2023

Contributions to data compression research for various media formats using particle swarm optimization and neural networks

Reetu Hooda

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

Recommended Citation
Hooda, Reetu, "Contributions to data compression research for various media formats using particle swarm optimization and neural networks" (2023). Dissertations. 274.
https://louis.uah.edu/uah-dissertations/274

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.
CONTRIBUTIONS TO DATA COMPRESSION
RESEARCH FOR VARIOUS MEDIA FORMATS USING
PARTICLE SWARM OPTIMIZATION AND NEURAL
NETWORKS

by

REETU HOODA

A DISSERTATION

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

HUNTSVILLE, ALABAMA

2023
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.

Reetu Hooda

03/01/2023
(date)
Submitted by Reetu Hooda 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.

Committee Chair

Dr. W. David Pan

(Date)

Dr. Vineetha Menon

(Date)

Dr. B. Earl Wells

(Date)

Dr. Sivaguru S. Ravindran

(Date)

Dr. Seong-Moo Yoo

(Date)

Dr. Aleksandar Milenkovic

(Date)

Department Chair

3/02/23

3/02/23

3/10/2023

3/3/2023

3/3/2023

3/21/2023
ABSTRACT

School of Graduate Studies
The University of Alabama in Huntsville

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

Name of Candidate Reetu Hooda

Title Contributions to Data Compression Research for Various Media Formats Using Particle Swarm Optimization and Neural Networks

Advancement in high resolution devices used to capture images and videos has put strain on data storage and transmission. For their practical usage, efficient data compression techniques becomes essential. In this context, we conducted a study on compression of two media formats, including ROI maps of hyperspectral images, and 3D point clouds. For lossless compression of ROI maps, we proposed novel schemes that can achieve high compression ratios by partitioning the intricate regions of the image into smaller blocks. Our analysis of a linear predictive model applied on the original images showed that the resulting residual images tend to have lower entropy. We then proposed to use the DPSO algorithm that searched for the best combination of scan directions to achieve better compression. Simulation results on various data sets showed that the proposed scheme outperformed JBIG2, the international standard for binary image compression. Our work also advanced the research on lossy attribute compression of
3D point clouds in two aspects. First, we proposed an adaptive scheme based on 3D Sobel filters that can switch between the RAHT and Dyadic RAHT to achieve early termination. We showed that the proposed method was able to achieve noticeable compression gains over the all-dyadic approach. Second, we further improved the adaptive method by training a neural networks so that the switching would no longer depend on a threshold. Compared to the GPCC standardized method that uses only Dyadic transform throughout the point cloud, the proposed scheme can achieve cumulative BR-rate gains on various publicly available datasets.
ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor and mentor, Dr. David Pan, for his unwavering support throughout my degrees. His patience and constant encouragement gave me confidence to pursue my research to completion. I would also like to thank my committee members Dr. Earl Wells, Dr. Vineetha Menon, Dr. Sivaguru S. Ravindran and Dr. Seeong-Moo Yoo for their insightful comments and help though my coursework with them.

I would like to express my gratitude to my family. It is their support, love and strong belief in me that have helped me to come to America and pursue my dreams. I would like to thank my friends Dr. Aditi Singh, Dr. Bernard Benson, Dr. Rajesh Reddy and Mr. Tamseel Mahmood Syed for being my family away from home. I would like to extend my special thanks to Ms. Aamira Indikar, Mr. Satyaki Goswami and Mr. James McNeill for always pushing me to do better. Without all these people, this journey would not have been possible.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>List of Figures</th>
<th>xii</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Tables</td>
<td>xv</td>
</tr>
<tr>
<td>List of Symbols</td>
<td>xvi</td>
</tr>
</tbody>
</table>

## Chapter

1 Introduction  

1.1 Need of Data Compression  

1.2 Types of Compression  

1.2.1 Lossless Compression  

1.2.2 Lossy Compression  

1.2.3 Near-Lossless Compression  

1.3 Data Compression Flow  

1.4 Media Formats  

1.4.1 Binary Images  

1.4.2 Point Clouds  

viii
1.5 International Compression Standards ..................................... 11
  1.5.1 Joint Bilevel Imaging Group 2 (JBIG2) .............................. 11
  1.5.2 Moving Picture Experts Group (MPEG) ............................ 12
1.6 Performance Metrics ............................................................. 14
1.7 Outline of Dissertation .......................................................... 16

2 Literature Review ..................................................................... 17
  2.1 Autoencoders ........................................................................ 19
    2.1.1 Point Cloud Geometry Autoencoder (PCG-AE) .................... 20
    2.1.2 RD Control Through Implicit and Explicit Quantization ..... 23
    2.1.3 Learning Convolutional Transforms (LCT) ....................... 26
    2.1.4 Learned Point Cloud Geometry Compression .................... 28
  2.2 Fully Connected Neural Network ............................................. 31
    2.2.1 Deep AE-based PCGC ................................................... 32
    2.2.2 FoldingNet: PC AE via Deep Grid Deformation ................. 34
  2.3 Recurrent Neural Network ..................................................... 37
  2.4 AI-based PC transmission .................................................... 39
  2.5 Conclusion ........................................................................... 42

3 Lossless Compression of Bi-Level ROI Maps ............................. 43
<table>
<thead>
<tr>
<th>4.2.2</th>
<th>3D edge detection</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.3</td>
<td>Simulation Results</td>
<td>86</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Conclusion</td>
<td>90</td>
</tr>
<tr>
<td>4.3</td>
<td>Threshold Dependency Removal using Neural Networks</td>
<td>91</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Problem Statement</td>
<td>93</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Neural Network Structure</td>
<td>97</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Results</td>
<td>100</td>
</tr>
<tr>
<td>4.3.4</td>
<td>Conclusion</td>
<td>106</td>
</tr>
</tbody>
</table>

5 Conclusion and Future Work

REFERENCES
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Data explosion</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Data compression flow</td>
<td>6</td>
</tr>
<tr>
<td>1.3 Binary image</td>
<td>7</td>
</tr>
<tr>
<td>1.4 Point cloud “RomanOilLight”</td>
<td>8</td>
</tr>
<tr>
<td>1.5 3D model of a human body</td>
<td>9</td>
</tr>
<tr>
<td>1.6 Example of a PLY file</td>
<td>10</td>
</tr>
<tr>
<td>1.7 G-PCC encoding scheme</td>
<td>12</td>
</tr>
<tr>
<td>1.8 BD-curves</td>
<td>15</td>
</tr>
<tr>
<td>2.1 Taxonomy of the proposed schemes in the literature.</td>
<td>19</td>
</tr>
<tr>
<td>2.2 PCG-AE coding solution.</td>
<td>21</td>
</tr>
<tr>
<td>2.3 DL-based PC coding architecture using implicit-explicit quantization.</td>
<td>23</td>
</tr>
<tr>
<td>2.4 LCT architecture.</td>
<td>27</td>
</tr>
<tr>
<td>2.5 Overview of learned PCGC.</td>
<td>29</td>
</tr>
<tr>
<td>2.6 Deep AE-based architecture.</td>
<td>32</td>
</tr>
<tr>
<td>2.7 FoldingNet architecture overview.</td>
<td>36</td>
</tr>
<tr>
<td>2.8 Residual calculation.</td>
<td>38</td>
</tr>
</tbody>
</table>
4.1 RAHT. ......................................................... 76
4.2 The forward and inverse RAHT Transform. ................. 78
4.3 Dyadic RAHT. ............................................... 79
4.4 Edge detection based on the neighbourhood. ............... 81
4.5 Sobel template masks. ...................................... 83
4.6 Sample solid cube with depth = 5. ........................ 85
4.7 Point cloud dataset. ........................................ 87
4.8 Two types of decomposition. ................................ 94
4.9 3D edge detection scheme. ................................ 95
4.10 T-SNE visualization for Pleno data. ....................... 97
4.11 Fully connected neural network (FCNN) architecture. .... 98
4.12 Point cloud dataset ....................................... 102
4.13 Training vs validation loss. ................................ 103
4.14 Training vs validation accuracy. .......................... 104
4.15 8iVFB dataset ............................................. 106
4.16 Confusion matrix of classification results. ................. 107
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison of machine learning based methods for PCC</td>
</tr>
<tr>
<td>3.1</td>
<td>Average bitrate in bpp</td>
</tr>
<tr>
<td>4.1</td>
<td>Cumulative gains of Dyadic RAHT w.r.t RAHT for sample cubes</td>
</tr>
<tr>
<td>4.2</td>
<td>BD-rate gains and approximate encoding-decoding time (s)</td>
</tr>
<tr>
<td>4.3</td>
<td>BD-rate cumulative gains</td>
</tr>
<tr>
<td>4.4</td>
<td>BD-rate gains for proposed scheme over Dyadic RAHT</td>
</tr>
<tr>
<td>4.5</td>
<td>Cumulative BD-rate gains for proposed scheme over Dyadic RAHT on 10 random frames (a)</td>
</tr>
<tr>
<td>4.6</td>
<td>Cumulative BD-rate gains for proposed scheme over Dyadic RAHT on 10 random frames (b)</td>
</tr>
<tr>
<td>4.7</td>
<td>BD-rate gains for proposed scheme over Dyadic RAHT on 8i voxelized full bodies (8iVFB) Dataset</td>
</tr>
</tbody>
</table>
### LIST OF SYMBOLS

<table>
<thead>
<tr>
<th>ACRONYMS</th>
<th>DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>2D</td>
<td>2-dimensional</td>
</tr>
<tr>
<td>3D</td>
<td>3-dimensional</td>
</tr>
<tr>
<td>8iVFB</td>
<td>8i voxelized full body dataset</td>
</tr>
<tr>
<td>AE</td>
<td>Autoencoder</td>
</tr>
<tr>
<td>AI</td>
<td>Artificial intelligence</td>
</tr>
<tr>
<td>ASCII</td>
<td>American standard code for information interchange</td>
</tr>
<tr>
<td>bpp</td>
<td>bits per pixel</td>
</tr>
<tr>
<td>bpip</td>
<td>bits per input points</td>
</tr>
<tr>
<td>BCE</td>
<td>Binary cross entropy</td>
</tr>
<tr>
<td>BDBR</td>
<td>Bjontegaard delta bitrate</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolution neural network</td>
</tr>
<tr>
<td>CR</td>
<td>Compression ratio</td>
</tr>
<tr>
<td>DL</td>
<td>Deep learning</td>
</tr>
</tbody>
</table>

xvi
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DNN</td>
<td>Deep neural network</td>
</tr>
<tr>
<td>DPSO</td>
<td>Discrete particle swarm optimization</td>
</tr>
<tr>
<td>FCNN</td>
<td>Fully connected neural network</td>
</tr>
<tr>
<td>GPCC</td>
<td>Geometry point cloud compression</td>
</tr>
<tr>
<td>JPEG2000</td>
<td>Joint photographic expert group 2000</td>
</tr>
<tr>
<td>JBIG2</td>
<td>Joint bi-level image expert group 2</td>
</tr>
<tr>
<td>LiDAR</td>
<td>Light detection and ranging</td>
</tr>
<tr>
<td>LSTM</td>
<td>Long short term memory</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi layer perceptron</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving picture expert group</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean square error</td>
</tr>
<tr>
<td>MVUB</td>
<td>Microsoft voxelized upper bodies dataset</td>
</tr>
<tr>
<td>PAE</td>
<td>Peak absolute error</td>
</tr>
<tr>
<td>PC</td>
<td>point cloud</td>
</tr>
<tr>
<td>PCC</td>
<td>point cloud compression</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>PCL</td>
<td>Point cloud library</td>
</tr>
<tr>
<td>PLY</td>
<td>Extension of the polygon file format</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak signal to noise ratio</td>
</tr>
<tr>
<td>QS</td>
<td>Quantization step</td>
</tr>
<tr>
<td>RAHT</td>
<td>Region adaptive hierarchical transform</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent neural network</td>
</tr>
<tr>
<td>ROI</td>
<td>Region of interest</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic gradient descent</td>
</tr>
<tr>
<td>SNNRMSE</td>
<td>Symmetric neighbour root mean square error</td>
</tr>
<tr>
<td>SVM</td>
<td>Support vector machine</td>
</tr>
<tr>
<td>VAE</td>
<td>Variational auto encoder</td>
</tr>
<tr>
<td>VPCC</td>
<td>Voxelized point cloud compression</td>
</tr>
<tr>
<td>VRN</td>
<td>Voxelation resnet</td>
</tr>
<tr>
<td>WBCE</td>
<td>Weighted binary cross entropy</td>
</tr>
</tbody>
</table>
Dedicated to my parents, Mr. S. K. Hooda, Mrs. Sheela Hooda, my brother, Erash Hooda and other struggling PhD students
CHAPTER 1

INTRODUCTION

...and God said, “let there be light.”

—Genesis 1:3

We have witnessed a prevalent and rapid development in the field of information technology merely in the last two decades but mainly the past decade has given birth to a data explosion. Due to the advent of portable and affordable wireless devices, there has been an enormous increase in the creation of digital data, majorly emanating from personal devices (mobile phones and personal computers). Data are now captured and shared at unprecedented levels [1]. Other portions of the data are relevant to transactional details generated by purchases at the point of sale associated with individuals via credit cards and/or store club card. It also includes surveillance data generated at banks, sports arenas, vending areas and on street corners/highways. Information relevant to medicine, seismological research and satellite also evidently contributes to the explosion of data. As reported in [2],
Figure 1.1 Data explosion

government and private responses to COVID-19 pandemic resulted in the generation and dissemination of personal data not previously available in the public sphere.

The number of active mobile devices surpassed the number of human beings back in 2014 [3]. Although not everyone in the world owns a mobile device, mobile phones are used by two-thirds of the global population [4]. Mobile devices are multiplying ten times faster than we are. With our population growing at a rate of about two people per second or 1.05% annually, Cisco estimates that there will be 3.6
connected devices per second and nearly 10 devices and connections per household by 2023 [5].

As of today, the world generates over 64 zettabytes of data in a year, that is, 64 trillion gigabytes of data, from a total of 23.8 billion connected devices. The data creation trend and prediction for next few years is shown in Figure 1.1. So, it is well established and accepted that the world is set to generate astronomical amount of digital information as highlighted in a The Wall Street Journal published on Dec 8, 2021 [6]. This rapidly growing voluminous data is commonly referred as ”The Data Explosion,” which poses a huge challenge and also a gigantic opportunity for engineers and researchers. One of the biggest challenge to address data explosion is data compression as detailed in the following section.

1.1 Need of Data Compression

Different types of data can be widely divided into audio, video and textual data. Among various categories, media information (images and videos) form the major contributing factor in causing the explosion. And with new advanced high resolution cameras, we capture images with all the intricate details and that leads to big overall file sizes that takes up all the space. This not only puts strain on storage, requiring large hard drives, memory cards, etc., but also strains processing
power resulting in slower devices. Therefore, for practical usage of media formats in aforementioned applications in the previous section, efficient compression has become an important concern.

1.2 Types of Compression

Different data compression techniques can be used to address the main challenge of limited storage space and transmission bandwidth. We can achieve significant reduction in the file sizes by removing irrelevance and redundancy in the data format. However, the type of compression to be used in the field of imaging has been a topic of discussion for a long time. Image compression techniques can be classified into three categories:

1.2.1 Lossless Compression

Lossless compression allows complete reconstruction of data without any loss of information. Although it is of limited use because of its modest compression performance, it is often preferred for applications that requires very high reconstruction quality such as compression of mammography images, which requires very high diagnostic accuracy and reversibility.
1.2.2 Lossy Compression

At the cost of degradation of image quality, lossy compression can provide a very high compression ratio and used in applications where losing some amount of image details is affordable. In case of medical imaging, it is sometimes required to state that lossy compression was used with information about approximate compression ratio (CR). But in other media applications such as entertainment, lossy compression offers a lot of advantages for example- switching to lower bit rates/quality in order to continue uninterrupted online streaming which has resulted in adoption of this type of compression by most companies including YouTube, Netflix and Amazon prime etc.

1.2.3 Near-Lossless Compression

This new compression technique was first established in the field of medical imaging to ensure the high diagnostic quality of compressed images. The visual quality achieved using near-lossless is as good as lossless as it produces imperceptible differences and therefore is often referred to as ”visually lossless”. It assumes that the peak of absolute error (PAE) between the original and reconstructed image should be user-defined in order to provide significant increase in compression rates
while providing quantitative estimates on the amount of distortion introduced by the compression algorithms.

1.3 Data Compression Flow

A basic data compression algorithm is an encoding-decoding system where the original media format (image or video) is fed to the encoder that produces a bitstream to represent the original media format in a compressed form with much smaller file size. The compressed file is transmitted over a network or channel and reconstructed at the decoder to produce the decoded data as close as possible to the original media format. The encoding algorithm involves preprocessing step, post processing step and an integral part where redundancies are exploited to squeeze out the important information. In contrast, the decoding algorithm performs the same steps as the encoding algorithm but in reverse order at the receiver. Figure 1.2 shows a basic data compression flow.

![Data compression flow](image)

**Figure 1.2** Data compression flow
1.4 Media Formats

We have considered contributions to compression of two different media formats which are discussed in the following sections.

1.4.1 Binary Images

A binary image is also known as a bi-level image, which consists of only two intensity values for each pixel. It is the simplest form of image which can be represented as a matrix with $m$ rows and $n$ columns comprising of zeroes and ones. It essentially becomes a series of black dots on a white background. This may not seem very useful, however, there are several good uses of bi-level images. They are used in a lot of applications such as ground truth in saliency detection, government agencies, banks, credit card companies, airline and insurance companies. A few
input output devices also use binary images which includes fax machines and some computer displays that can only handle binary images.

1.4.2 Point Clouds

A point cloud (PC) is an emerging media format which provides 3D representation of the world. A typical PC data contains up to a few millions of points scattered in 3D coordinate system similar to a pointillism art where each point has a location called geometry and color component called attributes. With the PC data being this voluminous, it is only natural for it to pose a huge challenge in transmitting and storing high quality PCs efficiently [7], [8].

![Figure 1.4 Point cloud “RomanOilLight”](image)

Figure 1.4 shows point cloud ”RomanOilLight” with different number of points in the set. The hull of Roman Oil Lamp becomes clearer as the number of points increases. The connectivity between the points in a PC can form an even more advanced media format called meshes as shown in Figure 1.5.
These fairly newer media formats can be created either synthetically or by capturing the real world scenes using stereo cameras in conjunction with kinetic cameras for smaller array of PCs usually in the indoor environment [9], [10]. On the other hand, LIDAR is commonly used in open environment scene capturing.

The fundamental element of PCs is a 3D extension of a pixel in 2D image called voxel which is a cube of size $1 \times 1 \times 1$. PCs are very sparse in nature as only 10% of voxels are occupied and are conveniently represented in a "polygon file format" with .ply extension that lists only the occupied voxel information as opposed to 2D image representation. A sample of a ply file is shown in Figure 1.6. It consists of a header that contains the number of points information (792192) and the bit depth of
Figure 1.6 Example of a PLY file with only first 27 lines

the point cloud followed by XYZ coordinates and RGB values. It can either be in a
binary format or ASCII format which requires more bits but is human readable [11].
1.5 International Compression Standards

A lot of international agencies have initiated several compression standardization efforts to address the data exploding problem over the past few decades. Different standards are used for different types of media formats. This section provides a brief summary with merits and demerits of two relevant compression standards that are used as a benchmark in this research.

1.5.1 Joint Bilevel Imaging Group 2 (JBIG2)

JBIG2 is a relatively new image compression standard that was first published in 2000. It offers both lossless and lossy compression and was primarily tailored for bi-tonal images. In JBIG2 compression algorithm, the bilevel image is segmented into three different regions (text, halftone and generic regions) and coded separately. Pattern matching & substitution (PM & S) and soft pattern matching (SPM) are used to encode textual data. Whereas halftone regions can be compressed using arithmetic coding. Finally, all three regions including generic regions are entropy coded using MQ coder. JBIG2 is capable of reducing files to as small as one fifth of its original size giving a compression ratio up to 100:1 when compared to other compression formats. While offering major advantages, one of the limitation of JBIG2 is that it can generate unexpected results in lossy mode.
1.5.2 Moving Picture Experts Group (MPEG)

MPEG issued a call of proposal back in 2017 which led to distinction between two compression technologies for point clouds i.e., video-based point cloud compression (V-PCC) and geometry-based point cloud compression (G-PCC). We have used G-PCC as the benchmark in our research. As mentioned in Section 1.4.2, the two main aspects of a PC is geometry and its corresponding attributes [12]. In G-PCC, the point compression task is divided into two main parts: Geometry Compression and Attribute Compression. The overview of G-PCC scheme is shown in Figure 1.7

![G-PCC encoding scheme](image)

**Figure 1.7** G-PCC encoding scheme

*Geometry Compression:* The first step in geometry compression is coordinate transformation as geometry points may be represented by floating point num-
bers. Quantization of floating points is performed in coordinate transformation which might result in duplicate points having the same geometry. Duplicate point removal is done but it raises an issue of attribute assignment which is addressed by averaging the attribute values corresponding to duplicate points. Then, the entire PC is contained in a 3D volumetric cube. Now, Voxelization is followed by coordinate transformation where the 3D volumetric cube is divided into 8 cubes with half the dimension. This process is repeated until the size of the voxel is reduces to $1 \times 1 \times 1$ and referred to as Octree decomposition. At each level, the occupancy of the cube is represented as '1' and unoccupied voxels are marked as '0'. Octree decomposition is coding the geometry in lossless manner. For lossy compression of the geometry, octree can be pruned followed by surface approximations which improve the connectivity information and quality in lower bitrates [13].

Attribute Compression: G-PCC offers three options for compression of attributes: 1) Region Adaptive Hierarchical Transform (RAHT), 2) Predicting Transform and 3) Lifting Transform. RAHT is a haar-inspired transform that starts from the highest level (leaves of octree) and move upwards to the lowest level (root of octree) [14]. The main idea behind this approach is to use the attribute values of higher octree level to predict the attribute values of lower level. However, the predicting transform employs nearest-neighbour prediction scheme. Lifting transform is an
extension of predicting transform with an extra update/lifting step. The transform
coefficients are further quantized as the last step.

Finally, the Octree bitstream and transform coefficients are fed to the arithmetic coder to generate final compressed bitstream of the point cloud.

1.6 Performance Metrics

The compression efficiency in case of a lossless binary image compression is usually measured in terms of bitrate (bits per pixel (bpp)) defined as follows:

\[
\text{Bitrate (bpp)} = \frac{\text{Size of compressed file}}{\text{Total no. of pixels in the original image}}
\]

and compression ratio (CR) defined as:

\[
\text{Compression ratio (CR)} = \frac{\text{Size of the original file}}{\text{Size of compressed file}}
\]

However, for lossy compression of point clouds, rate-distortion is used to evaluate the performance of a compression algorithm. The rate is the number of bits used to encode the point cloud expressed in terms of bits per input point (bpip) and distortion is the quality of the reconstructed point cloud measured in terms of peak signal to noise ratio (PSNR) defined as follows:
\[ PSNR = 10 \log_{10} \left( \frac{A_{\text{max}}^2}{MSE} \right) \]  

where \( A_{\text{max}} \) is the maximum signal amplitude and \( MSE \) is the mean square error between the original and the reconstructed point cloud. Since there are three channels in a point cloud, PSNR can be calculated for \( Y \), \( U \) or \( V \) channels, denoted as \( \text{PSNR}_Y \), \( \text{PSNR}_U \) and \( \text{PSNR}_V \) respectively. For rate-distortion curves, PSNR for only \( Y \) channel is used for simplicity as the three PSNRs are highly correlated. The \textit{pc_error} tool can be used to perform above mentioned calculation.

**Figure 1.8** BD-curves
Figure 1.8 shows BD curves performance comparison of three different methods i.e., A, B and C. For different rate points, PSNR is calculated and plotted on a graph followed by curve fitting through these rate-distortion (RD) points. Then, the PSNR difference between the methods is computed by calculating the area between the curves using integrals. A positive BD-PSNR value signifies gain as PSNR is higher for a given rate as for method C. On the other hand, the performance of method A and B are very similar.

1.7 Outline of Dissertation

The remainder of the dissertation is outlined in this section. Chapter 2 discusses few of the most recent advancements made in the field of point cloud compression using deep learning methods. Chapter 3 provides the details of our first main contribution where we proposed a smart algorithm to improve compression of bitonal images. Chapter 4 presents two-fold improvements in compression of point clouds. It entails experimental details supporting the motivation behind the idea which is also evident by offering substantial gain on variety of datasets. Finally, a general conclusion is given in chapter 5 along with some suggestion for future research directions.
CHAPTER 2

LITERATURE REVIEW

In the past decade, machine learning (ML) has become an emerging field that offers solutions to problems associated with image and video data [15]. These ML solutions also include data compression techniques with encouraging results that are comparable to traditional methods [16], [17]. Following the same research direction, ML techniques were also explored to offer potential solutions to point cloud compression. Additionally, the advances in artificial intelligence (AI) or ML has provided an opportunity to consider their use in real-time PC video streaming as well [18]. The transmission of PC data using AI-powered techniques provides more than 30 times compression ratio for their network conditions as highlighted in [18].

Although ML has the flexibility to exploit 3D correlations in the data, due to the non-uniformity and unstructured nature of point clouds, the application of ML techniques to point cloud compression (PCC) is not straightforward. Hence, most relevant ML methods first convert the 3D data into a corresponding 2D format as a
preprocessing step before feeding it to an ML-based compression algorithm [19]. This chapter provides a structured review of ML-based solutions to PCC in recent years. It includes the benchmark data sets with performance metrics used to evaluate previous contributions [20]. The comparison of all the reviewed methods is summarized in Table 2.1.

Given the fundamental representation of PC data, the research problems related to PCC are categorized into two groups:

1. Geometry Compression: Focused on compressing the locations of the points.

2. Attribute Compression: Aim at exploiting the redundancy in attribute values given the locations of the points.

Since ML approaches for PCC specifically pertaining to geometry compression [21] are relatively in their initial stages of development, a few recent contributions in this field form the main focus of this chapter. The taxonomy of schemes covered in this chapter is shown in Figure 2.1.

The rest of the chapter is organized as follows. Section 2.1 discusses the various CNN-based autoencoder implementations for PCC along with an evaluation of their merits and demerits. The use of fully connected convolutional neural network (FCNN) approach to PCC is presented in Section 2.2, and the recurrent neural network (RNN) based PCC approaches are provided in Section 2.3. The use of AI in
transmitting compressed PC data and potential advantages of using AI are discussed in Section 2.4. Section 2.5 summarizes and concludes the chapter.

2.1 Autoencoders

CNNs are known to be one of the most effective neural networks (NN) in extracting useful features in uniform structure sharing a pattern or correlation. These characteristics are mostly to be found in images and video resulting in wide usage of CNNs. And hence, most recently proposed deep learning (DL) based PC geometry compression techniques adopt autoencoder (AE) neural network design involving convolutional layers. This section describes some of the recently proposed PC geometry coding schemes using CNN based autoencoders.
**Table 2.1** Comparison of machine learning based methods for PCC

<table>
<thead>
<tr>
<th>Paper</th>
<th>Technique</th>
<th>Dataset</th>
<th>Benchmark</th>
<th>Metrics</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>[22]</td>
<td>CNN-AE</td>
<td>MPEG</td>
<td>PCL</td>
<td>RD curve, D1</td>
<td>SGD (BCE)</td>
</tr>
<tr>
<td>[23]</td>
<td>CNN-AE</td>
<td>MPEG</td>
<td>GPCC</td>
<td>RD curve, D1</td>
<td>RD Loss</td>
</tr>
<tr>
<td>[24]</td>
<td>3D CNN</td>
<td>ModelNet40</td>
<td>GPCC</td>
<td>D1, D2</td>
<td>RD Loss</td>
</tr>
<tr>
<td>[25]</td>
<td>Stacked CNN</td>
<td>ShapeNet</td>
<td>GPCC, PCL</td>
<td>D1, D2</td>
<td>RD Loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JPEG</td>
<td>VPCC</td>
<td></td>
<td>WBCE</td>
</tr>
<tr>
<td>[26]</td>
<td>FCNN MLP</td>
<td>ShapeNet</td>
<td>GPCC</td>
<td>D1</td>
<td>RD Loss</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JPEG</td>
<td></td>
<td></td>
<td>Chamfer Dist</td>
</tr>
<tr>
<td>[27]</td>
<td>Folding-based</td>
<td>ShapeNet</td>
<td>FC</td>
<td>-</td>
<td>Chamfer Dist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ModelNet10</td>
<td>Decoder</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[28]</td>
<td>RNN</td>
<td>TierIV</td>
<td>Octree, JPEG</td>
<td>SNNRMSE</td>
<td>-</td>
</tr>
</tbody>
</table>

**2.1.1 Point Cloud Geometry Autoencoder (PCG-AE)**

PCG-AE proposes a basic implementation of a DL approach for PCC to highlight the potential improvements of AE PC coding paradigm. The general 3D coordinate representation of PC is changed to a set of binary 3D blocks format where occupied voxels are signalled by either a '1' or '0'. Since points in PC are generally arranged in disorderly fashion, coding them using NN can be very challenging. Thus this type of formatting introduces some correlation similar to images and video to
be exploited by a CNN based AE in 3D domain. The underlying idea is to use an adaptive forward and inverse transform instead of using a fixed predefined transform such as DCT [22].

The details of the structure of transform is shown in Figure 2.2. First, the input PC geometry is divided into 3D binary voxeled blocks of size $32 \times 32 \times 32$ and the position of the 3D sub-blocks in original PC are sent as index in the final bitstream. Then AE is used to map PC geometry into the feature space and is addressed as latent representation as shown in Figure 2.2. So, the original 3D geometry is now represented by latents at the end of the encoding process with reduced dimensionality. These latents are then binarized using a certain threshold (for e.g. 0.5) to generate the bitstream.
At the decoder, the restored latent representation by decoding the incoming bit stream is then fed to the CNN AE with the aim of reconstructing the geometry as close as possible to the original geometry. The 3D blocks recovered the restored latents are then placed in their respective positions using the indices as side information to eventually reconstruct the entire PC.

**Adaptive Transform:** This implementation uses a very basic 3D CNN design which is an extension of the one typically used for 2D CNN image processing. It has 4 convolutional layers at the encoder and 4 at the decoder, each with width, depth and height of 3 and a stride of 2 to downsample and reduce the dimensionality of the latent representation. N represents the number of filters which is a hyperparameter and four models were trained for these N values: 32, 64, 96 and 128. N decides the reconstruction quality with the associated bitrate and hence affects the RD performance. N is tuned experimentally. All CNN layers use sigmoid as the activation function with stochastic gradient descent (SGD) as the convergence function and binary cross entropy (BCE) as the loss function to reconstruct the availability of voxels as accurately as possible. The decoder has a symmetric structure with downsampling replaced with upsampling by a factor of 2.

The training of the encoder and decoder is performed simultaneously. Point clouds from MPEG datasets were used for training and testing. Point cloud library
(PCL) codec was set as the benchmark to evaluate the performance of the proposed scheme. RD performance with ratio of reconstructed points and original points were used as the performance metric. It was concluded that using more filters (N) achieves higher quality as expected. Although for lower bitrates, PCL performs better than PCG-AE, but in case of medium and higher bitrates, PCG-AE outperforms PCL by a significant margin.

2.1.2 RD Control Through Implicit and Explicit Quantization

In traditional transform-based coding methods, RD loss function (more specifically Lagrangian multiplier $\lambda$) defined in (Equation 2.1) is used to control the RD trade-offs in DL-based coding solutions.

$$\text{Loss} = \text{dist} + \lambda \times \text{rate} \quad (2.1)$$

where $\text{dist}$ is the distortion between the original and reconstructed PC, and $\text{rate}$ is the entropy of the quantized latent representation. The extension of PCG-AE has

Figure 2.3 DL-based PC coding architecture using implicit-explicit quantization
been very recently proposed to provide flexible RD control functionality [23]. Typical RD control can be achieved using either of the following:

1. Implicit quantization which employs tuning between the entropy of latents shown in Figure 2.2 and the reconstruction error (controlled by $\lambda$).

2. Explicit quantization which employs a parameter quantization step (QS) to obtain different RD points.

Although the implicit quantization approach requires training of multiple DL models for each desired RD point, it performs substantially better than explicit quantization. The scheme in [2] suggests the use of the combination of implicit and explicit quantization, which reduces the training complexity and memory requirement to reach multiple RD points, when compared to implicit-only quantization coding. Therefore, RD trade-off is achieved using small number of trained DL models.

The overview of the scheme is depicted in Figure 2.3, where it uses the loss function defined in Equation 2.1 for training. Models trained for different values of $\lambda$ allow us to achieve various RD points without using explicit quantization. A new PC will use these trained models for compression.

Similar to the approach mentioned in the previous section, latent representation is entropy coded after quantization. But to have more efficient entropy coding,
variational autoencoder (VAE) is used to adapt the entropy model to current latent data.

When the model shown in Figure 2.3 is trained for $QS = 1$, it corresponds to implicit quantization even though we are explicitly specifying the amount of quantization desired. RD control can be achieved using higher value of QS ($QS > 1$ i.e., explicit quantization) to generate multiple RD points using the same DL model. The Scaling block down-scales the latent representation by QS, which is further entropy coded. At the decoder, the $Scaling^{-1}$ block up-scales the latent values by the same QS. And the entropy model adapts to a certain value of QS.

Now, to reach multiple RD points, single trained model can be used with

1. $QS = 1$ (implicit quantization)
2. $QS > 1$ (explicit quantization)

In addition, an implicit-explicit balance is to be made since the model was initially trained for $QS = 1$ and higher QS may incorporate some distortion. This happens because there is a certain limit to the number of RD points generated using single trained DL model.

Performance Evaluation: In this approach, unlike PCG-AE, PCs were divided into blocks of size $64 \times 64 \times 64$. The QS value was varied from 1 to 20. And for implicit-only scheme, $\lambda$ was varied from 50 to 20000. It was observed
that single model was unable to reach lower bitrate (as QS increases) when trained for higher bitrate and was unable to reach higher bitrate (as QS decreases) when trained for lower bitrates. Hence, implicit-explicit quantization scheme was not able to cover wide range of quality and rates unlike multiple DL models trained using implicit-only quantization. Therefore, to address this issue three models were used instead of a single model to cover a wide range of bitrates. Implicit-explicit quantization with fairly fewer models ($\lambda = 100, 1000, 10000$) could achieve similar performance (for some cases slightly better) as that of using 10 trained models ($\lambda = 100, 250, 500, 600, 750, 1000, 2000, 3000, 10000, 20000$) with implicit-only quantization.

### 2.1.3 Learning Convolutional Transforms (LCT)

LCT approach uses 3D convolutional autoencoder which directly operates on voxels. The decoding is interpreted as a classification problem to predict or infer the occupancy of the voxel by defining the geometry as a binary signal across the voxel grid. The architecture of the method is depicted in Figure 2.4, where $f_a$ and $f_s$ are the analysis and synthesis transform respectively. $N$ specifies the number of filters (which is 32 in the proposed scheme). $k^3$ and $s^3$ specify the filter size and strides in each direction respectively [24].
The latent values are represented as $y = f_a(x)$, which is further quantized using uniform quantization to obtain $\hat{y} = Q(y)$ and is decompressed to get $\hat{x} = f_s(\hat{y})$. $f_a$ and $f_s$ are learned during the training process. The encoder and the decoder use convolutions and transpose convolutions respectively with the same padding and stride of $2^3$. The $k$ value is 9 for the first layer, and 5 for the second and last layer with no activation function.

The scheme does not use any entropy coder, but instead uses a combination of LZ77 and Huffman coding to compress $Q(y)$. While decoding is considered as a classification problem, due to the sparsity of the point cloud, an imbalance is created between the filled and empty voxels. This issue is addressed using $\alpha$-balanced focal loss. However, the final loss is the same as defined in Equation 2.1.

The results showed that the model was able to generalize well even though it was trained using ModelNet40 mesh dataset (to cover a wide range of variety and quantity) and tested on completely different dataset (MVUB-Microsoft Voxelized
Upper Bodies dataset [29]). The proposed scheme outperformed MPEG anchor (described in [30], MPEG GPCC for static PCs and MPEG VPCC for video based PCs) for all the sequences and at all bitrates. For the similar bitrates, it achieves lower distortion and reconstructs more points compared to MPEG anchor, resulting in better quality. Different $\lambda$ values ($10^{-4}, 5 \times 10^{-5}, 10^{-5}, 5 \times 10^{-6}, 10^{-6}$) were used to generate RD points and different octree depths for the MPEG anchor. It improved the Bjontegaard-delta bitrate (BDBR) by 51.5% on average for the testing data.

2.1.4 Learned Point Cloud Geometry Compression

The LCT method was followed by other new approaches in the literature. One of them was Learned PCGC that uses DNN-based variational autoencoder. The overview of the scheme is illustrated in Figure 2.5.

**Preprocessing:** This module involves three steps: voxelization, scaling and partition. *Voxelization* is an optional step and is skipped if PC is already in 3D volumetric presentation. Cartesian coordinate system is used to perform voxelization, where a voxel is set to 1 if it is occupied and 0 otherwise. Inter-voxel correlation which can be exploited by 3D convolutions to generate very compact latent representation of a 3D block [25].
Similar to image down-sampling used in image and video compression, down-sampling was also used in PCGC in order to maintain the quality at lower bit rates. Scaling is incorporated to reduce the sparsity and hence increase the density of the PC. Doing so was found to improve the efficiency by a considerable amount, especially for sparser PC such as dynamically acquired point clouds (category 3) of MPEG dataset. The scaling is performed as follows:

$$\hat{X}_n = \text{round}(X_n \times s),$$  \hspace{1cm} (2.2)

where $(X_n \times s) = (i_n \times s, j_n \times s, k_n \times s)$, and $X_n$ is the original PC geometry for $n = 1, \ldots, N$. $X_n$ is scaled by $s$ where $s < 1$. Then the PC is partitioned into 3D non-overlapping cubes of size $W \times W \times W$. The respective position of the valid cube (at least one occupied voxel) with the number of occupied voxels in each cube is signaled explicitly, thereby adding a very small amount of overhead (metadata).
The 3D cubes are fed to the analysis transform one-by-one, which consists of 3D stacked CNNs, to generate a compact latent representation of size \((\text{channel, length, width, height})\) defined as \(y = f_e(x; \theta_e)\), where \(\theta_e\) is the set of convolutional weights. For decoding, the synthesis transform is so designed that quantized latent values \(\hat{y}\) are used to reconstruct the cubes, formulated as \(\tilde{x} = f_d(\hat{y}; \phi_d)\), where \(\phi_d\) is the set of parameters.

The basic 3D convolutional unit in the transforms used is Voxception-Resnet (VRN) because of its efficiency obtained due to the residual and inception network [31].

The scheme uses fairly smaller kernels \((3 \times 3 \times 3)\), offering lower complexity. Hyperprior is used for entropy modeling similar to the LCT approach.

**Post-processing:** It is comprised of classification, inverse scaling and extraction. Even though the median value of threshold is 0.5, it is shown to be not-optimal and hence, the Classification step uses adaptive thresholds to infer whether the voxels in the reconstructed cube \(\tilde{x}\) are present or not. The classification of the voxels in the cube is also influenced by the number of occupied points in the original cube. This information is sent as the metadata during encoding. Inverse scaling is implemented as the next step, which factors the reconstructed geometry by \(1/s\) for rendering and
display. *Extraction* is an optional step that changes the volumetric representation to raw file format such as polygon file format (.ply) or ASCII.

For model generalization, the scheme used ShapeNet as the training dataset, MPEG and JPEG pleno standardization datasets were used for testing. The models were trained from end-to-end for each bitrate by changing $\lambda$ (Lagrangian) and scaling factor s. Weighted binary cross entropy loss (WBCE) was used while training and adaptive thresholding was used for inference. Simulation results show that learned PCGC outperforms GPCC (octree) by 77\% & 69\%, GPCC (trisoup) by 67\% & 62\%, and it offers 88\% & 82\% gains against PCL for D1 (point to point) and D2 (point to plane), respectively.

### 2.2 Fully Connected Neural Network

CNNs not only retain the spatial relations in images and videos but also have the advantage of weight sharing, which significantly reduces the complexity of ML solutions. Therefore, CNNs have become the most common choice for solving vision-based tasks for their practical application, especially for advanced visual data representations such as PC and meshes [29]. Nonetheless, there are few approaches that directly work on the geometry coordinate of PC to achieve compression. Two
of those schemes are discussed in this section and compared against the previously summarized 3D CNN based AE solutions.

2.2.1 Deep AE-based PCGC

This scheme focuses on design of a codec that adapts to the features of PC for improved compression efficiency. The architecture of the proposed network is shown in Figure 2.6. It comprises of four main modules: a pointnet based encoder that acts as an analysis transform [32], a uniform quantizer, an entropy estimation block in the middle, and the decoder that acts as a non-linear synthesis transform. The geometry of a PC is represented as 3D points consisting of \( n \) points. These \( n \) points are downsampling first, to reduce the sparsity of the original PC. Sampling layer will result in duplicate points which are merged together resulting in fewer points \( (m) \), which are the subset of original sampled \( n \) points [26].

Next, the \( m \) points are fed to the encoder, which comprises of five 1D convolutional layers with kernel size of 1, followed by a \( ReLU \) and batch normalization.

![Figure 2.6 Deep AE-based architecture](image-url)
The filters in each layer is 64, 128, 128, 256 and $k$, respectively. Here $k$ is decided upon the number of input points. Then there is a max-pooling layer used to produce $k$-dimensional latent code $z$. The latent code is quantized and encoded by the entropy coder.

The decoder uses an FCN, which decodes the latent code $z$ using 3 fully-connected layers with hidden layers of 256, 256 and $n \times 3$ neurons to generate $n \times 3$ reconstructed PC. This whole process can be summarized using the following equation:

$$ z = D(Q(E(S(x)))) , $$

where $S$, $E$, $Q$ and $D$ are sampling layer, encoder, quantization and decoder, respectively. Analysis and synthesis transform are formulated as $z = f_e(x; \theta_e)$ and $\hat{x} = f_d(y; \phi_d)$, respectively, where $x$ is the original PC, $z$ is the compressive representation, and $\hat{x}$ is the reconstructed PC. $\theta_e$ and $\phi_d$ are the trainable parameters.

Quantization using rounding function makes the derivative 0 or undefined, rendering the loss non-differentiable. Therefore, the proposed scheme replaces quantization by an additive uniform noise.

$$ [f(x)] \approx f(x) + u $$

(2.4)
where \( u \) is the random noise. The distortion is computed using Chamfer Distance defined as follows:

\[
d_{CH}(S_1, S_2) = \sum_{x \in S_1} \min_{y \in S_2} ||x - y||_2^2 + \sum_{y \in S_2} \min_{x \in S_1} ||x - y||_2^2
\]

(2.5)

where \( S_1 \) is the original PC with \( n \) points and \( S_2 \) is the reconstructed PC with \( m \) points. The proposed scheme uses ShapeNet as the training and testing set from a single class with 90\% and 10\% split. The results are compared with MPEG anchor (TMC13). The model uses data from four categories: chair (train: 6101, test: 678), airplane (train: 3640, test: 405), table (train: 7658, test: 851) and car (train: 6747, test: 750). Deep AE-based PCGC achieves BD rate gain of 73.15\% on average over MPEG GPCC.

### 2.2.2 FoldingNet: PC AE via Deep Grid Deformation

A CNN requires its neighbors at a certain distance to share fixed spatial relation. Since PC does not exhibit such characteristics in raw format, voxelization is performed to make convolution operation on PC more meaningful. But voxelization sacrifices the details of the original representation of the PC. There are very few approaches that work on the original PC without voxelization. One of them is
FoldingNet recently proposed. It is a MLP based AE, which uses a 2D grid structure to construct a PC through Folding operation [27].

Figure 2.7 shows the FoldingNet architecture. The encoder comprises of MLP and graph based max-pooling layers. The geometry of the PC, \( n \times 3 \) (where \( n \) is the number of points) is the input to the encoder. Then the local covariance matrix of size \( 3 \times 3 \) for each point \( v \) is computed using the coordinates, which are one hop neighbor of \( v \) and vectorized to \( 1 \times 9 \). The covariances (\( n \times 9 \)) and geometry (\( n \times 3 \)) are concatenated to a matrix of size \( n \times 12 \) and fed to the 3 layer perceptron. The output of the perceptron is fed to two consecutive graph layers to eventually generate a codeword (\( 1 \times 512 \)), which maps a high-dimensional embedding of an input PC. The codeword is replicated \( m \) times and concatenated with fixed 2D grid points of size \( m \times 2 \) to generate a matrix of \( m \times 514 \). This matrix is the input to the 3-layer perceptron and generates the output of the first folding operation (\( m \times 3 \)). Then again, it is concatenated with replicated codewords and becomes the input to the second folding operation, which results in the reconstructed PC. Two folding operation can be perceived as a 2D to 3D transformation/mapping that produces elaborating surfaces. The first one folds the 2D grid to 3D space and the second one folds inside the 3D space. The number of output points \( m \) is not necessarily the same
as $n$. The distortion error between the input and the reconstructed PC is calculated using the Chamfer distance defined in previous section.

During training, the model starts with a random folding but eventually turns into a meaningful PC. The model was trained using 16 categories of ShapeNet dataset. One major advantage of Folding based decoder shown in Figure 2.7 is that it only uses 7% parameters of a decoder compared to that of a FCNN decoder and it is more focused towards solving the classification problem rather than compressing the PC. However, the idea could be adapted for PCC to validate the performance in comparison to international standardization. As far as classification problem is considered, the codewords generated at the end of the encoder are used to classify using SVM.

The folding-based decoder was compared against the FC decoder. During the training process, the reconstruction loss (Chamfer Distance) kept decreasing, which implies the reconstructed PC is becoming more and more similar to the original PC. And the classification accuracy of the linear SVM which was trained on the code-
words improved, implying that the codeword becomes more linearly separable while training. For validation, the ShapeNet trained model was used for transfer learning and tested on ModelNet dataset that achieved an 88.4% classification accuracy.

2.3 Recurrent Neural Network (RNN)

There are very limited ML approaches that use RNN for point cloud compression. One of them is discussed in this section that focuses on data from 3D LiDAR sensors. The conventional methods that are part of the standardization are not very successful in reducing the point cloud to a very low volume, especially 3D LiDAR sequences. Therefore, some approaches operate on the raw packet data from a 3D LiDAR, which is shown to be highly cost effective approach for compression. Although it is possible to transform raw packet data into 2D matrix losslessly, they are very irregular in nature and do not share any spatial correlation, if taken without calibrations. The advantage of doing so is reduced sparsity of the PC which is helpful for the further processing [28].

The proposed scheme targets static PC and introduces a RNN based compression method with residual blocks. First, 2D matrices are constructed by transforming the frames of raw packet data. 2D-matrix normalization is then performed as a preprocessing step before the data is fed to the network comprising of an encoder, a
Figure 2.8 Residual calculation

Binarizer and a decoder. As 2D format can have a much larger range of possible values due to the accuracy and long range LiDAR detection scanners, the scheme uses a few steps for normalization: The packet data are randomly chosen from 100 frames to calculate the mean $\mu$ and the histogram of distance values are created and the data matrix $R$ is normalized as follows: $(R - \mu)/\theta$ where threshold $\theta$ is set such that 95% of the distances fall under this value.

The encoder, binarizer and decoder in Figure 2.8 each comprise of a combination of convolutional layers and convolutional LSTM layers. The decoder also uses residual blocks because getting an accurate decompression can be challenging while using such a network. Specifically, the decoder iteratively uses incoming binary file to reconstruct the original input and the residual is computed by comparing the reconstructed data with the original PC. This residual becomes the input for next iteration. This process is repeated until the decompressed data are reasonably ac-
accurate. The process of generating this residue is also shown in Figure 2.8. The decompression loss is computed using SNNRMSE (symmetric neighbor root mean squared error) [28]. The scheme is tested on driving data from 11 areas of Japan and is shown to outperform the octree compression and JPEG compression benchmark methods.

### 2.4 AI-based PC transmission

Transmission of PC video data over resource constrained channels is a challenge. Moreover, real-time transmission of PC video data using the existing bandwidth of available networks is difficult. Hence the availability of a compression scheme for PC data is very useful to accomplish such PC video streaming applications.

As mentioned previously, the use of AI/ML to assist in compression of PC data yields promising results. Likewise, the use of AI/ML in assisting with the transmission of PC video data is discussed in this section. In [18], an AI enabled technique called AITransfer is proposed. This method uses an AI-powered adaptive transmission technique, which is bandwidth-aware. The reduction in bandwidth consumption and the alleviation of computations is achieved by only extracting and transferring the key features from the PC. Thereby, as mentioned in [18], one of the benefits of using the AITransfer technique is the incorporation of dynamic network
bandwidth in the design of an end-to-end architecture using fundamental concepts like feature extraction and reconstruction. Another contribution of AITransfer is the use of online adapter to sense the network bandwidth for a matching optimal inference model.

Figure 2.9 shows the architecture of AITransfer highlighting its components and system flow obtained from [18]. As can be seen in Figure 2.9, this technique extracts key features from the PC data collected by multiple view cameras using an...
AI-based neural network. Extracting just these features compresses the PC data and makes it more compatible to the networks bandwidth handling capacity. The feature extracted data are then transmitted using the existing network and reconstructed at the intended terminals. The feature extraction steps are presented in Figure 2.10.

From Figure 2.10, it can be seen that the various layers used in the AI-based network for extracting key features have a few abstraction steps, followed by convolutional layers, dimensionality changes and more convolutional layers, which yields the compressed PC output. These data are then transmitted according to the network conditions and subsequently reconstructed at the terminals like cell-phones,
VR headsets, etc. [18] presents a case study to evaluate the performance of AITransfer and shows that a compression ratio of almost over 30 to 1 can be attained.

### 2.5 Conclusion

This chapter presents a detailed survey on an emerging research area of point cloud compression (PCC) using deep learning and machine learning techniques. Due to early stage of development in this area, a summary of only 8 research articles was presented in this survey. In addition to the summary of the most recent developments in this area, we present a comparative performance study of the surveyed techniques.

The majority of PCC techniques using DL/ML proposed in the literature are related to the geometry compression, with very few techniques focused on attribute compression of PC and video compression. It was also seen that most of the approaches use convolutional based auto-encoders, multi-layer perceptrons and fully connected neural network decoders to achieve PCC. Most recently, the use of RNN based techniques to perform PC compression was also studied and one such method is also included in this chapter.

Transmission of PC or compressed PC data is also challenging, hence making it an emerging topic area. A most recent simultaneous AI based compression and transmission technique called AITransfer is also briefly discussed.
CHAPTER 3

LOSSLESS COMPRESSION OF BI-LEVEL ROI MAPS

3.1 Introduction

Hyperspectral imaging is used in a wide range of applications such as remote sensing, medical diagnose, biotechnology and surveillance. A hyperspectral image is a 3D cube with spectral and spatial information providing discrimination capabilities. The high volume of hyperspectral data collected by the sensors put strain on storage and transmission. Therefore, efficient compression of hyperspectral data have become an important concern [33] [34]. To address the main challenges of limited storage space and transmission bandwidth, we can achieve significant size reduction by preserving the Region of interest (ROI) [35]. A region of interest map indicates the location of pixels that belong to the region of specific interest to a certain applications such as medical imaging and nuclear medicine [36] [37]. We can define an ROI by creating a bi-level map, which is a binary image that is of the same size as
the original image. An entry of "1" in the map means that co-located pixel in the original hyperspectral image belongs to the ROI. A "0" entry means otherwise [38]. Due to irregular nature of the ROI maps, achieving high compression on such images can be challenging and very critical in applications with limited bandwidth.

Although entropy coding such as Huffman coding, run-length coding and context modelling form the basic steps employed by most state-of-the-art lossless compression techniques, but in the recent years, neural networks based solution have become popular in hyperspectral image compression field achieving results that are comparable to conventional methods. To reduce the training time complexity shallow NN have been used to perform adaptive filtering for better prediction [39]. LSTM could also be used to accommodate long-term data dependency in hyperspectral imagery for reducing prediction errors [40]. Among conventional methods, incorporation of a deep belief network has been used for parameter estimation of Golomb-rice coding that offers reduction in computations over brute force search methods [34]. But pure conventional method involves band reordering and regrouping to introduce spectral redundancy [41]. And since ROI maps fall under the category of binary images, international standards include JBIG2 and JPEG2000 [42]. Other popular schemes include rectangular partitioning based algorithm [43] [44] designed for
images containing graphs and tables, and prediction algorithms that are used to improve the redundancy to achieve higher compression gains [45].

To the best of our knowledge, the proposed scheme [46] is a first attempt tailored specifically for compression of ground truth, classification maps and ROI maps, which are employed in remote sensing applications, saliency detection etc. The main contribution of the proposed method are as follows:

1. A fixed prediction algorithm is used as a preprocessing step to exploit the spatial relationship of adjacent pixels in ROI maps to offer more compressibility.

2. Discrete Particle swarm Optimization (DPSO) is used. It adapts to the non-uniform nature of the ROI maps and finds the absolute optimal scanning direction for efficient compression.

We proposed to use prediction algorithm as a preprocessing step for exploiting spatial relationships of adjacent pixels in ROI maps. In our previous work, we proposed a tree-based algorithm that uses full search for the partition of the image into non-uniform blocks and the data file is compressed using Bzip2 [47]. Although full search technique is the most simplistic search method that guarantees to find the absolute optimal solution but it is an exhaustive method with large computational complexity, which has limited effect when compressing a single image [48] [49]. To
lower the time complexity, we introduced to use Discrete Particle Swarm Optimization [50] [51] as an efficient method for exploration in partitioned image.

The rest of the chapter is organized as follows. In Section 3.2, we illustrate the prediction scheme with details of the image grid patterns and scanning directions employed. Section 3.3 discusses the problem statement and Section 3.4 comprises of all the steps of the optimization algorithm used. In Section 3.5, the experimental results are provided. The chapter is concluded in Section 3.6.

3.2 Preprocessing

The correlation amongst the neighbouring pixel leads to redundancy, which is a common characteristics shared by most ROI images. Binary image compression techniques are concerned with exploiting these redundancies present in the image, which can be quantitatively described by "Entropy". An image with a higher entropy typically means the image is harder to compress [52]. Pixel prediction leads to residuals that tend to have lower entropy than the original image. Further compression is achieved by partitioning the image using an optimization algorithm. The preprocessing step is described in sections as follows:
3.2.1 Predictive Model

Predictive model utilizes the correlation between adjacent pixels in both spatial dimension. Although it is advantageous to have a high order predictor, for practical implementation we have limited the order to a small number (e.g., $N = 4$), to maintain lower complexity and less overhead [53]. The ordering of the pixels are based on Euclidean distance from the target pixel as shown in Figure 3.1. We use $x(i, j - 1), x(i - 1, j), x(i - 1, j - 1)$ and $x(i - 1, j + 1)$ (neighbouring pixels) to predict the target pixel $x(i, j)$. The predicted pixel $\hat{x}(i, j)$ is a linear combination of its causal pixels and a bias. In the case of $N^{th}$ order predictor we define $N + 1$ dimensional vector. From Figure 3.1, for $N = 4$ we consider 5 dimensional vector as 

$\{1, x(i, j - 1), x(i - 1, j), x(i - 1, j - 1), x(i - 1, j + 1)\}$.

From Figure 3.2, the predicted pixel can be expressed as: $\hat{x}(i, j) = \alpha_0 + \alpha_1 x(i, j - 1) + \alpha_2 x(i - 1, j) + \alpha_3 x(i - 1, j - 1) + \alpha_4 x(i - 1, j + 1)$ where $\alpha'$s are real numbers called prediction coefficients.
3.2.2 Numerical Solution

The LS (least square) optimization is used to calculate the optimal prediction coefficient [54] [55]. Given the size of the image being \( m \times n \), our objective is to find LS solution for the system:

\[
\begin{align*}
P \alpha &= y \\
&P^T P \alpha = P^T y
\end{align*}
\]

where \( P \) is an \((mn) \times (N + 1)\) matrix shown in Figure 3.3. In matrix \( P \), each row corresponds to 4 neighbouring pixels and a bias associated with each target pixel. \( \alpha = [\alpha_0, \alpha_1, \alpha_2, \alpha_3, \alpha_4]^T \) are prediction coefficients to be determined, and \( y = [x(1,1) \ldots x(i,j) \ldots x(m,n)]^T \) is an \( mn \) dimensional vector with target pixels. From Equation 3.1 we have:

\[
P^T P \alpha = P^T y
\]
Equation 3.2 has a unique solution:

$$\alpha = (P^T P)^{-1} P^T y$$  \hfill (3.3)

Using the optimal solution, $\hat{x}(i, j)$ where $1 \leq i \leq m, 1 \leq j \leq n$ is calculated, a predicted value for $x(i, j)$. Thereafter, the output $\hat{x}(i, j)$ of the predictor module is compared with the original value $x(i, j)$ to construct an error (residual) image as follows:

$$E(i, j) = \hat{x}(i, j) \oplus x(i, j)$$  \hfill (3.4)
where \( \oplus \) is modulo 2 addition, which is realized by an \( XOR \) operation. Finally, the lower entropy error image \( E(i, j) \) with optimal predictor coefficients \( \alpha \) is fed to the optimization algorithm discussed in the next section for further compression.

### 3.2.3 Grid Patterns

To take advantage of long-range and short-range correlations, the error image is divided into variable-size blocks based on its statistics. The idea is to partition highly actively changing portion of the image into smaller blocks and have larger blocks for less complicated regions in the image to divide pixel-concentration. First, the image is divided into 4 equally sized blocks and then further division takes place by partitioning each non-zero sub-block into 4 sub-blocks. We examine a total of 16 patterns as shown in Figure 3.4. An image can be segmented, at least into 4 blocks while the maximum number of blocks can go to 16.

### 3.2.4 Scanning Directions

Image pixels are scanned following a certain direction to convert the image block to a one-dimensional vector. As the distribution of pixels are non-homogeneous in ROI maps, three different scanning directions are used [56]. An image block can be scanned in Horizontal, Vertical or Zig-Zag raster scan direction, generating different intervals for different scanning path as shown in Fig. Figure 3.5.
Figure 3.4 16 possible grid patterns for partitioning an image
Interval Generation: The interval sequence generated affects the degree of compression. It is calculated as the distance between the previous and next occurrence of the same bit (referred to as Intervals) following a scanning direction mentioned above. Due to the complimentary nature of the bit values in binary images, distance coding focuses on symbol ”1” to generate the intervals.
3.3 Problem Statement

An image of size $m \times n$ is divided into $D$ blocks where $D \in \{4, 7, 10, 13, 16\}$ as shown in Figure 3.4 and each block can be scanned in one of the three scanning directions to exploit horizontal, vertical and diagonal redundancy in an image. Therefore, the search complexity increases as $3^D$. The value of $D$ determines the dimensions of the search space as $S \in \{3^4, 3^7, 3^{10}, 3^{13}, 3^{16}\}$. A full search method to find the optimal combination of scanning direction out of $S \in \{81, 2187, 59049, 1594323, 43046721\}$ combinations for maximum compression is an exhaustive process and time consuming. Discrete PSO algorithm can be employed to find the best combination of the scanning directions to obtain the largest compression on the intervals. Therefore, for a chosen pattern, each block is scanned in a particular direction as determined by the search algorithm.

Let us consider an illustrative example, where an image of size of $12 \times 12$ takes up one of the patterns mentioned in Section 3.2.3, by being divided into $D = 13$ blocks as depicted in Figure 3.6. The scan path for all the blocks are referred to as Direction Bits. Now, let us assume the direction bits shown in Figure 3.7(a), as the optimal one for this image. The direction bit is set to “0” if the horizontal direction is chosen for a block; a direction bit of “1” is set for vertical scanning
Figure 3.6 An image (12 × 12) divided into smaller blocks

Figure 3.7 Bitmap and interval generation
patterns; otherwise, a direction bit of “2” is set, implying zig-zag direction. The interval sequence is generated by scanning the blocks accordingly.

3.4 Algorithm

3.4.1 Overview

The goal is to search for the best combination of scanning direction using Discrete Particle Swarm Optimization (DPSO) to minimize the compressed file size. It was inspired by bird flocking or fish schooling behavior. In a DPSO system, each particle represents the candidate solution to the optimization problem and swarm of particles is considered to fly through the search space. The position or movement of the particles is guided by the experience of the neighboring particles and experience of the entire swarm [57]. A particle’s best position is termed as $P_{best}$ and the global best of the entire swarm is termed as $G_{best}$. Fitness functions are used to measure the performance of each particle [58].

3.4.2 Parameter Initialization

The particles in the DPSO model are initialized randomly with a vector representing the scanned path which are discrete in nature, since a block can be scanned in only three directions, i.e., H, V, or Z. The dimension of the vector depends on the
total number of blocks an image is partitioned into. Each particle records coordinates of its current position, coordinates of best position so far and velocity vector in a D-dimensional space. After initializing the particles with random initial positions and velocity, evaluation of all the candidate solutions is performed, which becomes the candidate solution fitness, known as $P_{best}$ and the best among all the candidate solution is the global best $G_{best}$. Hence, the first swarm is generated.

### 3.4.3 DPSO Model

DPSO is initialized with a swarm size of $N_{pop}$ particles. Each particle is treated as a point in the $D$-dimensional space, where $D$ is the number of blocks the image is divided into. The $i^{th}$ particle is represented as $x_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where $x_{id} \in \{0, 1, 2\}$ and $1 \leq i \leq N_{pop}, 1 \leq d \leq D$. The velocity of particle $i$ is represented as $v_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. The previous best position of the $i^{th}$ particle is represented as $p_i = (p_{i1}, p_{i2}, \ldots, p_{iD})$ and the best among the entire swarm is represented by $p_g = (p_{g1}, p_{g2}, \ldots, p_{gD})$.

At each step, DPSO changes the velocity of each particle towards its $P_{best}$ and $G_{best}$ location. The velocity of a particle can be updated according to the following equation:

$$v_{id} = \omega v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}), \quad (3.5)$$
where $\omega$ represents the inertial weight used to control the search capability of the DPSO algorithm. Thus, in order to improve both global and local search capability of a particles, we choose a value of $\omega = 0.998$ for the momentum term after some tests and tuning.

The position of the particle can be updated using:

$$
x_{id} = \begin{cases} 
0, & \text{if } \sigma(v_{id}) < 0.5 \\
1, & \text{if } \sigma(v_{id}) > 0.5 \\
2, & \text{if } \sigma(v_{id}) = 0.5 
\end{cases} \tag{3.6}
$$

where

$$
\sigma(v_{id}) = \frac{1}{1 + e^{v_{id}}}. \tag{3.7}
$$

In Equation 3.5, $c_1$ and $c_2$ are called acceleration coefficients that controls the effect of cognitive ($c_1 r_1 (p_{id} - x_{id})$) and social component ($c_2 r_2 (p_{gd} - x_{id})$) on the overall velocity, which are updated using the method shown in [59]. $r_1, r_2$ are random numbers uniformly distributed in the range of $[0, 1]$. $t$ is the current iteration and $Iter_{max}$ is the maximum number of iterations.

The velocity of a particle is influenced by three factors called momentum term ($\omega v_{id}$), cognitive component ($c_1 r_1 (p_{id} - x_{id})$) and social component ($c_2 r_2 (p_{gd} - x_{id})$)
Figure 3.8 History preservation scheme

\( x_{id} \)). \( \omega \) represents inertial weight used to control the search capability of the BPSO algorithm. Thus, in order to improve both global and local search capability of a particles, we choose a value of \( \omega = 0.998 \) for the momentum term after some tests and tuning.

3.4.4 Particle History Preservation

Due to the involvement of random numbers in evaluation of velocity in Equation 3.5, there are high chances that same positions can be encountered, which were visited in previous iterations. The proposed method records the history of all the
previously visited positions and their corresponding cost value to avoid re-evaluation of the fitness value for the same position. Particle history preservation scheme shown in Figure 3.8 would effectively reduce the overall computational complexity by avoiding redundant fitness calculations associated with candidate position that have been visited previously.

3.4.5 Fitness Function

The objective function also referred to as fitness function maps the search space to the function space. Figure 3.9 summarizes the fitness function used in the proposed method. The function space has only one output (1-D) providing a fitness value for each set of parameters (candidate solutions) which determines their optimality.

In the proposed method, a pattern is selected based on the statistics of the input image, for which we divide into non-uniform blocks. The blocks are scanned to generate intervals employing the directions recommended by the DPSO algorithm in its current iteration. If the direction bit is “0”, “1” or “2”, the block is scanned in Horizontal, Vertical or Zig-Zag direction respectively. Finally, the sequence of intervals and direction bits are combined into a data file and fed to an entropy coder (arithmetic coder in our implementation) to output the compressed file size as the
Figure 3.9 Flowchart of compression algorithm
output of the fitness function, i.e., mapping the search space into a function space. A header is also attached to the data file to synchronize the direction bit sequence with the sequence of the intervals. With every iteration, new direction bits are generated by the BPSO algorithm and the minimum file size ($G_{\text{best}}$) that can be achieved for an image is found out.

### 3.5 Results

To examine the effectiveness and performance of the proposed scheme, we tested our method on six various datasets listed in Table 4.2. Figure 3.10 shows SalinasA dataset with sample band, ground truth and the 7 ROI maps.

The proposed algorithm was applied to the ROI maps and compared against five other lossless image compression scheme which includes CCITT, Fax3, Fax4, JPEG2000 and an international standard for bi-level images (JBIG2). The simulation results for SalinasA is shown in Figure 3.11. It shows that our method achieves significantly higher compression for biased ROI maps (classes 1-6) and outperforms all the other methods on average. For non-biased ROI map (class 0), it gains compression efficiency comparable to JBIG2.

To more comprehensively evaluate the performance of the proposed scheme, we have included the average bitrate and computing time in Table 4.2 with
Figure 3.10 SalinasA dataset
Table 3.1 Average bitrate in bpp

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ROIs</th>
<th>Proposed</th>
<th>Time(s)</th>
<th>JBIG2</th>
<th>Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>IP</td>
<td>0:16</td>
<td>0.0166</td>
<td>∼ 0.21</td>
<td>0.0644</td>
<td>∼ 0.61</td>
</tr>
<tr>
<td>144 × 144</td>
<td>1:16</td>
<td>0.0093</td>
<td></td>
<td>0.0574</td>
<td></td>
</tr>
<tr>
<td>SalinasA</td>
<td>0:6</td>
<td>0.0740</td>
<td>∼ 0.22</td>
<td>0.1841</td>
<td>∼ 0.62</td>
</tr>
<tr>
<td>80 × 80</td>
<td>1:6</td>
<td>0.0423</td>
<td></td>
<td>0.1735</td>
<td></td>
</tr>
<tr>
<td>Salinas</td>
<td>0:16</td>
<td>0.0141</td>
<td>∼ 0.73</td>
<td>0.0154</td>
<td>∼ 0.63</td>
</tr>
<tr>
<td>512 × 208</td>
<td>1:16</td>
<td>0.0063</td>
<td></td>
<td>0.0133</td>
<td></td>
</tr>
<tr>
<td>PU</td>
<td>0:9</td>
<td>0.0397</td>
<td>∼ 2.47</td>
<td>0.0284</td>
<td>∼ 0.64</td>
</tr>
<tr>
<td>608 × 336</td>
<td>1:9</td>
<td>0.0217</td>
<td></td>
<td>0.0192</td>
<td></td>
</tr>
<tr>
<td>KSC</td>
<td>0:13</td>
<td>0.0056</td>
<td>∼ 3.78</td>
<td>0.0055</td>
<td>∼ 0.65</td>
</tr>
<tr>
<td>512 × 608</td>
<td>1:13</td>
<td>0.0013</td>
<td></td>
<td>0.0046</td>
<td></td>
</tr>
<tr>
<td>Botswana</td>
<td>0:14</td>
<td>0.0021</td>
<td>∼ 5.02</td>
<td>0.0046</td>
<td>∼ 0.66</td>
</tr>
<tr>
<td>1472 × 256</td>
<td>1:14</td>
<td>0.0011</td>
<td></td>
<td>0.0038</td>
<td></td>
</tr>
</tbody>
</table>

respect to the second best method (JBIG2). Since few of the ROI maps of Pavia University are not highly biased compared to other ROI maps shown in Figure 3.12, we expect that the proposed method will be a little less efficient and will be comparable to JBIG2 as demonstrated in Figure 3.13. It achieves reasonable compression on average compared to other lossless schemes. More specifically, for ROI map 0, 1 and 8, which are globally scattered and regionally unbiased, the proposed algorithm achieves the second best compression efficiency, followed by Fax4. Similar improvements were observed for the other ROI maps as demonstrated in Table 4.2 and simulation results are shown in Figure 3.14, Figure 3.15, Figure 3.16 and Fig-
Figure 3.11 Simulation results for SalinasA dataset

Figure 3.17. Lossless check was performed after reconstruction of the decoded images with the original image pixel by pixel.

Computational Complexity: Computational complexity of the proposed scheme can be divided into two sub-parts.
Figure 3.12 Pavia University dataset
Figure 3.13 Simulation results for Pavia University dataset
1. Preprocessing: Computation of optimal prediction coefficient (equation Equation 3.3) takes $O((N+1)^2mn)$ to multiply $P^TP$, $O((N+1)mn)$ to multiply $P^Ty$ and $O((N+1)^3)$ to compute $P^TP^{-1}$. Now we know that $O((N+1)^2mn)$ dominates $O((N+1)mn)$ and since $mn > N+1$, $O((N+1)^2mn)$ also dominates
And computation of error image $E$ has complexity of $O(mn)$. Therefore, the total time complexity of the preprocessing step is $O((N+1)^3mn)$.

2. DPSO: The complexity of DPSO can be estimated using computation cost of initialization ($T_{init}$), main search algorithm ($T_s$) and fitness evaluation ($T_{FE}$) step. Since initialization of positions, velocity and computing personal best
and global best is considered as one operation, $T_{\text{init}}$ is $O(N_{\text{pop}})$. For the search algorithm, DPSO algorithm has an inner loop when going through population $N_{\text{pop}}$ and an outer loop for iteration $t$. So the $T_s$ at extreme case is $O(N_{\text{pop}}^2t)$. Although the computation cost is relatively inexpensive, since $T_s$ is linear in
terms of $t$, the most computationally extensive part is the fitness evaluations ($FE$s). The complexity of fitness evaluation is dependent on dimensionality of the search space $D$ and size of the image ($mn$). Therefore, $T_{FE}$ is $O(D \times mn)$, where $D$ in our case $\in \{4, 7, 10, 13, 16\}$. As the size of the image increases, the overall time increases due to increase in $T_{FE}$, which is validated by recorded time in shown in Table 4.2.
3.6 Conclusion

We presented a new scheme for compression of ROI maps of hyperspectral images. It is comprised of a preprocessing step that takes advantage of the correlations between the neighboring pixels and improves the sparsity of the ROI maps. In the next step, we divide the image into smaller blocks and use an optimization algorithm to choose an optimal scanning direction for each block, thereby resulting in a significant improvement on compression efficiency. The simulation results on six hyperspectral datasets show that the proposed algorithm has compression performance that is comparable to or higher than other standard methods by significant percentages.
CHAPTER 4

LOSSY 3D POINT CLOUD COMPRESSION

4.1 Introduction

After 3D meshes, point cloud (PCs) are the most advanced media format used in data representation. They comprise of scattered points in space where each point is represented using a spatial coordinate \((x, y, z)\) called Geometry with color and/or reflectance information (normal vectors) related to it called Attributes [60], [61]. Although a mesh provides far more complex geometry and sub-metric inspection of an object, point clouds are more widely used due to some limiting factors of meshes, mainly linked to complexity. Moreover, dense PCs are used in the development of 3D meshes to generate their finely detailed faces, edges and vertices [62]. Therefore, PCs can also be perceived as building blocks of a mesh. In addition, point cloud is considered a powerful 3D visual representations by providing a very realistic and interactive experience to the users [63], [64]. Due to the popularity of PCs in recent
years, they are employed in various applications such as immersive media, geographical information system, medical tomography, autonomous driving, augmented reality (AR), robotics, etc.

With increasing usage of inexpensive 3D scanners and modern multibeam echosounders, there has been a rise in generation of very high volume of dense point cloud datasets [65], [66]. To efficiently represent the details entailing 3D scenes, an enormous amount of information is required. As the precision of the PC increases, the storage and the bandwidth requirements for transmission also increase [60], [67]. In addition, because of the unstructured nature of these PCs, unlike traditional 2D images/videos, PC compression can be extremely challenging [68], [69]. Therefore, in many practical applications such as smooth streaming with limited bandwidth, efficient PC coding solutions become essential [70], [71].

Paramount efforts have been made by researchers to improve the compression efficiency of point clouds. The MPEG and JPEG community have initiated activities (MPEG call for proposal (CfP) in January 2017) related to the standardization of PCC [60], [72], which is now widely used as a benchmark in academic and non-academic research [73], [74]. The two distinct technologies are Geometry-based PCC (G-PCC) and Video-based PCC (V-PCC). From the details of the codec architecture mentioned in [60], it can be concluded that due to PC's unstructured nature, the
geometry and attribute information are generally encoded separately. The geometry is encoded using a popularly known Octree approach whereas, the attribute information can be encoded using a Haar-inspired transform called RAHT [73], [75], [70]. For dynamic data, these technologies mainly comprises of 3D to 2D projection and rule-based traditional approaches [76]. Several recent works have validated the effectiveness of dyadic decomposition-based methods but are used only in clinical applications such as detection of glaucoma [77] and diagnosis of COVID-19 [78]. There were some improvements reported by shifting from RAHT to Dyadic RAHT for attribute encoding in PCs. Dyadic RAHT is a variation of RAHT which has been recently incorporated in the PCC standard [79], [80]. Dyadic RAHT transforms both low and high frequency components, thereby exploiting redundancies in both the sub-bands, whereas only low frequency components are transformed further in the case of RAHT [68], [81]. The existing standard has the limitation of using only one type of decomposition throughout the point cloud limiting the adaptability to its changing statistics which is resulting in reduced gains. We propose to improve to the adaptability of the transforms and hence improve the compression on PCs.

The rest of the chapter is divided into two subsections that provides the explanation for the two main contributions made for efficient attribute compression of 3D point clouds.
4.2 Early Termination of Dyadic RAHT

In this section, we elaborate one of the proposed schemes to switch between the two variants of RAHT based on the characteristics of the blocks to be transformed. Early termination in Dyadic RAHT leads to a change in the BD-rate performance. The idea is to use Dyadic RAHT for denser and highly varying areas, while using RAHT for sparser and uniform areas in the point cloud. 3D Sobel operator-based edge detection is employed in order to identify the characteristics of the block to be transformed [82], [83], [84]. The proposed method is tested on publicly available datasets and compared to all-Dyadic RAHT approach, which is an improvement on all-RAHT scheme.

The rest of the Section 4.2 is organized as follows: Section 4.2.1 provides detailed comparisons of the two types of decomposition used. Section 4.2.2 describes the switching scheme based on 3D edge detection. Section 4.2.3 includes the simulation results and analysis of the BD-rate performance on the PC datasets. Section 4.2 is summarized in Section 4.2.4.

4.2.1 Types of decomposition

At first, three distinct techniques were defined by the MPEG group for the compression of different categories of PC data: Video-based PCC (V-PCC) for dy-
namic content, Surface PCC (S-PCC) for static surfaces, and LiDAR PCC (L-PCC) for dynamically acquired LiDAR sequences. S-PCC and L-PCC shared some similarities due to which they were later on merged together and referred to as Geometry-based PCC (G-PCC) [65], [85]. In G-PCC, the geometry can be coded in a lossy manner by pruning the octree. For coding of attributes, one of the options in G-PCC is RAHT, which has been upgraded to Dyadic RAHT.

![Figure 4.1 Region Adaptive Hierarchical Transform (RAHT)](image)

**Figure 4.1** Region Adaptive Hierarchical Transform (RAHT)
RAHT is a low-complexity hierarchical transform that is an extension of Haar transform in 3D domain. Let us consider a signal $V$ containing $N$ elements. At decomposition level $l$, two adjacent voxels $V_{l+1,2n}$ and $V_{l+1,2n+1}$ are transformed to generate low- and high-pass coefficients $G_{l,n}$ and $H_{l,n}$ as follows:

$$
\begin{bmatrix}
G_{l,n} \\
H_{l,n}
\end{bmatrix} =
\begin{bmatrix}
\alpha & \beta \\
-\beta & \alpha
\end{bmatrix}
\begin{bmatrix}
V_{l+1,2n} \\
V_{l+1,2n+1}
\end{bmatrix},
$$

(4.1)

where

$$
\alpha = \sqrt{\frac{w_1}{w_1 + w_2}}, \quad \beta = \sqrt{\frac{w_2}{w_1 + w_2}},
$$

(4.2)

and $w_1$, $w_2$ are the respective weights of the voxels $V_{l+1,2n}$ and $V_{l+1,2n+1}$. Note that the transform is orthonormal, i.e., $\alpha^2 + \beta^2 = 1$. The butterfly diagram used to explain the forward transform is shown in Figure 4.2. Its corresponding inverse transform is also shown in Figure 4.2 and is defined as follows:

$$
\begin{bmatrix}
V_{l+1,2n} \\
V_{l+1,2n+1}
\end{bmatrix} =
\begin{bmatrix}
\alpha & -\beta \\
\beta & \alpha
\end{bmatrix}
\begin{bmatrix}
G_{l,n} \\
H_{l,n}
\end{bmatrix}.
$$

(4.3)

In G-PCC, the RAHT decomposition is performed on $2 \times 2 \times 2$ (=8) blocks in three steps. Since, two adjacent voxels are transformed at a time, the first
Figure 4.2 The forward and inverse RAHT Transform

step involves transforming 4 pairs of blocks to split the $2 \times 2 \times 2$ blocks into low (L) and high (H) frequency components along the $z$-direction each of dimension $2 \times 2 \times 1$. Only the L sub-band is further transformed along the $y$-direction in step 2 to produce LL and LH (two pairs of blocks are processed) of dimension $1 \times 2 \times 1$. Finally, one pair of blocks are transformed to generate one DC coefficient (LLL) of dimension $1 \times 1 \times 1$ and 7 AC coefficients highlighted in dashed box in Figure 4.1 and Figure 4.3. One operation of RAHT decomposition involves $7 \times (4 + 2 + 1)$ transformations.

The dyadic RAHT decomposition is also performed on 8 blocks at a time, but now the high frequency components are also transformed further along the $z, y$ and $x$ directions, making a total of $12 \times (4 + 4 + 4)$ transformations. The compression performance was improved by replacing RAHT with Dyadic RAHT decomposition,
which helps by decorrelating the high frequency coefficients. Thus dyadic RAHT was incorporated in the standard compression scheme.
4.2.2 3D edge detection

Let us consider a case where all the $2 \times 2 \times 2$ blocks are occupied, whose attribute values are to be transformed. Although the dyadic decomposition can lead to better performance, the number of transformations will now increase to 12 when compared to RAHT, which requires only 7 transformations. Therefore, the complexity increases by a factor of $12/7$ [79]. However, switching to dyadic decomposition will not lead to much increased complexity in lower occupancy cases.

To this end, we propose to study the characteristics of the neighbouring parent blocks in order to decide the type of decomposition needs to be employed in transforming the central block. The idea is to use RAHT for uniform areas and use dyadic decomposition for discontinuous region (which needs further breaking down of energy) in the point cloud. The goal is to improve the compression performance without much increased complexity.

Now, let us consider a point or block at a certain level in the octree which is to be transformed, referred to as the central block and surrounded by neighbouring blocks. The discontinuities in the central block shown in red in Figure 4.4 are identified using its 18 neighbour parent blocks that share a face or an edge with the central block to be transformed [86]. We have considered 18-neighbourhood of the central block, since it covers more patterns with fewer masks [82], [87]. To adapt
the decomposition to the characteristics of the PC, a 3D Sobel operator is utilized to determine the discontinuities in the central block. To evaluate the strength of the gradient $G(x, y, z)$, the proposed algorithm uses only the Luma component values.
of the 18 neighboring blocks for simplicity. The 3D Sobel operator is an orthogonal gradient operator. The gradient of a PC \( f \) at position \((x, y, z)\) is defined in terms of directionally oriented spatial derivatives as:

\[
\nabla f(x, y, z) = G(x, y, z) = [G_x G_y G_z]^T
\]

(4.4)

where

\[
G_x = \frac{\partial f(x, y, z)}{\partial x} = S_x(x, y, z) \ast f(x, y, z),
\]

(4.5)

\[
G_y = \frac{\partial f(x, y, z)}{\partial y} = S_y(x, y, z) \ast f(x, y, z),
\]

are gradients in \( x, y \) and \( z \) direction, which point to the direction of the maximum rate of change in \( f \) at coordinate \((x, y, z)\). The operator uses the 3D Sobel edge kernels \((S_x(x, y, z), S_y(x, y, z), S_z(x, y, z))\) (shown in Figure 4.5) to perform the convolution (denoted by \( \ast \) in (Equation 4.5)) in order to calculate the gradients \( G_x, G_y, G_z \).

The magnitude of the gradient is calculated as:

\[
\text{mag}(\nabla f) = \|\nabla f\| = \sqrt{G_x^2 + G_y^2 + G_z^2}.
\]

(4.6)
For faster computation, the $\text{mag}(\nabla f)$ can be approximated using absolute values defined as:

\[
\text{mag}(\nabla f) \approx |G_x| + |G_y| + |G_z|
\]  \hspace{1cm} (4.7)

**Figure 4.5** Sobel template masks
Then $\text{mag}(\nabla f)$ is normalized using the average of the available blocks to find a ratio $k$ (where $0 \leq k \leq 1$) as:

$$k = \frac{\text{mag}(\nabla f)}{\text{average}}. \quad (4.8)$$

If $k$ exceeds a certain threshold ($T$), we then decide central block to be continuous region and is to be transformed using RAHT decomposition; otherwise dyadic is used. The decomposition at point $(x, y, z)$ will be:

$$d(x, y, z) = \begin{cases} 
\text{RAHT}, & \text{if } k < \text{Threshold} \\
\text{Dyadic}, & \text{Otherwise}
\end{cases} \quad (4.9)$$

The research idea stems from the intuition that RAHT decomposition will be more efficient for flat regions and dyadic decomposition for discontinuous region in the point cloud. To demonstrate that, we generated three solid cubical point clouds with non-uniform region and depth $d = 4, 5,$ and $6$ consisting of $4096, 32768$ and $262144$ points respectively and can be generalized as $N \times N \times N$ where $N = 2^d$. A sample of one of the generated point clouds is shown in Figure 4.6. All-RAHT was considered as the baseline model for comparisons while we slowly start introducing dyadic decomposition by tuning the threshold ($T$). When the threshold is set to 0, it corresponds to All-RAHT, and when the threshold is set to 1, it corresponds to
All-Dyadic. Table 4.1 shows the cumulative gain over three-channels as threshold increases. As expected, we observe improvements in gains as we move towards more and more of dyadic decomposition for the sample cubes, confirming that dyadic is efficient for such cubes which mostly comprises of regions full of edges. Therefore, based on the properties of a certain region in the PC, suitable decomposition is used to exploit the redundancies while saving on the number of transformation by avoiding dyadic in the case of uniform areas in the PC.

Figure 4.6 Sample solid cube with depth = 5
Table 4.1 Cumulative gains of Dyadic RAHT w.r.t RAHT for sample cubes

<table>
<thead>
<tr>
<th>Depth</th>
<th>No. of Points</th>
<th>RAHT vs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>T=0.2</td>
</tr>
<tr>
<td>4</td>
<td>4096</td>
<td>-0.7%</td>
</tr>
<tr>
<td>5</td>
<td>32768</td>
<td>-3.1%</td>
</tr>
<tr>
<td>6</td>
<td>262144</td>
<td>-3.7%</td>
</tr>
</tbody>
</table>

4.2.3 Simulation Results

To validate the efficiency of the proposed method, we tested our algorithm on three types of point cloud datasets. One LiDAR (overpass sequence), six static objects (Boxer, Egyptian mask, Loot, Red and black, Thai dancer and Basketball player) were chosen from the MPEG dataset [88] and frames extracted from dynamic sequences of MVUB (Microsoft Voxelized Upper Body [89]) dataset. The point cloud test set is illustrated in Figure 4.7.

The geometry and attributes were encoded in lossy manner. Since dyadic RAHT was an improvement on RAHT, we have compared the proposed algorithm to dyadic RAHT. To incorporate both bitrate and quality measurement, the evaluation was based on bitrate-distortion (BD) performance. The bitrate is calculated as the total number of bits in the bitstream by the total number of input points in PC and
Figure 4.7 Point cloud dataset (from left to right, top to bottom): Boxer, Phil, Ricardo, Andrew, Sarah, Red and black, Basketball player, Thai dancer, Loot, Egyptian mask, overpass
the distortion is measured in terms of PSNR (peak signal to noise ratio) [63]. To make the comparison for wide range of bitrates, six point BD-rates were computed.

**Table 4.2** BD-rate gains and approximate recorded encoding-decoding time (s) for proposed scheme over Dyadic RAHT

<table>
<thead>
<tr>
<th>Test Sequences</th>
<th>No. of Points</th>
<th>BD-rate</th>
<th>All Dyadic</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Luma</td>
<td>Cb</td>
<td>Cr</td>
</tr>
<tr>
<td>Boxer</td>
<td>3493085</td>
<td>2.7%</td>
<td>-6.0%</td>
<td>-5.7%</td>
</tr>
<tr>
<td>EM (vox12)</td>
<td>272684</td>
<td>-0.9%</td>
<td>-5.0%</td>
<td>-3.5%</td>
</tr>
<tr>
<td>Loot</td>
<td>3017285</td>
<td>0.3%</td>
<td>-0.4%</td>
<td>-1.0%</td>
</tr>
<tr>
<td>Red and black</td>
<td>757691</td>
<td>-0.1%</td>
<td>-1.0%</td>
<td>0.1%</td>
</tr>
<tr>
<td>Thai dancer</td>
<td>3130215</td>
<td>-1.2%</td>
<td>-0.5%</td>
<td>-0.9%</td>
</tr>
<tr>
<td>EM (vox20)</td>
<td>272689</td>
<td>0.1%</td>
<td>-2.9%</td>
<td>-2.5%</td>
</tr>
<tr>
<td>Basketball player</td>
<td>2925514</td>
<td>3.1%</td>
<td>-9.2%</td>
<td>-4.9%</td>
</tr>
<tr>
<td>Andrew</td>
<td>279664</td>
<td>-0.7%</td>
<td>-9.0%</td>
<td>-1.1%</td>
</tr>
<tr>
<td>Phil</td>
<td>370798</td>
<td>0.7%</td>
<td>-2.3%</td>
<td>-0.2%</td>
</tr>
<tr>
<td>Ricardo</td>
<td>960703</td>
<td>-1.0%</td>
<td>-0.6%</td>
<td>-2.6%</td>
</tr>
<tr>
<td>Sarah</td>
<td>302437</td>
<td>-0.9%</td>
<td>-1.2%</td>
<td>-2.3%</td>
</tr>
<tr>
<td>overpass_q1mm</td>
<td>5255920</td>
<td>0.5%</td>
<td>-1.2%</td>
<td>-1.0%</td>
</tr>
</tbody>
</table>

The results comparing the proposed method and dyadic RAHT are summarized in Table 4.2. A positive value of BD-rate corresponds to loss and a negative value corresponds to PSNR gains at the same bitrate. After some tests and tuning, a threshold value of 0.2 (for Boxer, Basketball player and overpass), 0.4 (for Egyptian mask (vox20)) , 0.6 (for Egyptian mask (vox12), Phil and Ricardo) and 0.8 for the
remaining PC were used. From Table 4.3, we observe that a maximum cumulative gain of 11% BD-rate gain was achieved with minimum cumulative gain of 1.7% in the worst case. In a majority of the datasets, BD-rate improvements can be observed. Therefore, the results demonstrate that it is advantageous to switch between the two types of transform by determining the discontinuity in the central block using 3D edge detection, instead of using either only RAHT or dyadic RAHT.

Table 4.3 BD-rate cumulative gains for proposed scheme over Dyadic RAHT

<table>
<thead>
<tr>
<th>Test Sequences</th>
<th>No. of Points</th>
<th>Cumulative Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>T=0.2</td>
</tr>
<tr>
<td>Boxer</td>
<td>3493085</td>
<td>-9.0%</td>
</tr>
<tr>
<td>EM (vox12)</td>
<td>272684</td>
<td>3.2%</td>
</tr>
<tr>
<td>Loot</td>
<td>3017285</td>
<td>5.7%</td>
</tr>
<tr>
<td>Red and black</td>
<td>757691</td>
<td>2.0%</td>
</tr>
<tr>
<td>Thai dancer</td>
<td>3130215</td>
<td>6.7%</td>
</tr>
<tr>
<td>EM (vox20)</td>
<td>272689</td>
<td>-1.8%</td>
</tr>
<tr>
<td>Basketball player</td>
<td>2925514</td>
<td>-11.0%</td>
</tr>
<tr>
<td>Andrew</td>
<td>279664</td>
<td>0.4%</td>
</tr>
<tr>
<td>Phil</td>
<td>370798</td>
<td>6.6%</td>
</tr>
<tr>
<td>Ricardo</td>
<td>960703</td>
<td>8.0%</td>
</tr>
<tr>
<td>Sarah</td>
<td>302437</td>
<td>-1.4%</td>
</tr>
<tr>
<td>overpass_q1mm</td>
<td>5255920</td>
<td>-1.7%</td>
</tr>
</tbody>
</table>
Computational Complexity: The computational complexity of the proposed algorithm includes the computational cost of 3D edge detection, which is performed using convolution operation. Computation of 3D convolution for an image of dimension $N \times M \times K$ with filter sizes of $p \times q \times s$ is $O(N MKpq)$, which can be further reduced to $O((N+p)(M+q)(K+s) \log((N+p)(M+q)(K+s)))$ using Fast Fourier Transform with some memory cost. We have used the direct implementation with an extreme case of 3493085 voxels convolved with $p = q = s = 3$, i.e., $3 \times 3 \times 3$ filter, which in turn is expected to have a nominal effect on the encoding and decoding time as validated by the approximate recorded time shown in Table 4.3.

4.2.4 Conclusion

We have introduced a new scheme that allows switching between RAHT and Dyadic RAHT to adapt to the unordered nature of 3D point clouds. In order to effectively exploit the redundancies in uniform and non-uniform areas in a PC, the switch is based on the characteristics of the neighbourhood. The continuity and discontinuity are recognized by 3D edge detection using the Sobel operator. The experimental results on two different types of datasets show that the proposed algorithm achieves good BD-rate cumulative gains of up to 11% when compared to using only one type of transform throughout the point cloud without increasing the
computational complexity. The limitation of the presented work lies in its threshold dependency which is currently tuned.

In next section, we elaborate a framework that focused on addressing the threshold dependency of the switching scheme. The idea was to use the tuned threshold values from the proposed technique in Section 4.2 to train a neural network in order to achieve generalized applicability and eliminating the threshold dependency [90].

4.3 Threshold Dependency Removal using Neural Networks

In addition to the conventional coding solutions implemented on PCs, deep learning (DL) has also made its way in advance media compression with impressive preliminary results [22]. Most of the DL coding solutions are 3D CNN-based autoencoders (AE) [24], [25] with few fully-connected neural network (FCNN) approaches [26], [27] and even fewer recurrent neural network (RNN) [28] based techniques. Among the limited neural network based solutions, only a handful of them are end-to-end. In [91], one of the first few end-to-end framework for lossy attribute coding using AE is introduced. There are also few partitioning-based methods such as [92] which segments the PC into fine-grained patches. In contrast, [93] uses kd-tree based decomposition to efficiently divide the color distribution. The coding
gains of the above mentioned approaches are reported to be comparable and in some cases outperforms the MPEG-anchor. However, these approaches need to train large models to generate rate-distortion curves and they are also data dependent [94].

In G-PCC, attributes are encoded using Region Adaptive Hierarchical Transform (RAHT), separately from geometry [81]. Due to the effectiveness of Dyadic decomposition [75], Blackberry proposed to replace RAHT with Dyadic RAHT which was later adopted in the codec [79]. Then, as detailed in Section 4.2, it was observed that switching between the two types of decomposition was found to be more effective instead of using only one type of transform throughout the PC [95]. But the approach was threshold dependent, thereby hindering its practical usability. In this chapter, we discuss an approach to address the threshold tuning problem of the technique in [95] by training neural networks to learn the switch between RAHT and Dyadic RAHT. Experimental results show considerable compression gains while successfully eliminating the threshold dependency.

The remainder of the section is organized as follows: Section 4.3.1 provides the problem statement with details related to data preprocessing. Section 4.3.2 describes the shallow neural network (SNN) coding solution. Section 4.3.3 discusses the BD-rate performance of the proposed technique. Section 4.3.4 concludes Section 4.3.
4.3.1 Problem Statement

One of the options to encode the attribute values of a PC is RAHT, which is an adaptive variation of a Haar wavelet transform introduced in [73]. It is based on the hierarchical structure of the occupancy map of the PC called Octree. The transform is applied in three steps as shown in Figure 4.8.

The RAHT was later replaced by Dyadic decomposition. Changing the fundamental structure of RAHT to Dyadic showed $\sim 2.3\%$ average gain on the entire MPEG dataset and hence was adopted in the G-PCC codec.

It was found that BD-rate gains can be achieved by switching between RAHT and Dyadic RAHT transforms. The switching was performed based on the characteristics of the neighboring blocks [95]. The nature of the neighboring blocks was studied using the 3D Sobel filter kernels ($S_x$, $S_y$ and $S_z$) to output the strength of the edge ($\nabla f$) defined as follows:

$$\nabla f(x, y, z) = G(x, y, z) = [G_x G_y G_z]^T$$  \hspace{1cm} (4.10)

Where $G_x$, $G_y$ and $G_z$ are gradients in $x$, $y$ and $z$ direction respectively. The magnitude was approximated using the absolute values for faster computation as defined below:
Figure 4.8 Two types of decomposition: Let us consider the vertical rectangular block in the first step in the left represents the 8 attribute values that are decomposed (shown in dashed lines) to generate low-pass (L) and high-pass (H) components. In the second step, only low-pass coefficients from the first step are decomposed to output LL and LH. Finally, LL is transformed to generate LLL and LLH. In Dyadic decomposition shown in the right, high-pass components at each stage are also transformed.
| \( (\nabla f) \| \approx |G_x| + |G_y| + |G_z| \) \hspace{2cm} (4.11)

The magnitude of the edge (\( \| \nabla f \| \)) was normalized using the average (avg) to compute \( k \). This process can be perceived as the luma values (represented as \( L_i \) where \( 1 \leq i \leq 19 \)) of 18 neighbors and central block multiplied with constant weights (\( \alpha's \)) as shown in Figure 4.9, where \( \alpha's \) represents the fixed weights of the Sobel filter. Normalized value \( k \) was used to make a binary decision based on a threshold. The disadvantage of the proposed method is its threshold dependency on \( k \), which was selected by trial and error for each point cloud. Therefore, threshold dependency hinders the general applicability of this switching scheme.

Neural network (NN) based compression methods for PC have emerged recently with comparable compression gains, albeit with the necessity to store multiple
trained models to generate different rate-distortion trade-offs [23]. In this contribution, we address the switching problem by replacing the original 3D edge filter scheme with a shallow neural network. The problem now becomes a pattern classification task with two output classes (RAHT and Dyadic RAHT).

Data Preprocessing: Only Luma values of 18 neighboring blocks with the central block was used in the original scheme and tested for threshold values of $T = 0.2, 0.4, 0.6$ and $0.8$. The threshold with maximum cumulative gain was selected. The 19 features with the transform chosen (0 for RAHT and 1 for Dyadic) based on the manual tuning was written in a data file to prepare the training dataset. Data cleaning is performed by first separating the 0 (RAHT) samples with 1 (Dyadic RAHT) samples to study their distribution. Duplicate data samples were dropped from both the classes and concatenated together followed by random shuffling.

Data Visualization: We use t-SNE to visualize data in two dimensions. A random sample of 15000 data points is used to create the embedding which are then projected onto a two dimensional plane to make it easier for visualization [96]. Here we observe more of Dyadic points compared to RAHT making the data biased as shown in Figure 4.10. Since performing Dyadic decomposition majority of the time and using RAHT decomposition for very weak edges was found to be beneficial, it was expected for the data samples to be more biased towards the Dyadic class.
Figure 4.10 T-SNE visualization for Pleno data

4.3.2 Neural Network Structure

The PC geometry is encoded using the octree approach, where the PC is enclosed in a 3D volume of $D \times D \times D$ voxels. The 3D volume is divided into 8 sub-cubes of size $D/2 \times D/2 \times D/2$. Only occupied voxels are divided further and represented by '1', and '0' otherwise. This process is repeated until the dimension reduces to $1 \times 1 \times 1$. Since the occupancy information is required for the attribute compression method chosen by the user, the geometry is encoded first.
Figure 4.11 Fully connected neural network (FCNN) architecture
For attribute compression using either RAHT or Dyadic RAHT, the octree representation is also considered. Let us consider a certain region in a 3D PC. The block to be transformed is highlighted in orange as shown in Figure 4.11 and referred to as the central block. Now, similar to the prediction scheme [86], the original scheme of 3D edge detection also uses 18 neighbors around the central block. 19 Luma values (18 neighbors plus the central block) were used in detecting the edges in the central block to decide the type of decomposition to be used. These 19 values form features for the target class based on the original scheme used in [95].

Although most deep learning based coding solutions for PC compression use CNN-based architecture to retain 3D correlations and maintain lower complexity via weight sharing. They need to store multiple models to obtain different RD curves with extensive running time required in training large networks [27]. In our problem, we encounter a very localized region in the PC where at a time, a maximum of 19 Luma values are processed. Therefore, we have opted for a fully connected structure as shown in Figure 4.11. Instead of using a set of 3D blocks, directly fed to the neural network, we flatten the 18 neighbors and the central block and use 19-tuple feature vectors. The architecture has an input layer of 19 neurons with an output layer of 1 neuron for the target label ('0' is used for RAHT and '1' is used for Dyadic RAHT), which is essentially a binary classification problem. We have used 2 small hidden
layers of 12 and 5 neurons respectively, making it a total of 4-layer architecture including the input and output layers. All the layers are fully connected. A sigmoid function was used for the final layer and the ReLU activation function was used for the remaining three layers. The FCNN architecture as shown in Figure 4.11 is used for training with the data split of 70%, 15%, 15% to divide it into training, validation, and testing set respectively. The model was trained for 1000 iterations with the binary cross entropy (BCE) loss function. Adam optimizer with a learning rate of 0.01 and weight decay of $1 \times 10^{-6}$ was used for fast convergence. Regularization with $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and dropout with probability of 0.1 in the third layer was used to avoid over-fitting and to improve generalization on the biased data. Finally, the trained model was imported into the MPEG-GPCC codec replacing the original 3D edge detection scheme which was threshold dependent.

4.3.3 Results

This section presents the performance assessment of the proposed method to eliminate the threshold dependency. Learning based coding solutions are generally most effective for dataset that share some similarity that uses the adaptation from the training data onto the testing data. In this context, we used Microsoft Voxelized Upper Bodies (MVUB) dataset [89] from the open source JPEG Pleno database,
Table 4.4 BD-rate gains for proposed scheme over Dyadic RAHT

<table>
<thead>
<tr>
<th>Test Sequences</th>
<th>No. of Points</th>
<th>BD-rate</th>
<th>Cumulative Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>277038</td>
<td>-0.9%</td>
<td>-8.4%  -3.6% -12.9%</td>
</tr>
<tr>
<td>Phil</td>
<td>336323</td>
<td>0.2%</td>
<td>-5.4%  -2.1% -7.3%</td>
</tr>
<tr>
<td>Ricardo</td>
<td>952178</td>
<td>-1.2%</td>
<td>-3.8%  -3.8% -8.8%</td>
</tr>
<tr>
<td>Sarah</td>
<td>304528</td>
<td>-0.9%</td>
<td>-4.9%  -4.4% -10.2%</td>
</tr>
<tr>
<td>David</td>
<td>302584</td>
<td>-2.9%</td>
<td>-3.7%  -2.9% -9.5%</td>
</tr>
</tbody>
</table>

which is a dynamic point cloud dataset publicly available. Each sequence consists of multiple frames that share correlations between the frames within a sequence. We trained the neural network shown in Figure 4.11 with specifications provided at the end of the previous section.

Table 4.5 Cumulative BD-rate gains for proposed scheme over Dyadic RAHT on 10 random frames (a)

<table>
<thead>
<tr>
<th>Test Sequences</th>
<th>Cumulative Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Andrew</td>
<td>-12.9%</td>
</tr>
<tr>
<td>Phil</td>
<td>-7.3%</td>
</tr>
<tr>
<td>Ricardo</td>
<td>-8.8%</td>
</tr>
<tr>
<td>Sarah</td>
<td>-10.2%</td>
</tr>
<tr>
<td>David</td>
<td>-9.5%</td>
</tr>
</tbody>
</table>

101
The model was trained using only the first frame of each of the four sequence and tested on 10 random frames. Figure 4.12 shows the first frames of Phil, Ricardo, Sarah and Andrew sequence. Accuracy of 92.78%, 92.54%, 92.38% was achieved on the training, validation and testing data respectively. The loss curve is shown in Figure 4.13 and accuracy curve is shown in Figure 4.14 for training and validation set.
The Dyadic RAHT approach was used as the benchmark to assess the performance of the proposed technique and RD curves were used as the performance metrics to observe the gain across the three channels (Luma, Cb, Cr). Table 4.4 shows the gains over each channels for a random frame that achieved the highest gain. The confusion matrix of the classification results are shown in Figure 4.16. The instances or counts in the confusion matrix can also be expressed in terms of percentages. The proposed scheme achieved 92.37% of accuracy with high sensitiv-
ity, specificity and precision of 95.88%, 88.88% and 89.69% respectively, showing the accuracies obtained are not skewed by uneven test data.

To summarize the RD performance of the proposed scheme, the cumulative gain can be used, which is calculated by simply adding the gain or loss across the three channels. The cumulative gain on 10 random frames from the four dynamic sequences is tabulated in Table 4.5 and Table 4.6 arranged in decreasing order. Our method provides average cumulative gain of 8.53%, 3.75%, 5.06%, 5.72% and 5.87%
Table 4.6 Cumulative BD-rate gains for proposed scheme over Dyadic RAHT on 10 random frames (b)

<table>
<thead>
<tr>
<th>Test Sequences</th>
<th>Cumulative Gain</th>
<th>Average Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrew</td>
<td>-6.6% -6.3% -5.7% -5.3%</td>
<td>-8.53%</td>
</tr>
<tr>
<td>Phil</td>
<td>-2.7% -2.3% -2.0% -1.7%</td>
<td>-3.75%</td>
</tr>
<tr>
<td>Ricardo</td>
<td>-3.9% -3.5% -3.0% -2.7%</td>
<td>-5.06%</td>
</tr>
<tr>
<td>Sarah</td>
<td>-4.4% -4.2% -2.4% -2.2%</td>
<td>-5.72%</td>
</tr>
<tr>
<td>David</td>
<td>-5.2% -4.6% -2.7% -2.5%</td>
<td>-5.87%</td>
</tr>
</tbody>
</table>

for Andrew, Phil, Ricardo, Sarah and David sequence respectively over the Dyadic RAHT approach.

Table 4.7 BD-rate gains for proposed scheme over Dyadic RAHT on 8i voxelized full bodies (8iVFB) Dataset

<table>
<thead>
<tr>
<th>Test Sequences</th>
<th>Cumulative Gain</th>
<th>Average Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soldier</td>
<td>-4.8% -4.0% -3.8% -3.4%</td>
<td>-4.0%</td>
</tr>
<tr>
<td>Loot</td>
<td>-8.5% -6.9% -5.5% -3.9%</td>
<td>-6.2%</td>
</tr>
<tr>
<td>Long dress</td>
<td>-1.3% -1.0% -0.6% -0.4%</td>
<td>-0.825%</td>
</tr>
<tr>
<td>Red and Black</td>
<td>-3.0% -1.9% -1.6% -1.5%</td>
<td>-2.0%</td>
</tr>
</tbody>
</table>

In order to verify the robustness of the proposed scheme, we also tested it on the 8iVFB (8i Voxelized Full Bodies) JPEG Pleno dataset shown in Figure 4.15.

Our method not only provided an average cumulative gain of 4.0%, 6.2%, 0.825%
and 2.0% for Soldier, Loot, Long Dress and Red & Black sequence, respectively (as summarized in Table 4.7), but also eliminates the need to tune the threshold as in the original switching scheme.

4.3.4 Conclusion

In Section 4.3, we present a new neural network based coding approach that focuses on compression of static point cloud attributes. More precisely, we address
the threshold dependency problem to enable the generalized applicability of the transform switching technique based on the characteristics of different regions in a point cloud. The proposed neural network technique comprises of three main steps: collecting data from the 3D edge detection scheme, using the data to train a fairly simple shallow neural network and finally deploying the trained network to replace the original switching scheme. We have demonstrated the efficiency of the proposed method for point cloud attribute compression in terms of RD-performance, by comparing it to the MPEG-GPCC standardized method that uses only Dyadic transform
throughout the point cloud. Average cumulative gains of over 3% was achieved for all the frames from MVUB dataset from open source JPEG Pleno database.
CHAPTER 5

CONCLUSION AND FUTURE WORK

With this chapter, I summarize my conclusions and suggest future research directions. In this dissertation, I presented solution to compression of two different types of data i.e., binary images and point cloud that adds to the data exploding.

Firstly, I presented a new scheme for compression of ROI maps of hyperspectral images. It comprises of a preprocessing step that takes advantage of the correlations between the neighboring pixels and improves the sparsity of the ROI maps. In the next step, I divide the image into smaller blocks and use an optimization algorithm to choose an optimal scanning direction for each block, thereby resulting in a significant improvement in compression efficiency. The simulation results on six hyperspectral data sets show that the proposed algorithm has compression performance that is comparable to or higher than other standard methods by significant percentages. In the future, it would be of interest to use more scanning directions
to scan each block of the divided image to experiment with its effect on the final compression results.

With respect to point cloud compression, I have introduced a new scheme that allows to switch between RAHT and Dyadic RAHT to adapt to the unordered nature of 3D point clouds. In order to effectively exploit the redundancies in uniform and non-uniform areas in a PC, the switch is based on the characteristics of the neighbourhood. The continuity and discontinuity are recognized by 3D edge detection using the Sobel operator. The experimental results on two different types of datasets show that the proposed algorithm achieves good BD-rate cumulative gains of up to 11% when compared to using only one type of transform throughout the point cloud without increasing the computational complexity. The limitation of the this approach was in its threshold dependency which was previously tuned.

Next, I presented a new neural network based coding approach that focused on compression of static point cloud attributes. More precisely, I addressed the threshold dependency problem to enable the generalized applicability of the transform switching technique based on the characteristics of different regions in a point cloud. The proposed neural network technique comprised of three main steps: collecting data from the 3D edge detection scheme, using the data to train a fairly simple shallow neural network and finally deploying the trained network to replace the origi-
inal switching scheme. I have demonstrated the efficiency of proposed method for point cloud attribute compression in terms of RD-performance, by comparing it to MPEG-GPCC standardized method that uses only Dyadic transform throughout the point cloud. Average cumulative gains of over 3% was achieved on MVUB dataset, while showing minor gains on 8iVFB dataset.

In this framework, I have only used the Luma values for classification. In the future work, feature size could be increased to obtain higher training, validation and testing accuracy. The effect of these changes on the attribute compression of point clouds could be studied. Additionally, transfer learning could be employed to extend the generalized applicability of the switching technique.
REFERENCES


