Design and analysis of biased run-length coding methods and their applications in lossless compression of bi-level ROI maps in hyperspectral images

Amir Leon Liaghati

Follow this and additional works at: https://louis.uah.edu/uah-dissertations

Recommended Citation

This Dissertation is brought to you for free and open access by the UAH Electronic Theses and Dissertations at LOUIS. It has been accepted for inclusion in Dissertations by an authorized administrator of LOUIS.
DESIGN AND ANALYSIS OF BIASED RUN-LENGTH CODING METHODS AND THEIR APPLICATIONS IN LOSSLESS COMPRESSION OF BI-LEVEL ROI MAPS IN HYPERSPECTRAL IMAGES

by

AMIR LEON LIAGHATI

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in The Department of Computer and Electrical Engineering to The School of Graduate Studies of The University of Alabama in Huntsville

HUNTSVILLE, ALABAMA

2016
In presenting this dissertation in partial fulfillment of the requirements for a doctoral degree from The University of Alabama in Huntsville, I agree that the Library of this University shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by my advisor or, in his/her absence, by the Chair of the Department or the Dean of the School of Graduate Studies. It is also understood that due recognition shall be given to me and to The University of Alabama in Huntsville in any scholarly use which may be made of any material in this dissertation.

Amir Leon Liaghati

3/19/16
(date)
DISSERTATION APPROVAL FORM

Submitted by Amir Leon Liaghati in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering and accepted on behalf of the Faculty of the School of Graduate Studies by the dissertation committee.

We, the undersigned members of the Graduate Faculty of The University of Alabama in Huntsville, certify that we have advised and/or supervised the candidate of the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Electrical Engineering.

Dr. W. David Pan 3/18/16 (Date) Committee Chair

Dr. Craig Newborn 3/18/16 (Date)

Dr. Mark Tillman 3/18/16 (Date)

Dr. B. Earl Wells 3/18/16 (Date)

Dr. Dongsheng Wu 3/18/2016 (Date)

Dr. Ravi Gorur 3/30/16 (Date) Department Chair

Dr. Shankar Makkingam 3/31/16 (Date) College Dean

Dr. David Berkowitz 4/12/16 (Date) Graduate Dean
ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

Degree Doctor of Philosophy College/Dept. Engineering/Electrical and Computer Engineering

Name of Candidate Amir Leon Liaghati

Title Design and Analysis of Biased Run-Length Coding Methods and their Applications in Lossless Compression of Bi-Level ROI Maps in Hyperspectral Images

This work addresses efficient lossless compression of binary images. To this end, we designed several novel compression techniques, including one method known as the biased run-Length coding method. In this method, we first partition a binary image into equally sized blocks. We then convert the binary pixels within each block into a block symbol. In contrast to conventional approaches where all symbols are run-length coded, our method run-length codes only the most probable block symbols, followed by Huffman coding on the run-lengths. The other less probable block symbols will be coded with a separate Huffman code. Tests on NASA’s AVIRIS dataset showed that this method could provide significant improvements over various binary image compression techniques (including those based on JBIG2 and lossless JPEG 2000 standards) on regions-of-interest (ROI) maps of hyperspectral images. Furthermore, we provided a detail analysis on the biased run-length coding method using statistical models. The analytical results agreed very well with the results on empirical data. Lastly, we introduced a modified model based on Markov random fields, which can generate a large number of binary images iteratively. Simulation
results showed that the biased run-length coding method significantly outperformed the arithmetic code and JBIG2 method. This study also gave rise to a dual biased run-length method, which can provide further compression gains on images with a larger number of foreground objects.

Abstract Approval: Committee Chair

Dr. W. David Pan

Department Chair

Dr. Ravi Gorur

Graduate Dean

Dr. David Berkowitz
ACKNOWLEDGMENTS

First and foremost, I would like to thank my wonderful family wholeheartedly for their unconditional love and support throughout my life.

I would like to express my deepest appreciation to Dr. Pan. Over the past five years Dr. Pan has been one of the most influential person in my academic research. With his guidance I start innovating and developing new ideas that could improve the existing international standards. He has motivated me to patent my new ideas as well. To date, I have authored five patents, and several trade secrets of which some are in the area of data compression. He has been very enthusiastic about research and was able to steer my interest in the same research topic. He has been very understanding around my work schedule for which I am very thankful.

I would also like to express my sincere gratitude to my committee members Dr. Wells, Dr. Tillman, Dr. Newborn, and Dr. Wu for their time and insightful comments.
# TABLE OF CONTENTS

List of Figures x

List of Tables xiii

Chapter

1 Introduction 1

1.1 Binary Images 1

1.2 Conventional Distance Coding of Binary Images 3

2 Background 5

2.1 Interval Generation in Distance Coding 6

2.2 Adaptive Internal Generation 8

2.3 Discussions 12

3 Improved Distance Coding of Binary Images by Run Length Coding of the Most Probable Interval 15

3.1 Distribution of Sequence of intervals 16

3.2 The Proposed Method 17

3.2.1 Run-length of intervals with value being 1 17

3.2.2 Huffman coding for the run-lengths 21

3.2.3 Generation of the modified sequence of interval values 22
3.2.4 Huffman codes for the modified sequence of interval values . 23
3.2.5 Header information and direction Bits . . . . . . . . . . . . . 24
3.2.6 Decoder . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
3.2.7 Lossless Check . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
3.3 Simulation Results . . . . . . . . . . . . . . . . . . . . . . . . . . . 27
3.4 Conclusion . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31

4 An Efficient Method for Lossless Compression of Bi-level ROI Maps

of Hyperspectral Images 34
4.1 Hyperspectral Imaging . . . . . . . . . . . . . . . . . . . . . . . . . . 34
4.2 The Biased Run-Length Coding Method . . . . . . . . . . . . . . . 38
4.3 Simulation Results and Discussions . . . . . . . . . . . . . . . . . . . 42
4.4 Conclusions . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 45

5 Symbol Blocking and Biased Run Length Coding: Analysis and Ap-
plication in Compression of Hyperspectral Maps 47
5.1 Analysis of Biased Run-length Method on ROI binary Maps of Hyp-
spectral Images . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 47
5.1.1 Distribution of block symbols . . . . . . . . . . . . . . . . . . . 49
5.2 Mathematical Model for Biased Run-Length Coding Method . . . 53
5.2.1 Geometric distributions . . . . . . . . . . . . . . . . . . . . . . . 54
5.2.2 Markov model for distribution of the RLs . . . . . . . . . . . . . . 56
5.2.3 Construction of the new modified sequence . . . . . . . . . . . . 60
5.3 Results . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 62
5.4 Entropy Analysis of Biased Run-Length Coding ............... 65
5.5 Conclusion ....................................................... 65

6 Evaluation of The Biased Run-Length Coding Method Generated
by a Modified Ising Model ........................................... 68
6.1 Markov Random Field ........................................... 69
   6.1.1 Ising model ............................................... 70
6.2 Modified Ising Model ........................................... 73
   6.2.1 Proposed modified Ising model ......................... 73
   6.2.2 Results .................................................... 75
6.3 Dual Biased Run-Length Coding ................................. 77
6.4 Results on DB-RL .............................................. 78
6.5 Conclusion ....................................................... 79

7 Conclusion ............................................................. 86
7.1 Conclusion ......................................................... 86
7.2 Suggestions for Future Work .................................. 88

REFERENCES ............................................................. 90
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Image compression diagram example</td>
</tr>
<tr>
<td>2.1</td>
<td>A $5 \times 5$ binary image. (Left) A horizontal scan yields the interval sequence $[2, 3, 1, 1, 3, 2, 1, 4, 1, 1, 1, 1, 1]$. (Right) A vertical scan yields $[2, 3, 1, 1, 1, 1, 1, 3, 1, 5, 2, 1, 2]$.</td>
</tr>
<tr>
<td>2.2</td>
<td>A $10 \times 10$ image partitioned into four $5 \times 5$ blocks, on which scan-direction decision is made adaptively.</td>
</tr>
<tr>
<td>2.3</td>
<td>(Left) The direction bits for each of the blocks in Fig. 2 and (Right) the sequence of intervals for each of the blocks.</td>
</tr>
<tr>
<td>2.4</td>
<td>The encoder of the block-based adaptive interval-generation method.</td>
</tr>
<tr>
<td>3.1</td>
<td>(Left) Grayscale image of an asteroid; (Right) Binary image obtained after thresholding.</td>
</tr>
<tr>
<td>3.2</td>
<td>Distribution of the interval values.</td>
</tr>
<tr>
<td>3.3</td>
<td>The proposed encoder.</td>
</tr>
<tr>
<td>3.4</td>
<td>Block diagram for calculating the run-lengths of interval with value being 1.</td>
</tr>
<tr>
<td>3.5</td>
<td>The histogram of the run-lengths.</td>
</tr>
<tr>
<td>3.6</td>
<td>Huffman codeword lengths for the run-lengths of the most probable interval value. While only the 50 smallest possible run-lengths were shown, the maximum codeword length is 12 bits for all run-lengths.</td>
</tr>
<tr>
<td>3.7</td>
<td>Length of Huffman codewords used for the largest 50 run-lengths.</td>
</tr>
<tr>
<td>3.8</td>
<td>Diagram for modified sequence of intervals generation.</td>
</tr>
<tr>
<td>3.9</td>
<td>Lengths of the codewords for the 50 smallest possible hybrid interval values.</td>
</tr>
<tr>
<td>Section</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5.4</td>
<td>Entropy comparison between 1D and $3 \times 3$ block based for binary ROI images</td>
</tr>
<tr>
<td>5.5</td>
<td>Diagram for the Biased Run Length coding method using statistical model.</td>
</tr>
<tr>
<td>5.6</td>
<td>Transition Probabilities between block symbol 511 and other blocks.</td>
</tr>
<tr>
<td>5.7</td>
<td>Size comparison result between biased method and entropy of the original distribution for $\beta = 0.5$, different $\sigma^2$ and $q$.</td>
</tr>
<tr>
<td>5.8</td>
<td>Size comparison result between biased method and entropy of the original distribution for $\beta = 0.8$, different $\sigma^2$ and $q$.</td>
</tr>
<tr>
<td>5.9</td>
<td>Entropy calculation for Biased RL coding method.</td>
</tr>
<tr>
<td>5.10</td>
<td>Bitrate using entropy calculation for Biased RL, and conventional RL.</td>
</tr>
<tr>
<td>6.1</td>
<td>Binary images generated by the Ising Metropolis model.</td>
</tr>
<tr>
<td>6.2</td>
<td>An 8-niighborhood pixel (left) and a 4-bit neighborhood pixels for the proposed modified Ising model.</td>
</tr>
<tr>
<td>6.3</td>
<td>Binary images generated using the modified Ising model.</td>
</tr>
<tr>
<td>6.4</td>
<td>Bitrate result using modified Ising model.</td>
</tr>
<tr>
<td>6.5</td>
<td>Dual Biased Run-Length coding scheme</td>
</tr>
<tr>
<td>6.6</td>
<td>Ising model for higher iterations.</td>
</tr>
<tr>
<td>6.7</td>
<td>Bitrate result using the modified Ising model.</td>
</tr>
<tr>
<td>6.8</td>
<td>Bitrate result using the modified Ising model.</td>
</tr>
<tr>
<td>6.9</td>
<td>Entropy comparison using 1D and 2 D for different iterations.</td>
</tr>
<tr>
<td>TABLE</td>
<td>PAGE</td>
</tr>
<tr>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>4.1 Bitrate (bits/pixel, or bpp) of the compressed files.</td>
<td>44</td>
</tr>
<tr>
<td>5.1 Generation of probability distribution of RLs using Markov model.</td>
<td>58</td>
</tr>
<tr>
<td>5.2 511 blocks vs. New merged blocks</td>
<td>60</td>
</tr>
</tbody>
</table>
Dedication

To my wonderful parents

Dad, You have been my number one hero. Words cannot describe how much I miss you. I keep going as I always promised you, and I live to make you proud as long as I live.

Mom, without you I wouldn’t be the person I am today. Thank you for your endless love and support throughout my life.

To my lovely sisters

Mahsa, You are the best sister in the entire world, and you are the hardest working person I know. Thank you for all your inspiration and support in my life.

Yalda, as the oldest you have always inspired me to do better. Thank you for your endless love.
Keep going.

—Dad
1.1 Binary Images

Over the past decade, the need of image compression has evolved exponentially. In order to reduce the data bandwidth as well as storage needs, more efficient compression techniques are needed. Images generated in the fields of medicine, aerospace, and defense tend to be extremely large and need to be compressed more efficiently. There might also be redundant information in data or images. Image compression is the process of removing irrelevant, nonessential information to reduce the size of the data storage [1, 2]. Image compression falls into two categories lossless and lossy. Sometimes removing the nonessential information will not cause any loss of information in the image, therefore there is no error in compression techniques and the original image will be recovered perfectly. This type of image compression falls into the lossless category. Fig. 1.1 shows an example diagram of lossless compression in space applications. If there are losses in an image then the original image will not be recovered after the compression process. This type of compression falls into lossy category. In many cases, the loss of information is not noticeable to human eyes; therefore losses are acceptable. As a result, using lossless compression, the quantity of data and the
size of the storage are reduced even further. Also, more images can be transmitted over the same bandwidth or the required bandwidth becomes smaller. Depending on the application, lossy and lossless compression can be applied. This research aimed at improving the lossless compression of binary images. Binary images, also known as bi-level images, are images with pixel values of 1’s and 0’s. Although high resolution grayscale images are more useful for human eyes, binary images might be more suitable for some applications, where color or grayscale images may be converted to binary images. Examples include visual sensor networks [3], object detection [4], edge detection [5], morphological [6], and Region of Interest (ROI) [7] among many others.

Existing methods for compression of binary images include Huffman coding [8, 9], run-length coding [10,11], arithmetic coding [12], geometric-based coding [13,14],

**Figure 1.1:** Image compression diagram example
etc. In addition, international standards for binary image compression have been developed, such as JBIG [15], JBIG2 [16] and JPEG 2000 [17]. Also, distance coding methods [18–21] have been proposed to compress binary images efficiently. Some of the most recent binary image compression can be found in the references [22–27].

1.2 Conventional Distance Coding of Binary Images

In conventional distance coding methods [11, 12], one traverses an image in a certain scan (e.g., raster scan) order and calculates the intervals (distances) between the previous and current occurrence of the same symbol. Coding the sequence of these intervals might lead to a higher compression than coding the original binary image directly. Due to the binary nature of the symbols in the image, the locations of either 0s or 1s completely determine the original binary image. Thus the sequence of intervals of either symbol “0” or “1” can be used by the decoder to reconstruct the original sequence without any loss of information. Nevertheless, there are many possible scan orders in which one can traverse the symbols in an image, thereby leading to a large number of candidate sequences of intervals. An attempt to search through all these possible scan orders so as to find an order that yields a sequence of intervals with the largest compressibility would be computationally prohibitive. Hence, the challenge to improving the distance coding method would be to determine scan orders that can offer better compression than the conventional raster-scan order with only a modest increase of its computational complexity. To this end, in next chapter, we introduced a computationally efficient block-based interval generation
method, which scans through symbols in orders adaptive to local statistics, thereby leading to significantly higher compression than the raster-scan order.

This dissertation is arranged into six chapters. Chapter 2 presents the background in our earlier work, also known as the adaptive interval generation method, and its advantages over interval generation code in distance coding. Chapter 3 will discuss an improved distance coding method by introducing the new idea of run-length coding the most probable interval, and it will show improved results over previous distance coding methods and other binary code compressors. Chapter 4 will introduce a novel method for compression of binary images by run-length coding the most probable block symbol, also known as the biased run-length coding method. Chapter 5 will show a detail analysis of the biased run-length coding method and will propose a new mathematical model based on the Markov model. Lastly, Chapter 6 proposes a novel model to generate more binary images to evaluate the biased run-length coding method. This work also introduced a dual-biased run-length coding method for more efficient binary image compression.
CHAPTER 2

BACKGROUND

The following chapter is from our published work in\textsuperscript{1} [28]. We proposed an adaptive method for more efficient distance-coding of binary images. The proposed method partitions the image into blocks where the interval sequences of zeros or ones can be calculated, as opposed to the conventional method where intervals are calculated by following a fixed scan order. In the proposed method, one can adaptively choose either a horizontal or a vertical scan within a block, depending on criteria based on entropy values. The resulting intervals tend to have lower entropies than the conventional non-adaptive methods, thereby allowing for higher compression than distance coding using a lossless codec. Our simulations on various test images demonstrated that (i) the proposed method achieved significantly higher compression than the non-adaptive distance coding method; (ii) the proposed method can be used as an efficient preprocessor of a lossless coder, offering higher compression than directly coding on the original images.

\textsuperscript{1}©2015 SPIE
2.1 Interval Generation in Distance Coding

In conventional distance coding methods, one scans through the pixels in a binary image following the raster-scan order, i.e., from left to right and from top to bottom. The intervals between the current and previous occurrences of the same symbol (either “0” or “1”) are calculated. In the following, without loss of generality due to the complementary nature of the only two possible symbols, we focus on symbol “1” in generating the intervals. Fig. 2.1(left) shows the sequence of intervals for an example 5 × 5 image, where except for the first entry in the sequence, which is an absolute location index of the first occurrence of “1”, all other entries represents the difference (interval) of the location indices of two succeeding 1’s. For instance, an interval “3” means that there are two consecutive zeros between the previously seen and the current ones (i.e., “1001”) along the scan path; and “1” means that there are two consecutive ones, and so on. The distance coding method removes the constraint of the chain code methods [29–35] by allowing non-contiguous 0’s or 1’s in the scan path. Once the interval sequence is obtained, the problem of coding the original binary image is changed into the problem of coding the intervals, which are not bi-level any more. It also follows that the sequence of intervals thus generated will be dependent upon the scan path. As shown in Fig. 2.1 (right), a vertical raster scan generates a different sequence of intervals from that obtained by using the horizontal scan. In general, the number of orders in which one can traverse all the pixels in an image is equal to N!, where N is the number of symbol “1” in the image. While different sequences of intervals would have varying symbol statistics, and therefore
varying degrees of compressibility, it would be impractical computationally to sort through all these possible sequences, especially for large images, to find one that would offer the largest possible compression. Furthermore, a significant number of extra bits have to be spent on coding the side information about an irregular scan order (other than say, either a horizontal or a vertical raster scan) so that the decoder can reconstruct the original image losslessly. Hence, it would be desirable to have the capability to explore a large number of possible ways to scan through the image so as to identify sequences of intervals that are “easy” to compress, while limiting the number of possible scan orders in order to reduce the amount of side information. To this end, we introduce a novel adaptive interval generation method as discussed in the following section.

Figure 2.1: A $5 \times 5$ binary image. (Left) A horizontal scan yields the interval sequence $[2, 3, 1, 1, 3, 2, 1, 4, 1, 1, 1, 1, 1]$. (Right) A vertical scan yields $[2, 3, 1, 1, 1, 1, 1, 3, 1, 5, 2, 1, 2]$. 
Figure 2.2: A $10 \times 10$ image partitioned into four $5 \times 5$ blocks, on which scan-direction decision is made adaptively.

2.2 Adaptive Internal Generation

In order to explore the local statistics of the image, we partition the image into equally spaced blocks. Since the number of pixels in each block tends to be much smaller than the original image, the problem of exploring possible scan orders within a block becomes more tractable. Therefore, rather than scanning the image pixel by pixel following a fixed raster scan order as in conventional distance coding methods, the proposed method scans through the image on a block-by-block basis. Furthermore, to limit the amount the side information and keep the computational complexity low, we consider only two possible scan orders within each block: either horizontal raster scan or vertical raster scan. For each block, we choose between these two scan orders...
adaptively according to a criterion based on entropy comparison. More specifically, within each block, we traverse the 1’s following both horizontal and vertical directions. As a result, two sequences of intervals are obtained, one for the horizontal scan, the other for the vertical scan. Two empirical entropy values (one for horizontal and the other for vertical scan) are calculated based on the sequences of intervals. Here the empirical entropy is given as

\[
H(R) = - \sum_{k=1}^{L} P(r_k) \log P(r_k),
\]

(2.1)

where \( L \) is the number of distinct symbols \( r_k \) (with \( 1 \leq k \leq L \)) in the interval sequence, and \( P(r_k) \) is the probability of each symbol \( r_k \) occurring in the sequence. In principle, the smaller the entropy \( H(R) \), the higher the compression one could possibly achieve on the sequence. Hence, within each block, we will choose a scan direction that has lower entropy. If the horizontal direction is chosen then the direction bit is set to “1” for this block; otherwise, a direction bit of ‘0’ will be set. See Fig. 2.2 for an illustrative example, where we assume that after entropy comparisons as discussed above, it was decided that block 1 should be scanned vertically since the resulting sequence of intervals would have lower entropy. Thus a direction bit of “0” is assigned for this block. Also we assume that the remaining three blocks were found to be more suitable for horizontal scans after entropy comparisons. As a result, direction bits of 1’s are assigned for blocks 2, 3, and 4 (see Fig. 2.3 (left)). Thus for this 10 × 10 image, only a small amount of side information (4 direction bits) is required. Once the optimal scanning direction for each block is decided upon, the sequence of intervals for
each block can be generated (see Fig. 2.3 (right)). The blocks will then be traversed in
a raster-scan order (from left to right, then from top to bottom) such that the sequence
of intervals for each block can be concatenated to form a sequence of intervals for the
entire image. The resulting sequence will start with the absolute location index of
the first “1” in the first block of the image, followed by the sequence of intervals of
1s. Note that during a transition from the previous block to the current block, the
interval of the first “1” in the current block is calculated as the distance from the last
“1” in the previous block. For example, in see Fig. 2.3, the interval of the first “1” in
Block 3 is shown as “4” since it is 4 pixels away from the last “1” in the previous block
(Block 2). The proposed method, as summarized in Fig. 4, generates a sequence of

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 1</td>
<td>Block 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block 3</td>
<td>Block 4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Block 1: | [3 1 2 1 2 1 1 1 3 1 1 2 2 3] |
| Block 2: | [1 1 1 2 1 1 2 2 1 1 1 4 4 2 1] |
| Block 3: | [4 2 1 1 1 1 2 6 2 1 1 1 1 1] |
| Block 4: | [1 1 1 1 3 1 2 1 1 1 1 2 2 2 6] |

**Figure 2.3:** (Left) The direction bits for each of the blocks in Fig. 2 and (Right) the sequence of intervals for each of the blocks.

intervals and direction bits which will be combined into a data file and compressed by
a data compression utility such as bzip2. Since the original binary image can also be
directly compressed by the same data compressor, the proposed method can be used
as an “add on” component. In order to “synchronize” the direction bit sequence and
the interval sequence, information such as the block size and the number of blocks
in the image need also be conveyed to the decoder such that the intervals can be
Figure 2.4: The encoder of the block-based adaptive interval-generation method.
converted back to the location indices of the 1s in the corresponding block within the image. This extra information was incorporated into the header of the compressed files in our actual implementation of the proposed method. Simulation results given in the next section showed that higher compression can be achieved on binary images by using the proposed adaptive interval generation method.

2.3 Discussions

We proposed a block-based interval generation method that can adaptively choose between horizontal and vertical scans for each block. Simulation results in this preliminary study demonstrated that the proposed method achieved significantly higher compression than the non-adaptive distance coding method. Also, the proposed method can be used as a computationally efficient preprocessor of a lossless coder such as bzip2, offering higher compression than directly applying bzip2 on the original images. The proposed method can be applied not only to the compression of both binary and non-binary images (e.g., greyscale and color images, which can be converted to binary images by concatenating their bit planes), but also to the compression of binary data such as Internet traffics. While the use of block-based scan in horizontal or vertical directions for many image compression applications has been considered previously [36], the novelty of our proposed method is that the scan direction can be chosen within each block adaptively based on entropy values reflecting the local statistics. This means that the scan direction is not fixed like the method proposed in [37], where only one scan direction is followed by all the blocks within an image, based on which the best scanning path will be chosen ultimately. Similarly, in
the block-based scan method used in the 2-D integer Haar wavelet transform [38], the
scanning within each block is pre-determined and the best scan directions for these
regions are picked only after applying the chosen scanning direction. In contrast, our
method is adaptive and it will choose scan directions on the fly and there is no need
for a predefined scan order for each block. On the other hand, while adaptive scanning
methods exist (e.g. those based on space-filling curves in image compression in [36]),
most scanning patterns are two dimensional, thereby requiring side information (for
scan direction) of at least 2 bits per pixel for each direction. In contrast, our pro-
posed method has an overhead of only 1 bit per block to indicate either horizontal or
vertical scan. In terms of computational complexity, our method has the advantage
of using distance coding which reduces significantly the number of symbols to code
than coding the original binary image. Nevertheless, the proposed method suffers
from the limitation that the block size cannot be changed according to the varying
local statistics of the image once the symbol scan starts. Therefore, further adaptive
gains can potentially be achieved by allowing block size and shape to change on the
fly. However, the overhead associated with non-uniform blocks has to be taken into
account. Also, with the understanding that bzip2 was not intended to be used as
image compression algorithm, it was used in our work mainly to demonstrate that
the proposed method is able to reduce the entropy of the source and thus increase the
“compressibility” of the original binary image. In the next chapter, we will discuss
the improved distance coding method and will compare our method with JPEG-2000
for binary still images, and JBIG and JBIG2 for text images. Further development of
the technique is likely to result in a competitive alternative to the existing standard approaches, with much lower computational complexity.
CHAPTER 3

IMPROVED DISTANCE CODING OF BINARY IMAGES BY RUN LENGTH CODING OF THE MOST PROBABLE INTERVAL

The following chapter is from our published work in\textsuperscript{1} [39]. In this chapter we propose a new method to improve our previous work on efficient distance-coding of binary images, where we compressed a binary image by applying the bzip2 lossless data compressor on a sequence of intervals, which represent the distances between identical source symbols – either zeros or ones for binary images. Motivated by the observation that a majority of intervals tends to be one, we propose to run-length code this most probable interval independently from the rest of the intervals. Separate Huffman coding tables were used to code the run-lengths of the most probable interval versus other intervals. Consequently, this hybrid coding scheme allows a fraction of one bit to be assigned to the most probable interval on average, as opposed to at least one bit per interval without run-length coding, thereby contributing to about 17\% improvement on the compression ratios on some test images.

\textsuperscript{1}©2015 IEEE
Figure 3.1: (Left) Grayscale image of an asteroid; (Right) Binary image obtained after thresholding.

3.1 Distribution of Sequence of intervals

An 8-bit grayscale image of an asteroid is used as an illustrating example (Fig. 3.1 left). Thresholding operation was applied to the grayscale image to obtain a binary image based on the threshold $T$:

$$I(i,j) = \begin{cases} 
1, & \text{if } I(i,j) \geq T. \\
0, & \text{otherwise.}
\end{cases} \quad (3.1)$$

where $I(i,j)$ is the pixel values at point $(i,j)$. Next, the block-based adaptive interval-generation method was applied to obtain the sequence of intervals and the direction bits. The histogram of the interval sequence is shown in Fig. 3.2. In most of the binary images we studied, the interval value of '1' occurs more than 90% of time, intervals greater than one are much less likely. This is to be expected since either black or white pixels tend to cluster in binary images. Since 90% of the intervals have values being '1', the entropy of the sequence tends to be very small, thereby allowing for efficient
compression of the original binary image. If we apply Huffman coding directly on the interval sequence, the most probable interval '1' will be assigned one bit per interval. In the case of binary asteroid image, a total of 60,386 interval values equals to '1', accounting for about 91.43% of the entries in the interval sequence. Applying the Huffman code directly on this sequence will generate at least 60,386 bits for those most probable intervals with value being '1', which is source of inefficiency due to its failure to capture the correlations between these identical entries in the sequence.

3.2 The Proposed Method

The idea is to treat the intervals of '1's separately from the rest of the intervals in the sequence with values greater than 1. More specifically, we count the run-lengths of those most probable intervals (with value being '1') and apply Huffman code on the run lengths. For all other intervals with values greater than one, we concatenate them and the pointers to the most probable intervals to form a modified sequence of intervals, before applying a separate Huffman code to this modified sequence. Fig. 3.3 shows the block diagram for the proposed method.

3.2.1 Run-length of intervals with value being 1

The procedure for calculating the run-lengths of '1's in the sequence of intervals is shown in Fig. 3.4. It starts by checking whether the current interval is '1', followed by a check on the next interval. If the next interval is also a '1', the counter variable “count” will increase by 1. If this is the last interval in the sequence, then the count will be stored in an array called “Count” where all the run-lengths will be stored and
that is the end of the loop. If the next interval is not a ‘1’, then the count will be stored in the array “Count”, and the procedure repeats. If the current interval is not equal to ‘1’, then we will check for the next interval, and if the next interval is the last interval, it will check whether it is a ‘1’. If it is a ‘1’, a value ‘1’ will be stored in the current array Count. The case where the last interval is a ‘1’ is a special case. If the interval is not the last interval, the process is repeated until the last interval in the sequence is visited. Also, if the next interval is the last and is not a ‘1’, the procedure will be terminated without counting any ‘1’s. The array “Count” now contains all

Figure 3.2: Distribution of the interval values.
Figure 3.3: The proposed encoder.
Figure 3.4: Block diagram for calculating the run-lengths of interval with value being 1.

An example distribution of run-lengths of intervals with values being ‘1’s is shown in Fig. 3.5. We will code the run-lengths with Huffman codes. Since the distribution of the run-lengths depends on the source statistics of the original binary image, other compression methods such as Golomb codes can be considered for the coding of run-lengths as well. In this preliminary study, we trained the Huffman code by using the
sequence of run-lengths derived from a single image (Fig. 3.1), the coding table thus obtained would be optimum for this particular sequence only. In practice, we can train the Huffman code based on a set of sequences of run-lengths derived from a large set of test images to achieve some robustness of the coding tables.

### 3.2.2 Huffman coding for the run-lengths

Fig. 3.6 shows the Huffman codeword lengths for some run-lengths. It can be seen that as run-lengths exceed 6, the codeword lengths become smaller than the run-
lengths. This means run-length coding allow for compression gains on the most probable interval whenever there is more than 6 consecutive intervals with value being 1. For example, when run-length is 48, 10 ² bits are required to code 48 consecutive intervals with value being 1, which requires at least 48 bits to code if applying Huffman coding directly on the sequence of intervals without using run-length coding. Fig. 3.7 shows the lengths of the codewords used to code the largest 50 run-lengths for the example image, where a large run-length of 2,989 requires only 12 bits to code, representing a huge saving.

3.2.3 Generation of the modified sequence of interval values

For those intervals with value great than 1, we combine them into a modified sequence of intervals, among which one flag bit is interspersed. Fig. 3.8 shows the block diagram for generating the modified sequence of interval values. For example, given an original sequence of interval values being [2, 1, 1, 1, 1, 5], the modified sequence is [2, 1, 5], where the ‘1’ in the middle is indeed a pointer to an entry in the sequence of run-lengths as described previously in a subsection. The entry pointed to has a value of 4, meaning four consecutive intervals with identical value being 1 in the original sequence of interval values. This modified sequence of hybrid interval values can be unambiguously reconstructed by the decoder, since any entry equal to 1 means it is a flag bit, which is to be replaced by the actual number of '1' intervals as indicated by

²Note: In actual implementation, a total of 11 bits would be required, i.e., 10 bits of Huffman codeword, plus 1 flag bit that serves as the pointer to the run-lengths in the modified sequence of block symbols.
the corresponding run-lengths, whereas an entry greater than 1 means it is already an interval value.

### 3.2.4 Huffman codes for the modified sequence of interval values

We also Huffman coded the modified sequence of interval values to achieve data compression, with some example codeword lengths shown in Fig. 3.9, where the interval value being 1 means it is a flag bit, which requires exactly one bit codeword.
Figure 3.7: Length of Huffman codewords used for the largest 50 run-lengths.

3.2.5 Header information and direction Bits

The compressed image file should have a header, which contains the block sizes used in the block-based interval-generation distance coding method, as well as overhead bits used to keep track of the optimal scanning direction chosen for each block can. As an option, the header can be compressed and decompressed losslessly. However, it is advisable to compress the direction bits, which increase significantly as block sizes become smaller.
3.2.6 Decoder

As shown in Fig. 3.10, the decoder proceeds by scanning through the modified sequence of hybrid interval values, including values greater than 1 and the flag bits. If a flag bit of ‘1’ is encountered, the decoder will refer to the corresponding decoded “Count” array which contains the run-lengths of ‘1’ intervals, and expand the flag bit into the actual number of interval values as determined by the corresponding run-
Figure 3.9: Lengths of the codewords for the 50 smallest possible hybrid interval values.

lengths. On the other hand, if values greater than 1 are encountered, the decoder knows that these values are original interval values and thus no expansion operation is needed. As a result, the decoder can reconstruct the original sequence of interval values, based on which the entire image can be reconstructed block by block.
3.2.7 Lossless Check

Lossless check was performed at the end of the decoder to make sure the reconstructed binary image is exactly the same as the original image pixel by pixel.

3.3 Simulation Results

To test the proposed method, a set of 100 binary images were obtained by converting grayscale images to binary images using thresholding from the first 100 frames of the video sequence called “Table Tennis” [40]. Fig. 3.11 shows 9 selected frames for the purpose of demonstration. As shown in Fig. 3.12 and Fig. 3.13, the proposed hybrid coding scheme (the “New Method”) improves the compression of the block-
Figure 3.11: Diagram for reconstructing the original sequence of intervals.

based (method without run-length coding) in [28] called the “Old Method” by about 17%. Similar improvements were also obtained for some other binary images used in the test, thereby demonstrating the advantage of coding the most probable interval separately using run-length coding. We have also applied the new method on different types of images and compare its performance against the JBIG2 standard. As an example, we try to compress binary image used in soccer player detection [41]. Fig. 3.14 (left) shows a video frame from the World Cup 2014 final game between Germany and Argentina [42]. Next, we used a basic color detection method to detect
the players in blue inside the field. Then the image is converted to a binary image, where only players in blue are shown in black as foreground, the rest of the image is converted to white as background as shown in Fig. 3.14 (right). We then applied both our method and the JBIG2 standard codec on the image. 4 different block sizes were used to compress the image using our proposed method, and every one shows improvement over JBIG2 as shown in Fig. 3.15. The block size of $9 \times 12$ provides the best improvement (up to 23%) over the JBIG2 standard codec [43]. Binary images can also be very useful in medical imaging. For instance, the exact location of a tumor

**Figure 3.12:** Compressed file sizes.
Figure 3.13: The overall compressed file sizes for 100 binary images.

Figure 3.14: A video frame (720 x 1272) selected from the scene of the Final Game of World Cup 2014 (left), The binary version (right).
in the brain is extremely important and need to be identified as shown in Fig. 3.16 (left) [44]. This image is an MRI of brain with tumor region. Only for demonstration reason, we have converted the grayscale image of the MRI to a binary image (see Fig. 3.16 (right)). 2 different block sizes were used for comparison. The compression result is shown in Fig. 3.17, block size $51 \times 39$ shows the best compression over JBIG2 by 22.44\% improvement.

### 3.4 Conclusion

We proposed a new method to improve our previous work on distance-coding of binary images, by run-length coding the most probable interval value independently
Figure 3.16: MRI image of a brain tumor (left); the binary image (right).

Figure 3.17: Compressed file sizes (the New Method vs. JBIG2 codec).
from the rest of the interval values. Separate Huffman coding tables were used to
code the run-lengths of the most probable interval versus other intervals. Conse-
quently, this hybrid coding scheme allows a fractional bit to be assigned to the most
probable interval value on average, as opposed to at least one bit per interval without
run-length coding, thereby contributing to significant improvement on the compres-
sion efficiency. In next chapter, we will introduce a new method also known as biased
run-length coding method for more efficient compression of binary images, as well as
comparison with coding efficiencies of the standard codec such as JBIG, JBIG2, and
JPEG 2000.
CHAPTER 4

AN EFFICIENT METHOD FOR LOSSLESS COMPRESSION OF
BI-LEVEL ROI MAPS OF HYPERSPECTRAL IMAGES

4.1 Hyperspectral Imaging

![Sample dataset](image)

(a) Spectral band 30.  (b) 17 ROI’s identified.

Figure 4.1: Sample dataset (“indian_pines_corrected”) and ROI’s.

The following chapter is from our published work in\textsuperscript{1} [45]. While one can achieve very large size reduction on a hyperspectral image dataset by preserving only some regions-of-interest (ROI’s), the bi-level map that describes the locations of the

\textsuperscript{1}$\copyright 2016$ IEEE
Figure 4.2: Individual ROI maps (ROI pixels are shown in white while non-ROI pixels in black). Note: ROI 0 (Fig. 4.2a) can be viewed as the background, which contains all the pixels outside the other 16 ROI’s.
ROI pixels tend to defy efficient compression due to the somewhat “random” nature of ROI pixel locations. To this end, we proposed a novel method for lossless compression of these ROI maps. In this method, we first partitioned a bi-level map into equally sized blocks. We then converted the bi-level pixels within each block into a block symbol. Based on the observation that the most probable blocks tend to contain either all zeros or all ones, we chose to run-length code these most probable block symbols before applying Huffman code in order to achieve high compression, whereas we applied a separate Huffman code on other less probable block symbols. Thus, this biased run-length coding method differs from conventional approaches where all symbols are run-length coded. Tests on NASA’s AVIRIS dataset showed that the proposed method could provide significant improvements over other bi-level image compression techniques (including JBIG2 and lossless JPEG 2000) on the ROI maps.

Hyperspectral imaging techniques have been used in a wide array of applications such as aerospace, astronomy, agriculture, mineralogy and surveillance [46–65]. A hyperspectral image is often organized as a three-dimensional dataset with two spatial dimensions and one spectral dimension. As an example, see Fig. 4.1(a), which is the 30th spectral band (out of a total of 220 bands) from NASA’s Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) hyperspectral image dataset [66]. Due to the enormous amount of data produced by hyperspectral imagers, data compression techniques are crucial to data size reduction, which leads to more efficient data storage and transmission, especially for real-time applications. Since most applications of hyperspectral imaging cannot tolerate any inaccuracy due to data loss, lossless compression is generally preferred over lossy methods. Unfortunately, state-of-the-
art lossless compression methods often offer an order-of-magnitude lower compression than their lossy counterparts on hyperspectral images. However, if only a small region in the image is deemed “important”, as determined by a certain application, then one can apply lossless compression on this region of interest (ROI) instead of the entire image, thereby achieving a very large reduction on the original image without losing any important information.

ROI’s can be defined by a human user, or they can be automatically identified by some machine learning methods such as the Support Vector Machine (SVM) [67]. In general, one cannot assume a ROI to have a regular shape. For example, Fig. 4.1(b) displays a color-coded map indicating a total 17 ROI’s detected using a SVM classifier, where it can be seen that some ROI’s have irregular shapes with isolated dots and “holes”. Note that in JPEG 2000 image compression standard [68], a method known as Maxshift allows wavelet coefficients of ROI pixels to be scaled up and placed in higher bit-planes than the coefficients for the non-ROI pixels. At the decoder, the ROI coefficients and the non-ROI coefficients can be separated by comparing their magnitudes, thereby eliminating the need to tell the decoder explicitly about the shape information of the ROI’s. Nevertheless, the Maxshift technique would not be suitable for the problem we are addressing here, where we lossless code only the ROI pixels, while dropping all non-ROI pixels, in order to achieve largest possible compression on the original image. In the absence of non-ROI pixels, the shape information in the form of ROI map (or mask) has to be conveyed explicitly to the decoder in order for the reconstructed ROI pixel values to be placed back in the right locations. For this sake, we have proposed an efficient lossless compression method
Figure 4.3: The major components of the lossless ROI compression algorithm for hyperspectral images with arbitrarily shaped ROI’s in a prior work [69]. Before individual ROI compression, the overall ROI map as shown in Fig. 4.1b was first converted to a set of bi-level ROI maps: one for each individual ROI, all of which are shown in Fig. 4.2. The major steps of the individual ROI compression algorithm of [69] are shown in Fig. 4.3. For ROI map compression, arithmetic coding was applied to a sequence of bi-level symbols after we converted a two-dimensional ROI map into a one-dimensional vector following a raster scan. While being straightforward, this method failed to effectively capture two-dimensional correlations existing in the source. Therefore we seek to develop more efficient bi-level image compression methods for ROI maps, which in turn will lead to better compression on the ROI’s.

4.2 The Biased Run-Length Coding Method

In order to exploit the pixel correlations in both horizontal and vertical directions, we adopted a symbol packing approach, where a binary image was first partitioned into $3 \times 3$ blocks, with each block being scanned in a raster-scan order. The resulting sequence of bi-level symbols was then converted to a 9-bit binary representation of that block\(^2\). For instance, an all-one block will be converted to the block symbol ‘511’,

---

\(^2\)Assume that the image dimensions are divisible by 3; otherwise, rows and columns of 1’s can be padded to make the image size divisible by 3.
the decimal value of the binary representation \((111111111)_{b}\), as shown in Fig. 4.4 (left). Given the choice of \(3 \times 3\) blocks, there are a total of 512 possible block symbols. Using larger blocks will allow us to better exploit the spatial correlations in the source image; however, a large alphabet of block symbols would make the actual implementation of coding (e.g., Huffman coding) overly complicated.

![Figure 4.4: Examples of converting a block of nine bi-level symbols to a single block symbol.](image)

We observed that in many bi-level images, either the all-1 or the all-0 block symbol tends to be the most probable one among all possible symbols. For example, Fig. 4.5 shows that in ROI maps 1 through 16, the all-1 symbol occurs more than 80% of the time. This is to be expected since the black backgrounds in these ROI maps occupy a majority of the pixels (with value being 1). In contrast, in ROI 0, the all-0 block symbol occurs about 38% of the time and thus is the most probable symbol (see Fig. 4.6), where the all-1 block symbol (with a frequency of about 36%) is slightly less probable than the all-0 symbol. This can be explained by the fact that all-black and all-white blocks appear to be much more “evenly” distributed in the ROI-0 map than other ROI’s.
To take advantage of the redundancy associated with the most probable symbols, we introduced a biased run-length encoding method, which run-length codes only the most probable block symbol. For example, given a sequence of block symbols $[2, 511, 511, 511, 511, 5]$, a modified sequence is generated as $[2, 511, 5]$, where the ‘$511$’ in the middle is a pointer to an entry in another sequence recording the run-lengths the ‘$511$’ symbol ever encountered. In this example, the entry in the run-length sequence being pointed to has a value of 4, corresponding to four consecutive
symbols with identical values of ‘511’ in the original sequence of block symbols. This modified sequence of hybrid symbols values can be unambiguously reconstructed by the decoder, since any entry equal to ‘511’ means it is a flag bit, which is to be replaced by the actual number of ‘511’ block symbols as indicated by the corresponding run-lengths, whereas an entry less than 511 means it is already a block symbol value. It is important to note that if the conventional run-length coding is applied, then all the block symbols will be run-length coded, which would lead to a large number

**Figure 4.6:** Distribution of block symbols for ROI 0, with the all-0 block symbol being the most probable symbol.
of very short runs – a major source of inefficiency in run-length coding. However, our biased method solves this problem by run-length coding only the most probable block symbol. Thus the main novelty of our method lies in its special treatment of the most probable symbol from the rest of the block symbols. The block diagram of this biased run-length coding method is shown in Fig. 4.7, where the run-lengths of the most probable block symbol is entropy coded using a variable-length code such as the Huffman code. On the other hand, we use a separate Huffman code to compress the modified sequence of block symbol values.

### 4.3 Simulation Results and Discussions

We compressed all 17 ROI maps using the proposed method. We also tested six other lossless bi-level image compressors, including arithmetic coding, CCITT compression and its variants (FAX 3 and FAX 4), lossless JPEG 2000, and JBIG2. The results are summarized in Table 4.1 and also plotted in Fig. 4.8 for easy comparison.

We can see that for compression of ROI maps 1 through 16, the proposed method outperforms all other compressors including the state-of-the-art JBIG2 standard. More specifically, for maps of very sparse ROI’s with irregular shapes (e.g., ROI’s 4 and 15), JBIG2 standard can achieve the second best compression followed by either arithmetic code or FAX 4. For other less sparse ROI’s with “randomly” distributed dots (e.g., ROI’s 3, 10, and 11), again, our method achieves the highest compression, with JBIG2 being the second best, followed by either lossless JPEG 2000 or FAX 4.
Figure 4.7: The proposed biased run-length coding scheme.
Table 4.1: Bitrate (bits/pixel, or bpp) of the compressed files.

<table>
<thead>
<tr>
<th>ROI #</th>
<th>JPEG 2000</th>
<th>CCITT</th>
<th>FAX3</th>
<th>FAX4</th>
<th>Arithmetic Coding</th>
<th>Proposed</th>
<th>JBIG2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.371</td>
<td>0.625</td>
<td>0.550</td>
<td>0.265</td>
<td>1.001</td>
<td>0.271</td>
<td>0.180</td>
</tr>
<tr>
<td>1</td>
<td>0.085</td>
<td>0.316</td>
<td>0.321</td>
<td>0.112</td>
<td>0.023</td>
<td>0.002</td>
<td>0.047</td>
</tr>
<tr>
<td>2</td>
<td>0.213</td>
<td>0.434</td>
<td>0.444</td>
<td>0.234</td>
<td>0.364</td>
<td>0.093</td>
<td>0.137</td>
</tr>
<tr>
<td>3</td>
<td>0.173</td>
<td>0.366</td>
<td>0.377</td>
<td>0.169</td>
<td>0.235</td>
<td>0.042</td>
<td>0.094</td>
</tr>
<tr>
<td>4</td>
<td>0.105</td>
<td>0.316</td>
<td>0.321</td>
<td>0.113</td>
<td>0.091</td>
<td>0.007</td>
<td>0.050</td>
</tr>
<tr>
<td>5</td>
<td>0.115</td>
<td>0.347</td>
<td>0.351</td>
<td>0.132</td>
<td>0.161</td>
<td>0.025</td>
<td>0.064</td>
</tr>
<tr>
<td>6</td>
<td>0.13</td>
<td>0.363</td>
<td>0.363</td>
<td>0.142</td>
<td>0.224</td>
<td>0.033</td>
<td>0.072</td>
</tr>
<tr>
<td>7</td>
<td>0.080</td>
<td>0.313</td>
<td>0.319</td>
<td>0.110</td>
<td>0.016</td>
<td>0.001</td>
<td>0.044</td>
</tr>
<tr>
<td>8</td>
<td>0.102</td>
<td>0.322</td>
<td>0.329</td>
<td>0.118</td>
<td>0.160</td>
<td>0.010</td>
<td>0.052</td>
</tr>
<tr>
<td>9</td>
<td>0.081</td>
<td>0.316</td>
<td>0.321</td>
<td>0.110</td>
<td>0.012</td>
<td>0.001</td>
<td>0.044</td>
</tr>
<tr>
<td>10</td>
<td>0.184</td>
<td>0.407</td>
<td>0.410</td>
<td>0.198</td>
<td>0.273</td>
<td>0.067</td>
<td>0.113</td>
</tr>
<tr>
<td>11</td>
<td>0.263</td>
<td>0.497</td>
<td>0.489</td>
<td>0.268</td>
<td>0.527</td>
<td>0.132</td>
<td>0.166</td>
</tr>
<tr>
<td>12</td>
<td>0.151</td>
<td>0.355</td>
<td>0.361</td>
<td>0.154</td>
<td>0.191</td>
<td>0.035</td>
<td>0.082</td>
</tr>
<tr>
<td>13</td>
<td>0.088</td>
<td>0.314</td>
<td>0.319</td>
<td>0.110</td>
<td>0.081</td>
<td>0.004</td>
<td>0.046</td>
</tr>
<tr>
<td>14</td>
<td>0.122</td>
<td>0.345</td>
<td>0.345</td>
<td>0.124</td>
<td>0.331</td>
<td>0.022</td>
<td>0.061</td>
</tr>
<tr>
<td>15</td>
<td>0.102</td>
<td>0.328</td>
<td>0.332</td>
<td>0.119</td>
<td>0.134</td>
<td>0.011</td>
<td>0.051</td>
</tr>
<tr>
<td>16</td>
<td>0.097</td>
<td>0.319</td>
<td>0.323</td>
<td>0.113</td>
<td>0.042</td>
<td>0.004</td>
<td>0.050</td>
</tr>
</tbody>
</table>

For ROI 0, since the all-1 block symbol accounts for only about 36%, which is less probable than the all-0 block symbol (refer back to Fig. 4.6), we expect the proposed method would become less efficient by run-length coding the all-1 block symbol. This is indeed the case. However, while our method is only very slightly less efficient than JBIG2 (by about 0.1 bpp) and FAX4 (by 0.06 bpp), it could still compress more efficiently than arithmetic code (which resulted in data expansion rather than compression), CCITT, FAX 3, and JPEG 2000. If we run-length code the all-0 block symbol, which is the most probable symbol for the ROI-0 map, we can raise the compression ratio of the proposed method to the second best after JBIG2 (with a narrower gap of only 0.06 bpp).
4.4 Conclusions

We have demonstrated that the proposed biased run-length coding method achieved outstanding compression on 16 out of 17 ROI maps in hyperspectral images, with higher compression efficiency than various state-of-the-art techniques including JPEG 2000 and JBIG2. While these bi-level maps have widely ranging shape patterns, they share a common feature of having a dominant most probable block symbol. For the image source with a less biased most probable symbol, our method could still...
perform very closely to the best compressor. In next chapter, we will introduce a new
mathematical model based on Markov models for analyzing of the biased run-length
coding method.
5.1 Analysis of Biased Run-length Method on ROI binary Maps of Hyperspectral Images

In this chapter, we show a detail analysis on our previously proposed method also known as Biased Run-length coding (BRL). This consists of three parts: 1) the effectiveness of the proposed block-symbol based coding technique versus the 1-D coding. 2) Introducing a novel model based on Markov and Capon models for the BRL method, and effectiveness of the BRL coding using both the statistical model and empirical data. 3) Entropy comparison between the BRL and Conventional Run-length coding method. The results for both empirical data and statistical data show significant improvement over the original entropy of block symbols by taking advantage of the 2D correlations and special treatments of the most probable blocks.

In this section, we will analyze in detail the significance of the biased run-length coding method. In order to exploit the pixel correlations in both horizontal and vertical directions, we proposed a symbol packing approach. The motivation is
**Figure 5.1:** Most Probable blocks in order (from left to right).
to pack more pixels in a block symbol, because the background appears to contain the majority of the pixels. In addition, the objects usually have “thickness”, meaning there are more groups of pixel with values ‘0’ in the image than ‘0’ by itself (random dots). To show that this is indeed the case, we calculated the distribution of each block symbol for all the ROI binary maps we studied shown in Fig. 4.2. Fig. 5.1 shows the first nine most probable block symbols in order from left to right. Block symbol ‘511’ is the most probable block symbol with probability of 0.9192. Block symbol ‘0’ is the next most probable block with probability of 0.0408. It is clearly shown that the probability of the block symbols with same pixel values grouped together (group of white/black pixels) are higher than block symbols with random pattern (black or white dots). In other words, there are more neighboring pixels with the same values in most probable blocks for the ROI maps we studied.

5.1.1 Distribution of block symbols

In this section, we will examine when the block symbol-based coding is beneficial using statistical model. It was shown that the distribution of integers can be modeled using the generalized Gaussian distribution (GGD) [70].

We assume the probability of block symbols can be shown in Discrete Gaussian distribution and since the block symbols are always positive integers, only the right side of the Gaussian function is chosen. Note that the normalization is required so that the sum of right-sided discrete Gaussian function is still one. The probability $G$ at given $k/\beta$ can be calculated using the following formula, where $k$ is an integer, $\beta$ is the quantization level, and $n$ is the maximum number of block symbols.
Figure 5.2: Discrete Gaussian Distribution probabilities for different $\beta$ for $\sigma^2 = 0.5$. 


\[ G(x = k\beta) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(k\beta - \mu)^2}{2\sigma^2}\right) \sum_{x=0}^{n\beta} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right). \] (5.1)

Fig. 5.2 shows the discrete Gaussian Distribution probabilities for different \( \beta \) values.

We can see that as \( \beta \) increases, the distribution changes as well, and as a result the entropy varies. The entropy using the block-symbol technique can be calculated using the following equation:

\[ \text{Entropy of block based} = -\frac{\sum_{n=1}^{2^N} G_n \log G_n}{N}, \] (5.2)

where \( N \) is the number of pixels per block, and \( 2^N \) is the number of block symbols.

Note that top part of the equation is the average amount of pixels per block which needs to be divided by the number of pixels per block to calculate the average amount of bit per pixel.

Given the statistical probabilities of block symbols \( G_n \), we propose a unique way to extract the probability of 1 from the source probabilities of block symbols. In other words, using the statistical distribution of block symbols, we can obtain the original binary source. In order to achieve this, first we need to find the probability of 1 in each block symbol then rearrange it from the most probable to the least probable symbol block and store it in an array ‘A’ which contains \( 2^N \) elements. Fig. 5.1 shows an example of nine first most probable block symbols using the ROI binary maps.

For instance, the most probable block symbol is 511 which consists of nine 1’s so the
probability of 1 is 1.00. Next is block symbol 0, which has zero 1’s, and therefore
the probability of 1 is 0. Next is 219 which contains six 1’s, and as a result, the
probability of 1 is 0.6667 and this happens for the entire block symbols. As a result,
the probability of 1 \( p \) can be calculated as following equation:

\[
p = \sum_{n=1}^{2N} A_n G_n
\]  \hspace{1cm} (5.3)

Once the probability of 1 is extracted from block symbols, the entropy of the
new extracted binary source can be calculated using the following equation:

\[
H = -(p \log_2 p + (1 - p) \log_2(1 - p)).
\]  \hspace{1cm} (5.4)

Next, we compared the entropy values between the block-symbol technique
and the new binary source when block symbols can be represent as one-sided discrete
Gaussian function with different \( \sigma \) values. It can be seen in Fig. 5.3 that as \( \sigma^2 \)
decreases the entropy decreases as well. For each \( \beta \) there are entropies calculated for
the probability distribution of symbol blocks known as 2D and the extracted binary
source also known as 1D. We can see that the corresponding 1D has higher entropy in
for such distributions in the figure, which makes the block based technique a superior
method over non-blocking techniques. As \( \beta \) decreases the entropy for both 1D and
2D gets closer to each other. In order to validate the behavior of the above model, we
also calculated the entropy values for both the block based and the original binary
ROI images. The main difference here is that the probability of 1 for each image now
is based on the actual distribution of block symbol per ROI image. We can clearly see that the behavior of the entropies is very similar to the ones based on model. When the entropy of the block based is rather high, there is a larger entropy difference with its corresponding 1D entropy. For instance, ROI 0 has an entropy around 0.3 for the block based, however, the entropy for the 1D is close to 1. This is to be expected, since the distribution of block symbol 511 and 0 are very close. We can see that the gap is rather large, but when entropy of the block based is smaller, the gap between 1D and 2D is smaller. For example, ROI 1, 7, and 9 have very small difference between the 1D and 2D method. Similarly, in Fig. 5.3 as entropy is smaller, the gap between them become smaller. Fig. 5.4 shows the entropy comparison between entropy of the original binary image and the entropy of block based method using 3×3 blocks.

5.2 Mathematical Model for Biased Run-Length Coding Method

First the Right-sided discrete Gaussian distribution is chosen as the original distribution of the block symbol. In the original biased run-length coding method, the original distribution is segmented into two different distributions. First distribution consists of the of the run-lengths of the most probable block symbol. This distribution can be defined as the Markov model using the probability of the most probable block symbol extracted from the original distribution of block symbols. Fig. 5.5 shows the diagram for the biased run-length coding method using the statistical models.
Figure 5.3: Entropy comparison between 1D and 3×3 block based for different and \( \beta \) and \( \sigma^2 \).

5.2.1 Geometric distributions

Run lengths of binary source can normally be modeled using the geometric distribution as following

\[
G(n) = p^n q = p^n (1 - p),
\]

(5.5)

where \( p \) is the probability of “1”, and \( n \) is the run length of “1”. However, for the case of the biased run-length coding method, since the most probable block symbol
Figure 5.4: Entropy comparison between 1D and $3 \times 3$ block based for binary ROI images

Figure 5.5: Diagram for the Biased Run Length coding method using statistical model.
is only run-length encoded, $p$ is now the probability of the most probable block symbol, and $n$ is the run lengths of the most probable block symbol. However, there are disadvantages associated with using the geometric distribution model, this distribution does not capture the transition probability between the most probable block symbol and other block symbols. As a result, we introduce a new Markov model based on Capon model [71], which contains the transition probabilities between the most probable block symbol and other symbol blocks for the distribution of the run-lengths.

5.2.2 Markov model for distribution of the RLs

![Diagram](image)

**Figure 5.6**: Transition Probabilities between block symbol 511 and other blocks.
In this section first order Markov model is used. Fig. 5.6 shows the transition probabilities between the most probable block symbol and other symbol blocks. \( p \) is the probability of the most probable block symbol occurs, \( 1 - p \) is the probability that other blocks symbols occur. Let \( A \) represents the state for most probable block symbol, and \( B \) represents the state for other block symbols. Given the probability is in state \( A \), the probability of next block symbol is also in \( A \) is \( q \). As a result, the probability of next block is in state \( B \) given the current state is in \( A \) is \( 1 - q \). Similarly, the probability of next state is \( B \) given the current state is also \( B \) is \( r \), and the probability of next state is \( A \), given the current state is \( B \) is \( 1 - r \). The following inequality must be satisfied in order to maintain the stability for transitions between states. Moreover, it needs to be emphasized that in Capon’s model the probabilities of runs of white and black pixels are assumed to have the same distribution in the picture. However, our mathematical model is uniquely derived using the biased distribution between the most probable block symbol and rest of the block symbols. In our following model, probability of transition from the most probable block symbol to other blocks and staying in that state plays a major role which was not derived in based on Capon’s earlier model.

\[
p(1 - q) = (1 - p)(1 - r) \tag{5.6}
\]

\[
0 < \frac{p(1 - q)}{1 - p} < 1
\]

\[
q > 2 - \frac{1}{p}
\]
Table 5.1 shows the probability distribution of Run-lengths of the most probable block symbol using the Markov model. “n” is the run-length of the most probable block symbol. It is important to note that for demonstration purposes, codeword is shown in binary format, where indeed 1 represents the most probable block symbol, and 0 represents other block symbols.

Table 5.1: Generation of probability distribution of RLs using Markov model.

<table>
<thead>
<tr>
<th>n</th>
<th>codeword</th>
<th>probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>( p(1 - q) )</td>
</tr>
<tr>
<td>2</td>
<td>110</td>
<td>( pq(1 - q) )</td>
</tr>
<tr>
<td>3</td>
<td>1110</td>
<td>( pq^2(1 - q) )</td>
</tr>
<tr>
<td>4</td>
<td>11110</td>
<td>( pq^3(1 - q) )</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>n</td>
<td>11...0</td>
<td>( pq^{(n-1)}(1 - q) )</td>
</tr>
</tbody>
</table>

As a result, the Markov model for run-lengths can be defined as the following equation:

\[
M(n) = pq^{(n-1)}(1 - q) \tag{5.7}
\]

The Markov distribution will be used to calculate the reduced probability of the most probable block symbol using the original probability. Before calculating the entropy, the Markov probability distribution needs to be normalized so the sum of probabilities is equal to one.

\[
M_{\text{normalized}}(n) = \frac{M(n)}{\sum_{n=1}^{\infty} M(n)} \tag{5.8}
\]
The entropy of the Markov distribution of the run-lengths can be calculated using the following.

\[ H_M = - \sum_{n=1}^{\infty} M_{\text{normalized}}(n) \log_2[M_{\text{normalized}}(n)] \]  

(5.9)

After the biased run-length coding, the number of merged most probable block will be smaller than the original number of the most probable blocks. For example, if the run-length of a ‘511’ blocks is five, then five consecutive ‘511’ blocks will be reduced to a single merged-block symbol, with its value being five. The average shrinking factor (SF) can be found as:

\[ SF = \frac{1}{\hat{E}[n]} = \frac{1}{\sum_{n=1}^{\infty} nq^{n-1}(1-q)} = 1 - q. \]  

(5.10)

In practical simulations with only finite run-lengths being observed, the average shrinking factor can be estimated as:

\[ \overline{SF} = \frac{1}{\hat{E}[n]} = \frac{1}{\sum_{R} n \cdot \text{Prob}[n]}, \]  

(5.11)

where \( \hat{E}[n] \) is the sample mean of the set \( R \) of distinct run-lengths \( n \) ever encountered, and \( \text{Prob}[n] \) represents the empirical distribution of the run-lengths.

As a result, the length of the merged-block sequence can be found as:

\[ \text{Sequence Length} = N \cdot p \cdot \overline{SF}, \]  

(5.12)
where $N$ is the length of the original block symbol sequence.

**Table 5.2**: 511 blocks vs. New merged blocks

<table>
<thead>
<tr>
<th>n</th>
<th>original blocks</th>
<th>merged blocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>511</td>
<td>511</td>
</tr>
<tr>
<td>2</td>
<td>511 511</td>
<td>511</td>
</tr>
<tr>
<td>3</td>
<td>511 511 511</td>
<td>511</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>M</td>
<td>511...511</td>
<td>511</td>
</tr>
</tbody>
</table>

Table 5.2 shows the original symbol blocks 511 versus the new merged blocks for different run-lengths of block symbol 511.

The total number of bits for the run-lengths can be calculated by multiplying the entropy of the new Markov model in Eq. (8) and the total number of run-lengths in Eq. (11).

\[
\text{Total number of bits} = H_M \times N \cdot p \cdot SF
\]  \hspace{1cm} (5.13)

### 5.2.3 Construction of the new modified sequence

To construct the modified sequence, we use the original right-sided discrete Gaussian distribution of block symbols in addition to the distribution of RLs. To obtain the original number of each of block symbols $B$, the original right-sided discrete Gaussian distribution in Eq. (1) will be multiplied by “N” the total number of blocks as shown in the following:
\[ B = G \times N. \] (5.14)

Then the new modified sequence will have the same distribution of blocks except the most probable block symbol. In other words, the original number of most probable block symbol will be replaced by the new number of block symbols which is the same number as the sequence length calculated in Eq.(12) as shown below:

\[ B(1) \text{ (New number of the most probable block symbol)} = N \cdot p \cdot SF. \] (5.15)

Note, \( B \) is an array of probabilities, thus \( B(1) \) is the first probability value in the array. Once the new distribution of block symbols is constructed, the new “Modified Sequence” (\( MS \)) can be calculated by normalizing the block distributions as following:

\[ MS = \frac{B}{\sum_{n=1}^{\infty} B}. \] (5.16)

Next, the entropy of the modified sequence will be calculated. To calculate the total number of bits for the modified sequence, the entropy of the modified sequence should be multiplied by the number of symbols in the modified sequence. The total number of symbols in the modified distribution can be calculated using the following:
Total number of symbols in the modified sequence =

\[ N - Np + N \cdot p \cdot SF \]

\[ = N - Np(1 - SF) \]

\[ = N(1 - p(1 - SF)) \] (5.17)

Next, the entropy of the modified sequence will be multiplied by the total number of symbols in the modified sequence as shown below:

\[
\text{Total number of bits} = -\sum_{n=1}^{\infty} MS \log_2(MS) \times N(1 - p(1 - SF))
\] (5.18)

Lastly, the total number of bits for the biased method can be calculated by adding the total number of bits for the run-lengths (Eq. (12)) and the modified sequence (Eq. (17)).

5.3 Results

Fig. 5.7 shows comparison results between our proposed biased RL model and the entropy of the original distribution. For this example “N” is 2304, \( \beta \) is 0.5, and \( \sigma^2 \)
and $q$ are variables. It is clearly shown that for higher $q$ values, the biased run-length coding method is superior over the entropy of the original distribution.

Fig. 5.8 shows comparison results between our proposed biased RL model and the entropy of the original distribution. For this example “N” is 2304, $\beta$ is 0.8, and $\sigma^2$ and $q$ are variables. For this example, we used the minimum possible values for $q$ for each $\sigma^2$. We can see that when $\sigma^2$ is very small, even when $q$ is the minimum possible value, our BRL coding method can be beneficial over entropy of the original
Figure 5.8: Size comparison result between biased method and entropy of the original distribution for $\beta = 0.8$, different $\sigma^2$ and $q$.

distribution. As $\sigma^2$ increases, there is a margin for $q$ where BRL coding method is beneficial. In other words, when $q$ is larger than some threshold, our method is superior over the entropy of the original distribution of block symbols. We can conclude that the proposed mathematical model is a very useful tool in determining when the proposed BRL coding method is beneficial which can be used to classify the original ROI binary sources.
5.4 Entropy Analysis of Biased Run-Length Coding

To show the advantage of using the proposed biased method, we compared the theoretical size using the entropy value between the biased run-length coding and the conventional run-length coding method.

Fig. 5.9 shows the diagram for calculating the theoretical compressed size using the entropy value of the biased RL method. Since the biased method contains sequence for run-lengths and the modified sequence, the entropy for each is calculated independently. Next, the entropy for each sequence is multiplied by the length of each sequence. “Entropy 1” is the entropy of the run-lengths, and “Entropy 2” is the entropy of the modified sequence (MS). Next, “Entropy 1” is multiplied by the length of the run-lengths, and “Entropy 2” is multiplied by the length of the MS. Finally, both values are added to show the theoretical compressed size. The bitrate comparison between biased RL and conventional RL is shown in Fig. 5.10.

The result shows that our proposed biased method could achieve higher compression than conventional run length coding in all the images using the theoretical entropy calculation.

5.5 Conclusion

This chapter focused on analysis of the biased run length coding. It first explains the advantage of symbol packing techniques versus 1D scanning. Next, it shows the detail analysis of biased run length coding with its advantages using a novel model based on Markov model. The results for both empirical data and statistical data
Figure 5.9: Entropy calculation for Biased RL coding method.
Figure 5.10: Bitrate using entropy calculation for Biased RL, and conventional RL.

show the significance of the proposed method and when it will be beneficial over the conventional run-length coding techniques.
CHAPTER 6

EVALUATION OF THE BIAS ED RUN-LENGTH CODING METHOD
GENERATED BY A MODIFIED ISING MODEL

In this chapter based on our paper\(^1\) [72], we evaluate the performance of several lossless compression methods on binary images simulated by a statistical model. This work was motivated by our previous study on compression of regions-of-interest maps in hyperspectral images, where a biased run-length coding method was found to provide good compression on just a few binary images with most pixels being in the background. In order to discover the trend on the compression performance, we introduced a modified Ising model of Markov random fields by Metropolis, which can generate a large number of binary images iteratively, so that we can study how the compression ratios of the above method change gradually when the contents of the images get varied slightly with iterations. Simulation results showed that the biased run-length coding method significantly outperformed the arithmetic code and JBIG2 method. The study also gave rise to a dual biased run-length method, which can provide further compression gains on images with more foreground objects.

\(^1\)©2016 IEEE
6.1 Markov Random Field

*Markov random field* has many applications in image processing, synthetic aperture radar (SAR), information theory, bio-medical imaging, hyperspectral imaging, object tracking, image segmentation, image compression among many others [73–97]. In a two-dimensional (2D) *Markov random field*, the probability distribution of $X_n$ depends solely on the neighbors $X_{n-1}$ and $X_{n+1}$. In another words, assuming $S$ is an $m \times n$ matrix where

$$s(i, j), 1 \leq i \leq m, 1 \leq j \leq n.$$ 

For a fixed index matrix also known as “site”, there is a neighborhood $N(s)$. For instance, the neighborhood for location index $S(i, j)$, can be defined as $N(i, j) = \{(i - 1, j), (i + 1, j), (i, j - 1), (i, j + 1)\}$.

Since the probability is dependent only on the neighborhoods, the conditional Markov property can be described as

$$Pr\{X(s) = x(s)|X(N_s) = x(N_s)\}. \quad (6.1)$$

A random value $X(S)$ is a *Gibbs Random Field* and its distribution is in the following form

$$P(x) = \frac{1}{Z} \exp(U(x)), \quad (6.2)$$
Where $Z$ is a normalization constant, and $U$ is the energy function. The energy function is described as the following

$$U(x) = \sum_{c \in C} V_c(X),$$  \hspace{1cm} (6.3)

where $V_c$ is the clique potential function over all possible set of cliques $C$.

### 6.1.1 Ising model

*Ising Model* describes the interaction between particles in ferromagnetic materials [98, 99]. Similar to a bi-level images, each particle can have two values. For the case of ferromagnetic material these values can be $+1$ or $-1$. Each particle only interacts with its neighborhood. The orientation of each particle determines the total energy in the system as following

$$E = -J \sum_{x_i \sim x_j} x_i x_j,$$ \hspace{1cm} (6.4)

where $J$ represents the strength of interaction. The probability for any particular pixel over the neighborhood pixels can be defined as

$$P(x) = \frac{1}{Z} \exp(-E(x)),$$ \hspace{1cm} (6.5)
where $Z$ is the normalization constant. replacing $E(x)$ by previous equation, we get the following result

$$P(x) = \frac{1}{Z} \exp(J \sum_{x_i \sim x_j} x_i x_j)$$ (6.6)

The following algorithm is an Ising model by Metropolis [100].

Generate a random binary image $A$

$[M \ N] = \text{size}(A)$

for $i = 0$ to iteration do

pick a random point $a$ with location $(i, j)$

$N_{\text{agree}} = \sum(\text{neighborhoods} = A(i, j))$

$N_{\text{disagree}} = \sum(\text{neighborhoods} \neq A(i, j))$

diff $= N_{\text{agree}} - N_{\text{disagree}}$

pick a random number $b$, $0 < b < 1$

if $b < \exp(-2J \text{diff})$ then

if $A(i, j) = 0$

$A(i, j) = 1$

else

$A(i, j) = 0$

end if

end for

Fig. 6.1 shows an example for different iterations.
Figure 6.1: Binary images generated by the Ising Metropolis model.
6.2 Modified Ising Model

As shown in previous section, the binary images based on Ising model tend to have blobs of black and white with very similar distributions of 1’s and 0’s. Motivated by our previous studies on ROI imaging of hyperspectral images where we observed that in a class of images the majorities of pixels are the background pixels, while the detected object has the lower probability of pixels, we propose to modify the Ising model to model this class of binary images with biased distribution of 1’s and 0’s.

6.2.1 Proposed modified Ising model

There are two main reasons to use the proposed method. One is that since in most of the binary ROI maps we studied, the shapes of the objects are not the same and the sizes vary each time, we do not have control over the size of the object detected. Therefore, we decided to use models to generate binary images, where we have control over the growth of the arbitrary shape objects and study the trend of compression achievable based on their growth rate.

The second reason is that most ROI binary maps share a common condition, the background pixels which are 1’s are the majorities of the pixels values, while the detected object which pixels values are 0’s are less probable.

As a result, we generate a new model based on modified Ising model. As opposed to the Ising model, the initial binary image is not a random binary image with same probability of 1’s and 0’s. We propose to pick a random point in an all-ones image, and replace it with a black pixel 0. Next, inside each iteration, another
random point will be chosen, if the point is 0, the neighborhoods become zeroes as well. Neighborhood definition can vary, for example, it can be a 4-neighborhood pixels or an 8-neighborhood pixels, or any other patterns as shown in Fig. 6.2. For simplicity and faster growth rate, the 8-neighborhood configuration is chosen. Fig. 6.3 shows ten different binary images generated by the proposed model.

Generate an all-ones image $A$

$[M \ N] = \text{size}(A)$

pick a random point $a$ with location $(i, j)$

$A(i, j) = 0$

for $i = 0$ to iteration do

pick a random point $b$ with location $(x, y)$

if $A(x, y) = 0$

$A(x - 1, y) = 0;$
$A(x + 1, y) = 0;$
$A(x, y - 1) = 0;$
$A(x, y + 1) = 0;$
$A(x - 1, y - 1) = 0;$
$A(x - 1, y + 1) = 0;$
$A(x + 1, y - 1) = 0;$
$A(x + 1, y + 1) = 0;$

end

end for
6.2.2 Results

We applied the B-RL coding method on binary images generated by the modified model shown in Fig. 6.3. Besides B-RL, Arithmetic coding (AC) and JBIG2 methods were also applied on these images. See the compression results in Fig. 6.4. It clearly shows that as the foreground object grows the resulting bit rate is increasing as expected. The B-RL coding method outperformed other methods significantly because it takes advantages of the background pixels and packed more 1’s by run-length coding only the most probable block. As the foreground object grows, the fraction of the most probable blocks is decreasing, and as a result, their run-lengths become smaller. In next section we proposed a new method for efficient compression of binary images with larger objects.

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
$A(x-1,y-1)$ & $A(x-1,y)$ & $A(x-1,y+1)$ \\
\hline
$A(x,y-1)$ & $A(x,y)$ & $A(x,y+1)$ \\
\hline
$A(x+1,y-1)$ & $A(x+1,y)$ & $A(x+1,y+1)$ \\
\hline
\end{tabular}
\end{figure}

\begin{figure}[h]
\centering
\begin{tabular}{|c|c|}
\hline
$A(x-1,y)$ & $A(x+1,y)$ & $A(x-1,y+1)$ \\
\hline
$A(x+1,y-1)$ & $A(x,y)$ & $A(x+1,y+1)$ \\
\hline
$A(x+1,y)$ \\
\hline
\end{tabular}
\end{figure}

\textbf{Figure 6.2}: An 8-neighorhood pixel (left) and a 4-bit neighborhood pixels for the proposed modified Ising model.
Figure 6.3: Binary images generated using the modified Ising model.
6.3 Dual Biased Run-Length Coding

As the object gets larger and larger, the probability of block symbols ‘511’ decreases and the probability of block symbol ‘0’ increases. In this new modified scheme, we run-length code the top two most probable block symbols independently from the rest of the block symbols. The diagram for the new dual biased run length coding (DB-RL) is shown in Fig. 6.5. In this method, the modified sequence from the original biased Rl method will be processed further so that the run lengths for block symbol ‘0’
will be calculated. The new modified sequence now has two pointers, one representing block symbol ‘511’, and other representing ‘0’. The output run-lengths for ‘0’ will be Huffman coded in addition to ‘511’ and the new modified sequence.

To evaluate the performance of the DB-RL coding, we generated a new sets of binary images with much more iterations. These images are shown in Fig. 6.6, where objects are much larger than those in Fig. 6.3. We then applied the DB-RL coding method, B-RL coding method, along with the arithmetic code and JBIG2 method to compare the bit rates of the compressed images.

6.4 Results on DB-RL

The compression results is shown in Fig. 6.7. It is shown that both the DB-RL and B-RL coding methods outperform JBIG2 and arithmetic coding for all the images. As the object is growing for each iteration, the DB-RL coding outperform B-RL coding by taking advantage of run-length coding the second most probable symbol block which is symbol block 0. This can be seen in Fig. 6.8. It is important to note that there are 512 possible symbol blocks that can occur in an image, however, only those top two most probable blocks were run-length coded for background and object.

By looking at Fig. 6.7 and Fig. 6.8 we can see that the graphs for our methods and for JBIG2 method are not as smooth as the results in previous simulations. For example, image 20 is compressed slightly better than image 19. We might expect as the number of iteration increases, the compression result become smaller, however, this might not be the case for all the iterations. One of the main reasons is that the distribution of random block symbols (non-all zeros and non-all ones block symbols)
are more in image 20 (46 blocks) than in image 19 (52 blocks). This by itself might cause extra bits for random block symbols. In other words, even though the object is growing the random blocks at the edges appear to be more. In first simulation the object appears to be growing in all the direction, since the object is close to the middle of the image. However, in the second simulation the object is close to the edge of the image which decreases the growth rate in the direction of the edge of the image. Decreasing the growth rate can cause less change in the compression rate between the two consecutive images. despite these special cases, in general, as the number of iteration increases, the compression size decreases.

To this end, we calculated the entropy of the block symbols for all 20 images. Fig. 6.9 shows the entropy which is an average amount of bits per pixel. We can see that the entropy curve is an increasing function except at few points. For instance, image 20 has higher entropy than image 19 as we discussed previously. In addition, we calculated the first order entropy of the original binary images which is shown in the same figure. We can see that this plot is smooth like the arithmetic coding method, because it does not capture the 2D correlation between pixels as opposed to symbol blocking techniques which use the 2D correlations. This shows the advantage of using symbol block techniques for more efficient compression of binary images for higher iterations.

6.5 Conclusion

In our previous studies on binary maps of ROI images, we discovered that the background pixels contain the majority of the pixels in an image. However, each image
contains arbitrary shape objects within the background and each has different shape and size. In order to uncover any possible trend of compression ratio of the proposed coding methods on simulated images whose contents can be varied gradually, we propose a new model by modifying the Ising model of Markov Random Field by Metropolis. We also proposed a Dual Biased Run-Length coding by modifying the previous method known as Biased Run-Length coding method. This new method was found to outperform the original biased run-length coding method and the arithmetic code as well as the JBIG2 method on all the simulated images. In the new method, we run-length code the two most probable symbol block for more efficient compression for larger objects.
Figure 6.5: Dual Biased Run-Length coding scheme
Figure 6.6: Ising model for higher iterations.
Figure 6.7: Bitrate result using the modified Ising model.
Figure 6.8: Bitrate result using the modified Ising model.
Figure 6.9: Entropy comparison using 1D and 2D for different iterations.
CHAPTER 7

CONCLUSION

7.1 Conclusion

Over the past decade, the need of image compression has increased significantly due to the higher resolution of images in area of aerospace, and medical imaging among many others. Limited telemetry bandwidth and storage is one of most challenging problems, therefore, more efficient compression techniques are needed. In this thesis, we proposed several novel methods for lossless compression of bi-level images.

In Chapter 3, we improved our previous work on distance-coding of binary images by introducing the idea of run-length coding the most probable interval. This hybrid coding method allowed a fraction of one bit to be assigned to the most probable interval as opposed to an entropy coder such as Huffman code, which requires at least one bit for the most probable interval. As a result, this method contributed about 17% improvement over the previous method.

In Chapter 4, motivated by our previous work which was described in chapter 3, we proposed a novel scheme known as the Biased Run-Length Coding method. In order to exploit more pixel correlation in 2D directions, we adopted a symbol packing approach. In this approach, we first partitioned a binary image into blocks, and
then scanned them in raster-scan order. The new sequence of binary symbols was converted to a binary representation of each block. The statistical distribution of these blocks symbols is biased towards the all-one block. As a result, we decided to only run-length code the most probable block symbol. The symbol packing scheme by itself allows assigning only fraction of a bit for one block, which typically requires 9 bits to represent a $3 \times 3$ block size. On top of this, we run-length the most probable block symbol which allows much more compression. In B-RL method, it only requires one bit for the pointer and a few bits for the run-length, however, if we use entropy coder such as Huffman code, it requires at least one bit per the most probable block symbol. More importantly, run-length coding the most probable block is in a way a variable block size coding without spending more overhead bits on the dimensionalities of the variable block sizes. Simulation results demonstrated that the proposed biased run-length coding method achieved outstanding compression on most ROI maps in hyperspectral images, with higher compression than very efficient and powerful compression standards such as JBIG2 and JPEG2000.

In Chapter 5, we showed a detail analysis on the Biased Run-Length coding method. The analysis revealed the effectiveness of using the symbol-packing technique versus the 1D coding. In addition, we introduced a novel model based on Markov model for the BRL method, and showed the effectiveness of BRL method using both the statistical model and emperical data.

Lastly, in Chapter 6 we evaluated the performance of several lossless compression techniques including the BRL coding method on binary images simulated by the new statistical model. In order to discover the trend on the compression performance,
we proposed a new modified Ising model of Markov random fields, which can generate a large set of binary images iteratively. Consequently, this allows us to study the compression ratios as the content of the objects get varied slightly with iterations. In addition, this work proposed a Dual Biased Run-Length coding method, which run-length codes the two most probable block symbols. The compression results show that both the B-RL and DB-RL outperform JBIG2 and arithmetic doing for all the images generated by the new statistical model.

7.2 Suggestions for Future Work

The work in this dissertation can be a baseline for future work. In our B-RL coding method, we capture the 2D correlation in a fixed raster-scan order as our baseline. This work can be extended using different scanning method such as zig zag, or vertical scanning. In addition, the block sizes for the B-RL coding method were chosen so that the Huffman table does not get over complicated. For instance, for a $4 \times 4$ block size, there would be 65536 different block symbols, however we are not limited to a $3 \times 3$ block size.

More effectively, the B-RL coding method can be extended to scan in 3D direction. If two or more consecutive ROI maps of hyperspectral images were very close in nature, and the probabilities of neighboring pixels in the current image and next image are the same, then the 3D scanning can generate larger run-length of the most probable block symbol. As a result, this can increase the compression significantly.
In our future work on models based on Markov random field of Ising, we will consider adding multiple objects and noise to the model and evaluate different compression methods with these simulated images. This work can be a baseline for many future models. For instance, the growth rate of an object or the growth path direction can be varied. This model can also be used to generate models of object growing in three-dimensional space.
REFERENCES


[42] https://www.youtube.com/watch?v=NDEq44IEtSY.


detection on sar images using triplet Markov field model,” *IEEE Geoscience

[74] X. Lian, Y. Wu, W. Zhao, F. Wang, Q. Zhang, and M. Li, “Unsupervised sar im-
age segmentation based on conditional triplet Markov fields,” *IEEE Geoscience

radar image segmentation using fuzzy label field-based triplet Markov fields

[76] Y. Wu, F. Wang, Q. Zhang, F. Niu, and M. Li, “Fast algorithm based on
superpixel-level conditional triplet Markov field for successive-approximation
resistor image segmentation,” *IET Radar, Sonar Navigation*, vol. 9, no. 8, pp.

[77] Y. Baryshnikov, J. Duda, and W. Szpankowski, “Markov field types and
tilings,” in *2014 IEEE International Symposium on Information Theory (ISIT)*,
June 2014, pp. 2639–2643.

[78] Y. Hame, E. Angelini, E. Hoffman, R. Barr, and A. Laine, “Robust quantifica-
tion of pulmonary emphysema with a hidden Markov measure field model,” in *2013 IEEE 10th International Symposium on Biomedical Imaging (ISBI)*, April
2013, pp. 382–385.

segmentation based on triplet Markov fields with graph cuts,” *IEEE Geoscience

[80] M. Pereyra, N. Dobigeon, H. Batatia, and J. Tourneret, “Estimating the gran-
ularity coefficient of a Potts-Markov random field within a Markov chain monte
2385–2397, June 2013.

[81] F. Caravetta, “2-d-recursive modelling of homogeneous discrete Gaussian
1198–1203, May 2011.

[82] K. Xu, W. Yang, G. Liu, and H. Sun, “Unsupervised satellite image classifi-
cation using Markov field topic model,” *IEEE Geoscience and Remote Sensing

[83] L. Cordero-Grande, G. Vegas-Sanchez-Ferrero, P. Casaseca-de-la Higuera, and
C. Alberola-Lopez, “A Markov random field approach for topology-preserving
registration: Application to object-based tomographic image interpolation,” *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 2047–2061, April
2012.


