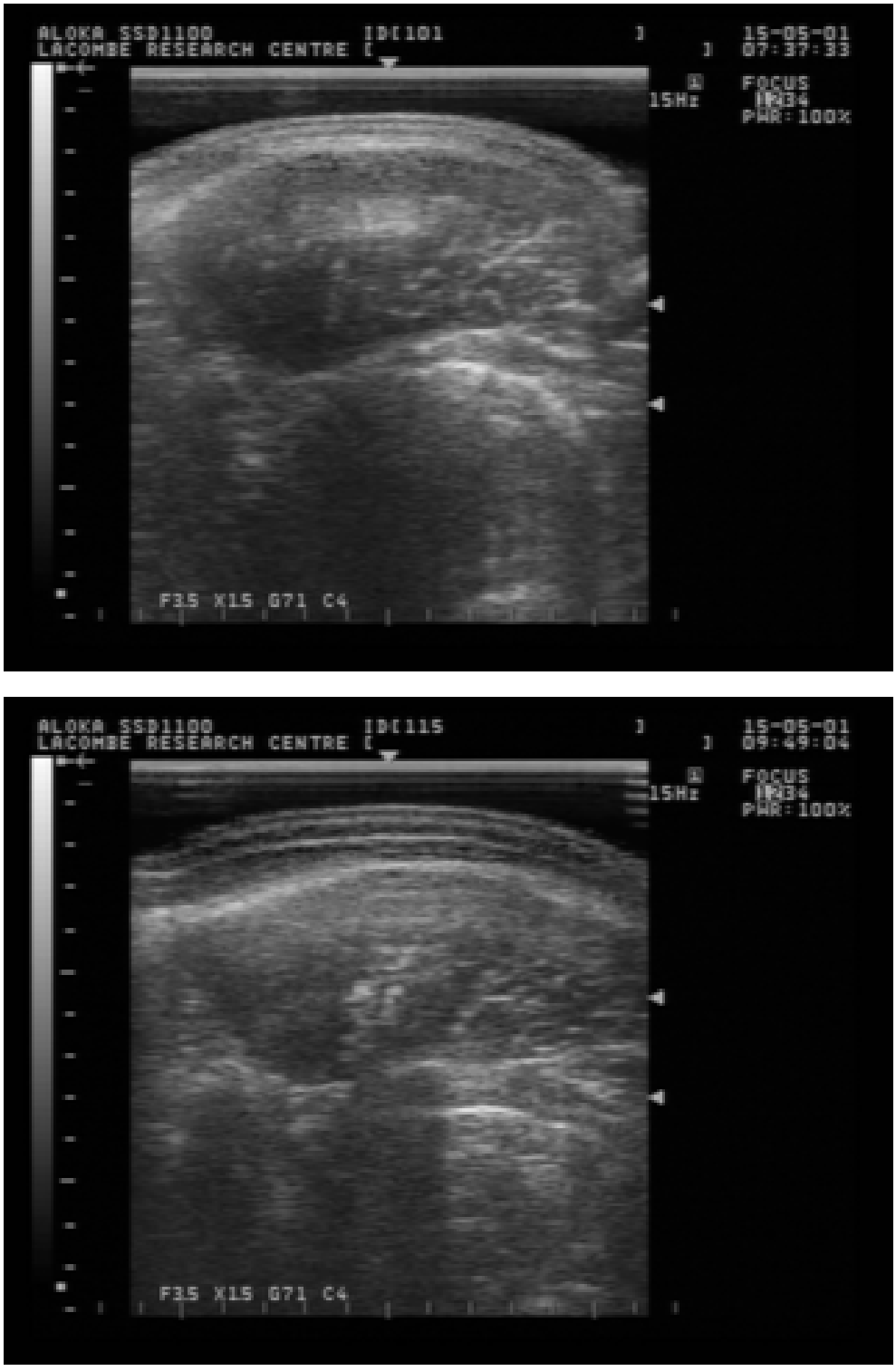


Figure 5.2: Typical images from the Pig set.

Figure 5.3: Typical images from the Joe set.

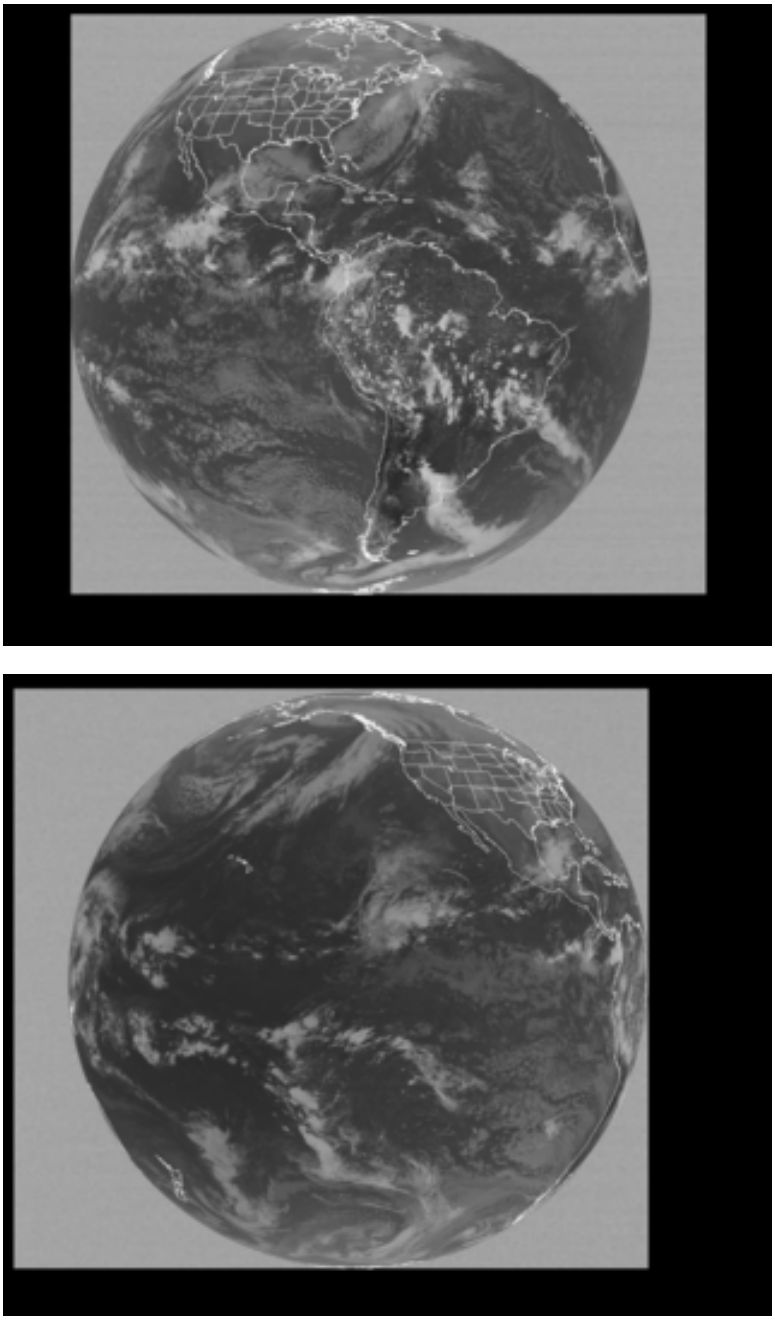


Figure 5.4: Typical images from the GOES set.

5.1.4 GOES

Geosynchronous satellite images of earth from the GOES Project were used for the fourth image set [19]. This set contained 128 thermal infrared images that are 1200×1000 pixels in size. The images were from two satellites, and each satellite produced a pair of images from its left and right sensors. Although the exact date for the images is unknown, the set was the same used in [38, 39]. The GOES 8 satellite provided the GOES-EAST images, which focused on eastern North America. The GOES 9 satellite focused on western North America and provided the GOES-WEST images.

5.1.5 Combination

The final set of test images was composed of the Galway image set combined with the first 29 images from the Pig set. The images from these two sets were chosen because they have the same dimensions. The goal was to test the framework with two clusters of images that have no relation to one another.

5.2 Implementation Details

Software was implemented to test the graph-theoretical framework proposed in the previous chapter. The software was written in the C and C++ programming languages [29, 48] and tested on a Dell server equipped with dual 2.7 GHz Intel Xeon processors running on the Fedora Core 6 distribution of the Linux operating system. The software allowed for the testing of each of the strategies presented in Section 4.5. The software also performs the graph construction and the MST computation required in the MST strategies. The Netpbm and libtiff image libraries [37, 50] were used by the software to read and write images.

The software also allowed various compression algorithms, such as JPEG2000 [1, 9], to be used with in the framework to compress the difference images.

5.3 Lossless Compression Results

For lossless compression, the actual compression results of the difference images were used as the edge weights in the graph. The total number of bytes required to store the image set was reported. A smaller total indicates a better compression strategy. In these results, the difference images were encoded as 9-bit images. The experiments compared the four strategies using JPEG2000 [1, 9] and JPEG-LS [56] compression algorithms to compress the difference images.

The performance of the framework can be measured by comparing the results of each of the compression strategies for each image set. The JPEG2000 results are given in Table 5.1 and the JPEG-LS results are shown in Table 5.2. The best results are highlighted for each set. Note that the minimum was always achieved by one of the two MST strategies. The experimental results show the effectiveness of the proposed framework in adapting to any given image set. Although JPEG-LS slightly outperformed JPEG2000, both tools reflected the same trend in the results. Table 5.3 shows the number of each type of tree edge chosen by the MST_a strategy. In this table *zero* refers to a tree edge with the zero image I_z as one of the vertices, *average* refers to a tree edge connecting the average image I_a and an image in the image set S_n, and *inter-image* represents a tree edge whose vertices are both images in S_n. Some remarks on the lossless compression results for each image set are given below.

Table 5.1: Lossless compression results (in bytes) using JPEG2000.

Strategy	Galway	Pig	Joe	GOES	Combination
Traditional	3,793,591	31,311,189	4,673,678	**56,244,059**	**6,863,327**
Centroid	**3,710,070**	29,109,705	4,562,428	60,049,815	7,818,852
MST	3,793,591	31,131,843	**4,434,391**	**56,244,059**	**6,863,327**
MST$_a$	**3,710,070**	**29,019,727**	4,451,691	56,511,969	6,960,466

Table 5.2: Lossless compression results (in bytes) using JPEG-LS.

Strategy	Galway	Pig	Joe	GOES	Combination
Traditional	3,604,582	29,689,794	4,285,473	**54,500,444**	**6,528,427**
Centroid	**3,550,729**	28,686,186	4,200,547	58,415,933	7,478,822
MST	3,604,582	29,600,067	**4,127,463**	**54,500,444**	**6,528,427**
MST$_a$	**3,550,729**	**28,574,336**	4,132,007	54,747,216	6,613,346

Table 5.3: Types of tree edges chosen by MST$_a$ using JPEG2000 for edge weights.

Type	Galway	Pig	Joe	GOES	Combination
Zero	1	1	5	129	58
Average	28	303	24	0	0
Inter-Image	0	1	135	0	0

5.3.1 Galway

Because of the large amount of redundancy among the images, the results showed that the Centroid strategy performed the best with this set. Of course, the MST_a strategy cannot perform worse than the Centroid strategy. In the MST_a case, all the images were joined through the average image as the set forms a tight cluster.

5.3.2 Pig

Here the MST_a strategy provided the best performance. Using JPEG2000 resulted in a 7% improvement by the MST_a strategy over the traditional strategy. The majority of images in the set were connected through the average image in the MST computed, thus indicating a clustering of the images within the set. The proposed framework allows the best compression strategy to be chosen locally in an image set. This is an improvement over using either strategy independently.

5.3.3 Joe

The MST strategy provided the best result for this image set. For sets of images that do not form a tight cluster but have small inter-image differences, MST without an average image provides the best encoding method. This set has a wide range of lighting conditions due to the movement of the sun and changing weather conditions, and this resulted in many small clusters of images grouped around various intensity levels. Since the image set contained many small clusters, the cost of storing an extra average image outweighs the redundancy removed from the few images in the cluster. Even if multiple clusters were used, the Centroid strategy would not perform as well as the MST method without average

images. Figure 5.5 shows a portion of the MST from this image set.

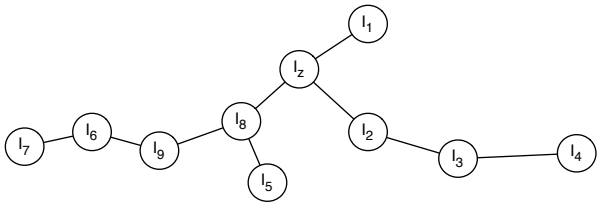


Figure 5.5: A portion of the computed MST for the Joe image set.

5.3.4 GOES

Here, the traditional strategy provided the best performance because the difference images required more storage than the original images using both JPEG2000 and JPEG-LS. Thus, it is not always better to compress difference images instead of the original ones. In this case, the fact that no average edges were used means the average image can be removed from the set.

5.3.5 Combination

As expected, the Centroid strategy with a single cluster did not perform well on this set. On the other hand, the MST_a strategy provided significant improvement over the Centroid strategy since it is the optimal strategy when the average image is included. Again, the MST strategy automatically recognized that the traditional strategy is best for this set of images.

Table 5.4: Galway entropy results.

Strategy	*Bits per Pixel*		
	1^{st} Order	2^{nd} Order	4^{th} Order
Traditional	6.857	5.437	3.821
Centroid	5.064	4.481	3.641
MST	**4.898**	**4.462**	**3.615**
MST$_a$	5.000	4.476	3.640

Table 5.5: Pig entropy results.

Strategy	*Bits per Pixel*		
	1^{st} Order	2^{nd} Order	4^{th} Order
Traditional	5.434	3.959	2.986
Centroid	4.707	3.589	2.913
MST	4.664	3.666	2.907
MST$_a$	**4.639**	**3.575**	**2.891**

5.4 Estimating Lossless Performance with Entropy

Computing the edge weights of the graph using the actual byte totals of the difference images is computationally intensive. The difference images must be computed, compressed, and stored. This requires the compression algorithm to be run which incurs a significant cost. In the case of the MST and MST$_a$ strategies, this results in $O(n^2)$ calls to the compression algorithm. Using entropy measures for the edge weights can reduce the time required to construct the graph by requiring only $O(n)$ calls to the compression algorithm to compress the chosen difference images. To further reinforce the higher cost of compressing a difference image over finding its entropy, the actual computation time required to complete each action for 50 difference images from the GOES set was measured. Using JPEG2000 to compress 50 difference images required 29 seconds, while the entropy computations only required 3 seconds. The entropy represents the potential of compression for each difference image, but it is not clear how well entropy would predict the performance

Table 5.6: Joe entropy results.

Strategy	*Bits per Pixel*		
	1^{st} Order	2^{nd} Order	4^{th} Order
Traditional	6.436	4.598	2.986
Centroid	5.854	4.414	3.021
MST	**5.208**	**4.093**	**2.882**
MST$_a$	5.235	4.113	2.896

Table 5.7: GOES entropy results.

Strategy	*Bits per Pixel*		
	1^{st} Order	2^{nd} Order	4^{th} Order
Traditional	5.826	4.460	3.285
Centroid	5.923	4.672	3.476
MST	**4.934**	**4.221**	**3.283**
MST$_a$	4.977	4.249	3.302

of compression algorithms such as JPEG-LS and JPEG2000.

To explore how well the entropy measure could predict the performance of the compression algorithms, experiments were completed using entropy instead of the actual byte totals for the edge weights of the graph. For each image set, a spanning tree was generated for each of the four compression schemes presented in Section 4.5. Using first-, second-, and fourth-order entropy as a measure, the *bit per pixel* cost of encoding each image set is reported in Tables 5.4 to 5.8. The best results in each table are highlighted. The bit per pixel (bpp) value has been calculated based on the total number of images in the original set S_n. For schemes that employ an average image, the cost of the average image was added to the cost for the entire set. Similar to the results in Section 5.3, the Centroid strategy used only a single average image to form a single cluster.

In all cases, the results demonstrated that either the MST or MST$_a$ strategy predicts the best compression for each image set. Interestingly, the results from the Galway set (Table 5.4) predicted the MST strategy as the best yet the actual results in Table 5.2 show

Table 5.8: Combination entropy results.

Strategy	***Bits per Pixel***		
	1^{st} Order	2^{nd} Order	4^{th} Order
Traditional	6.131	4.662	3.379
Centroid	7.315	5.662	4.040
MST	**4.838**	**4.089**	**3.280**
MST$_a$	4.957	4.173	3.338

Table 5.9: Types of tree edges chosen by the MST$_a$ strategy using entropy as edge weights.

Edges	**Galway**	**Pig**	**Joe**	**GOES**	**Combo**
Zero	1	1	4	6	3
Average	24	280	8	0	0
Inter-Image	4	24	151	123	56

that the MST$_a$ strategy gives the optimal results. For the remaining image sets the entropy results predicted the same strategy as shown in the results in Section 5.3. This indicates that the entropy measures are reasonably accurate in predicting the performance of JPEG-LS and JPEG2000 for the entire set in most cases.

On the other hand, these results indicate that entropy measures do not always accurately predict which edges will be chosen in the MST. Using the MST$_a$ strategy, the types of tree edges chosen (Section 5.3) in the JPEG2000 results are shown in Table 5.3, while the entropy results are in Table 5.9. Note that the entropy results do not closely match the actual results. Specifically, the entropy results for the GOES and combination sets indicate that their minimum spanning trees would contain mostly inter-image edges. Yet in the actual results, the minimum spanning trees for both sets are predominately composed of zero edges. The results show that the MST predicted by entropy was not good enough when using either JPEG2000 or JPEG-LS for difference image compression. Also, these results indicated that the JPEG-LS and JPEG2000 algorithms may not be the best compression algorithms for difference images. This is also an issue in lossy compression, and

Table 5.10: Bitrate achieved by the traditional and MST_a strategies using JPEG2000 compression at various average RMSE values.

RMSE	Galway		Pig		Joe		GOES		Combination	
	Trad	**MST_a**	**Trad**	**MST_a**	**Trad**	**MST_a**	**Trad**	**MST_a**	**Trad**	**MST_a**
4.0	0.566	0.363	0.342	0.250	0.561	0.782	0.369	0.646	0.434	0.527
5.0	0.397	0.191	0.256	0.166	0.448	0.562	0.251	0.413	0.313	0.348
6.0	0.284	0.088	0.193	0.109	0.379	0.416	0.190	0.254	0.235	0.232

it will be explored further in the next section.

5.5 Lossy Compression Results

For lossy compression, the experiments used RMSE for the edge weight function as in [38, 39]. First, a spanning tree was computed for each of the four strategies (Section 4.5). Then, the difference images associated with the edges chosen in the spanning tree were compressed using an algorithm such as JPEG2000. The difference images were quantized to 8-bit images since not all the software packages chosen handled 9-bit images. A quantized difference image has pixel values in the range of $[-127, 127]$, but the image compression software used unsigned integer to store the pixel values for a gray scale image. Therefore, all the pixel values were shifted by 128 to create a range of $[0, 255]$. The results are given in graphs showing the bitrate and the average distortion (RMSE) in the reconstructed images for each image set. A curve in the graph that is lower and to the left of another curve indicates a better compression strategy.

Table 5.11: Bitrate achieved by the traditional and MST strategies using JPEG2000 compression at various average RMSE values.

RMSE	Galway		Pig		Joe		GOES		Combination	
	Trad	**MST**	**Trad**	**MST**	**Trad**	**MST**	**Trad**	**MST**	**Trad**	**MST**
4.0	0.566	0.709	0.342	0.587	0.561	0.877	0.369	0.715	0.434	0.625
5.0	0.397	0.434	0.256	0.437	0.448	0.639	0.251	0.480	0.313	0.438
6.0	0.284	0.272	0.193	0.332	0.379	0.510	0.190	0.301	0.235	0.317

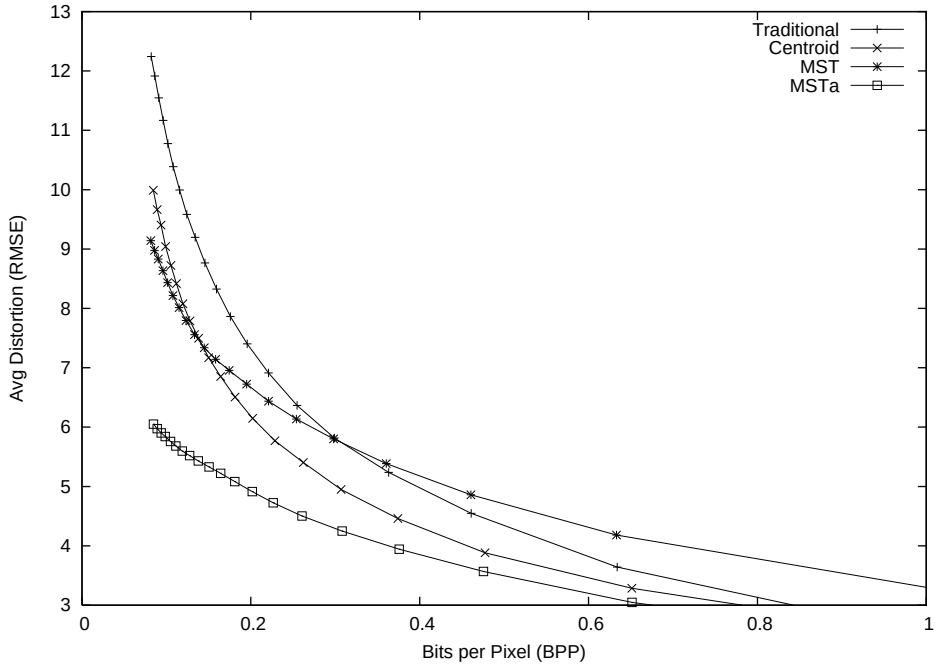


Figure 5.6: Lossy compression performance using JPEG2000 on the Galway set.

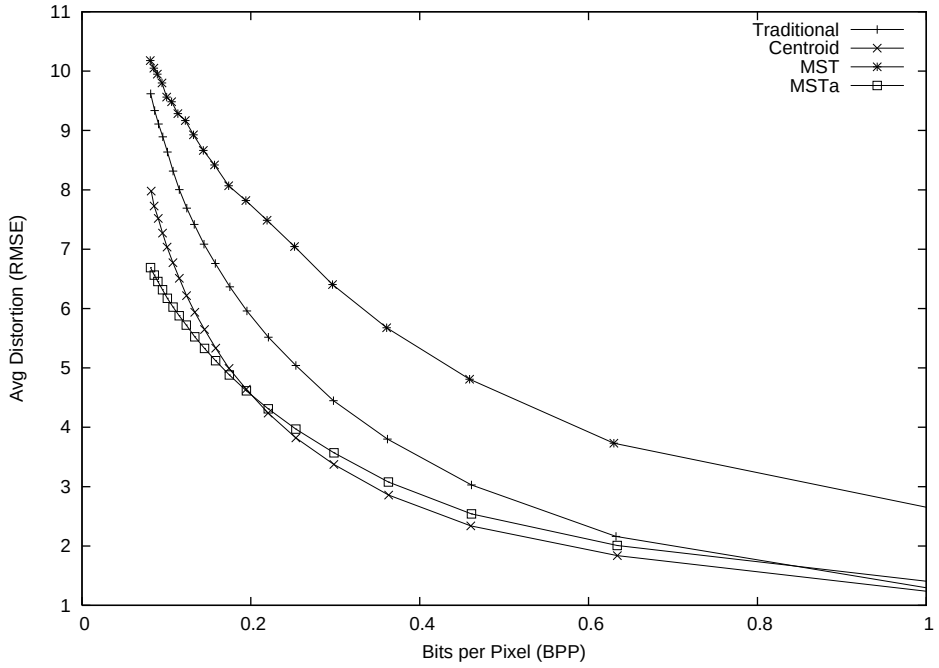


Figure 5.7: Lossy compression performance using JPEG2000 on the Pig set.

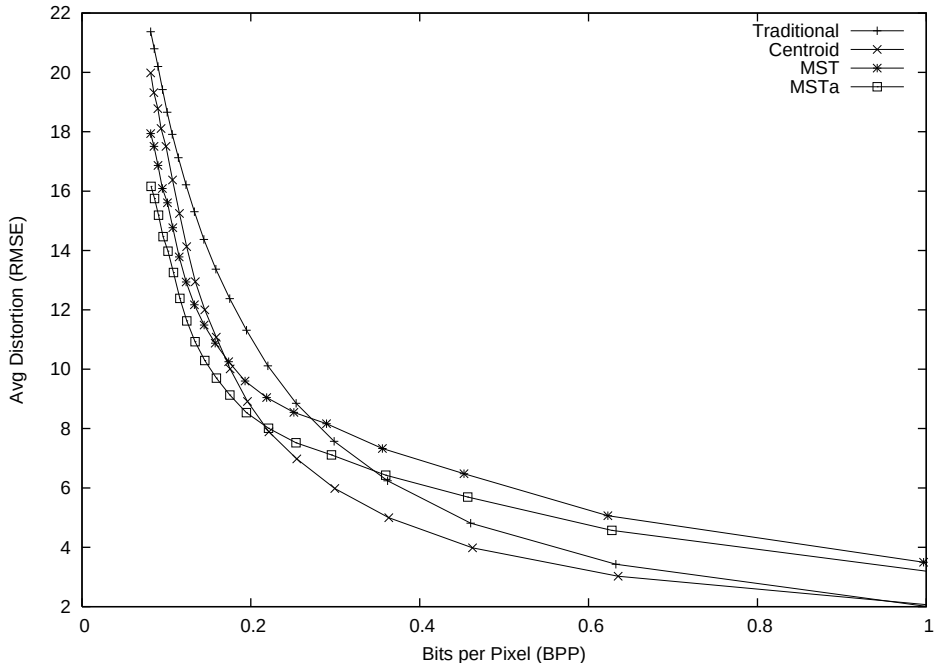


Figure 5.8: Lossy compression performance using JPEG2000 on the Joe set.

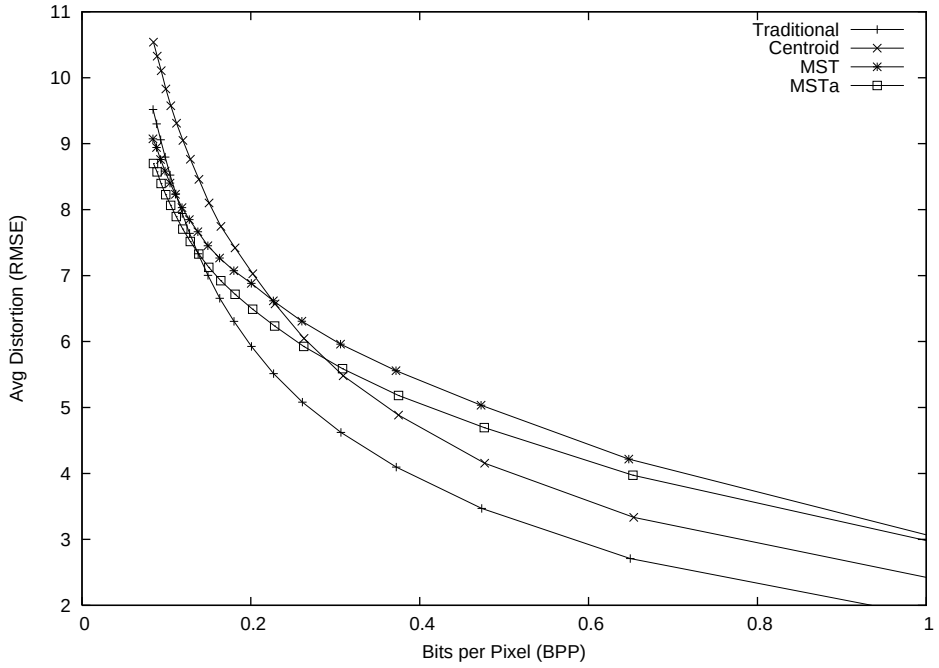


Figure 5.9: Lossy compression performance using JPEG2000 on the GOES set.

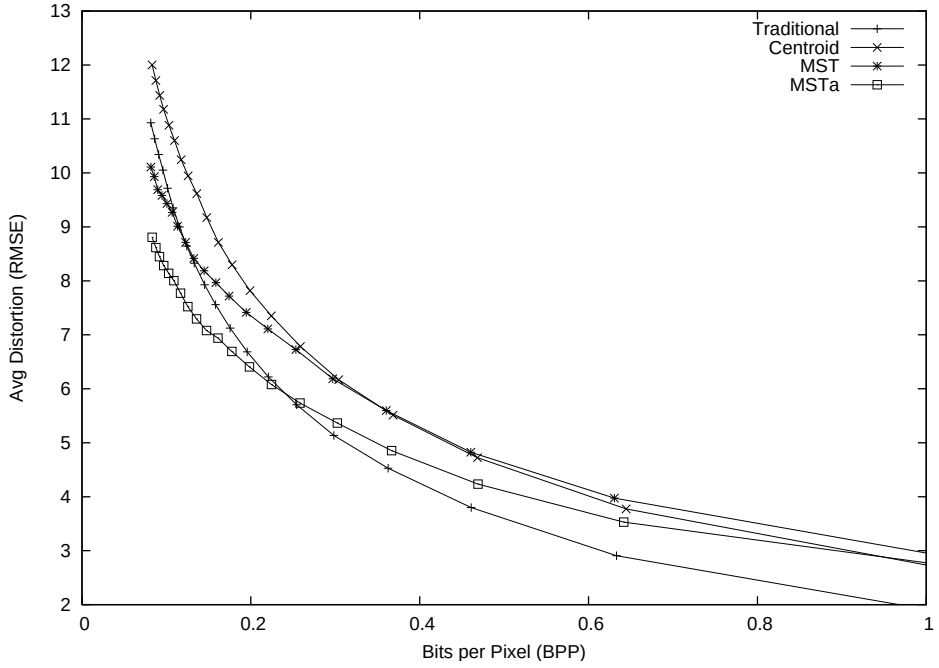


Figure 5.10: Lossy compression performance using JPEG2000 on the Combination set.

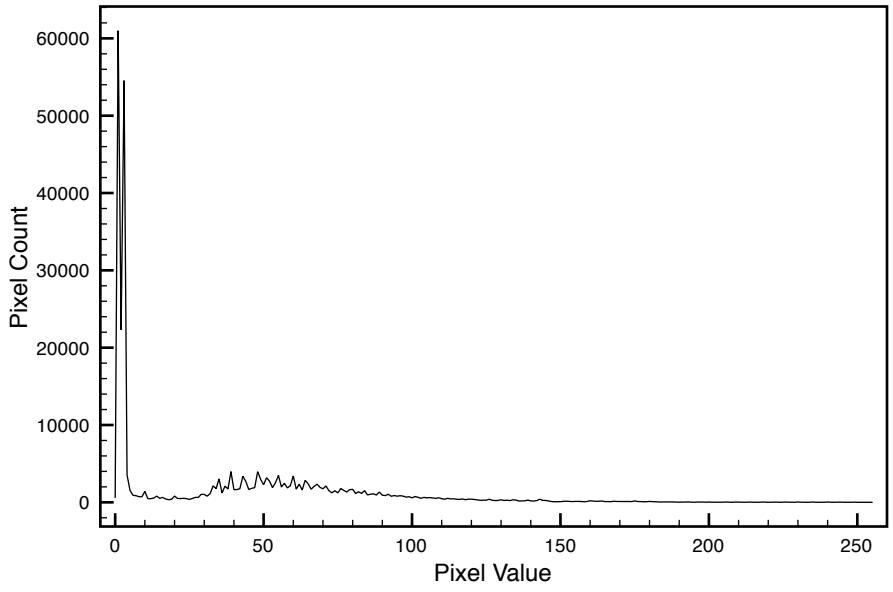
5.5.1 JPEG2000 Results

The initial lossy compression experiments used JPEG2000 as the underlying compression algorithm, and the results are summarized in Table 5.10. The table compares the bitrate between the traditional and MST_a strategies at various RMSE values. For the Galway and Pig sets, the MST_a strategy outperformed the traditional strategy and these results mirrored those predicted by the analysis. On the other hand, the results for the remaining three image sets did not reflect the predicted results. The complete results using JPEG2000 are shown in Figures 5.6 to 5.10.

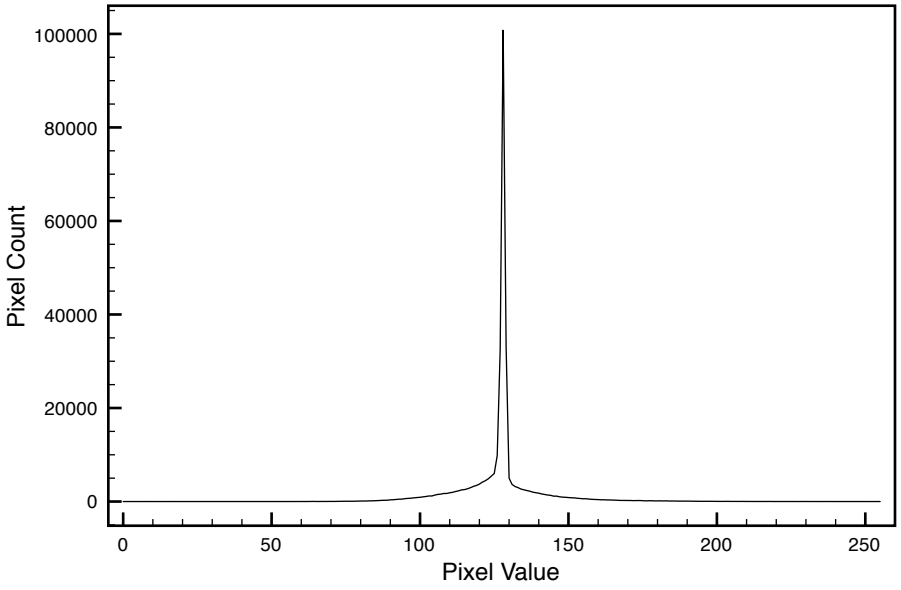
Figures 5.6 and 5.7 show that the MST_a algorithm outperformed the traditional strategy on Galway and Pig sets, as these two sets are known to form a tight cluster. On the other hand, the MST and MST_a strategies performed very poorly on the remaining three image sets compared to the other strategies. From the prediction results in Section 5.4, the MST strategy is best for these image sets as none of them form one tight cluster, but rather form multiple smaller clusters. From the MST strategy results in Table 5.11, it can be seen that the MST strategy actually performed worse than the traditional and MST_a strategies. The only exception occurred in the Galway set at an RMSE value of 6.0, where the MST strategy performed slightly better than the traditional strategy but not better than the MST_a strategy.

5.5.2 Wavelet Packets

To understand the discrepancy between the results predicted by RMSE and the actual compression results, the properties of the difference images were examined. Examining the histograms of a regular image and a difference image (Figure 5.11) shows that the data contained within each type of image is very different. The difference image contains



(a) Regular image



(b) Difference image

Figure 5.11: Histograms of a regular image and a difference image from the Pig set.

a large number of pixels with values close to 127, which corresponds to a difference of zero. Most of the information contained in a regular photographic image is found in the low pass coefficients (the LL quadrant in Figure 2.4), and JPEG2000 is tuned to compress this type of image. However, this is not necessarily true for a difference image, whose important data can be contained within the details. Using a wavelet packet algorithm, the basis chosen for a difference image can be very different (Figure 5.12) from the "pyramid basis" chosen by a wavelet compression algorithm (Figure 2.4). Therefore, a compression algorithm using wavelet packets should be more effective in compressing difference images than traditional wavelet algorithms such as JPEG2000.

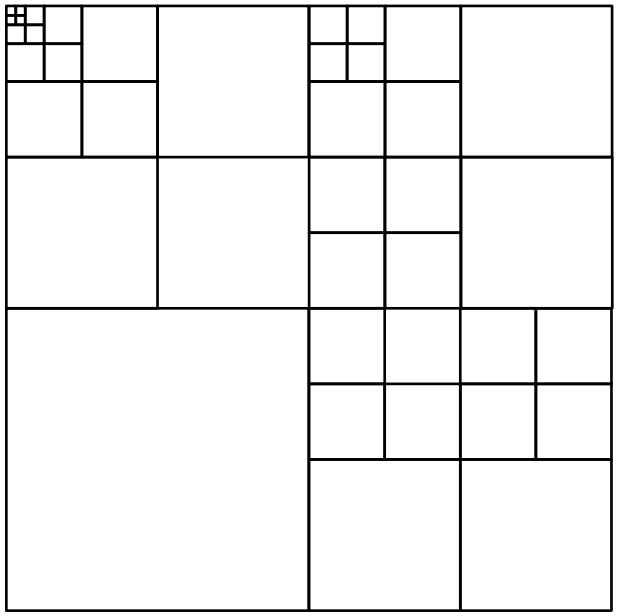


Figure 5.12: The best basis chosen for a difference image from the GOES set.

These results show that as the average distortion (RMSE) increased, the MST_a strategy performed better than the traditional strategy in Table 5.12. For example, in the Galway set the traditional strategy achieved a bitrate of 0.336 bpp while the MST_a strategy achieved a bitrate of 0.094 bpp at an average RMSE value of approximately 6.00 (a 72% reduction). The complete results are presented in Figures 5.13 to 5.17.

Table 5.12: Bitrate achieved by the traditional and MST$_a$ strategies using wavelet packet compression at various average RMSE values.

RMSE	**Galway**		**Pig**		**Joe**		**GOES**		**Combination**	
	Trad	**MST$_a$**	**Trad**	**MST$_a$**	**Trad**	**MST$_a$**	**Trad**	**MST$_a$**	**Trad**	**MST$_a$**
4.0	0.619	0.351	0.351	0.230	0.598	0.529	0.435	0.449	0.462	0.413
5.0	0.450	0.180	0.266	0.163	0.468	0.385	0.316	0.297	0.343	0.282
6.0	0.336	0.094	0.203	0.116	0.387	0.296	0.236	0.205	0.263	0.204

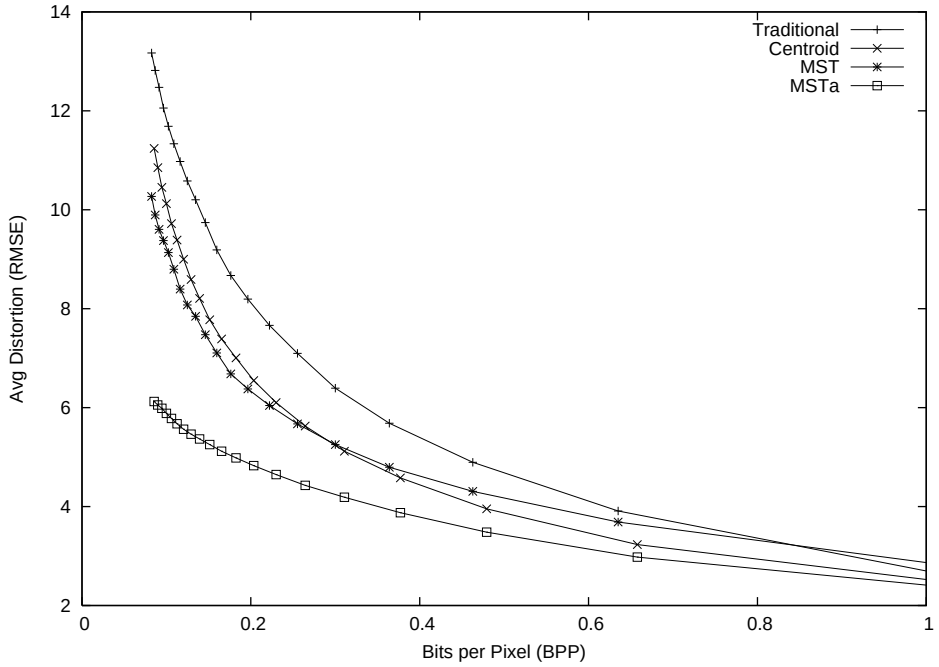


Figure 5.13: Lossy compression performance using wavelet packets on the Galway set.

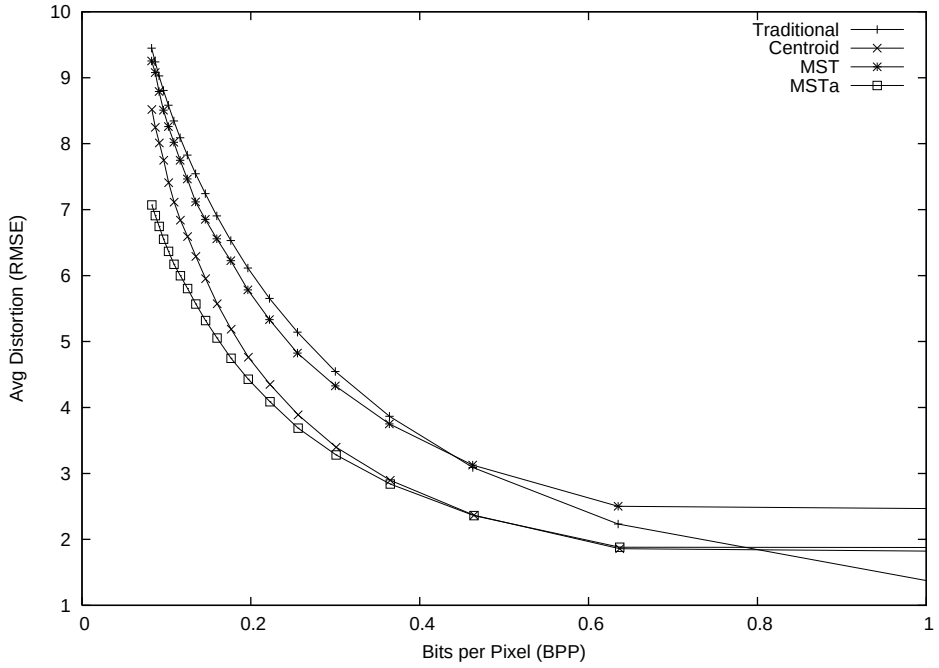


Figure 5.14: Lossy compression performance using wavelet packets on the Pig set.

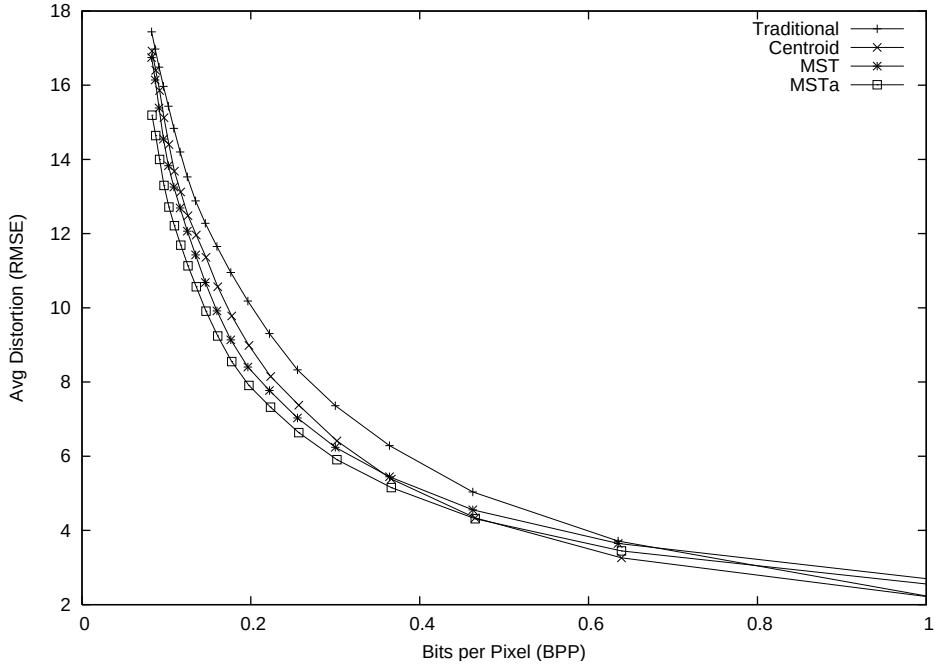


Figure 5.15: Lossy compression performance using wavelet packets on the Joe set.

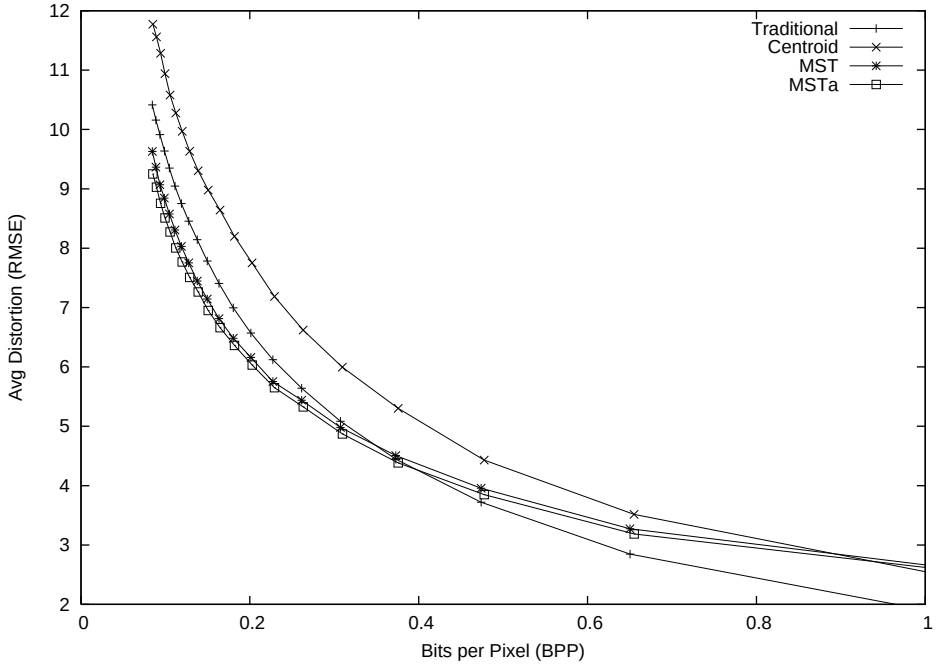


Figure 5.16: Lossy compression performance using wavelet packets on the GOES set.

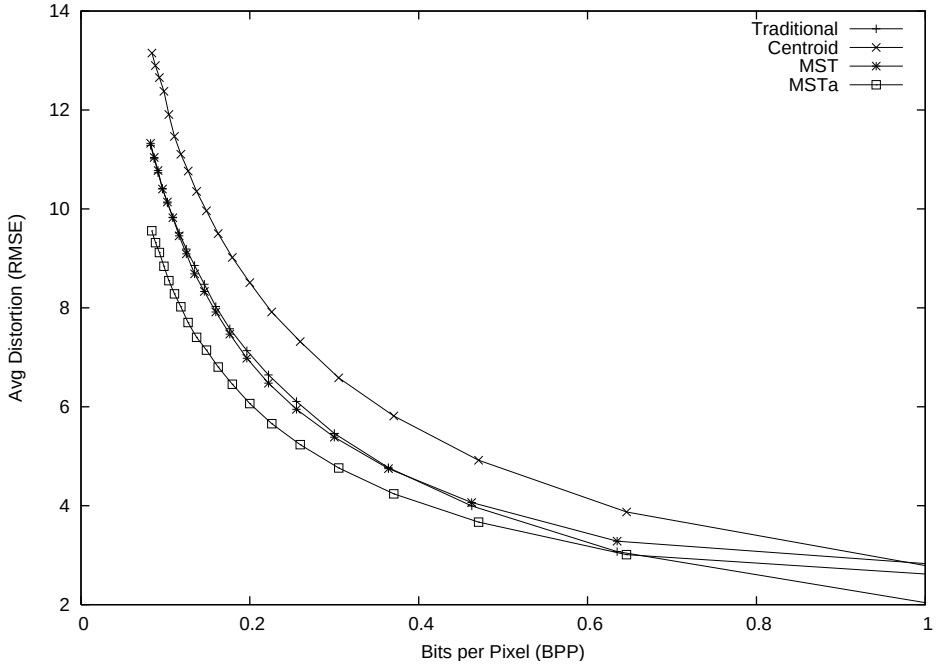


Figure 5.17: Lossy compression performance using wavelet packets on the Combination set.

From the results for both JPEG2000 and wavelet packets compression algorithms, three observations were made. First, the choice of the underlying compression algorithm affected the performance of the proposed scheme. The results showed that a wavelet packet algorithm compressed difference images much better than JPEG2000. The actual compression results using a wavelet packet algorithm reflect those predicted by the analysis. On the other hand, the JPEG2000 results (Section 5.5.1) did not perform as the RMSE edge weights predicted, and in many cases performed worse than the traditional strategy.

Second, the results show that an MST strategy was better than the traditional strategy when using a wavelet packet algorithm to compress the difference images. At lower bitrates, MST_a was better than all other compression strategies. This is clear in the Galway (Figure 5.13) and Pig (Figure 5.14) image sets, but it can also be seen in the Joe (Figure 5.15), GOES (Figure 5.16), and combination (Figure 5.17) image sets. The analysis indicated that there should be improvement as the average distortion decreases, but instead the results showed that the MST strategies got worse. This result was caused by the quantization of the difference images to 8-bit. Also, as the average RMSE decreased, the bitrate approached lossless compression (see Tables 5.1 and 5.2). In cases where the optimal lossless strategy was also achieved by a non-MST strategy, the quantization caused the MST strategies to perform worse.

Finally, the results comparing JPEG2000 and wavelet packet algorithms were not conclusive as other parts of the compression algorithms, such as the symbol encoder, were not considered. For example, the JPEG2000 results for the traditional strategy (Table 5.10) performed better than the wavelet packet algorithm for the traditional strategy (Table 5.12). Furthermore, in some cases the traditional strategy using JPEG2000 outperformed the MST and MST_a strategies using wavelet packets (Figure 5.21), while the analysis indicated that the MST or MST_a strategies should be no worse than the traditional or the Centroid strategies. Figure 5.18 to 5.22 show the complete results comparing the use of

the JPEG2000 algorithm for the traditional strategy and the wavelet packet algorithm for the MST and MST$_a$ strategies.

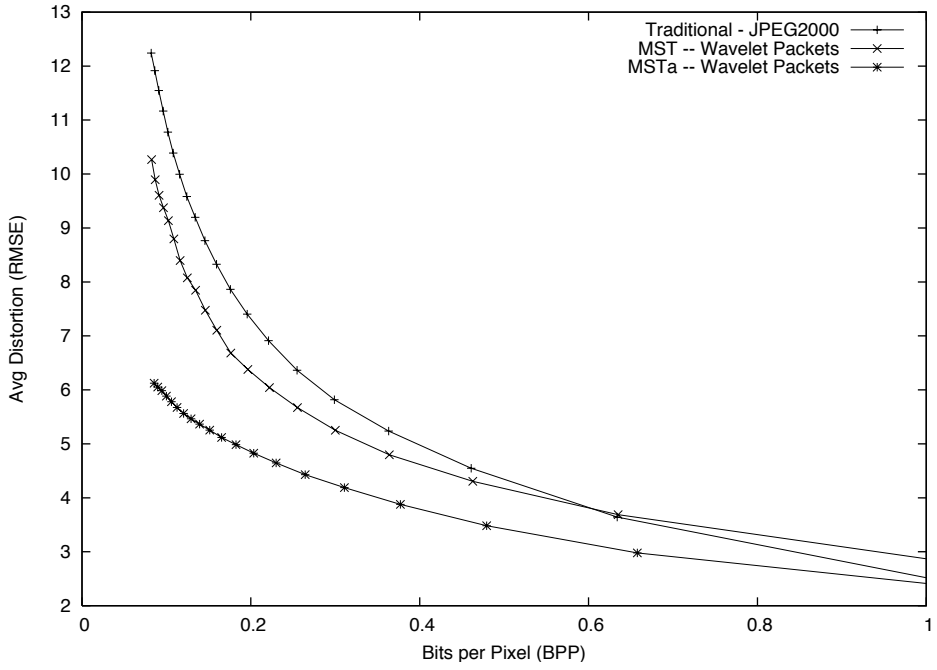


Figure 5.18: Lossy compression performance comparing JPEG2000 and wavelet packets on the Galway set.

5.5.3 9-bit Lossy Compression

Most of the information contained within a difference image is contained within the range of $[-127, 127]$. To avoid the quantization problem shown above, experiments were performed using an approach similar to the one used by Nielsen *et al.* [38, 39]. This method involved dividing the difference image into two images. One image contained the core information, which was pixels with a gray level value within the range of $[-127, 127]$. The second image contained the outlier values for those pixels that contained values outside of the core range.

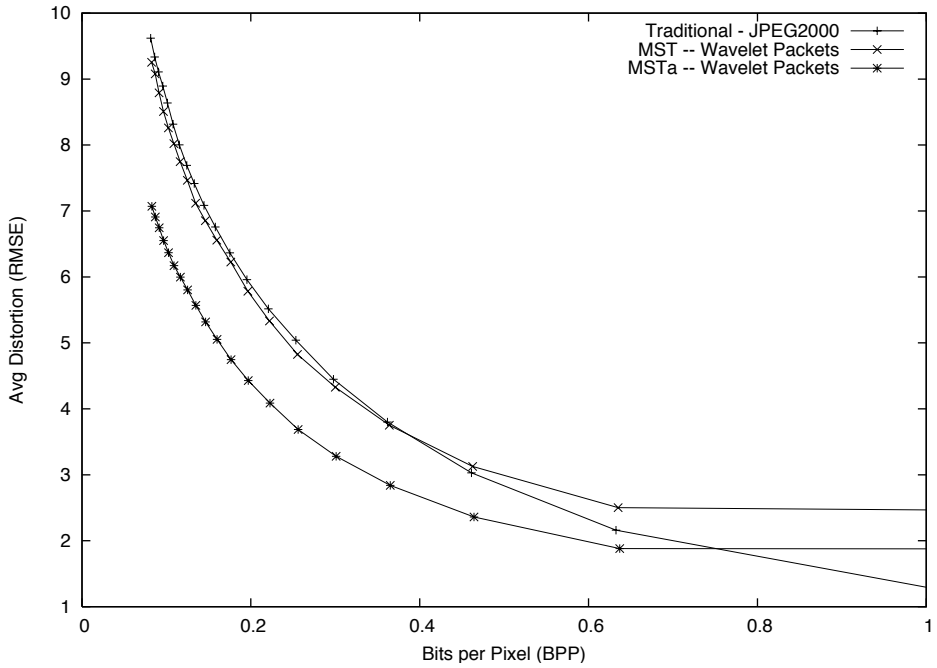


Figure 5.19: Lossy compression performance comparing JPEG2000 and wavelet packets on the Pig set.

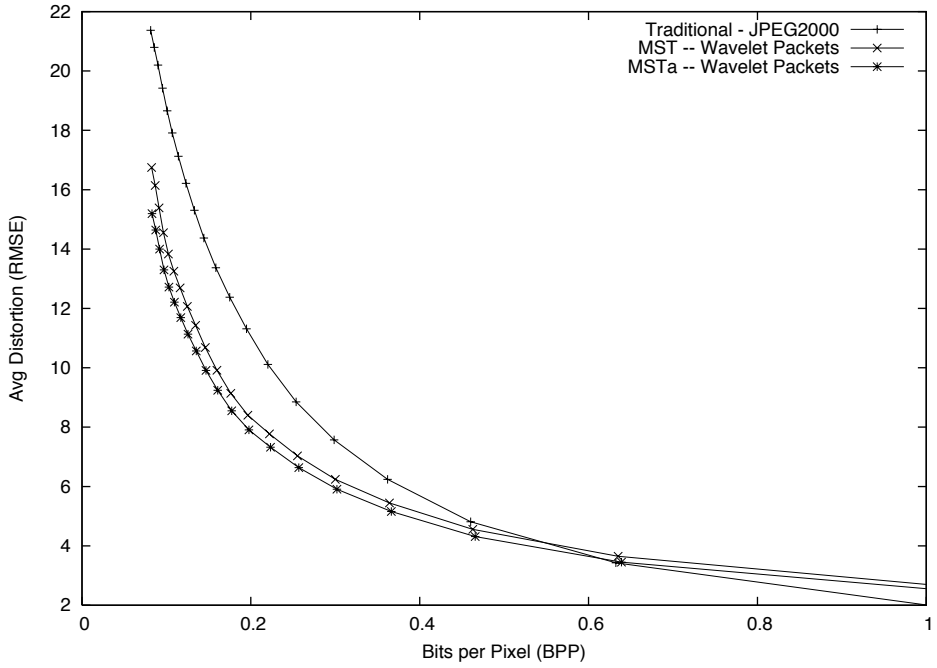


Figure 5.20: Lossy compression performance comparing JPEG2000 and wavelet packets on the Joe set.

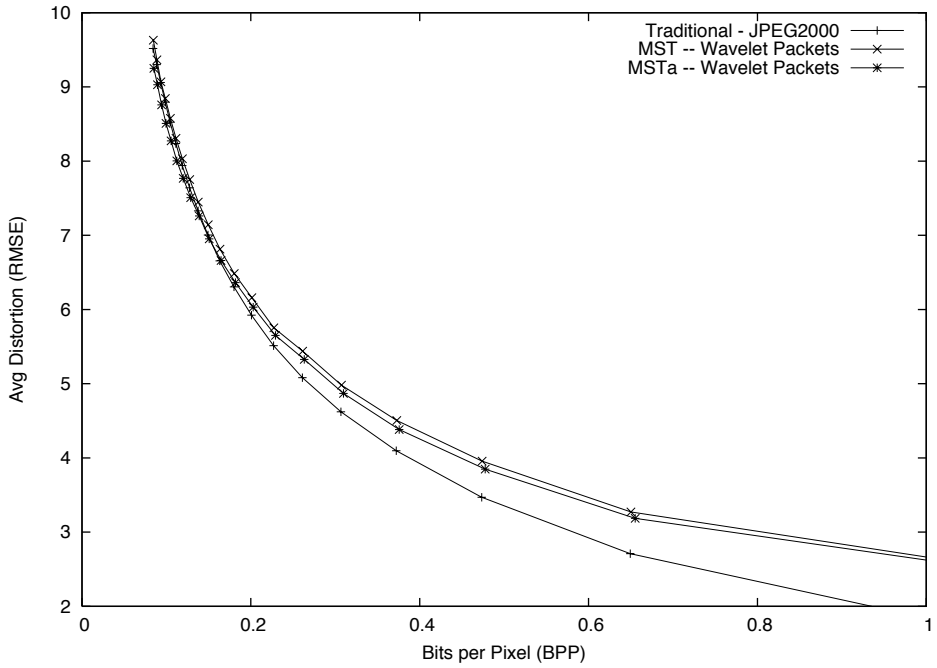


Figure 5.21: Lossy compression performance comparing JPEG2000 and wavelet packets on the GOES set.

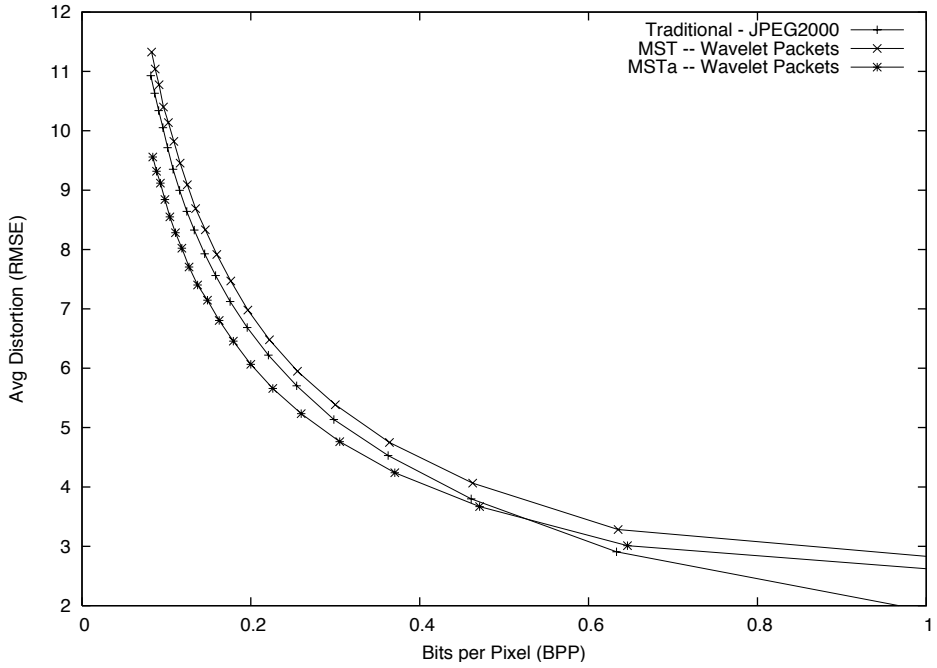


Figure 5.22: Lossy compression performance comparing JPEG2000 and wavelet packets on the combination set.

The results showed this method was not good for compressing the difference images as 9-bit images. Because the two images are correlated, the average distortion at particular bitrates was too high. To make this method of image splitting work properly, an analysis needs to be performed to allocate the appropriate bitrate for each image. Therefore, the best option is to use software that appropriately deals with 9-bit difference images. For these experiments, this option was not available in the software packages found, and the software chosen [33] was a closed project where no source code was available.

5.6 Summary

The theoretical framework proposed in Chapter 4 was implemented and tested. The software facilitated testing of lossless and lossy compression, and its interface allowed multiple underlying compression algorithms to be interchanged. The experiments were completed using five different image sets.

The lossless experiments were completed using the actual byte cost values of the difference images as the edge weight values in the graph. The JPEG2000 and JPEG-LS algorithms were each tested as the underlying compression algorithm. In all cases, the results showed that either the MST or MST$_a$ strategies always provided the best compression strategy for an image set.

The computation of edge weights using the actual byte values for lossless compression is expensive. In order to improve performance speed, experiments were completed using entropy for the edge weight value in the graph as it was faster to compute. The results showed that entropy was a good predictor for overall lossless performance of the framework but not as good for predicting which edges were chosen in the resulting MST.

The lossy experiments were completed using the RMSE value of the difference images as the edge weight values and the MST was computed for this graph. Then, the chosen

edges were compressed using the underlying compression algorithm. JPEG2000 and a wavelet packet algorithm were used as the underlying compression algorithms for these experiments. The initial lossy tests were done using JPEG2000 and the results did not reflect the predicted results. Further experiments were completed using a wavelet packet algorithm as the underlying compression algorithm, and the results followed the predicted results. Therefore, the choice of underlying compression algorithm for the framework was critical to the lossy compression performance of the proposed framework. Difference images have different properties than the original photographic images, and algorithms such as JPEG2000 may not be the best choice to compress difference images.

Chapter 6

Conclusion

In this thesis, an automatic compression scheme that adapts to any image set was investigated. Using a unified graph theoretical framework that allows comparison of all compression strategies that take advantage of inter-image redundancies between pairs of images, the performance of the proposed compression scheme was analyzed both theoretically and experimentally. Regardless of the properties of the image set, the framework automatically selects the optimal strategy for lossless compression or a near-optimal strategy for lossy compression.

Using this compression scheme, the optimal lossless compression strategy was computed and it is guaranteed to be no worse than any previously proposed strategy. In the lossy case, it was shown that computing the optimal strategy is difficult because of errors introduced by lossy compression. However, a strategy that is near optimal can be obtained and a performance guarantee was given. In some cases an improvement of up to 72% over traditional compression strategies was observed. The experimental results also showed that the choice of the underlying compression algorithm used to compress difference images is important. The results demonstrated that using a wavelet packet algorithm is more suitable than JPEG2000 for lossy compression of these types of images.

Since the proposed framework provides the optimal or a near optimal compression strategy for an image set, it is also useful for evaluating other image set compression strategies. The framework allows for a quantitative evaluation of other strategies in relation to the optimal or near optimal scheme. Furthermore, it allows for the study of the trade-off between compression performance and compression speed for an image set.

6.1 Limitations

The proposed framework focuses on improving the storage requirements for a set of images, but it is not effective for the transmission of an individual image over a network. For example, an image may be located far away from the root in the spanning tree. In order to transmit this image over a network, all the difference images along the path from the root to the desired image must be transmitted to correctly reconstruct the desired image.

To compress an image set, the proposed framework uses a complete graph when computing an MST. This is computationally expensive ($O(n^2)$ edge weight calculations), and it is not practical for very large image sets. But the proposed framework is still faster than using PCA to compress sets of images [35]. Furthermore, the framework assumes that image sets are static, and therefore it does not allow for dynamic updates of a compressed image set. The framework is not easily scalable to large dynamic image sets.

6.2 Future Directions

Image set compression offers many directions to explore for future research and a few potential ideas are presented below:

Compression algorithms for difference images.

Results in this thesis showed that a wavelet packet compression algorithm performed better than a wavelet compression algorithm for lossy compression. However, the wavelet packet algorithm used in the experiments for this thesis may not be the optimal algorithm for compressing difference images. More work is needed to understand the properties of difference images in order to further improve the performance of both lossless and lossy image set compression. Furthermore, an extensive study of suitable lossless compression

algorithms for difference images was not done in this thesis. Frajka and Zeger proposed a method to improve the compression of pairs of stereo images [15], and their method may be a good starting point to the more general difference images discussed in this thesis.

Multiple cluster set compression.

Previous works by other researchers suggest that compression performance may be improved further through the use of multiple clusters [27, 28, 39]. Using such a strategy, improved compression performance may be obtained on the Combination set and the GOES set presented in this thesis. A disadvantage to using multiple clusters is an increase in computational complexity added to the algorithm to classify images into clusters. On the other hand, using a multiple cluster strategy may allow for the optimal scheme to be approximated without constructing the complete graph. This could be achieved by ignoring certain edges between images in different clusters.

Automatic updating of an image set.

The proposed framework assumes that image sets are static, but many applications require dynamic updates to an image set. It is not clear how dynamic updates can be handled within the framework, but using dynamic data structures such as those proposed in [22] may allow this capability to be integrated into the framework.

Integration into a database system.

The storage and retrieval of digital images is an important problem, and this thesis is only concerned with reducing the storage requirements of image sets. One possible direction would be to explore how the inter-image relations discovered by the proposed framework can be used for image database systems that use content-based retrieval. For example, Nielsen and Li suggested that the MST structure is a good structure to be used

with content-based image retrieval [38].

Compression of color image sets.

Traditionally, image compression algorithms are presented using gray scale images. It is common to extend such algorithms to color images by treating each color component as a separate gray scale image and compressing them accordingly. In the case of set compression, color images may contain redundancy within their color components and any compression strategy would have to consider the relationship between color redundancy and inter-image redundancy.

Inter-image relationships between groups of images.

This thesis discusses the relationship between pairs of images. However, inter-image relationships among groups of more than two images may facilitate further improvements to set compression. To utilize such relationships, methods to discover and reduce the appropriate redundancies must be explored.

Bibliography

[1] M. Adams. JasPer project. http://www.ece.uvic.ca/~mdadams/jasper/.

[2] S. Ait-Aoudia and A. Gabis. A comparison of set redundancy compression techniques. *EURASIP Journal on Applied Signal Processing*, 2006:1–13, 2006.

[3] C. D. Aliprantis and O. Burnkinshaw. *Principles of Real Analysis*, pages 26–27. Academic Press, 1990.

[4] J.P. Cheiney and A. Touir. Fi-quadtree: A new data structure for content-oriented retrieval and fuzzy search. In *Proceedings of the International Symposium on Large /Spatial Databases*, pages 23–32, 1991.

[5] C.-P. Chen, C.-S. Chen, K.-L. Chung, H.-I. Lu, and G. Tang. Image set compression through minimal-cost prediction structures. In *Proceedings of the IEEE International Conference on Image Processing*, pages 1289–1292, 2004.

[6] H. Cheng. Partial encryption for image and video commmunication. Master's thesis, University of Alberta, 1998.

[7] H. Cheng and X. Li. On the application of image decomposition to image compression and encryption. In P. Horster, editor, *Communications and Multimedia Security II*, pages 116–127. Chapman & Hill, 1996.

[8] R. R. Coifman and M. V. Wickerhauser. Entropy-based algorithms for best basis selection. *IEEE Transactions on Information Theory*, 38(2):713–718, 1992.

[9] The JPEG Committee. JPEG 2000. http://www.jpeg.org/jpeg2000/index.html.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.

[11] I. Daubechies. *Ten lectures on wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1992.

[12] R. Diestel. *Graph Theory*. Springer-Verlag, third edition, 2005.

[13] Y. El-Sonbaty, M. Hamza, and G. Basily. Compressing sets of similar medical images using multilevel centroid technique. In *Proceedings of Digital Image Computing: Techniques and Applications*, pages 791–800, 2003.

[14] R. A. Finkel and J. L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4:1–9, 1974.

[15] T. Frajka and K. Zeger. Residual image coding for stereo image compression. *Optical Engineering*, 42(1):182–188, 2003.

[16] Galway.net. http://www.galway.net/webcam/images.shtml.

[17] B. Gergel, H. Cheng, and X. Li. A unified framework for lossless image set compression. In *Proceedings of the Data Compression Conference*, page 448, 2006.

[18] B. Gergel, H. Cheng, C. Nielsen, and X. Li. A unified framework for image set compression. In *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition*, pages 417–423, 2006.

[19] GOES project science. http://rsd.gsfc.nasa.gov/goes.

[20] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1993.

[21] P. Goupillaud, A. Grossmann, and J. Morlet. Cycle-Octave and related transforms in seismic signal analysis. *Geoexploration*, 23:85–102, 1984.

[22] M. R. Henzinger and K. King. Maintaining minimum spanning trees in dynamic graphs. In *Proceedings of the International Colloquium on Automata, Languages and Programming*, pages 594–604, 1997.

[23] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441,498–520, 1933.

[24] A. Jensen and A. la Cour-Harbo. *Ripples in Mathmatics, The Discrete Wavelet Transform*. Springer-Verlag, 2001.

[25] JoeTourist Infosystems. http://www.JoeTourist.net.

[26] G. Jomier, M. Manouvrier, and M. Rukoz. Storage and management of similar images. *Journal of the Brazilian Computer Society*, 6(3):13–25, 2000.

[27] K. Karadimitriou. *Set redundancy, the enhanced compression model, and methods for compressing sets of similar images*. PhD thesis, Louisiana State University, 1996.

[28] K. Karadimitriou and J. M. Tyler. The centroid method for compressing sets of similar images. *Pattern Recognition Letters*, 19(7):585–593, 1998.

[29] B. W. Kernighan and D. M. Ritchie. *The C programming language*. Prentice Hall Press, 1988.

[30] M. Kirby and L. Sirovich. Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):103–108, 1990.

[31] J. Knipe. A comparison of improved spatial and transform domain compression schemes. Master's thesis, University of Alberta, 1996.

[32] K. Kosmas Karadimitriou and J. M. Tyler. Min-max compression methods for medical image databases. *ACM Special Interest Group on Management of Data*, 26(1):47–52, 1997.

[33] F. Meyer, A. Averbuch, and J. Strömberg. Fast adaptive wavelet packet image compression. *IEEE Transactions on Image Processing*, 9(5):792–800, 2000.

[34] R. Murenzi. *Wavelets*. Springer Berlin, 1998.

[35] Y. S. Musatenko and V. N. Kurashov. Correlated image set compression system based on new fast efficient algorithm of Karhunen-Loeve transform. In C.-C. J. Kuo, S.-F. Chang, and S. Panchanathan, editors, *Proceedings of The International Society for Optical Engineering: Multimedia Storage and Archiving Systems III*, pages 518–529, October 1998.

[36] H. Nagata and H. Tanaka. Estimate of total volume of medical data in a year in Japan. In *Proceedings of the APAMI and CJKMI-KOSMI Conference*, pages 117–121, 2003.

[37] Netpbm. `http://netpbm.sourceforge.net`.

[38] C. Nielsen and X. Li. MST for lossy compression on image sets. In *Proceedings of the Data Compression Conference*, page 463, 2006.

[39] C. Nielsen, X. Li, and K. Abma. Methods of grouping similar images for compression coding. In *Proceedings of the International Conference on Computer Vision (WCAC'05)*, pages 93–99, Las Vegas, June 20–23, 2005.

[40] X. Qi and J. M. Tyler. A progressive transmission capable diagnostically lossless compression scheme for 3D medical image sets. *Information Sciences*, 175(3):217–243, 2005.

[41] X. Qi, J. M. Tyler, and O. S. Pianykh. Integer wavelet transformations with predictive coding improves 3D similar image set compression. In H. H. Szu, D. L. Donoho, A. W. Lohmann, W. J. Campbell, and J. R. Buss, editors, *Proceeding of The International Society for Optical Engineering: Wavelet Applications VIII*, pages 238–249, March 2001.

[42] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16(2):187–260, 1984.

[43] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423 and 623–656, July and October 1948.

[44] J. M. Shapiro. An embedded hierarchical image coder using zerotrees of wavelet coefficients. In *Proceedings of the Data Compression Conference*, pages 214–223, 1993.

[45] E. Shusterman and M. Feder. Image compression via improved quadtree decomposition algorithms. *IEEE Transactions on Image Processing*, 3(2):207–215, 1994.

[46] A. Skodras, C. Christopoulos, and T. Ebrahimi. The JPEG2000 still image compression standard. *Pattern Recognition Letters*, 22(12):1337–1345, 2001.

[47] P. Strobach. Quadtree-structured recursive plane decomposition coding of images. *IEEE Transactions on Signal Processing*, 39(6):1380–1397, June 1991.

[48] B. Stroustrup. *The C++ Programming Language (Special 3rd Edition)*. Addison-Wesley Professional, 2000.

[49] R. Tashakkori. *Medical Image Set Compression Using Wavelet and Lifting Combined with New Scanning Techniques*. PhD thesis, Louisiana State University, 2001.

[50] TIFF libraries and utilities. `http://www.remotesensing.org/libtiff`.

[51] P. N. Topiwala, editor. *Wavelet Image and Video Compression*. Kluwer Academic Publishers, 1998.

[52] M. A. Turk and A. P. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.

[53] M. Vassilakopoulos and Y. Manolopoulos. Dynamic inverted quadtree: a structure for pictorial databases. *Information Systems*, 20(6):483–500, 1995.

[54] M. Vassilakopoulos, Y. Manolopoulos, and N. Economou. Overlapping quadtrees for the representation of similar images. *Image and Vision Computing*, 11(5):257–262, 1993.

[55] J. Wang and H. Yan. Form image compression using template extraction and matching. In *Proceedings of Visual Information Processing*, December 2000.

[56] M. J. Weinberger, G. Seroussi, and G. Sapiro. The LOCO-I lossless image compression algorithm: Principles and standardization into JPEG-LS. *IEEE Transactions on Image Processing*, 9(8):1309–1324, 2000.