Lossless Compression for On-Board Satellite Imaging

S. Atek

Submitted for the Degree of Doctor of Philosophy from the University of Surrey

University of Surrey
Surrey Space Centre
School of Electronics and Physical Sciences
University of Surrey
Guildford, Surrey GU2 5XH, UK

March 2004

© S. Atek 2004
Summary

Although new generations of small satellites have power imaging capabilities that make them comparable with large satellites; they suffer from a limited downlink transmission. The use of data compression on-board the spacecraft will allow the reduction of the size of images onboard and shorten the transmission time.

The adopted approach to lossless image compression employs predictive neural networks, integer wavelet transforms and Peano-Hilbert scan. The benchmark results show that the compression ratios obtained with the neural network-based method combined with a Peano Hilbert scan are higher than those obtained with JPEG2000 and the Rice algorithm.

Ways of speeding up the processing time of the proposed algorithm have been undertaken. It has been shown that the number of neural network sets and the size of image tiles affect the processing speed of the neural network predictor. A 3-set configuration represents the best trade-off in terms of performance and complexity. A hardware implementation targeted at FPGA platform has been studied and simulated using a VHDL model. The concept of our proposed neural compressor is based on multiple neural processors performing the compression of a certain number of image tiles in parallel.

A decision support scheme was introduced, which makes the compression adaptive to the image zero-order entropy and aims to minimize the on-board memory usage. The concept behind the adaptive scheme is to allow an individual NN processor to execute a 2-set prediction instead of a 3-set one provided the resulting loss in performance is tolerable.

A solution to incorporating the novel compression technique in an on-board satellite image system is outlined. A new hardware system (SSONICS) is proposed realising lossless image compression in an independent decentralized way without using the resources of the on-board computer fully. The system can easily be extended to comprise additional image-processing techniques allowing decompression and recompression without changing the internal structure of the hardware completely. This novel system can facilitate a faster way of data transmission to ground terminals and therefore can offset the data link problem.

Key words: Onboard image compression, Lossless image compression, Neural network prediction, Decision support scheme, Hardware compression system.
MATERIAL REDACTED AT REQUEST OF UNIVERSITY
Acknowledgments

I would like to express my warm and sincere gratitude to all those who contributed to this research program. I wish to express my appreciation of working at the Surrey Space Centre (SSC) where I could find the right environment to fulfil my research and to the Centre National des Techniques Spatiales (CNTS), which sponsored me. I would also like to express my thanks to the Surrey Satellite Technology Ltd (SSTL) for sponsoring my attendance to an unforgettable Space Congress in Bremen where I could meet great figures of space science and technology.

First, I would like to thank my supervisors: Dr Tanya Vladimirova and Dr Peter Sweeney for their very helpful and useful guidance and supervision throughout my research project.

I wish to express my gratitude to Sir Professor Martin Sweeting for the support and also for giving a great example that I wish one day to follow.

Last but not least I would like to thank my family and my friends and those who gave me their support and help. I would like to thank Anne Irving, Alex daSilva Curiel, Jeff Ward and warmly Karen Collar.
# Contents

Summary ................................................................................................................................................... ii  
Acknowledgments ................................................................................................................................... iv  
Contents ..................................................................................................................................................... v  
List of Figures ........................................................................................................................................ viii  
List of Tables ........................................................................................................................................... xi  
Abbreviations .......................................................................................................................................... xii  
Presentations ........................................................................................................................................... xv  
Contest ...................................................................................................................................................... xv  
Publications ............................................................................................................................................. xvi  

1 Introduction ......................................................................................................................................... 1  
1.1 Data Link Problem ................................................................. 1  
1.2 Small Satellite Renaissance .................................................. 2  
1.3 The Need for Image Compression ........................................... 3  
1.4 Lossy or Lossless Compression .............................................. 6  
1.5 Structure of Thesis ............................................................... 8  
1.6 Novel Work Undertaken ....................................................... 8  

2 Lossless Image Compression: Theory and Techniques ......................................................................... 10  
2.1 Digital Images ............................................................................. 11  
2.2 Introduction to Information Theory ........................................ 12  
2.2.1 The Entropy ........................................................................... 12  
2.2.2 The Discrete Memoryless Source Model ................................ 12  
2.2.3 The Markov K-th order Source Model ................................. 12  
2.3 Data Compression Fundamentals ......................................... 13  
2.4 Entropy Coding Methods ....................................................... 18  
2.4.1 Run Length Encoding ......................................................... 19  
2.4.2 Lempel Ziv Welch Encoding ............................................... 19  
2.4.3 Huffman Encoding ............................................................. 19  
2.4.4 Arithmetic Compression ..................................................... 20  
2.5 Predictive Methods ............................................................... 21  
2.5.1 Context-Based Adaptive Lossless Image Compression: CALIC ......................... 25
2.5.2 The Rice Algorithm .............................................................................................27
2.5.3 Lossless JPEG Compression: JPEG-LS .................................................................32
2.6 Integer Wavelet Transform-Based Compression .......................................................33
  2.6.1 The Lifting Scheme ..............................................................................................34
  2.6.2 The Two-dimensional Integer Wavelet Transform ..........................................39
  2.6.3 The Integer Wavelet Transform-Based Image Compressor .........................40
  2.6.4 JPEG2000 Fundamentals and Standard .........................................................40
2.7 Pixel Arrangement Problem ..................................................................................43
  2.7.1.1 Space Filling Curves ......................................................................................44
  2.7.1.2 Peano Hilbert Curves ....................................................................................44
2.8 Multispectral Image Compression .......................................................................47
2.9 Error Protection of Compressed Images ..................................................................48
2.10 Conclusion .............................................................................................................48

4 Neural Network-Based Image Compression ............................................................50
  4.1 Image Compression using Neural Networks ..........................................................50
  4.2 Predictive Coding using Neural Networks .............................................................55
    4.2.1 Off-Line Approach to the Neural Network Prediction ....................................56
    4.2.2 On-line Approach to the Neural Network Prediction ......................................58
  4.3 On-line Neural Network Predictor Applied to Image Compression ......................60
  4.4 Investigation of the Effect of Peano-Hilbert Scan ..............................................71
  4.5 Using the Integer Wavelet Transform ..................................................................74
    4.5.1 IWT-Based Neural Compressor .................................................................76
  4.6 Comparison of the Neural Network-based Methods Against the Rice Algorithm and
     JPEG2000 ..............................................................................................................77
  4.7 Conclusions .........................................................................................................79

5 Hardware Implementation of the Neural Network Image Compressor ...................80
  5.1 Software or Hardware Compression Solution ......................................................80
  5.2 Complexity Versus Performance ........................................................................81
  5.3 Implementation of the NN Compressor in Hardware using Parallel Processing ....84
    5.3.1 Neural Network Processor Design ...............................................................88
    5.3.2 Execution Time ............................................................................................100
  5.4 Functional Simulation Results ..................................................................................97
  5.5 Decision Support Scheme of Active Neural Sets ................................................101
  5.6 SSONICS: Small Satellite Onboard Neural-Based Image Compression System ....106
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Spectrum of SSTL Small Satellite Solutions</td>
<td>3</td>
</tr>
<tr>
<td>2-1</td>
<td>Example of a digital image</td>
<td>11</td>
</tr>
<tr>
<td>2-2</td>
<td>General configuration of a compression/decompression algorithm</td>
<td>14</td>
</tr>
<tr>
<td>2-3</td>
<td>Scalar quantization example</td>
<td>15</td>
</tr>
<tr>
<td>2-4</td>
<td>Classification of lossless compression techniques</td>
<td>18</td>
</tr>
<tr>
<td>2-5</td>
<td>Example of Huffman tree building</td>
<td>20</td>
</tr>
<tr>
<td>2-6</td>
<td>Block diagram of a predictive scheme</td>
<td>22</td>
</tr>
<tr>
<td>2-7</td>
<td>Representation of a window of neighbouring pixels</td>
<td>22</td>
</tr>
<tr>
<td>2-8</td>
<td>(a) Picture “Lena”; (b) Probability mass function of Lena; (c) Probability mass function of the prediction errors of Lena using a unit delay predictor</td>
<td>23</td>
</tr>
<tr>
<td>2-9</td>
<td>(a) Alsat-1 satellite image; (b) Probability mass functions of ALSAT-1 image; (c) Probability mass function of the prediction error of Alsat-1 using a unit delay predictor</td>
<td>24</td>
</tr>
<tr>
<td>2-10</td>
<td>Prediction algorithm used in the CALIC algorithm</td>
<td>26</td>
</tr>
<tr>
<td>2-11</td>
<td>The encoder architecture of the Rice Algorithm</td>
<td>28</td>
</tr>
<tr>
<td>2-12</td>
<td>Block diagram of the Unit-delay Predictive Preprocessor</td>
<td>28</td>
</tr>
<tr>
<td>2-13</td>
<td>Performance curve for various values of k</td>
<td>30</td>
</tr>
<tr>
<td>2-14</td>
<td>JPEG-LS predictive schemes</td>
<td>32</td>
</tr>
<tr>
<td>2-15</td>
<td>Prediction used in the LOCO-I algorithm</td>
<td>33</td>
</tr>
<tr>
<td>2-16</td>
<td>Basic lifting based wavelet decomposition</td>
<td>35</td>
</tr>
<tr>
<td>2-17</td>
<td>IWT based on the lifting scheme</td>
<td>35</td>
</tr>
<tr>
<td>2-18</td>
<td>2-Dimensional integer wavelet decomposition of Lena</td>
<td>39</td>
</tr>
<tr>
<td>2-19</td>
<td>The IWT based compressor</td>
<td>40</td>
</tr>
<tr>
<td>2-20</td>
<td>JPEG2000 block diagram</td>
<td>42</td>
</tr>
<tr>
<td>2-21</td>
<td>Peano-Hilbert scan of a 16x16 square</td>
<td>45</td>
</tr>
<tr>
<td>2-22</td>
<td>Mapping of the horizontal coordinates of the Peano-Hilbert curve</td>
<td>46</td>
</tr>
<tr>
<td>2-23</td>
<td>Mapping of the vertical coordinates of the Peano-Hilbert curve</td>
<td>46</td>
</tr>
<tr>
<td>2-24</td>
<td>Spiral scan of a 5x5 square</td>
<td>47</td>
</tr>
<tr>
<td>3-1</td>
<td>Back propagation neural network</td>
<td>51</td>
</tr>
<tr>
<td>3-2</td>
<td>Hierarchical neural network structure</td>
<td>52</td>
</tr>
<tr>
<td>3-3</td>
<td>The neural configuration of the off-line approach</td>
<td>57</td>
</tr>
<tr>
<td>3-4</td>
<td>Overview of the Neural Network Predictor with 3 input sets</td>
<td>61</td>
</tr>
<tr>
<td>3-5</td>
<td>Test images set</td>
<td>69</td>
</tr>
<tr>
<td>3-6</td>
<td>Block diagram of the method</td>
<td>71</td>
</tr>
</tbody>
</table>
Figure 3-7: Comparison of the different compression ratios .......................................................... 73
Figure 3-8: Lena scanned with a Peano Hilbert scan ................................................................. 73
Figure 3-9: Low resolution image browser of the images stored in the satellite memory .......... 75
Figure 3-10: Low resolution image browser of the images stored in the satellite memory .... 75
Figure 3-11: Block diagram of the proposed image compression method .................................. 76
Figure 3-12: Compression performances of the NN-based methods compared with the RICE Algorithm and JPEG2000 ................................................................. 77
Figure 4-1: Processing times of the NN coder with different numbers of inputs .................... 81
Figure 4-2: Compression ratios of the NN coder using different numbers of inputs sets .......... 82
Figure 4-3: Memory usage of the NN coder as a function of the numbers of inputs sets .... 83
Figure 4-4: Proposed approach for a parallel processing ........................................................... 85
Figure 4-5: Processing times needed to compress sub-image tiles of different sizes .......... 85
Figure 4-6: Irreversible propagation of the errors in the decompressed image .................... 87
Figure 4-7: Confinement of the errors into one block ................................................................. 87
Figure 4-8: Hardware design organisation .................................................................................. 89
Figure 4-9: The NN processor block diagram .............................................................................. 90
Figure 4-10: Active Input Selection Block ................................................................................ 91
Figure 4-11: Values of the sigmoid function ............................................................................ 92
Figure 4-12: Block-diagram of the Weights Update module for a 3-set configuration ............. 92
Figure 4-13: Block-diagram of the Weights Update module for a 6-set configuration ............. 93
Figure 4-14: The arithmetic encoder block .............................................................................. 94
Figure 4-15: Example of the arithmetic encoding process ....................................................... 94
Figure 4-16: Illustration of the implementation order of each block of the neural processor ....... 95
Figure 4-17: Illustration of the processing times as a function of the FPGA frequency for a 1-set configuration ....................................................................................... 96
Figure 4-18: Illustration of the processing times as a function of the FPGA frequency for a 3-set configuration ....................................................................................... 97
Figure 4-19: Waveform of the simulation of the Active Input block ........................................... 98
Figure 4-20: Waveform of the simulation of the Arithmetic Encoder Block ............................... 98
Figure 4-21: Waveform of the simulation of the Activation Function block ............................. 99
Figure 4-22: Waveform of the simulation of the Activation Function block ............................. 99
Figure 4-23: Waveform of the simulation of the Weight Update block ..................................... 100
Figure 4-24: Simulation results of the top-level program of the NN-based technique ............. 100
Figure 4-25: Values of $R_{2,3}$ for different compression ratios $R_3$ of UoSat-12 multispectral images .......................................................... 103
Figure 4-26: Compression ratios $R_3$ and $R_{10}$ of satellite multispectral images .................. 104
Figure 4-27: Compression ratios $R_3(R_{10})$ of satellite multispectral images ......................... 104
Figure 4-28: A schematic representation of the decision support scheme .................................................. 106
Figure 4-29: Block diagram of SSONICS ..................................................................................................... 107
Figure 4-30: SSONICS within the on-board subsystems ............................................................................ 108
List of Tables

Table 1-1: Comparison between small and big remote sensing satellites resolutions .........................4
Table 2-1: Fundamental Sequence Encoding ..........................................................................................30
Table 2-2: The zero-block option .........................................................................................................31
Table 3-1: Values of contexts and active inputs for a 4-Set configuration ...........................................59
Table 3-2: Compression ratios using the predictors NN1 and NN2 ..........................................................70
Table 3-3: Compression ratios after applying the different scans and the Burrow Wheeler transform .....72
Table 3-4: Compression ratios after applying a (2,2) IWT for 2-levels of decomposition .......................76
Table 3-5: Compression ratios of the proposed NN-based methods against the RICE algorithm and JPEG2000. ........................................................................................................................................78
Table 4-1: Processing times required to compress some SSTL satellite images (in seconds) ..............82
Table 4-2: Compression ratios obtained after compressing SSTL satellite images ............................83
Table 4-3: Functionality of the design signals .......................................................................................101
Table 4-4: Compression ratios of panchromatic images using 2 different neural configurations .............102
Table 4-5: Number of image tiles compressed with a 2-set configuration Adaptive Neural Processor .....105
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR</td>
<td>Auto-Regressive</td>
</tr>
<tr>
<td>ARMA</td>
<td>Auto-Regressive Moving-Average</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application specific Integrated Circuit</td>
</tr>
<tr>
<td>BWT</td>
<td>Burrow Wheeler Transform</td>
</tr>
<tr>
<td>CALIC</td>
<td>Context Based Adaptive Lossless Image Compression</td>
</tr>
<tr>
<td>CAN</td>
<td>Controller Area Network</td>
</tr>
<tr>
<td>CCD</td>
<td>Charge Coupled Device</td>
</tr>
<tr>
<td>CCSDS</td>
<td>Consultative Committee for Space Data Systems</td>
</tr>
<tr>
<td>CNES</td>
<td>Centre National d'Etudes Spatiales</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial-Off-The-Shelf</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DMS</td>
<td>Discrete Memoryless Source Models</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processor</td>
</tr>
<tr>
<td>DWT</td>
<td>Discrete Wavelet Transform</td>
</tr>
<tr>
<td>EC</td>
<td>Entropy Coding</td>
</tr>
<tr>
<td>ESA</td>
<td>European Space Agency</td>
</tr>
<tr>
<td>ESTEC</td>
<td>European Space Research &amp; Technology Centre</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>FS</td>
<td>Fundamental Sequence</td>
</tr>
<tr>
<td>GAP</td>
<td>Gradient Adjusted Predictor</td>
</tr>
<tr>
<td>GIS</td>
<td>Generalized Iterative Scaling</td>
</tr>
<tr>
<td>GSD</td>
<td>Ground Sampling Distance</td>
</tr>
<tr>
<td>ICT</td>
<td>Irreversible Component Transformation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>ISA</td>
<td>Industrial Standards Architecture</td>
</tr>
<tr>
<td>IWT</td>
<td>Integer Wavelet Transform</td>
</tr>
<tr>
<td>JPEG</td>
<td>Joint Photographic Expert Group</td>
</tr>
<tr>
<td>KL</td>
<td>Kullback-Leiber</td>
</tr>
<tr>
<td>LEO</td>
<td>Low Earth Orbit</td>
</tr>
<tr>
<td>LS</td>
<td>Lifting Scheme</td>
</tr>
<tr>
<td>LZW</td>
<td>Lempel Ziv Welch</td>
</tr>
<tr>
<td>MAC</td>
<td>Multiply Accumulate Cycles</td>
</tr>
<tr>
<td>MAE</td>
<td>Mean Absolute Error</td>
</tr>
<tr>
<td>MKS</td>
<td>Markov K-th order Source</td>
</tr>
<tr>
<td>MPGA</td>
<td>Masked Programmed Gate Array</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean Square Error</td>
</tr>
<tr>
<td>NEO</td>
<td>Near Earth Objects</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network</td>
</tr>
<tr>
<td>NOAA</td>
<td>National Oceanic and Atmospheric Administration</td>
</tr>
<tr>
<td>OBC</td>
<td>On-Board Computer</td>
</tr>
<tr>
<td>OBDH</td>
<td>On-Board Data Handling</td>
</tr>
<tr>
<td>PH</td>
<td>Peano-Hilbert</td>
</tr>
<tr>
<td>PLD</td>
<td>Programmable Logic Device</td>
</tr>
<tr>
<td>PMF</td>
<td>Probability Mass Function</td>
</tr>
<tr>
<td>PPM</td>
<td>Prediction-by-Partial Match</td>
</tr>
<tr>
<td>PPP</td>
<td>Progressively Predictive Pyramid</td>
</tr>
<tr>
<td>PR</td>
<td>Perfect Reconstruction</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal to Noise Ratio</td>
</tr>
<tr>
<td>RCT</td>
<td>Reversible Component Transformation</td>
</tr>
<tr>
<td>RD</td>
<td>Rate Distortion</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>RLE</td>
<td>Run-Length Encoding</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SFC</td>
<td>Space-filling Curve</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SSI</td>
<td>Small Scale Integration</td>
</tr>
<tr>
<td>SSONICS</td>
<td>Small Satellite Onboard Neural-based Image Compression System</td>
</tr>
<tr>
<td>SSTL</td>
<td>Surrey Satellite Technology Ltd</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
</tbody>
</table>
Presentations


2. 9th International Workshop on Systems, Signals and Image Processing IWSSIP'02, November 7-8, 2002 - Manchester, United Kingdom.

3. The British Festival of Space, July 10th -12th, 2003, University of Surrey, Guildford, Surrey, United Kingdom.


Contest

1. 10th Satellite Design Contest, October 27, 2002- Tokyo Metropolitan College of Aeronautical Engineering, Tokyo, Japan. Awarded the Second Price in the small satellite design category.
Publications


3 "Lossless Compression for Small Satellite On-board Imaging", Sofiane Atek, Tanya Vladimirova, Peter Sweeney, 9th International Workshop on Systems, Signals and Image Processing IWSSIP'02, November 7-8, 2002 - Manchester, United Kingdom.

4 "Neural Network Based On-board Lossless Image Compression", Sofiane Atek, Tanya Vladimirova, Peter Sweeney, 9th International Workshop on Systems, Signals and Image Processing IWSSIP'02, November 7-8, 2002 - Manchester, United Kingdom.


Chapter 1

1 Introduction

In this thesis, the problem of the limited bandwidth transmission of small satellites is addressed. This problem is due to many constraints such as limited power budget and short station contact time. The objective of this research is to provide an efficient solution to offset this problem by designing a novel lossless compression scheme targeted at a hardware implementation.

1.1 Data Link problem

Fundamental physical limits on the transmission bandwidth represent a serious issue for satellite telecommunications in Low Earth Orbit. The practical bandwidth of a satellite moving over a ground station can be reduced to half the allocated band. This phenomenon is the result of the Doppler shift that is provoked by the satellite velocity. For example an S-band link is allocated a 2 MHz Band and the Doppler shift may be as much as 1 MHz.

Satellites can have different applications and thus different orbits. Satellites on a geostationary orbit have the ability to appear fixed above a point in the equator. They are sited to be permanently in view from their groundstations. For all other orbits, the satellites move across the sky and consequently may be out of sight of any particular groundstation. Satellite altitudes may vary consequently, and thus the horizon line of each of them depends on the orbit altitude. The lower the altitude, the lower is the contact time and the more it is a problem for a satellite to be seen. Consequently some spacecrafts in low orbits need to store a lot of data before transmitting to ground.

The purpose of this research is to propose an efficient way to offset the data link problem allowing a class of satellites, “small satellites”, to store more imaging data on-board and transmit them in less time.
1.2 Small Satellite Renaissance

The first big step for unmanned orbiting spacecrafts, commonly called satellites, was successfully accomplished with the advent of micro satellite Sputnik 1. But since its launch in 1957, the trends toward developing and designing bigger scale satellites went increasingly wider on demand from commercial and scientific community [Coxhill02].

Nowadays, commercial-off-the-shelf (COTS) components and innovative design processes have made smaller satellites a viable alternative to conventional ones. The reduction of the huge costs and long development periods needed to design a small satellite provides a cost-effective solution to Earth observation missions at a time when space budgets are facing severe cuts.

Recently, the trends go towards building constellations of small satellites that provide many applications in addition to remote sensing missions: voice and data communications for mobile telecommunications and Near Earth Objects (NEO) [Kennedy02] and planetary explorations [Space03]. These successful implementations by small satellites lead many governments, universities and other organisations to develop their own small satellites programs.

Although small satellites have powerful on-board processors and a large size of memory on-board, there are technical limitations that require further attention. They only have low bandwidth downlink, which is in general the bottleneck of the Low Earth Observation small spacecrafts.

This conflict between the considerable amount of data to transmit to ground and the limited downlink bandwidth will prevent from fast transmission. The use of compression on-board the spacecraft will allow the reduction of the size of the data onboard and shorten the transmission time in this way.
Figure 1-1 shows the spectrum of small satellite solutions developed by the Surrey Satellite Technology Ltd.

![Figure 1-1: Spectrum of SSTL Small Satellite Solutions](image)

Figure 1-2 shows SNAP-1 nanosatellite, which has been designed by Surrey Satellite Technology Ltd (SSTL). Figure 1-3 represents Alsat-1, which is an Algerian remote sensing satellite for disaster monitoring.

![Figure 1-2: Surrey Satellite Technology Ltd Nanosatellite SNAP-1 [SSTL04].](image)

![Figure 1-3: Alsat-1 Enhanced Microsatellite Microsat-100 [SSTL04].](image)

### 1.3 The Need for Image Compression

Data compression has gained an increasingly important role in the fields of medical, commercial and government-related computing. Compression makes it possible for system designers to exploit limited storage resources and communication channel bandwidth. Other application areas of data compression can be remote sensing,
commercial data processing and storage, medical imaging and military imaging, communications and computing.

In spacecraft applications, compression schemes became more popular when new more efficient embedded processing systems were introduced.

Recently small spacecraft payloads benefited from spectacular advances. Remote sensing micro-satellites can be equipped with very advanced onboard sensor devices and their development knows an era of fast improvement and better image handling quality. New generations of Low Earth Orbit (LEO) satellites have emerged with imaging capabilities that are very close to large satellite imaging devices and the storage on board is more able to deal with a large amount of data.

Table 1-1 shows a comparison between conventional satellites and small satellite camera resolutions (highlighted in red). This table shows that some recent small spacecrafts are now able to handle images with an improved ground sampling distance (GSD) that makes them perform even better than some large satellites.

<table>
<thead>
<tr>
<th>Satellite</th>
<th>Country</th>
<th>Launch</th>
<th>Resolution (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>QuickBird-2</td>
<td>US</td>
<td>2001</td>
<td>0.6</td>
</tr>
<tr>
<td>IKONOS-2</td>
<td>US</td>
<td>1999</td>
<td>1.0</td>
</tr>
<tr>
<td>IRS TESS</td>
<td>India</td>
<td>2001</td>
<td>1.0</td>
</tr>
<tr>
<td>EROS A1</td>
<td>Israel</td>
<td>2000</td>
<td>1.8</td>
</tr>
<tr>
<td>SPOT-5</td>
<td>France</td>
<td>2002</td>
<td>2.5</td>
</tr>
<tr>
<td>TopSat</td>
<td>UK</td>
<td>2005</td>
<td>2.5</td>
</tr>
<tr>
<td>Ziyan-ZY-2B</td>
<td>China</td>
<td>2002</td>
<td>2.5</td>
</tr>
<tr>
<td>RapidEye-A</td>
<td>Germany</td>
<td>2005</td>
<td>6.5</td>
</tr>
<tr>
<td>KOMPSAT-1</td>
<td>Korea</td>
<td>1999</td>
<td>6.6</td>
</tr>
<tr>
<td>Proba</td>
<td>ESA</td>
<td>2001</td>
<td>8.0</td>
</tr>
<tr>
<td>SPOT-4</td>
<td>France</td>
<td>1998</td>
<td>10.0</td>
</tr>
<tr>
<td>DMC Bilsat</td>
<td>Turkey</td>
<td>2003</td>
<td>12.0</td>
</tr>
<tr>
<td>Landsat 7</td>
<td>US</td>
<td>1999</td>
<td>15.0</td>
</tr>
<tr>
<td>DMC Alsat-1</td>
<td>Algeria</td>
<td>2002</td>
<td>32.0</td>
</tr>
<tr>
<td>DMC NigeriaSat-1</td>
<td>Nigeria</td>
<td>2003</td>
<td>32.0</td>
</tr>
<tr>
<td>DMC UK</td>
<td>UK</td>
<td>2003</td>
<td>32.0</td>
</tr>
</tbody>
</table>

Table 1-1: Comparison between small and big remote sensing satellites resolutions [NOAA04].
But the actual capacity of the onboard processing resources has some bottlenecks and limitations that will make it difficult to manage the large amounts of data. The reasons are:

(a) A limited transmission bandwidth.

(b) Dealing with a higher volume of data will lead to higher power consumption and a longer contact time with the ground.

(c) LEO satellites have limited ground coverage, and therefore a limited contact time (about 10 minutes).

In commercial satellite applications a typical imaging system is composed of an imager (Figure 1-4), data storage devices (Figure 1-5), data buses and a transmission system. First, an operator instruction is uplinked through a low bit-rate channel to the spacecraft. The satellite will operate to take an image of the specified area following a certain schedule. After the on-board camera has captured the image, the data is stored in the on-board RAM disk. When the satellite can be seen again by the ground station, the image is downloaded via UHF/VHF or S-band channels after one or more orbits around the Earth, depending on the size of the image [Palmer01].

Figure 1-4: Small satellite imager [SSTL04].
The purpose of this work is to design an image compression scheme operating onboard the spacecraft that will overcome the difficulties mentioned above, and optimise the satellite resources such as power budget, CPU consumption, memory usage and execution time.

1.4 Lossy or Lossless Compression

The primary objective of lossless image compression is to represent the original image with a minimised number of bits without losing any information [Taubman01]. Indeed, in many applications, the exact reconstruction of the image is required. For instance medical applications require lossless compression to avoid legal disagreement on the significance of errors in images. In satellite imagery, very often, images need to be processed through complex algorithms in order to extract features that cannot be seen by the human eye. Besides, where multiple compression operations are needed, a lossless scheme can be more appropriate to prevent accumulation of errors from several lossy compression operations [Taubman01].

However, for small satellite imaging, mainly lossy compression schemes have been used [Peixin99]. Some reasons for this are listed as follows:

(a) Lossless compression achieves relatively modest ratios from 1.5 to 4 [Weijmans96].

(b) Lossy compression can achieve higher compression ratios, and as a consequence, shorter time will be required for transmission to the ground-station. A lossless
scheme will require more number of passes to download the images compared with a lossy one [Brewer95].

(c) The human visual system can tolerate a significant amount of loss without interfering with the image content interpretation.

(d) Small satellites employ commercial-off-the-shelf (COTS) charged coupled device (CCD) cameras, which are not high fidelity sensors, and therefore are highly sensitive to different noises [Peixin99] such as the black current. The digital input to the compression system as perceived by the imager is already an imperfect representation of the real-world scene [Taubman01] where intrinsic quantization is also applied to the digital output of the CCD.

As an example, most of small satellites developed by SSTL employed lossy compression. However, little work has been done on lossless algorithms, such as Huffman compression [Montpetit93].

It is obvious that a scheme based on lossless image compression contains many limitations, but it is possible to overcome them. Even if the compression ratio is bounded by a mathematical limit, an effective compression scheme will reach closer to this limit, which can be at the best case about 4:1 [Weijmans96]. If we improve an existing compressor that can bring a compression ratio of 1.5:1 to the value 2:1, that means we divide the number of passes by two instead of 1.5. Moreover, recent satellites are equipped with more accurate CCD sensors. As a consequence, less information is lost during the image capture by the intrinsic quantization.

The use of LEO constellations of small satellites enables sharing the data among individual satellites, which increases the passes over one ground station.

One of the advantages of lossless compression is that images could be used by different users for different purposes. When a loss of information is tolerated by one, it may not be the case by the other. This is why lossless compression is essential.

Current technologies are able to provide manufacturers of small satellites with better data processing platforms. For instance, enhanced micro satellites developed by SSTL use S-band downlink channels that allow a high rate transmission up to 8 Mbps [SSTL04]. However, the technological advances in on-board data handling systems lead to increase remote sensing images, which can reach up to several Gigabytes.
1.5 Structure of Thesis

In chapter 2, we briefly give some definitions about digital images. An overview of information theory and data compression fundamentals is given. We discuss the background of lossless image compression and the main methods representing the state-of-the-art of the field. We group these techniques in three categories: entropy coding methods, predictive methods and Integer Wavelet Transform (IWT) based methods. We discuss examples of recent techniques for each category.

In chapter 3, we present a new method for satellite image compression. This novel technique applied to compression of satellite images is based on neural network prediction and pre-processing techniques of data such as the Peano-Hilbert scan and integer wavelet transforms. Compression ratios based on a set of satellite and natural images will be shown and analysed.

In chapter 4, experimental work is shown to address the complexity constraint of our proposed method in terms of execution time and memory usage. A 3-set configuration of the neural network compressor is adopted as a good trade-off between complexity and performance. A hardware implementation of our technique is explained and a new configuration using parallel neural processors is proposed. A decision support mechanism combined with the neural network compressor allows the compressor to use less on-board memory by switching from a 3-set configuration to a 2-set configuration when the entropy of the input image tile is less than an empirical threshold. A new lossless image compression system targeted at small satellite on-board implementation is proposed. The proposed system performs compression independently from the on-board computers and manages the capture, compression/decompression and storage of the images.

In chapter 5, we draw the main conclusions of our research and discuss directions for future work.

1.6 Novel Work Undertaken

The goal of this research project is to develop a new lossless image compression algorithm to incorporate compression for on-board small satellite storage and transmission.

The research resulted in the following novel work:
• A novel algorithm implemented on satellite images based on a neural network scheme that was initially used for text compression. The scheme has been modified by applying empirical parameters that improved the compression ratios. The algorithm has been combined with a (2,2) integer wavelet transform in one implementation and a Peano-Hilbert scan in another, which achieved significant compression ratios compared to the Rice algorithm and JPEG2000.

• An optimisation study of the novel neural network scheme in order to lower its complexity and address the performing time constraint.

• A new hardware configuration of the neural network based compressor, where multiple neural network processors are compressing tiles of the image input in a parallel way.

• A novel design of the neural network processor was captured in a hardware description language. This design has been simulated and tested.

• A new decision support mechanism was introduced, which makes our compression algorithm adaptable to the zero-entropy of the image in order to lower the memory consumption.

• A new hardware compression system based on our novel algorithm was proposed.
Chapter 2

2 Lossless Image Compression: Theory and Techniques

The issue of using lossless image compression was emphasized in the previous chapter as a practical solution to offset the data link problem. The objective of this chapter is to introduce lossless image compression and give an insight of the work that has been carried out in the field. This chapter is focused on giving an overview of the most widely used algorithms for lossless image compression.

General definitions about digital images are presented in section 2.1. In section 2.2 and 2.3, an overview of information theory and data compression fundamentals is given. Entropy is presented as a mathematical limit to information coding and two types of information models are given. General definitions about compression techniques are presented where the difference between lossless compression and lossy compression is shown. In addition, some quantitative measures to assess compression performances are given. Lossless compression techniques are grouped in three main categories: entropy coding techniques, predictive techniques and transform-based techniques. Section 2.4 presents the most common entropy coding techniques. Section 2.5 talks about predictive techniques and transform-based techniques are discussed in 2.6.

The importance of the pixel arrangement in image compression is discussed in section 2.7. Section 2.8 considers some issues about compression of multispectral images and section 2.9 gives an overview of error protection of compressed images. Finally, The conclusions are drawn in section 2.10.
2.1 Digital Images

In order to obtain a digital image, a discretization is applied to the image on both the spatial coordinates and the brightness intensity. The digitized brightness value is called the gray level [Gonzalez92].

A gray level image is a 2-dimensional integer function of light intensity \( F(x, y) \). In this function \( x \) and \( y \) represent the spatial coordinates and the value of this function \( F \) is proportional to the brightness of the image at the corresponding coordinates \((x, y)\).

A pixel is basically each element of the matrix \( F \) and each pixel is encoded with a certain number of bits \( M \). The digitized image can be represented by the following matrix:

\[
F = \begin{bmatrix}
    f(0,0) & f(0,1) & \ldots & f(0,N-1) \\
    f(1,0) & f(1,1) & \ldots & \ldots \\
    \ldots & \ldots & \ldots & \ldots \\
    f(L-1,0) & \ldots & \ldots & f(L-1,N-1)
\end{bmatrix}
\]  

(2.1)

With \( 0 \leq f(x, y) \leq 2^M - 1 \), where \( L \times N \) represents the size of the image.

Figure 2-1 shows more clearly the pixels of the image and the different gray level intensities after applying a zooming (5:1) on a small area of the picture “Lena”.

![Figure 2-1: Example of a digital image.](image-url)
2.2 Introduction to Information Theory

Before introducing the data compression field we briefly give an overview of some important definitions in information theory.

2.2.1 The Entropy

Shannon established that there is a fundamental limit to lossless data compression. This limit is called the Entropy rate, denoted \( H \). The value of \( H \) depends on the statistical nature of the source. The quantity \( H \) is also defined as the average of the self-information [Sayood00].

The statistical nature of the information will define the models on which the design of the code will depend. There are many models to map data we outline below the most known ones. In the following, we assume \( A \) an alphabet of symbols \( \{x_1, x_2, ..., x_m\} \) and \( \text{card}(X) \) the size of the alphabet \( X \).

2.2.2 The Discrete Memoryless Source Model

In the Discrete Memoryless Source Models (DMS) the successive symbols are statistically independent, i.e. the current symbol doesn’t depend on the previous symbols. The entropy is defined as follow [Sayood00]:

\[
H(X) = - \sum_{i=1}^{\text{card}(X)} p_i \log_2 p_i 
\]  

(2.2)

Where \( p_i \) is the probability of occurrence of the \( i^{th} \) symbol.

2.2.3 The Markov K-th order Source Model

We have statistical dependencies between the symbols, in such a way that the present symbol depends on the \( k \) previous ones.

We can express the Markov K-th Source model (MKS) by the following conditional probability:

\[
p(x_i \mid x_{i-1}, x_{i-2}, ..., x_{i-k}) = p(x_i = a_k \mid x_{i-1}, x_{i-2}, ..., x_{i-k}, ...) \quad \forall i \in \{k+1, ..., \text{card}(X)\} \]  

(2.3)

with \( k \in \{1, ..., \text{card}(X)\} \).
The entropy $H(X)$ of a MKS is defined as:

$$H(X) = \sum_{S_k} p(x_{i-1}, x_{i-2}, ..., x_{i-k}) H(x | x_{i-1}, x_{i-2}, ..., x_{i-k})$$  \hspace{1cm} (2.4)

Where $H(x | x_{i-1}, x_{i-2}, ..., x_{i-k})$ is the conditional entropy defined below:

$$H(x | x_{i-1}, x_{i-2}, ..., x_{i-k}) = \sum_i p(x_i = a_k | x_{i-1}, x_{i-2}, ..., x_{i-k}) \log_2(p(x_i = a_k | x_{i-1}, x_{i-2}, ..., x_{i-k}))$$  \hspace{1cm} (2.5)

$S_k$ denotes all possible realizations $\{x_{i-1}, x_{i-2}, ..., x_{i-k}\}$ $\forall$ $i,k$

In general for two sources with identical alphabet and symbol probabilities we have:

$$H(X)_{MKS} < H(X)_{DMS}$$  \hspace{1cm} (2.6)

The MKS model is more realistic for images and videos, because images are correlated in the spatial domain and videos are correlated in the spatial and temporal domain.

In the following section, we give a brief introduction to data compression fundamentals.

### 2.3 Data Compression Fundamentals

Data compression deals with removing most, if not all, the redundancies that can be found in data. The degree of data reduction can be measured with a quantitative value that is called the compression ratio. It represents the relationship between the amount of compressed data and the quantity of the original data [Sayood00].

$$\text{Compression Ratio (CR)} = \frac{\text{Length of original data string}}{\text{Length of compressed data string}}$$  \hspace{1cm} (2.7)

Compression techniques can be classified into two main categories: lossless methods and lossy methods.

- Lossless compression techniques are completely reversible. Data are compressed so that, when decompressed, the original data can be perfectly reconstructed. Lossless compression can also be referred as “reversible” and non-destructive compression.

- Lossy compression techniques are mainly used on images, audio and videos. With this type of compression the data is not completely reproducible. Even though, after decompression, the data may not be the exact duplication of the original data, the difference between the original and the reconstructed data may be so minor as to be hardly perceptible.
In Figure 2-2, we present a general block diagram showing the main processing steps in a data compression/decompression algorithm [Belbachir04]. The lossy compression part is formed by a decorrelation stage and a quantization stage. Lossless data compression is formed by the entropy coding stage. The decompression part consists of decoding and applying an inverse transform of the compressed data stream in order to reconstruct the original data.

![Diagram of compression and decompression](image)

Figure 2-2: General configuration of a compression/decompression algorithm.

The task of quantization in lossy compression is to produce an approximation of the decomposed image data performing a minimal loss in image quality. The symbols that are processed during the quantization step are the output of the decorrelation stage. Quantization is essential to control the amount of information loss and to play a great role in adjusting the compression ratio in lossy compression schemes. For example, a transform-based scheme such as the discrete cosine transform will decorrelate an image so that the low frequency coefficients will gather more information and energy than the high frequencies. Quantization will have the task to maximize the image quality and minimize the compression bit rate while altering the high frequency coefficient values and preserving the low frequency coefficient values where the essential part of the image information is kept [Peixin99].
There are mainly two types of quantization: scalar quantization and vector quantization. The difference between the two techniques is that scalar quantization uses the signal input individually while vector quantization quantizes vector of signals. Figure 2-3 shows an example of a scalar quantization. Its task is to map real numbers to a discrete set of reconstruction levels, based on ordered decision levels.

To assess or “measure” the performance of a data compression technique, a parameter known as reconstruction error is generally adopted. This parameter is used to quantify the data distortion as a function of the difference between the original and reconstructed data. This difference can be expressed using several quantitative measures. The usual measures used for the performance assessment are:

- The mean absolute error (MAE): This measure is generally adopted for simplicity purposes. The mean absolute error for the original data $S$ and the reconstructed data $S_R$ is defined as:

$$MAE(S, S_R) = \frac{1}{N} \sum_{x=1}^{N} |S(x) - S_R(x)|$$

(2.8)

- The root mean square error (RMSE): The RMSE differs from the MAE in the fact that the squares of the differences are averaged then the square root is applied to the result. In
other words, it is calculated as the standard deviation of all reconstructed data set relative to the original data. The RMSE is defined as:

$$RMSE(S, S_R) = \sqrt{\frac{1}{N} \sum_{x=1}^{N} (S(x) - S_R(x))^2}$$ (2.9)

- The signal-to-noise ratio (SNR): The SNR can be seen as an improvement of the RMSE equation by taking the intensity of the reference into account. In fact, the total reference power is divided by the total error power. The logarithm function is then adopted to reduce the range of the values.

The SNR is defined as:

$$SNR(S, S_R) = 10 \log_{10} \frac{\sum_{x=1}^{N} S(x)^2}{\sum_{x=1}^{N} (S(x) - S_R(x))^2}$$ (2.10)

We notice that this measure is inversely proportional to the previous two.

- The peak signal-to-noise ratio (PSNR): This parameter is widely used in the performance assessment of compression techniques. The PSNR is given by:

$$PSNR(S, S_R) = 10 \log_{10} \frac{(\max(S(x)) - \min(S(x)))^2}{\frac{1}{N} \sum_{x=1}^{N} (S(x) - S_R(x))^2}$$ (2.11)

These criteria are mainly used for lossy compression but they can also be used to assess the performance of a lossless predictive scheme (see section 2.5) as a comparison tool between the original image and the predicted one.

The performance of data compression techniques can be assessed using different important criteria apart from the reconstruction error. The most relevant of these criteria or parameters are:

- The compression ratio: The compression ratio is the criterion that characterises the capability of a method to find and remove the redundancy in the data. The compression ratio increases when more redundancy is removed.
Chapter 2: Lossless Image Compression: Theory and Techniques

- The complexity of the method: The previous criteria are very important in the design of compression techniques. However, the complexity of the algorithm is even more important because it defines the feasibility of the method. This criterion has to be taken into account at the first stage of the design, and it must be adopted as a criterion to assess the compression method.

- The execution time: This criterion depends on the algorithm implementation and on the machine on which it is running. There is a strong dependence between the algorithm implementation and the execution time but the relationship is far from trivial when the way the method is implemented and the hardware description are not well known.

- Channel errors [Brewer95]:

When channel errors occur, some algorithms can be very sensitive to them. The whole image can be corrupted if one pixel or even a bit is flipped. A good algorithm should show good robustness to bit errors.

- The thumbnail image [Hercus97]:

A thumbnail image is an image that is at less than full resolution. This might be used to provide a preview of the scene to the user before decompressing the whole image. If the user decides not to transfer the complete image then the total transmission time will be significantly reduced. On the other hand, if the full image is needed, then it would help the system to utilise the low-resolution image already transmitted so that it reduces the amount of the data which is requested to be transmitted.

Lossless compression techniques can be classified into three categories: entropy coding methods, predictive methods and transform based methods. The first category comprises methods, which emerged historically before the other two categories. They are used in all compression algorithms as the last processing step before compression is entirely finished. The other two categories comprise recent methods. Predictive methods are based on the prediction of the pixel value using its neighbourhood, while integer wavelet transform based methods transform the image and decorrelate it using the property of energy packing.
In Figure 2-4 we show some examples of lossless compression schemes belonging to each category. A detailed description of these compression techniques is presented in sections 2.4, 2.5 and 2.6 below as indicated in figure 2-4.

![Figure 2-4: Classification of lossless compression techniques.](image)

### 2.4 Entropy Coding Methods

These methods make use exclusively of the redundancy of the data. They try to represent symbols with fewer bits using shorter codewords. The process of entropy coding (EC) uses two consecutive operations: modelling and coding. Modelling allots a set of probabilities to the symbols, and coding uses these probabilities to construct a bit sequence that represents the compressed data. Two of the most known entropy encoding techniques are Huffman coding and Arithmetic encoding. In this section we will give a brief overview of the most common entropy coding techniques.
2.4.1 Run Length Encoding

This coding scheme works with regard to repetition of characters. It reduces the physical size of the repeating string by reporting the number of repetitions near to the recurring character separated by a flag. An approach known as byte-stuffing [Steinmetz95] is used in order to differentiate between the flag and the character belonging to the data stream.

2.4.2 Lempel Ziv Welch Encoding

This was proposed by Abraham Lempel, Jacob Ziv in 1977 and Terry Welch in 1984. This scheme can be either a variable-to-fixed code or a variable-to-variable code.

Under the LZW algorithm, the encoder starts with an initial dictionary of single byte strings, which have a correspondence to indices that range from 0 to 255. The compression follows these two steps:

- If the string is in the dictionary, then the algorithm outputs the corresponding index.
- If the string consisting of that particular string and the following single character is unknown to the algorithm, then it is stored in the dictionary with a new index.

The LZW algorithm uses a numeric code to indicate the position of a string in the dictionary. Theoretically the size of the dictionary can grow infinitely. Practically, the size of the dictionary is limited. Welch had recommended a size of 4096 [Phamdo00] that means each index can be coded with 12 bits.

The length of the index may vary. For example we can encode the first index with one bit, the second and the third with two bits, the fourth the fifth with three bits and so on. This is a variable-to-variable length version of the LZW.

2.4.3 Huffman Encoding [Sayood00]

The basic idea is to assign short codewords to inputs with high probabilities and long codewords to those with low probabilities. The codewords are stored as a conversion table. Huffman encoding is designed by summing the two least probable characters and repeating this process until there are two symbols remaining. A code tree is generated and the Huffman code is obtained from the labelling of the code tree. The ones must be on one side of the tree while the zeros are on the other side.
Figure 2-5, shows an example of a Huffman tree building. In the example the alphabet is composed of 5 letters \(\{a,b,c,d,e\}\). Each letter has got a number of occurrences that is represented inside the boxes. In this example we use the number of occurrence instead of the probabilities of the symbols. The process consists of summing the two least numbers of occurrences and iterating that process until we build a tree. We put the ones on one side of the tree, and the zeros on the other side. The codeword is generated by combining the bits that are written along the branch of the tree following the path from the top of the tree towards the symbol position within the tree.

![Huffman Tree Building Diagram]

2.4.4 Arithmetic Compression

While the Huffman algorithm associates for each character of the source a binary code, arithmetic compression deals with the whole sequence by giving it a unique binary fraction between 0 and 1. It assigns a full code to the entire sequence of characters that has to be encoded.

The coding space is viewed as the semi interval \([0,1]\) containing all arbitrary length binary fractions. Any such point in the space is a codeword. The length of a subinterval corresponding to a sequence is equal to its probability of occurrence. Retaining at each iteration the probability that corresponds to the next input symbol iteratively reduces the
interval. This operation is repeated for the entire sequence of characters. It is very hard to implement this kind of encoding because the precision required, when subdividing the intervals, increases with the length of the sequences to be encoded. The algorithm for arithmetic compression is detailed below [Nelson91]:

1) A set of probabilities is assigned to the symbols being encoded.
2) The individual symbols are assigned a range along a probability line, which is nominally [0, 1[, using the set of probabilities obtained from step 1.
3) The encoding is then performed according to the following procedure:

   Set Low to 0.0
   Set High to 1.0
   While there are still input symbols do
      Get an input symbol
      Range = High - Low.
      High = Low + Range*High_range(symbol)
      Low = Low + Range*Low_range(symbol)
   End of While
   Output Low
4) The final code will be set to the final Low value obtained from step 3.

Low_range and High_range represent the boundaries of the range of the symbol within the probability line [0,1[. The values Low and High are the boundaries of the new subdivided interval.

There is no 1:1 correspondence between symbols and codewords. Alternatively statistics partitioning can be determined by a first scan of the file. It is best suited for sources whose alphabet is small and which have highly unbalanced statistics.

It is shown that Arithmetic coding can easily adapt to non-stationary source statistics [Peixin99]. In practice, arithmetic coding can perform up to 10% better in compression rate than Huffman coding [Vlachos01].

2.5 Predictive Methods

Predictive methods can be used to carry out compression without any loss of information. These methods exploit the redundancy between a pixel and its neighbours. A function of
prediction allows estimating the value of a pixel according to the value of the neighbouring pixels. We then encode the prediction error, which is the difference between the true value and the predicted value. The function of prediction can be more or less complex according to [Moz98]:

- The prediction order: the number of symbols involved in the calculation of prediction.
- The topology: position of the neighbouring symbols used in calculation.
- The weights: weights assigned to the symbols of the calculation of prediction according to their relative position.

Generally a good prediction has more to do with the topology than with right values of the weights [Moz98]. A typical lossless predictive compression scheme is built up using these different steps:

1. **Prediction Step**: where the value of the current pixel is predicted from the different previous values.
2. **Mapping Step**: The predicted value is converted to a value range.
3. **Coding Step**: We use entropy encoding to carry out this function.

Figure 2-6 illustrates the general block diagram of predictive schemes.

![Block diagram of a predictive scheme.](image)

Figure 2-7 represents a window of neighbouring pixels in an image. The pixel $p$ is the pixel to be predicted by a function of the surrounding pixels (a,b,c,d,e) intensity values.

![Representation of a window of neighbouring pixels.](image)
Entropy coding is known to be very efficient in coding sources that are distributed less evenly. When the Probability mass function (PMF) of a given source is skewed or when some symbols occur more often than others, the source can be compressed in a very efficient way using entropy coding [Sayood03].

Unfortunately, most images do not have this property, and do not have such skewed PMFs. However, it is possible to extract sequences, which can have a PMF skewed in shape, using prediction [Sayood03].

Figure 2-8 and Figure 2-9 show the difference in shape between the probability mass functions of the image and its prediction errors using a unit delay predictor. As we can see, the prediction error PMFs have a skewed shape and therefore the prediction errors of these images present more redundancies than the original ones. As a consequence, the entropy coding of these prediction errors will be more efficient.

![Figure 2-8: (a) Picture “Lena”; (b) Probability mass function of Lena; (c) Probability mass function of the prediction errors of Lena using a unit delay predictor.](image-url)
Predictive methods are used in many lossless compression applications. For space applications, for instance, the Consultative Committee for Space Data Systems (CCSDS) has adopted a standard for lossless data compression in space applications [Calzolari98]. This one is based on an extended version of the RICE algorithm, which is discussed in section 2.4.2. A study [Calzolari98] pointed out its good combination of speed, adaptivity and compression performance over a wide variety of data.

In the following sections some existing predictive-based methods are detailed.

---

1 Alsat-1 is an Algerian remote sensing microsatellite launched in 2003.
2.5.1 Context-Based Adaptive Lossless Image Compression: CALIC

The Context-based Adaptive Lossless Image Compression method (CALIC) uses a new gradient-based non-linear prediction design called Gradient-Adjusted Predictor (GAP). This latter adjusts the coefficients of the predictor after an estimation of the local gradients [Wu96]. The model context is composed of a combination of quantized local gradient and texture pattern. These two features are characteristic of the error behaviour.

In its initial evaluation, in 1995, the CALIC algorithm produced the lowest bit rates in six of seven image classes: medical, aerial, pre-press, scanned, video and compound document. It produced the third lowest bit rate in the class of computer-generated images [Wu96].

2.5.1.1 Algorithm Description

The CALIC algorithm uses raster scan order with a single pass through the image. The scheme can be applied on bi-level or continuous-tone images. The algorithm has four major integrated components: prediction; context selection and quantization; context modelling of prediction errors; and finally, entropy coding of prediction errors [Wu96].

2.5.1.2 GAP-Gradient Adjusted Predictor

GAP differs from the existing predictors. This non-linear adaptive predictor has the ability to tune itself to the intensity gradients near the predicted pixel \( \hat{x}(i, j) \).

The gradient of the intensity functions near the predicted pixel is computed as follows:

\[
d_h = |x(i-1, j) - x(i-2, j)| + |x(i, j-1) - x(i-1, j-1)| + |x(i+1, j-1) - x(i, j-1)|
\]

\[
d_v = |x(i-1, j) - x(i+1, j-2)| + |x(i, j-1) - x(i, j-2)| + |x(i, j-1) - x(i, j-2)|
\]

where

\( x(i, j) \) is the current pixel at the position \((i, j)\).

\( d_v \) and \( d_h \) are the intensity gradients.
The prediction is then processed as shown in Figure 2-10, where:

\[ n = x(i,j-1), w = x(i-1,j), ne = x(i+1,j-1), nw = x(i-1,j-1), nn = x(i,j-2), \]
\[ w = x(i-2,j). \]

If \((d_h - d_v) > 80\)

\[ \hat{x}(i,j) = n \]

else if \((d_v - d_h) > -80\)

\[ \hat{x}(i,j) = w \]

else

\[
\begin{cases}
\hat{x}(i,j) = (n + w)/2 + (ne - nw)/4 & \text{if } (d_h - d_v) > 32 \\
\hat{x}(i,j) = (\hat{x}(i,j) + n)/2 & \text{else if } (d_v - d_h) > 32 \\
\hat{x}(i,j) = (\hat{x}(i,j) + w)/2 & \text{else if } (d_h - d_v) > 8 \\
\hat{x}(i,j) = (3\hat{x}(i,j) + n)/4 & \text{else if } (d_v - d_h) > 8 \\
\hat{x}(i,j) = (3\hat{x}(i,j) + w)/4 & \text{else}
\end{cases}
\]

Figure 2-10: Prediction algorithm used in the CALIC algorithm.

where \( \hat{x}(i,j) \) is the predicted pixel.

2.5.1.3 Coding Context Selection and Quantization

In this step we form eight context errors called error energy contexts. They are formed by a quantization of the error energy estimator \( \Psi \) into 8 bins. \( \Psi \) is defined as follows:

\[
\Psi = d_h + d_v + 2|n - \hat{n}|
\]  

(2.15)

Where \( \hat{n} = \hat{x}(i-1,j) \), \( d_h \) and \( d_v \) are defined from (2.13) and (2.14). Practically, \( \Psi \) is quantized by \( Q \) to eight levels i.e. \( Q(\Psi) = \{0,1,\ldots,7\} \).

It has been found that an image independent \( \Psi \) quantizer with the bins:

\[
( q_1 = 5, q_2 = 15, q_3 = 25, q_4 = 42, q_5 = 60, q_6 = 85, q_7 = 140 )
\]

performed as well as the optimal one designed by choosing \( 0 = q_0 < q_1 < \ldots < q_{L-1} < q_L = \infty \) that minimizes

\[
\sum_{d=0}^{L-1} \sum_{\Psi < q_{d+1}} p(\Delta) \log p(\Delta)
\]

(2.16)
in an off-line calculation [Wu96], where $\Delta$ is the prediction error so that $\Delta = x - \hat{x}$.

### 2.5.1.4 Context Modelling of Prediction Errors and Error Feedback

To further improve the coding performance, 144 texture contexts were inserted into four error energies ($\Psi/2$) [Wu96]. A total of 576 compound contexts are obtained.

The algorithm realizes the texture context by applying a quantization on a vector $C$ whose components are elements of the neighbourhood.

$$C = \{ y_1, y_2, \ldots, y_7 \} = \{ n, w, nw, ne, nn, ww, 2n - nn, 2w - ww \}$$

(2.17)

The result of the quantization is an 8-bit binary number $B = b_7 b_6 \ldots b_0$ such that:

$$b_k = \begin{cases} 0 & \text{if } y_k \geq \hat{x}(i, j) \\ 1 & \text{if } y_k < \hat{x}(i, j) \end{cases}, \quad 0 \leq k < K = 8$$

(2.18)

A compound modelling context $C(Q(\Psi)/2, B)$ is formed, which is composed of the combination of the quantized error energy and the quantized texture pattern.

The bias in the prediction can be corrected by feeding back $\overline{\Delta}(\delta, \beta)$ to $\hat{x}$ in order to obtain an improved predictor $\bar{x} = \hat{x} + \overline{\Delta}(\delta, \beta)$. Where $\overline{\Delta}(\delta, \beta)$ is the $C(\delta, \beta)$ conditional mean.

### 2.5.2 The Rice Algorithm [CCSDS97, Calzolari98]

Here we will explain the functions of the Rice Coder. Basically, the data passes through a preprocessing step that will decorrelate it and map the negative values of the decorrelated output into positive values. Then an entropy coding is performed on the preprocessor output data. We will see that this encoder is an optimal entropy coder [CCSDS97].

#### 2.5.2.1 Model Description

The Rice Coder works by using several different codes and transmitting the code identifier. The Rice algorithm has the ability to adapt the coding process from low entropies to high ones. The algorithm consists of a pre-processor, which can be split into a predictor step and a mapping step followed by an adaptive entropy coder block. Rice coding requires the data to be split into $n$-sample blocks of $J$-bit samples. The parameter $J$ can be either 8 or 16 samples per block (the maximum value is 32 [Coco00]). The preferable value has been found to be 16 [CCSDS97].
Figure 2-11, shows the block diagram of the Rice compressor.

![Figure 2-11: The encoder architecture of the Rice Algorithm [CCSDS97].](image1)

The main blocks of the diagram are explained in the following sections.

**2.5.2.2 The Preprocessor**

The role of the preprocessor is to decorrelate the data samples and subsequently map them into symbols that can be more efficiently compressed by the Adaptive Entropy Coder. Figure 2-12 shows the block diagram of the preprocessor.

![Figure 2-12: Block diagram of the unit-delay predictive preprocessor [CCSDS97].](image2)
2.5.2.3 The Predictor

The predictor transforms the original data so that it removes the correlations between the samples in the input data block. The predictor used in the Rice Algorithm is a linear first-order unit-delay predictor (see Figure 2-12).

There are also other predictors that can be used. They can be a one-dimensional first order predictor, or a two-dimensional second order predictor where the predicted value \( \hat{x}(i, j) \) is the average of the adjacent pixels \( x(i, j-1), x(i-1, j) \) or a two-dimensional third order predictor. In the last case, the predicted value \( \hat{x}(i, j) \) is equal to a weighted combination of the neighbouring values of \( x(i, j-1), x(i-1, j), x(i-1, j-1) \).

The recommended predictor by the CCSDS [CCSDS97] is a one-dimensional first order predictor because of its simple implementation. It is, however, possible to use other higher order predictor types using application-specific predictors [Coco00].

2.5.2.4 The Mapper

The mapper has the role of converting each prediction error \( \Delta_i \) to a non-negative n-bit integer value. To make sure that more probable symbols are encoded with shorter codewords, the preprocessed symbol \( \delta_i \) should satisfy:

\[
p_0 \geq p_1 \geq p_2 \geq ... p_j \geq ... p_{(2^n-1)}
\]  

(2.19)

Where \( p_j \) is the probability that \( \delta_i \) equals the integer \( j \), for \( i = \{0,1,2,...,2^n-1\} \). The prediction error mapping function is:

\[
\delta_i = \begin{cases} 
2\Delta_i & 0 \leq \Delta_i \leq \theta \\
2|\Delta_i| & -1 - \theta \leq \Delta_i \leq 0 \\
\theta + |\Delta_i| & \text{otherwise}
\end{cases}
\]

(2.20)

where \( \theta = \min(\hat{x} - x_{\min}, x_{\max} - \hat{x}) \), This mapping function has the property that \( p_i < p_j, \forall |\Delta_i| > |\Delta_j| \).

2.5.2.5 The Entropy Encoder

This is a module composed of a set of variable-length codes that operates in parallel on a block of \( J \) processed samples. When a coding option achieves the higher compression ratio, it is then selected for transmission. To identify the winner selection, an
identification (ID) bit pattern is used.

**a. The Fundamental Sequence Encoding**

In this code we determine the transmitted symbol by the number of preceding zeros. Each ‘1’ digit indicates the end of a codeword. Table 2-1 explains the fundamental sequence encoding.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Code</th>
<th>Code 2 (FS Code)</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₁</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>s₂</td>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>s₃</td>
<td>10</td>
<td>001</td>
</tr>
<tr>
<td>s₄</td>
<td>11</td>
<td>0001</td>
</tr>
</tbody>
</table>

*Table 2-1: Fundamental Sequence Encoding.*

**b. The Split-Sample Options**

The $k^{th}$ split-sample option works by splitting off the $k$ least significant bits from the $J$ processed data sample, and then encoding the remaining higher order bits with a Fundamental Sequence (FS) codeword. These options are designed to encode the data from $k+1.5$ to $k+2.5$ bits/sample. $k=0$ is the fundamental sequence option. The curves in Figure 2-13 show the relationship between the data entropy and the ratio for each $k^{th}$ split-sample option.

*Figure 2-13: Performance curve for various values of $k$ [CCSDS97].*

We can notice that the entropy encoder is optimal for entropies higher than 1.5
c. The Second Extension Option

This is a low entropy coder. It compresses the data in a range of 0.5 bits/sample to 1.5 bits/sample. It works by gathering first, consecutive pairs of pre-processed samples of the \( J \)-sample block. The pairs are then transformed into a new value.

\[
\gamma = (\delta_i + \delta_{i+1}) \frac{(\delta_i + \delta_{i+1}+1)}{2} + \delta_{i+1}
\]

(2.21)

This value is coded using an FS codeword.

d. The Zero-Block Option

When one or more blocks are composed of zeros, this option is selected. We then encode the all-zeros pre-processed blocks by an FS codeword.

<table>
<thead>
<tr>
<th>Number of All-Zeros Blocks</th>
<th>FS Codeword</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>01</td>
</tr>
<tr>
<td>3</td>
<td>001</td>
</tr>
<tr>
<td>4</td>
<td>0001</td>
</tr>
<tr>
<td>ROS</td>
<td>00001</td>
</tr>
<tr>
<td>5</td>
<td>000001</td>
</tr>
<tr>
<td>6</td>
<td>0000001</td>
</tr>
<tr>
<td>7</td>
<td>00000001</td>
</tr>
<tr>
<td>8</td>
<td>000000001</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>63</td>
<td>0000 \ldots 0000000001</td>
</tr>
</tbody>
</table>

(63 0s and a 1)

Table 2-2: The zero-block option.

Some improvements of the Rice coder were provided by Coco, D’Arrigo and Giunta
For calculation economy, the trio determined a method that selects the optimal value $k$ of the $k$-split option. This method is based on the calculation of the correlation between the $k_{opt}$ and an estimate of the expectation of the magnitude of data output by the preprocessor. After finding the value $k_{opt}$, a comparison of the needed bits for the $k_{opt}$-split option and the needed bits for the other options is made before choosing the encoding. For $k_{opt} \geq 4$ the 2nd extension option can be skipped.

### 2.5.3 Lossless JPEG Compression: JPEG-LS

A very well known compression standard produced by the Joint Photographic Expert Group (JPEG) is the lossy image compression standard. The group developed also a lossless standard for image compression. Actually there are two standards: an "old" one and a "new" or "current" one [Sayood00].

#### 2.5.3.1 The Old Standard Method

This provides eight different predictive schemes shown in Figure 2-14 below [Sayood00]. The first scheme doesn't make any prediction. Three of the seven remaining are one-dimensional predictors. The last four are two-dimensional prediction schemes. $x(i,j)$ is the $(i,j)$ pixel of the original image and $\hat{x}(i, j)$ is the predicted pixel value at the coordinates $(i,j)$.

1. $\hat{x}(i, j) = x(i, j)$
2. $\hat{x}(i, j) = x(i, j - 1)$
3. $\hat{x}(i, j) = x(i - 1, j)$
4. $\hat{x}(i, j) = x(i - 1, j - 1)$
5. $\hat{x}(i, j) = x(i, j - 1) + x(i - 1, j) - x(i - 1, j - 1)$
6. $\hat{x}(i, j) = x(i, j - 1) + (x(i - 1, j) - x(i - 1, j - 1))/2$
7. $\hat{x}(i, j) = x(i - 1, j) + (x(i, j - 1) - x(i - 1, j - 1))/2$
8. $\hat{x}(i, j) = (x(i, j - 1) + x(i - 1, j))/2$

**Figure 2-14: JPEG-LS predictive schemes.**
Following different image topologies any of those seven modes of prediction can be used to perform a better prediction. After the prediction and the mapping steps are performed, one of the two entropy coders can be used: Huffman coding scheme or arithmetic coding. This method achieves compression ratios of about 22% less than the ones achieved by the CALIC method [Memon97] [Memon98] [Wu96].

2.5.3.2 The Current Standard Method

The current standard method is similar to the CALIC compression method presented in section 2.4.1, in the predictive phase. Because the CALIC algorithm performed best of all when compared to other recent algorithms [Sayood00], a team from Hewlett Packard designed an algorithm inspired from CALIC but with less complexity. They called it LOCO-I (for Low Complexity). A brief description of the predictive step is given in Figure 2-15 [Weinberger98], where:

\[ n = x(i, j-1), w = x(i-1, j), ne = x(i+1, j-1), nw = x(i-1, j-1), nn = x(i, j-2), ww = x(i-2, j). \]

\[
\begin{align*}
\text{if} \quad nw & \geq \max(w, n) \\
\hat{x}(x, y) &= \max(w, n) \\
\text{else} \\
\{ \\
\quad \text{if} \quad nw & \leq \min(w, n) \\
\quad \hat{x}(x, y) &= \min(w, n) \\
\quad \text{else} \\
\quad \hat{x}(x, y) &= w + n - nw
\}
\end{align*}
\]

Figure 2-15: Prediction used in the LOCO-I algorithm.

The average compression ratio of this algorithm is about 4% less than the average compression ratio obtained with the CALIC method [Memon97].

2.6 Integer Wavelet Transform-Based Compression

Mathematical transforms are commonly used in data compression, mostly in compression of sensory data such as audio, image, and video.
Although sensory signals are directly available to users in the spatial/time domain, a direct signal representation in that domain creates an enormous volume of data with excessive redundancy.

Transform coding aims to map signal samples from spatial/time domain into another domain. Typically in frequency or joint time-frequency domain where statistical and subjective redundancies can be better exploited and removed.

The main benefit of transform coding is its property of energy packing. By an adequate transform it is possible to transfer the majority of the signal energy into a few transform coefficients. This results in a large number of zero and non-zero coefficients that are easier to encode using an entropy coding technique. The probability distribution of a transformed image is more biased than that of the original image. Ideally, the more biased the distribution, the higher compression ratios it is possible to obtain [Rao01].

The IWT is a wavelet transform that constructs for an integer input an integer wavelet coefficient output. The oldest IWT is an integer version of the Haar Transform, which is called the S transform. The filters used are [Calderbank96, Heer90]:

\[
\begin{align*}
s[n] &= x[2n] + \left\lfloor \frac{d[n]}{2} \right\rfloor
\end{align*}
\] (2.25) (2.26)

Where \( x \) is the input vector, \( s \) the low-pass output vector and \( d \) the high-pass vector.

Said and Pearlman came up with an improved method that performs the S transform plus a Prediction step in order to generate a new set of high-pass coefficients. This transform is called the S+P transform [Said96].

### 2.6.1 The Lifting Scheme

The Lifting Scheme (LS) is a new method to construct biorthogonal wavelets [Sweldens95]. The latter are necessary to obtain a perfect reconstruction of the image [Stoffel98].
Figure 2-16 describes a lifting scheme process. First, the input data is split into even and odd samples. The even signal is convolved with a low pass filter \( H_1 \), and is added to the odd signal. The result is then convolved with a high pass filter \( H_2 \) and then added to the even signal. We can apply more lifting steps. In the output of the process we will have one high pass signal \( d^{(n)} \) and a low pass signal \( s^{(n)} \).

The lifting steps that use the low pass signal are called “prediction steps”, and those, which use the high pass signal, the “update steps” [Reichel01]. To reconstruct the original signal, the same structure is used but in the reverse order.

It was shown in [Reichel01] that LS ensures a perfect reconstruction of the signal. One important property is that the linear filters can be replaced by non-linear ones and preserve the Perfect Reconstruction (PR) properties. By introducing a rounding operation it is possible to generate the IWT coefficients (Figure 2-17).

Figure 2-16: Basic lifting based wavelet decomposition [Reichel01].

Figure 2-17: IWT based on the lifting scheme [Reichel01].
The Lifting steps are summarized in what follows [Sheng98]:

1. The input \( x[n] \) is split into even and odd indexed samples:

\[
\begin{align*}
    s^{(0)}[n] &= x[2n] \\
    d^{(0)}[n] &= x[2n+1]
\end{align*}
\]  

(2.27)  

(2.28)

2. \( M \) alternating lifting steps are applied:

\[
\begin{align*}
    d^{(i)}[n] &= d^{(i-1)}[n] - \left( \sum_k p^{(i)}[k] s^{(i-1)}[n-k] \right) + \frac{1}{2} \\
    s^{(i)}[n] &= s^{(i-1)}[n] + \left( \sum_k u^{(i)}[k] d^{(i)}[n-k] \right) + \frac{1}{2} 
\end{align*}
\]  

(2.29)  

(2.30)

for \( i = 1, \ldots, M \), where \( p^{(i)}[k] \) is a high pass IWT filter and \( u^{(i)}[k] \) is a low pass IWT filter.

3. Finally, a scaling factor is applied:

\[
\begin{align*}
    d[n] &= Kd^{(M)}[n] \\
    s[n] &= s^{(M)}[n]/K
\end{align*}
\]  

(2.31)  

(2.32)

In the literature [Calderbank96], [Memon98], [Sheng98], [Adams99], [Calderbanks98] many transforms are used to perform the IWT. Below are given the most common transforms. Here the notation \((N, \tilde{N})\) represents a transform with \( N \) and \( \tilde{N} \) vanishing moments in the analysis and synthesis high pass filters respectively:
• The (2,2) transform [Calderbank98]:

\[ d[n] = x[2n + 1] - \frac{1}{2} (x[2n] + x[2n + 2]) + \frac{1}{2} \]  
\[ s[n] = x[2n] + \frac{1}{4} (d[n - 1] + d[n]) + \frac{1}{2} \]  

(2.33)

(2.34)

• The (4,2) transform [Calderbank98]:

\[ d[n] = x[2n + 1] - \frac{1}{16} (x[2n] + x[2n + 2]) - \frac{1}{16} (x[2n - 2] + x[2n + 4]) + \frac{1}{2} \]  
\[ s[n] = x[2n] + \frac{1}{4} (d[n - 1] + d[n]) + \frac{1}{2} \]  

(2.35)

(2.36)

• The (4,4) transform [Calderbank98]:

\[ d[n] = x[2n + 1] - \frac{9}{16} (x[2n] + x[2n + 2]) - \frac{1}{16} (x[2n - 2] + x[2n + 4]) + \frac{1}{2} \]  
\[ s[n] = x[2n] - \frac{1}{16} (d[n - 1] + d[n]) - \frac{1}{16} (d[n - 2] + d[n + 1]) + \frac{1}{2} \]  

(2.37)

(2.38)

• The (2,4) transform [Calderbank98]:

\[ d[n] = x[2n + 1] - \frac{1}{2} (x[2n] + x[2n + 2]) + \frac{1}{2} \]  
\[ s[n] = x[2n] - \frac{19}{64} (d[n - 1] + d[n]) - \frac{3}{64} (d[n - 2] + d[n + 1]) + \frac{1}{2} \]  

(2.39)

(2.40)
Chapter 2: Lossless Image Compression: Theory and Techniques

- The (6,2) transform [Calderbank98]:

\[
d[n] = x[2n+1] - \frac{75}{128} (x[2n] + x[2n+2]) - \frac{25}{256} (x[2n-2] + x[2n+4]) + \frac{3}{256} (x[2n-4] + x[2n+6]) + \frac{1}{2} \tag{2.41}
\]

\[
s[n] = x[2n] + \frac{1}{4} (d[n-1] + d[n]) + \frac{1}{2} \tag{2.42}
\]

- The (2+2,2) transform [Calderbank98]:

\[
d^{(0)}[n] = x[2n+1] - \frac{1}{2} (x[2n] + x[2n+2]) + \frac{1}{2} \tag{2.43}
\]

\[
s[n] = x[2n] + \frac{1}{4} (d^{(0)}[n-1] + d^{(0)}[n]) + \frac{1}{2} \tag{2.44}
\]

\[
d[n] = d^{(0)}[n] - \frac{1}{16} (-s[n-1] + s[n] + s[n+1] - s[n+2]) + \frac{1}{2} \tag{2.45}
\]

- The Crew algorithm with 2 coefficients in the low-pass analysis and 10 coefficients in the high pass filter [Calderbank98]:

\[
d^{(0)}[n] = x[2n+1] - x[2n] \tag{2.46}
\]

\[
s[n] = x[2n] + \left\lfloor \frac{d^{(0)}[n]}{2} \right\rfloor \tag{2.47}
\]

\[
d[n] = d^{(0)}[n] - \frac{1}{64} (22(s[n+1] - s[n-1]) + 3(s[n-2] - s[n+2]) + \frac{1}{2} \tag{2.48}
\]
The S+P transform [Calderbank98]:

\[
d^{(i)}[n] = x[2n + 1] - x[2n]
\]

\[
s[n] = x[2n] + \left\lfloor \frac{d^{(i)}[n]}{2} \right\rfloor
\]

\[
d[n] = d^{(i)}[n] + \left\lfloor \frac{2}{8} (s[n - 1] - s[n]) + \frac{3}{8} s[n] - s[n + 1] + \frac{2}{8} d^{(i)}[n + 1] + \frac{1}{2} \right\rfloor
\]

2.6.2 The Two-dimensional Integer Wavelet Transform

To apply the integer wavelet transform on images, two steps are basically computed [Majani97]. First a one-level forward IWT is applied on the horizontal dimension. The result of this will be two integer output subimages. Another one-level forward IWT in the vertical dimension is then used. The first operation in the horizontal direction produces one low frequency image L and a high frequency image H. The second operation produces four subimages: one low-frequency subimage LL and three high-frequency subimages LH, HL and HH. The next level of the IWT will process the LL subimage using the same procedure.

Figure 2-18 shows an example of a 1-level decomposition of Lena.
2.6.3 The Integer Wavelet Transform-Based Image Compressor

The difference between IWT based compressors and predictive ones lies in the functions that precede the entropy coder where an integer wavelet transform is used instead of a predictor. The different IWT based compressors are distinguished by their filters. The selection of the filter will mostly depend on the application requirements. In most cases the IWT based compressors have the form presented in Figure 2-19.

![Figure 2-19: The IWT based compressor.](image)

2.6.4 JPEG2000 Fundamentals and Standard [Taubman01]

The ISO standardisation of JPEG2000 (ISO15444) was achieved through efforts from academia and industry (including Motorola, Kodak, Canon, Texas Instruments, HP, Sharp, Sony, Nokia and Ericsson). Several image compression technologies were submitted by JPEG2000 committee members, and in 1997, the committee met to select a wavelet transform based solution. The participants have continued to work together to improve the JPEG2000 compression system, which at low bit-rates, beats the performance of the current JPEG. It is particularly well-suited to gray-level, colour, bi-level and multi-component still images and allows the compression of a range of spatial resolutions (in pixels) of images, from iconic to huge sizes suitable for satellite applications.

In the JPEG2000 compression technique, the pixels of the original image undergo a specifically selected mathematical transformation where the information content of the pixels of the input image is rearranged so that they are represented more compactly.

In this way, less visually significant transformed coefficients can be discarded during the quantization process. An important advantage of the new JPEG2000, lies in the fact that the transformation is achieved via the DWT. This offers more flexible image compression and better manipulation of the transformed coefficients to achieve desirable features.
including resolution scalability, and quality scalability. In addition, when it is necessary to achieve a high compression ratio, the DWT offers a better starting point than the discrete cosine transform (DCT) used by the current JPEG standard.

Another advantage of the DWT over the DCT is the way an integer wavelet transformation can map an integer number to another integer value. In this way, the DWT based compression system used in the JPEG2000 can be truly lossless. In contrast, the DCT gives floating-point numbers, which can result in rounding errors common to finite arithmetic representation in a computer.

The JPEG2000 compression system offers a variety of compression and efficient code stream access without full decoding made possible by the structures it imposes on the original image. The first structure subdivides the digital input image into rectangular units called ‘tiles’ which do not overlap. This allows each tile component to be coded and decoded independently then passed to the transformation sub-system where a DWT decomposes it into several levels of sub-bands.

The subband coefficients are then quantized to achieve data reduction. Since the DWT decomposition is non-expansive, the number of coefficients generated is the same as the number of samples in the tile by decoding the appropriate number of sub-bands component. The dyadic decomposition into levels of subbands that can be achieved by the DWT yields a lower resolution image.

Another reason why code stream access is made more flexible is the way the quantized sub-band data is divided into code-blocks. It is this code-block structure or data unit that is passed onto the entropy coder, which performs a three-pass individual bit-plane coding of the quantized coefficients. The entropy encoder is based on the arithmetic coder. This three-pass coding process allows access to compressed data at this level by grouping the bit-stream resulting from each pass into layers. Therefore, by decoding successive layers of the bit-stream, increasingly high quality images can be obtained.

The last structure is the packet, i.e. another independent access unit of a spatial region of the image from the code stream. It is made up of a partition of one layer from a decomposition level of a tile component.

Using this framework, JPEG2000 offers features, in an integrated code stream, that is not available with the current JPEG image compression standard. The fact that individual tile components can be accessed from the code stream means that different spatial regions of
the image can be decoded and displayed. For example, it is not necessary to decode large satellite images in their entirety. They can be browsed to select a specific spatial area to be decoded, using a region of interest mask to encode a user-defined foreground at a higher quality than the background. In addition, errors in the bit stream can be concealed using an error recovery strategy, which makes JPEG2000 highly suited to error-prone wireless channels.

Another important feature of the JPEG2000 compression system is its resolution scalability. In other words, its ability to decode different spatial resolutions of the image from a single code stream.

The block diagram of JPEG2000 is represented in Figure 2-20.

![JPEG2000 block diagram](image)

**Figure 2-20: JPEG2000 block diagram.**

JPEG2000 is equipped with a lossless mode, which performs naturally in the course of progressive decoding. The standard has also the property of creating embedded bitstream and allows progressive lossy to lossless build-up [Christopoulos00]. JPEG2000 uses two types of discrete wavelet filters: the Daubechies 9/7 and the Daubechies 5/3 filters [Jung03]. The 5/3 filter allows repetitive encoding and decoding of an image using integer arithmetic, which makes it suitable for a lossless compression mode. The 9/7 filter
outputs coefficients given as floating-point numbers. This filter is suited to high visual quality compression but uses floating-point arithmetic, which leads to rounding errors.

Quantization is lossy in JPEG2000, but when the lossless mode is used the quantization step is set to 1.

JPEG2000 supports multiple-component images using two distinct transforms: the Reversible Component Transformation (RCT) and the Irreversible Component Transformation (ICT). For reversible compression the RCT is used and can only be combined with the 5/3 reversible wavelet transform. The RCT is a transformation that applies a decorrelation to the three first components of an image [Christopoulos00].

The performances of the lossless mode of JPEG2000 has been found in literature to achieve 20% less than the recent JPEG-LS standard [Jung03].

2.7 Pixel Arrangement Problem

The pixel arrangement problem of naturally ordered data lies in finding a mapping of the data objects \( \{ a_i^k, \ldots, a_n^k \} \) to a subwindow of size \((w \times h)\), i.e., a bijective mapping

\[
\mathbf{f}: \{1 \ldots n\} \rightarrow \{1 \ldots w\} \times \{1 \ldots h\} \text{ such that } \sum_{i=1}^{n} \sum_{j=1}^{n} d(f(i), f(j)) = \sum_{i=1}^{n} \sum_{j=1}^{n} \left| d(f(i), f(j)) - d((0,0), (\frac{|i-j|}{n}, \frac{|i-j|}{n})) \right| \tag{2.52}
\]

is minimal. Where we assume having an ordered sequence of \(n\) data objects \(\{a_1, \ldots, a_n\}\), each consisting of \(k\) data values \(\{a_1^k, \ldots, a_n^k\}\). In each of the sub windows, we want to present all the values \(\{a_1^k, \ldots, a_n^k\}\).

The distance \(d(f(i), f(j))\) is referred as the \(L^p\)-distance of the pixels belonging to \(a_i\) and \(a_j\). As we can see the pixel arrangement problem tries to determine the arrangement of pixels which best preserves the distance of the one-dimensional ordering in the two dimensional arrangement by solving the optimisation formula (2.52).
Mappings of ordered one-dimensional data sets to two dimensions was long before computers era a matter of attention for mathematicians. Space-filling curves try to bring a solution to the above optimisation problem. The Peano-Hilbert curve provides the best optimisation of the formula (2.52) among other space filling curves.

### 2.7.1 Space Filling Curves

A space-filling curve (SFC) is a continuous scan, which by changing direction repeatedly [Dafner00] manages to cross every pixel of an image once. There are many SFC curves and they are all defined recursively. A typical definition of a space-filling curve is the limit of the sequence curves $C_0, C_1$, where $C_0$ is a simple curve, and $C_1$ is a more complex curve constructed from $C_0$.

In 1878, Cantor found that any two finite-dimensional smooth manifolds, whatever their dimensions, have the same cardinality. From this he was able to demonstrate that the interval can be mapped bijectively onto the square. In 1879, Netto showed that a bijective map from $[0,1]$ to $[0,1]^2$ is necessarily discontinuous. Then in 1890, Peano found a continuous surjective map from the interval onto the square, by abandoning one-to-one mapping, and in this way he constructed the first space-filling curve. Further examples were proposed by Hilbert, Moore, Lebesgue, Sierpiniski, Pólya, and others [Soljanin02].

Hilbert used a geometric approach to constructing space-filling curves. He postulated that if the interval can be mapped continuously onto the square, then partitioned into four congruent subintervals and into four congruent subsquares, it would follow that each subinterval could be mapped continuously onto one of the subsquares. Hilbert showed that the subsquares could be arranged so that adjacent subintervals correspond to adjacent subsquares with an edge shared between them. SFC's appeal to many image-space algorithms based on the spatial coherence of neighbouring pixels. The most popular recursive SFC is the Peano-Hilbert curve [Dafner00].

#### 2.7.1.1 Peano Hilbert Curves

The property that makes the Peano-Hilbert curve so popular is its strong locality. At any level of refinement, it never leaves its current quadrant before traversing all the pixels in that quadrant [Dafner00]. Figure 2-21 shows an example of a Peano Hilbert curve.
This property of strong locality was used by Lempel and Ziv in their quest to achieve lossless compression of images [Ziv86]. They investigated compression schemes where the image is scanned enabling the image data to be read in a sequence, which is then compressed by a finite state encoder. The compressibility of any picture is lower-bounded by the sequence derived from it using the Hilbert scan [Soljanin02].

It has been shown by Lempel and Ziv that the entropy of the pixel sequence that has been scanned by the Peano-Hilbert curve converges asymptotically to the two-dimensional entropy of the image [Ziv86], for images generated by suitably random sources.

Figure 2-22 and 2-23 show the horizontal and vertical coordinates that the Peano-Hilbert curve follows in a subwindow of 256 by 256.
Figure 2-22: Mapping of the horizontal coordinates of the Peano-Hilbert curve.

Figure 2-23: Mapping of the vertical coordinates of the Peano-Hilbert curve.
2.7.1.2 Spiral Scan

Spiral scan is a curve that scans the image in a spiral path. It works by first scanning the outside image segments forming a square pattern. The scan moves inward in a spiral pattern, scanning smaller and smaller squares until the centre of the image is reached.

![Spiral scan of a 5x5 square.](image)

2.8 Multispectral Image Compression

The spectral redundancies can be exploited in a predictive coding scheme. In this context, one or more spectral images are used to predict the behaviour of the current image.

Taking into consideration that a remote sensing image provides information that are invisible to the human vision, the criterion used to judge a multispectral image is not the same as panchromatic image. Those images are often analysed using computers. They bring information which is to be processed for interpretation of earth features, whether they are in the visible spectrum or not. This is why it is essential to apply a lossless compression scheme to minimize the size of the huge data without any loss. Predictive methods are usually used successfully in that way [Hu97]. There are also other methods that deal with multispectral images, using transforms like KLT, 3D wavelets, etc [Tretter00].

A research was undertaken by Gerard Mozelle [Moz98] on multispectral compression, and began with a comparative study from the reversible methods described in the literature in order to define the method of optimal reversible compression. It transpires that the optimal predictors combined with an arithmetic coder with contextual modelling provide the highest compression ratios (on average of command 2.74 for SPOT-2 and SPOT-3 images and of 2.11 for SPOT-5 simulations) [Moz98].
2.9 Error Protection of Compressed Images

The transmission of errors constitutes a crucial problem for the compressed data. It is easy to see that the errors, which corrupt the data produced by an entropy coder, can have a catastrophic effect on the rebuilt images. The error will thus be propagated losing all the synchronization of data transmission.

Several approaches are possible to protect information. Conventional techniques of protecting coding, standard CRC, turbo code, etc are applicable; and by adding bits of redundancy to the data, detection and correction of a limited number of errors is enabled. The compression ratio is obviously decreased, but this is the price to pay to keep a good quality of the rebuilt image. The effectiveness of protection is sometimes more significant than the quality of compression to determine the final quality: it can happen that an image compressed with relatively low quality and transmitted using a code of effective protection will be better than the image compressed with excellent quality but with a weak protection coding.

The approaches of coding known as "source/channel" seek to find the optimal case: under the constraint of a maximum given flow of compression, they decide how to allocate the bits between source coding (compression) and channel coding (protection) [Czih099]. Another simple technique of protection consists of, more or less frequently, inserting single words of synchronization in the stream of data. This makes it possible neither to detect nor to correct the errors, but prevents the effects of error propagation throughout the data. The codes of protection and resynchronization decrease the final rate of compression. But it is also possible to use a technique only at the level of the decoder, which seeks to detect, and even to correct the errors, just by analyzing the received data.

2.10 Conclusion

When data compression was a new discipline, the algorithms were designed for multipurpose coding: texts, images, etc. These algorithms were general and worked well on all kinds of data but this was not sufficient since they didn’t exploit characteristics that make an image different from any other data. Since then, scientific researchers have directed their work to change the statistical characteristics of images by applying filters and transforms before the coding step.
Chapter 2: Lossless Image Compression: Theory and Techniques

Nowadays, work is oriented more towards finding the best way to decorrelate images than to code them. We can find in the literature many lossless compressors but one of the most known and efficient ones is the CALIC algorithm represented in section 2.4.1. It has an adaptive predictive preprocessing while most of predictive schemes use fixed weights and context. This property allows it to exploit the best pixel neighbourhood correlations and redundancies and outperform the existing methods. However, this algorithm has a higher degree of complexity and demands a lot of calculations.

The CCSDS communication protocol for space applications has adopted the Rice algorithm. This technique has proven to be fast and suitable for on-board lossless compression. However the method uses a unit delay predictor, which does not perform optimal prediction. It has the property to compress efficiently in one dimension while, in an image; correlations are spread along two dimensions.

Wavelets in their initial applications were used for lossy compression. With the advent of Sweldens and Daubechies work on integer wavelets a novel algorithm based on lifting schemes was developed. The main benefit of transform coding is its property of energy packing. By an adequate transform it is possible to transfer the majority of the signal energy into a few transform coefficients. JPEG2000 as a new standard is based on wavelet transforms. It has the ability to operate in lossless mode as well as in lossy mode. The JPEG2000 compression system offers a variety of compression and efficient code stream access without full decoding. Individual tile components can be accessed from the code stream and can be browsed to select a specific spatial area to be decoded. Another important feature of the JPEG2000 compression system is its resolution scalability, which gives the ability to decode different spatial resolutions of an image from a single code stream.

In the literature it has been found that, in terms of performances, ratios of IWT based compression are not competitive with CALIC ratios. However, the multiresolution property of IWT compression are attractive for our space-based application and unavailable in CALIC, JPEG-LS and LOCO-I. IWT permits the progressive transmission of the image and indeed the extraction of lossy versions of the image during transmission. The best algorithm will be the one that can extract from each method the best features.
Chapter 3

3 Neural Network-Based Image Compression

The previous chapter presented the results of literature reviews detailing state-of-the-art lossless image compression algorithms.

This chapter presents a novel technique for lossless image compression for satellite images. We will discuss the advantages of this technique and in particular the high compression ratios that are found to outperform the ratios obtained by some state-of-the-art methods.

In section 3.1 an introduction to image compression using neural networks is shown. Predictive coding using neural networks is discussed in section 3.2. In section 3.3, an application of an on-line neural network predictor to image compression is described. Two variants of the neural network based technique are discussed. Sections 3.4 and 3.5 give a description of the neural compressor combined with a Peano-Hilbert scan and with an integer wavelet transform. In section 3.6 we compare the performances of the two approaches with JPEG2000. The conclusions are drawn in 3.7.

3.1 Image Compression using Neural Networks

Recent resurgence of interest in neural networks has resulted in a large number of parallel techniques and models for real-world applications. Neural networks systems are suited to image compression due to the parallel structure and arrangement of all the neurones within each layer. This parallel structure of simple processing units can be used to implement computationally complex tasks and is ideal to be realized on general purpose parallel processing architectures with programmable capacity to change their structures and therefore their functionality [Jiang99]. In addition, the neural network is a good model to simulate linear and non-linear functions. This property represents a promising approach in optimising non-linear filters used to model the inherent nature of images.

Different approaches exist for the implementation of neural networks in image compression. These approaches are classified into three categories according to the nature of their applications and design [Jiang99]. The first category includes neural network
learning algorithms that have been directly developed to perform image compression. The second class of algorithms refers to neural network implementation of existing image compression schemes and finally the last category includes indirect application of neural networks to assist with those existing image compression methods [Jiang99].

Back-propagation algorithms are directly applied to image compression coding. The compression is performed in two phases: training and coding.

During the training phase, a set of images is used to train the network, which uses each input vector as the desired output. This phase aims to compress the input pixels into the narrow channel consisting of the hidden layer and to retrieve the input pixels to the output layer via the hidden layer (Figure 3-1). The entropy coding of the state vectors at the hidden layer represents the second phase of the compression process.

![Figure 3-1: Back propagation neural network [Jiang99].](image)

This type of neural network algorithms is very suitable for the Karhunen-Loeve transform (KLT) based compression technique, that offers an optimum solution for all linear channel types of image compression neural networks [Jiang99].

The KLT applies a decorrelation to images by mapping them into a new vector space where the covariance matrix of the new vectors is a diagonal matrix, the elements of which along the diagonal are eigenvalues of the covariance matrix of the original input vectors.
Another variant of the back-propagation algorithm uses hierarchical neural network algorithms.

For this type of neural network, more hidden layers are added to the scheme presented in Figure 3-1. The compression layer is preceded by an inner hidden layer called the combiner layer that exploits correlations between pixels (Figure 3-2). An outer hidden layer called the decombiner layer is situated after the compression layer, which exploits correlations between blocks of pixels.

![Hierarchical neural network structure](image)

*Figure 3-2: Hierarchical neural network structure [Jiang99].*

Hebbian learning has also been used to extract principal components, which are the basis vectors for the optimal KL transform. These learning algorithms have computation advantages over standard eigen-decomposition techniques. They use iterative calculations and have an ability to adapt to changes in the input signal.

Another model, the self-organized feature map (SOFM), has been used to design vector quantization (VQ) in lossy techniques.

State-of-the-art image compression algorithms can be implemented by extended neural network structures, such as wavelets, fractals and predictive coding.
In particular, a predictive coding technique has been adopted in this thesis due to its efficiency in de-correlating input data where a high degree of correlation is embedded among neighbouring data samples.

Although general predictive coding is classified into various models such as Auto-Regressive (AR) and Auto-Regressive Moving-Average (ARMA), etc., the predictive coding used in this thesis is based on a totally different approach where the predicted value represents the probability of occurrence of a bit taking into account the occurrence of previous values.

Conventional technology provides a mature environment and well developed theory for predictive coding, which is represented by LPC (linear predictive coding), PCM (pulse code modulation), DPCM (delta PCM) or their modified variations. Non-linear predictive coding, however, involves difficulties in optimising the coefficients extraction to obtain the best possible predictive values. Under these circumstances, neural networks represent a very promising approach in optimising non-linear predictive coding [Jiang99].

For further improvement on conventional image coding and compression algorithms, indirect neural network applications are developed to assist with traditional techniques. This type of application has been typified by significant research work on image pattern recognition, feature extraction and classification by neural networks. When traditional compression technology is applied to those pre-processed patterns and features, it is expected to achieve improvement by using neural networks since their applications in these areas are well established. Hence, image processing based compression technology could be one of the major research directions in the next stage of image compression development.

In addition to the techniques mentioned previously, many lossless image compression methods using neural networks have been implemented.

For instance, a lossless compression technique based on competitive learning network is described in [Jiang95a]. This method is developed from the traditional Ziv-Lempel data compression technique. In this algorithm, the input data set is split up into input vectors of 16 bits. A competitive learning algorithm is applied at the first hidden layer to select the possible winners, which match the input vector.

The functionality of the second hidden layer is to process the case where two winners are selected on the basis that each of the neurons matches the input vector partially by 8 bits.
Each weight is given a certain number of chances to be a winner before it is taken off the network. A new weight is fed into the network from the input vector whenever the network fails to find a winner.

It has been shown through experiments that the compression performance of this type of neural network configuration is competitive to LZW, Huffman and arithmetic encoding ratios, using zero-order Markov models [Jiang95b].

A low entropy pyramidal image data structure applied to lossless compression called progressively predictive pyramid (PPP) is proposed in [Qiu99]. The lossless image neural network compressor is based on the well-known Laplacian pyramid. By placing interresolution neural network predictors into the original Laplacian pyramid, the entropy level in the original pyramid can be reduced significantly [Qiu99].

The Laplacian pyramid offers a major advantage in the sense that the entropy of the difference image between the original one and the predicted one is lower than the entropy of the original image. In fact, in the PPP algorithm, a new laplacian pyramid is proposed where hierarchical neural network predictors are used into the original Laplacian pyramid to reduce its entropy level [Qiu99].

An adaptive non-linear method for the predictive coding of images using multilayer perceptrons was presented in [Marusic99].

The neural networks uses causal and localised training on the actual data being coded, rather than training separate data. With this configuration the network weights are continuously updated. This results in a highly adaptive predictor, with localised optimisation based on the stochastic gradient learning.

The neural prediction technique is combined with an arithmetic coding scheme. It has been found that this technique outperforms the CALIC algorithm.

The neural network has the property to be adaptive to dynamically changing image data, which results in a maximum gain from the prediction stage. As images are non-stationary, the algorithm performs an analysis of local information, which can be considered stationary in certain areas, and can then be exploited to adapt the neural prediction. The neural network is composed of single input, hidden and output layers respectively, made up of perceptrons. Each perceptron uses the same non-linear sigmoid function. The networks perform multiple non-linear predictions, which are combined
Chapter 3: Neural Network-Based Image Compression

through a non-linear output prediction. The network weights are optimised for a local causal training area using the error back-propagation technique.

Another approach for lossless image compression using neural networks is proposed in [Rizvi99]. The technique called modular differential pulse code modulation (MDPCM) consists of a classifier implemented by vector quantizer (VQ) codebook and several neural network class predictors. The classifier uses the four previously encoded pixels to identify the class of the current pixel. The current pixel is the predicted by corresponding class predictor.

Other lossless data compression neural network methods can be applied for image compression [Jiang99], if we consider each pixel as an individual symbol. Theoretical results [Romaniuk94a] based on Kolmogorov’s mapping neural network existence theorem [Kurkova91] show that a C gray level image of n x n can be completely described by a three-layer neural network with $2^\lceil \log n \rceil$ inputs, $4\lceil \log n \rceil + 2$ hidden neurones and $\lceil \log C \rceil$ output neurones [Romaniuk94]. This results in a storage of full connections represented by: $8^2 \log^2 n + (1 + \lceil \log C \rceil)4 \log n + 2\lceil \log C \rceil$ [Romaniuk94b], while the original image requires $\lceil \log C \rceil n^2$ bits. As a result, a theoretical lossless compression that is based on Kolmogorov’s mapping is possible [Jiang99].

3.2 Predictive Coding using Neural Networks

It is well known that lossless image compression treats the image as an inductive inference problem in order to achieve the best possible compression performance. Prediction techniques aim to decorrelate the data so that the residue of the prediction becomes an independent random data set. The entropy of the error data set can always be minimized if better inferences can be made so that the probabilities allocated to error entries are maximized. A measure to evaluate the performance of the prediction techniques is the mean square error (MSE) function. Many linear adaptive prediction schemes try to minimize the MSE of images, and these techniques are known to produce very good compression ratios. Another criterion to perform very good compression ratios is to minimize the entropy of images. It is proven that this criterion proves to work better than the MSE based prediction [Jiang93]. Bearing in mind that images do not, in general, offer correlations optimal for linear predictors, the use of linear schemes does not
necessary imply a better spatial prediction and thus MSE prediction is not consequently an optimal based prediction since it does not yield optimal entropy of the predicted errors. In contrast with conventional prediction techniques, the method presented in this chapter consists of using a neural network predictor NN to approximate the conditional probability distribution of a single bit given \( n \) previous bits. Initially this technique was used for text compression since connectionist neural models are well suited for modelling language constraints [Feldman82]. Yet in this thesis, a new application of the NN algorithm to image compression will be realized. There are two variants known to this method: an off-line and on-line approach. The on-line approach will be investigated and improved. The use of this approach is justified by the fact that this particular type of neural network offers an efficient way of predicting the bit probability using the maximum entropy principle. In addition, it offers a very practical implementation using a hardware solution since it does not require the use of the total number of neurons involved in the prediction. This scheme works on the basis that only a very limited number of neurons are active at the same time and thus only a set of virtual neurons are designed to process the weights of the active neurons within the total number of neurons.

### 3.2.1 Off-Line Approach to the Neural Network Prediction

In this approach Schmidhuber and Heil replaced a Prediction-by-Partial Match PPM predictor with a 3-layer neural network. They trained the NN using back propagation algorithm to allocate character probabilities within a given context of input [Schmidhuber96].

In this variant, the neural network predictor \( P \) uses a set \( F \) of training files to perform the training. After the training process, the weights cannot be subjected to any changes and the same version of \( P \) is used for the coding and decoding process.

#### 3.2.1.1 Prediction of Conditional Probabilities

We assume that \( X \) is an alphabet that contains \( k \) possible symbols \( z_1, z_2, \ldots, z_k \). The elements \( z_i \) can be represented locally by a binary \( k \)-dimensional vector \( r(z_i) \) with exactly one non-zero component located at the \( i \)-th position.

We build our neural network predictor with \( n \times k \) input units and \( k \) output units. Where \( n \) is described as the “the time window size”.
The standard procedure begins by inserting $n$ default characters $z_0$ at the beginning of each file $f$ of the Set $F$. The character $r(z_0)$ is represented as a default by the $k$-dimensional zero-vector. The $m$-th character of the file $f$ is denoted by $C^f_m$.

The input of the neural network predictor $P$ is formed by the concatenation of the $n$ vectors

$$r(z_0): r(C^f_{m-n}) \circ r(C^f_{m-n+1}) \circ \ldots \circ r(C^f_{m-1})$$

for all $f$ taken from the set $F$ and for all possible $m>n$, where the symbol "\circ" is the concatenation operator for vectors. The output of the neural network predictor is a $k$-dimensional vector denoted by $P^f_m$. Figure 3-3 describes the architecture of the neural network predictor $P$.

![Figure 3-3: The neural configuration of the off-line approach.](image)

The weights updates are performed using a back-propagation algorithm that minimizes the optimization function $J$:

$$J = \frac{1}{2} \sum_{f \in F} \sum_{m>n} \left\| r(C^f_m) - P^f_m \right\|^2$$

The function $J$ reaches a minimum when $P^f_m$ is equal to the conditional expectation of $r(C^f_m)$ given the input $r(C^f_{m-n}) \circ r(C^f_{m-n+1}) \circ \ldots \circ r(C^f_{m-1})$ [Schmidhuber96]:

$$P^f_m = E(r(C^f_m) \mid C^f_{m-n}, \ldots, C^f_{m-1})$$
Schmidhuber and Heil have shown that \((P^f_m)_i\) is equal to the conditional probability

\[
P_r(C_m^f = z_i | C_{m-n}^f, \ldots, C_{m-1}^f)
\]

for all \(f\) and for all appropriate \(m>n\). \((P^f_m)_i\) denotes the \(j\)-th component of the vector.

Unfortunately this algorithm was too slow to be used in practical compression software. Training of 10K to 20K files needed several days of computation on HP700 workstation and the prediction phase was running 1000 times slower than the standard methods [Mahoney00]. An online approach based on a 2-layer network was proposed to overcome the previous network shortcomings. This method is able to learn and predict in a single pass typically achieving better compression ratios.

### 3.2.2 On-line Approach to the Neural Network Prediction

In this variant of the method Mahoney follows the approach of Schmidhuber and Heil (1996). The method is based on using on-line neural network prediction followed by arithmetic encoding [Mahoney00].

#### 3.2.2.1 On-line Neural Network Predictor

When modelling a particular data distribution we derive the joint distribution \(P(y | x)\), where \(y\) is the following bit and \(x\) the previous sequence. According to the maximum entropy principle we have [Mahoney00]:

\[
P(x, y) = \frac{1}{Z} \prod_i \alpha_i^{f(x,y)}
\]  

(3.4)

where \(f(x,y)\) is an arbitrary "feature" function equal to \(x_i\) in this case, and \(\alpha_i\) are parameters that are found by generalized iterative scaling (GIS) [Mahoney00]. \(Z\) is a normalization constant to make the probabilities sum to 1.

The principle of maximum entropy is used to determine the most likely joint distribution, given a partial set of constraints, which is the one with the highest entropy, or the most uniform distribution [Mahoney00].

When we take the log of the expression (3.4) and employ the property \(P(y | x) = P(x, y) / P(x)\), we obtain the following expression [Mahoney00]:

\[
P(y | x) = \frac{1}{Z'} \exp(\sum_i \log(\alpha_i) x_i)
\]  

(3.5)
where $Z$ is set so that

$$P(y = 0 \mid x) + P(y = 1 \mid x) = 1$$

(3.6)

When simplifying (3.4) we obtain:

$$P(y \mid x) = g \left( \sum_i w_i x_i \right)$$

(3.7)

where $w_i = \log(\alpha_i)$ and $g(x) = 1/(1 + e^{-x})$.

This is a two-layer perceptron neural network with $N$ input units $x_i$ and one output $P(y=1 \mid x)$.

The values of $x_i$ represent the active inputs from the active contexts. In other words if a context of previous values is present then the value of $x_i$ of that context would be 1 and the other values of $x_j$ where $j \neq i$ would be 0.

For any given pixel $P$ a context will consist of a combination of previous possible values preceding the pixel $P$. If we have four input sets that would mean that the previous values will range from one pixel to 3 pixels.

Each of these combinations will have an input context value $x_i$ that will be set to 1 if the correspondent context occurred before the pixel $P$.

For example, if we have the sequence: 120, 50, 243, 200, where $P=200$ is our pixel to be predicted, the contexts and the active input will consist of the values shown in table 3-1.

<table>
<thead>
<tr>
<th>Contexts</th>
<th>$x_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>No previous pixel</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>..</td>
<td>0</td>
</tr>
<tr>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>..</td>
<td>0</td>
</tr>
<tr>
<td>1, 1</td>
<td>0</td>
</tr>
<tr>
<td>1, 2</td>
<td>0</td>
</tr>
<tr>
<td>..</td>
<td>0</td>
</tr>
<tr>
<td>120, 50</td>
<td>1</td>
</tr>
<tr>
<td>..</td>
<td>0</td>
</tr>
<tr>
<td>1, 1, 1</td>
<td>0</td>
</tr>
<tr>
<td>1, 1, 2</td>
<td>0</td>
</tr>
<tr>
<td>..</td>
<td>0</td>
</tr>
<tr>
<td>120, 20, 243</td>
<td>1</td>
</tr>
<tr>
<td>..</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3-1: Values of contexts and active inputs for a 4-Set configuration.
When training the network, only the weights of the active contexts are computed, which means that the algorithm computes the probability:

\[ P(200|120,50,243) = g(w(200) + w(243,200) + w(50,243,200) + w(120,50,243,200)) \] (3.8)

The network is trained on-line by adjusting the weights \( w_i \) so that we minimize the error \( E = y - P(y) \), between the real output \( y \) and the predicted value \( P(y) \). We update the weights using the expressions:

\[ w_{i+1} = w_i + \Delta w_i \] (3.9)

\[ \Delta w_i = \eta x_i E \] (3.10)

where \( \eta \) is the learning rate.

The technique described in this section serves as the base on which the work presented in sections 3.3, 3.4 and 3.5 is relying on. The novelty involved here is based on finding a new hybrid structure to a lossless image compressor involving an improved version of the neural network by applying a better tuning of its parameters combined with a lossless decorrelation stage presented in section 3.5 and a Peano-Hilbert scan presented in section 3.4.

### 3.3 On-line Neural Network Predictor Applied to Image Compression

The neural network inputs are subdivided into 6 context sets. The total number of inputs is \( 4 \times 10^6 \) where one input from each set is active at any time. The total number of all possible values, which represents \( 10^{12} \), has not been considered in Mahoney’s work [Mahoney00]. The reason behind it is that not all contexts occur and the total number of possible contexts is very large to be stored in a RAM.

These contexts are similar to the ones used by PPM encoders, where the last pixels are used to predict the incoming pixel [Mahoney00].
Figure 3-4 represents the network configuration with 3 input context sets.

![Network Configuration Diagram](image)

**Figure 3-4: Overview of the Neural Network Predictor with 3 input sets.**

The context set selection for a 6 set configuration is carried out according to the following:

Set 1: The last byte and current 0-7 bits with a leading 1, concatenated to a 16 bits number.

Set 2: The last 2 bytes plus current bits and position, hashed to a 22 bits number.

Set 3: The last 3 bytes plus current bits and position, hashed to a 22 bits number.

Set 4: The last 4 bytes plus current bits and position, hashed to a 22 bits number.

Set 5: The last 5 bytes plus current bits and position, hashed to a 22 bits number.

Set 6: The last 6 bytes plus current bits and position, hashed to a 22 bits number.

The weight update is carried out according to the following expression:

\[ w_i^{t+1} = w_i^t + (R_s + R_l / \sigma^2) x_i E \]  

(3.11)

where \( R_s \) and \( R_l \) are the short and long term learning rates.

The variance of the training data in context \( x_i \) is given by

\[ \sigma^2 = \frac{(C_0 + d)(C_1 + d)}{(C_0 + C_1 + 2d)} \]  

(3.12)

where \( C_0 \) and \( C_1 \) are the counts of \( y=0 \) and \( y=1 \) in context \( x_i \). The value \( d \) is a parameter between 0 and 1 so that it avoids a division by 0.
Chapter 3: Neural Network-Based Image Compression

After the prediction step is carried out, an arithmetic encoder begins with a range \([0, 1)\), and divides the range into two sub-ranges for each input bit. The use of an arithmetic encoder is justified by the fact that it performs better than the other known classic compressors [Mahoney00]. It has been shown that the arithmetic encoder is optimal within one bit of the entropy [Mahoney00].

In our experiments we use two-layer neural network structured predictors NN1 and NN2 that have different parameters. The parameter values of NN1 were taken from [Mahoney00], where \( R_s = 0.02, R_L = 0.38, d = 0.5 \) and \( \lambda = 9.76 \times 10^{-4} \). These empirical parameters were chosen to work well on text compression. Compression of images is, in fact, another case where these parameters may not offer the same compression efficiency in term of ratio as to text compression. To adapt this neural network method to the problem of image compression, one approach would consist of trying to find suitable parameters than can improve the ratios.

To obtain these parameters, experiments have been carried out. The reason for using an empirical approach to that problem lies in the fact that, in general, images do not have the same probabilistic structures and therefore it is impossible to use a theoretical approach to model images. As a consequence, it is rather difficult to find optimal parameters to that neural network using optimization solutions. In practice, an improvement of relatively 5% can be considered as a good performance. And therefore this criterion is used to judge the performance of NN2.

To perform our experiments a set of 8-bit encoded gray level images was compiled which was used throughout the work presented in this thesis (Figure 3-5).

The first twelve images are 1024 by 1020 pixels satellite images from the SSTL mini-satellite UoSat-12. The first eight images are 10 m resolution panchromatic images: pan1, pan2, pan3, pan4, pan5, pan6, pan7, pan8, while the other ones are 32m resolution multispectral images: multi11, multi12, multi13, multi14, multi21, multi22, multi23, multi24, multi31, multi32, multi33, multi34, multi41, multi42, multi43, multi44. These images were taken in four spectral bands, which are blue, green, red and near infra-red [Sun01].
The satellite images were chosen so that they could give a representative overview of the different landscapes that are imaged by satellites. We can find some images that represent urban areas, mountainous, rocky and desert landscapes.

In addition to these satellite images we also use 10 natural images: (512x512 pixels), Barbara (512x512 pixels), Goldhill (512x512 pixels), F16 (512x512 pixels), Baboon (512x512 pixels), Peppers (512x512 pixels), Boats (720x576 pixels), Cwheel (800x600 pixels), Splash (512x512 pixels) and House (256x256 pixels).

Those natural images are well known standard images often used in the area of image processing and compression by researchers and engineers for benchmarking purposes. The natural images were used in order to compare the performances of our compression technique with the different results obtained from the literature. They were used before compiling a set of satellite images.
Chapter 3: Neural Network-Based Image Compression

Pan7

Pan8

Multi11

Multi12

Multi13

Multi14
Chapter 3: Neural Network-Based Image Compression

Multi33

Multi34

Multi41

Multi42

Multi43

Multi44
Figure 3-5: Test images set.
In sections 3.3, 3.4 and 3.5 only pan1, pan2, pan3, pan4, multi11, multi12, multi13, multi14, multi21, multi22, multi23, multi24 and the 10 natural images are used. The other images were introduced in a later stage of our research and have been used in 4.5.

To program the neural network compressor, the Visual C++ language has been used. Other programming languages might have been chosen such as JAVA, Delphi and C++ builder but with the resources available and the author's background, the Visual C++ was the most appropriate to implement the algorithm. The parameter values of the improved version of the neural network NN2 that realizes the 5% criteria are $R_S=0.02$, $R_L=0.148$, $d=0.5$ and $\lambda=4.76 \times 10^{-4}$.

Compression ratios for the NN1 and NN2 compressors and the RICE algorithm are presented in Table 3-2. The satellite testing set consists of four panchromatic images and eight multispectral images taken from the test images set.

<table>
<thead>
<tr>
<th>Images</th>
<th>NN2</th>
<th>NN1</th>
<th>RICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan1</td>
<td>1.601</td>
<td>1.441</td>
<td>1.306</td>
</tr>
<tr>
<td>Pan2</td>
<td>1.603</td>
<td>1.451</td>
<td>1.308</td>
</tr>
<tr>
<td>Pan3</td>
<td>1.563</td>
<td>1.400</td>
<td>1.299</td>
</tr>
<tr>
<td>Pan4</td>
<td>1.567</td>
<td>1.408</td>
<td>1.306</td>
</tr>
<tr>
<td>Multi11</td>
<td>1.631</td>
<td>1.504</td>
<td>1.327</td>
</tr>
<tr>
<td>Multi12</td>
<td>1.803</td>
<td>1.713</td>
<td>1.388</td>
</tr>
<tr>
<td>Multi13</td>
<td>1.978</td>
<td>1.923</td>
<td>1.423</td>
</tr>
<tr>
<td>Multi14</td>
<td>2.073</td>
<td>2.012</td>
<td>1.445</td>
</tr>
<tr>
<td>Multi21</td>
<td>1.790</td>
<td>1.660</td>
<td>1.401</td>
</tr>
<tr>
<td>Multi22</td>
<td>1.991</td>
<td>1.910</td>
<td>1.501</td>
</tr>
<tr>
<td>Multi23</td>
<td>2.416</td>
<td>2.387</td>
<td>1.713</td>
</tr>
<tr>
<td>Multi24</td>
<td>2.636</td>
<td>2.629</td>
<td>1.839</td>
</tr>
<tr>
<td>Average</td>
<td>1.888</td>
<td>1.787</td>
<td>1.438</td>
</tr>
</tbody>
</table>

Table 3-2: Compression ratios using the predictors NN1 and NN2.

Table 3-2 shows that NN2 compresses 5.7% better than NN1. The parameters used for NN2 are not optimal but performed best among a set of parameters. NN2 gave a better prediction because the combination of the short and long term learning rates was better made to favour using the most recent inputs over the whole input history in order to adapt to the change of the image statistics. This means that the satellite images involved in the experiments contain pixels that are more dependent to close neighbouring pixels.
In the next subsections we discuss the effect of combining the on-line network scheme with some preprocessing techniques such as the Peano-Hilbert scan and the integer wavelet transforms.

### 3.4 Investigation of the Effect of Peano-Hilbert Scan

In section 2.7, the Peano-Hilbert scan has been shown to provide the best optimisation of the formula (2.52) among other space filling curves. This performance is realized thanks to its property of strong locality. The Peano-Hilbert scan tries to reorder the image pixels into a one-dimensional vector that exploits the spatial coherence of neighbouring pixels. Nevertheless, the neural network scheme presented in section 3.1 performs its prediction in one pass using a line-by-line scan as for text compression. This functionality does not allow the neural network compression algorithm to exploit the two-dimensional coherence of the image but treats it as a one-dimensional vector input. A way to adapt the compressor to a two-dimensional input data is investigated in this section and in section 3.5.

To investigate the effect of Peano-Hilbert scan on the compression ratios of the neural network-based method, we carried out some experiments on well known natural images described in section 3.3, and compared the ratios obtained using Peano-Hilbert scan with the raster scan, spiral scan (S) and the Burrow Wheeler Transform (BWT) [Sayood00]; used as a decorrelation stage and also with the CALIC algorithm. The CALIC software was provided by the University of Western Ontario’s Department of Computer Science for research and evaluation purposes. The scans were programmed using Matlab. Figure 3-6 shows a block diagram of the described method.

![Figure 3-6: Block diagram of the method.](image-url)
The results of the compression are listed in Table 3-3.

<table>
<thead>
<tr>
<th>Images</th>
<th>NN1+PH</th>
<th>NN1+S</th>
<th>NN1+BWT</th>
<th>NN1</th>
<th>CALIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lena</td>
<td>4.560</td>
<td>2.962</td>
<td>3.997</td>
<td>2.924</td>
<td>2.062</td>
</tr>
<tr>
<td>Barbara</td>
<td>3.054</td>
<td>3.039</td>
<td>2.789</td>
<td>2.910</td>
<td>1.545</td>
</tr>
<tr>
<td>Goldhill</td>
<td>3.550</td>
<td>3.483</td>
<td>3.344</td>
<td>3.504</td>
<td>1.565</td>
</tr>
<tr>
<td>F16</td>
<td>4.701</td>
<td>4.572</td>
<td>4.761</td>
<td>4.773</td>
<td>1.861</td>
</tr>
<tr>
<td>Baboon</td>
<td>1.169</td>
<td>1.135</td>
<td>1.168</td>
<td>1.179</td>
<td>1.392</td>
</tr>
<tr>
<td>Peppers</td>
<td>1.469</td>
<td>1.441</td>
<td>1.412</td>
<td>1.427</td>
<td>1.778</td>
</tr>
<tr>
<td>Splash</td>
<td>1.795</td>
<td>1.837</td>
<td>1.769</td>
<td>1.790</td>
<td>2.172</td>
</tr>
<tr>
<td>House</td>
<td>1.676</td>
<td>1.672</td>
<td>1.708</td>
<td>1.735</td>
<td>2.073</td>
</tr>
<tr>
<td>Average</td>
<td>2.747</td>
<td>2.518</td>
<td>2.619</td>
<td>2.530</td>
<td>1.806</td>
</tr>
</tbody>
</table>

Table 3-3: Compression ratios after applying the different scans and the Burrow Wheeler transform

In this experiment we used the NN1 predictor with the parameters $R_S=0.02$, $R_L=0.38$ and $\lambda=9.76 \times 10^4$. The scan performed for NN1 is the raster scan. Table 3-3 shows that the Peano-Hilbert scan brought an improvement of 8.5% to the neural network-based method. The Spiral scan degraded the ratio by 4.9%. The BWT combined with NN1 achieved a very good compression ratio with Lena, but did not perform well on the other images when compared to the other methods.

The PH+NN1 exploits well the correlations and smoothness that exist within some images such as Lena, Barbara, Goldhill and F16 resulting in high compression ratios. However for images, such as Baboon, which are characterized by their non-smoothness, the method did not yield high compression ratios.

We applied the PH+NN2 method to the testing set of satellite images, and compared the results to the ratios obtained with the RICE algorithm [Atek02a]. Contacts have been established with ESA/ESTEC (Software Engineer Raffaele Vitulli) as a result, of which the executable of the full version of the RICE software has been acquired and has been used for experimental work.
Figure 3-7 shows that the PH+NN2 method outperforms the RICE algorithm with an average ratio of 2.425.

![Comparison of compression ratios](image)

**Figure 3-7: Comparison of the different compression ratios.**

The positive effect of Peano-Hilbert scan can easily be observed from the experiment results. This result comes along with the analysis made in section 2.7. The Peano-Hilbert scan succeeds to determine a good arrangement of pixels which best preserves the distance of the one-dimensional ordering in the two dimensional arrangement.

Figure 3-8 shows the effect of the application of Peano-Hilbert scan to the popular grayscale image Lena.

![Lena scanned with a Peano Hilbert scan](image)

**Figure 3-8: Lena scanned with a Peano Hilbert scan.**
The Peano-Hilbert scan of Lena shows that the pixels along the rows are very similar in brightness intensity. This effect helps the neural network compressor to perform better since it works on the basis of one-dimensional scan of pixels. This original NN scheme has been improved in that way by combining it with an efficient scan. Now, thanks to that, the neural network technique is adapted to solve image compression problems.

3.5 Using the Integer Wavelet Transform

Satellite images may be corrupted due to errors that occur during image capture. A useful capability is to be able to preview the transmitted data at any time of the transmission. Consequently, only exploitable images can be fully downloaded saving in this way transmission time. Such a capability can be provided by wavelet transforms [Atek02c]. IWTs can construct integer outputs given integer inputs and guarantee a perfect reconstruction of the image. The use of DWT as a decorrelation stage instead of the IWT, leads to data loss since the property of perfect reconstruction is not guaranteed [Reichel01]. Another advantage of Integer Wavelet Transform lies in the fact that it allows a progressive transmission of images. It is possible to extract at any level of transmission a lossy version of the transmitted data. IWT can allow transmitting first the thumbnails. One can choose the downloading order of the images depending on the importance and the priority to their application. For example, if the selected image in
Figure 3-9 represents a flood disaster, this image will have the priority to be downloaded first.

With the multiresolution property of the IWT it is also possible to extract any interesting area of the image (Figure 3-10). As a result, the main information of the image is directly exploited: it saves time by transmitting only targeted data and allows more images to be sent to the ground station.
3.5.1 IWT-Based Neural Compressor

After decomposing the images using a (2,2) filter for 1-level of decomposition we applied the NN predictor with the parameters used in NN2. We noticed that the NN method performed better in high frequency sub-images. We increased the level of decomposition and compared the results. The best performances were obtained for \( n=2 \). This value gives a good trade-off between the ratio and the calculation complexity \([Atek02b]\). The (2,2) filter was selected because it is the simplest filter taken from 2.6.1 in terms of calculation complexity. Figure 3-11 shows a block diagram of the proposed method.

![Block diagram of the proposed image compression method.](image)

Table 3-4 lists the ratios obtained after applying a 2-level IWT using a (2,2) filter (IWTNN).

<table>
<thead>
<tr>
<th>Images</th>
<th>IWTNN</th>
<th>NN</th>
<th>RICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan1</td>
<td>1.430</td>
<td>1.441</td>
<td>1.306</td>
</tr>
<tr>
<td>Pan2</td>
<td>1.367</td>
<td>1.451</td>
<td>1.308</td>
</tr>
<tr>
<td>Pan3</td>
<td>1.418</td>
<td>1.400</td>
<td>1.299</td>
</tr>
<tr>
<td>Pan4</td>
<td>1.426</td>
<td>1.408</td>
<td>1.306</td>
</tr>
<tr>
<td>Multi11</td>
<td>1.472</td>
<td>1.504</td>
<td>1.327</td>
</tr>
<tr>
<td>Multi12</td>
<td>1.588</td>
<td>1.713</td>
<td>1.388</td>
</tr>
<tr>
<td>Multi13</td>
<td>1.727</td>
<td>1.923</td>
<td>1.423</td>
</tr>
<tr>
<td>Multi14</td>
<td>1.794</td>
<td>2.012</td>
<td>1.445</td>
</tr>
<tr>
<td>Multi21</td>
<td>1.537</td>
<td>1.660</td>
<td>1.401</td>
</tr>
<tr>
<td>Multi22</td>
<td>1.651</td>
<td>1.910</td>
<td>1.501</td>
</tr>
<tr>
<td>Multi23</td>
<td>1.975</td>
<td>2.387</td>
<td>1.713</td>
</tr>
<tr>
<td>Multi24</td>
<td>2.199</td>
<td>2.629</td>
<td>1.839</td>
</tr>
<tr>
<td>Average</td>
<td>1.632</td>
<td>1.787</td>
<td>1.438</td>
</tr>
</tbody>
</table>

Table 3-4: Compression ratios after applying a (2,2) IWT for 2-levels of decomposition.
“IWTNN” stands for the neural network predictor based method (NN) with an IWT decorrelation stage and “RICE” stands for the Rice algorithm.

From Table 3-4 we can notice that the use of IWT with the NN compressor outperformed the RICE algorithm. The results were not better than those obtained using only the NN predictor and the arithmetic compressor, but the use of the IWT gives additional functionality allowing us to progressively transmit the image.

### 3.6 Comparison of the Neural Network-based Methods Against the Rice Algorithm and JPEG2000

The NN-based method was also compared to JPEG2000. The results of the Rice algorithm derived from sections 3.3, 3.4 and 3.5 are also reported in this section. The average compression ratio obtained with JPEG2000 is 39.5% less than the PHNN method and 5.6% less than the IWTNN technique. Although JPEG2000 is generally good for Lossless compression, it works better for lossy and near-lossless compression [Taubman01]. Figure 3-12 shows the different compression ratios obtained using the NN-based techniques against JPEG2000 [Atek03a].

![Figure 3-12: Compression performances of the NN-based methods compared with the RICE Algorithm and JPEG2000.](image-url)
The JPEG2000 software was obtained from the “JPEG2000: Image Compression Fundamentals, Standards, and Practice” book’s CD-ROM [Taubman01]. In this experiment, JPEG2000 uses a 5/3 wavelet filter. The quantization was set to step 1 and the component transformation was not used since compression was applied on gray-level images. It is possible to observe that the Rice compressor is not achieving very high compression ratios (Table 3-5). One of the reasons is because the predictor in this method is a 1-DPCM predictor, which does not take into account the correlations of the pixels in the plane domain. One improvement to the algorithm can be to change the 1-D predictor by a 2-D filter that gives a better approximation of the pixel errors. It was shown that it is possible to use higher predictor types in [Coco00]. In addition, the mapper can also be bypassed by a more suitable one.

Table 3-5 shows the results in details obtained in Figure 3-12.

<table>
<thead>
<tr>
<th></th>
<th>RICE</th>
<th>JPEG2K</th>
<th>IWTNN</th>
<th>PHNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan1</td>
<td>1.306</td>
<td>1.333</td>
<td>1.430</td>
<td>1.964</td>
</tr>
<tr>
<td>Pan2</td>
<td>1.308</td>
<td>1.330</td>
<td>1.367</td>
<td>1.964</td>
</tr>
<tr>
<td>Pan3</td>
<td>1.299</td>
<td>1.324</td>
<td>1.418</td>
<td>1.902</td>
</tr>
<tr>
<td>Pan4</td>
<td>1.306</td>
<td>1.334</td>
<td>1.426</td>
<td>1.910</td>
</tr>
<tr>
<td>Multi11</td>
<td>1.327</td>
<td>1.439</td>
<td>1.472</td>
<td>2.090</td>
</tr>
<tr>
<td>Multi12</td>
<td>1.388</td>
<td>1.517</td>
<td>1.588</td>
<td>2.389</td>
</tr>
<tr>
<td>Multi13</td>
<td>1.423</td>
<td>1.586</td>
<td>1.727</td>
<td>2.773</td>
</tr>
<tr>
<td>Multi14</td>
<td>1.445</td>
<td>1.620</td>
<td>1.794</td>
<td>3.005</td>
</tr>
<tr>
<td>Multi21</td>
<td>1.401</td>
<td>1.515</td>
<td>1.537</td>
<td>2.401</td>
</tr>
<tr>
<td>Multi22</td>
<td>1.501</td>
<td>1.604</td>
<td>1.651</td>
<td>2.724</td>
</tr>
<tr>
<td>Multi23</td>
<td>1.713</td>
<td>1.849</td>
<td>1.975</td>
<td>3.558</td>
</tr>
<tr>
<td>Multi24</td>
<td>1.839</td>
<td>2.020</td>
<td>2.199</td>
<td>3.883</td>
</tr>
<tr>
<td>Average</td>
<td>1.438</td>
<td>1.539</td>
<td>1.632</td>
<td>2.547</td>
</tr>
</tbody>
</table>

Table 3-5: Compression ratios of the proposed NN-based methods against the RICE algorithm and JPEG2000.
In figure 3-12 and table 3-5, "PHNN" stands for the neural network predictor based method with a Peano-Hilbert scan, "JPEG2K" stands for the JPEG2000 algorithm, "IWTNN" stands for the neural network predictor based method (NN) with an IWT decorrelation stage and "RICE" stands for the Rice algorithm.

3.7 Conclusions

In this chapter we have presented two new image compression schemes for small satellite on-board imaging. The compressor is based on a neural network prediction and an arithmetic encoder combined with a Peano-Hilbert scan or an integer wavelet transform. The advantage of this predictive method is that the neural network can learn and predict in one pass. It gives the best prediction when short and long learning rates are combined to balance, between using the whole input history and the most recent inputs. This process allows the neural network to adapt to the change of the image statistics.

The neural network predictor was improved using empirically adjusted parameter values. The testing results obtained over the satellite images showed that the improved neural network method outperformed the Rice algorithm by 31.2%.

A Peano-Hilbert scan was introduced to exploit the correlations along two dimensions. The results indicated that the method based on the Peano-Hilbert scan combined with the improved neural network (PH+NN2) largely outperformed the Rice algorithm with an average compression ratio of 2.425.

A (2,2) integer wavelet filter was combined with the neural network compressor. This type of wavelet transform is used because it preserves the property of perfect reconstruction, which is essential in lossless compression. The IWTs are very useful for satellite applications because they can enable progressive transmission of images creating thumbnails and assessing image quality.

The IWTNN technique yielded a compression ratio of 1.632 over the test images.

A comparison of the NN-based techniques (IWTNN and PHNN) against JPEG2000 showed that these novel techniques altogether outperformed JPEG2000 on average by 13.8%.
Chapter 4

4 Hardware Implementation of the Neural Network Image Compressor

In the previous chapter we detailed the design of a novel neural network compressor. The proposed NN-based compression scheme yields high compression ratios as discussed in chapter 3, however the processing time and the memory usage of this neural model needs to be improved. In this chapter we propose an optimised design of the neural network compressor and the arithmetic encoder in order to offset the complexity problems.

In section 4.1, two approaches are discussed: hardware approach and software approach. Section 4.2 discusses the complexity problem of our proposed method. The implementation of the neural compressor in hardware is presented in section 4.3. The functional simulation results are presented in section 4.4. In section 4.5 a decision support mechanism is described. A new onboard image compression system is proposed in 4.6. Finally, the conclusions are drawn in section 4.6.

4.1 Software or Hardware Compression Solution

Software solutions are easy to design but unfortunately they are slow and therefore can be inefficient when real-time compression is required. In addition, the Von Neumann computation style fails to exploit the intrinsic parallelism in an algorithm. For example, a DSP implementation of a 30-tap FIR filter would require 30 Multiply Accumulate Cycles (MAC) [Mathen96], while a programmable logic device can perform the 30 MAC operations in parallel over a single cycle. Thus, it is always attractive to use hardware solutions to satisfy the high-speed requirements of real-time digital systems. When high-speed processing is desired, massively parallel VLSI implementation can provide a great potential of high performance. Nevertheless conventional hardware systems such as ASICs present high development costs and cannot offer flexibility for post-launch modifications [Economou94].
To surmount these limitations, a better option can be to implement the image compression algorithm on a reconfigurable platform that can be reprogrammable while on orbit. Such a hardware solution will lead to a faster implementation than a conventional software approach, and in the same way will allow small satellites to be equipped with more affordable On-board Data Handling (OBDH) systems and upgradeable onboard computing for a wider field of application [FRY01].

With the advent of Field Programmable Gate Arrays (FPGAs) in the market, it is now possible to realize custom hardware implementation without any physical modification, and therefore making satellites flexible with new mission requirements [FRY01].

4.2 Complexity Versus Performance

This section presents results of experimental work, which aims to investigate how the complexity of the neural network configuration affects the performance of the NN scheme (without a preprocessing stage) that was described in section 3.2. The experiments are carried out using a personal computer (PC) with a Pentium III 750MHz microprocessor. The graph in Figure 4-1 shows the relationship between the number of neurons and the processing time of the NN, which is almost a linear dependence. The numbers of neurons were derived from Mahoney’s work [Mahoney00].

![Figure 4-1: Processing times of the NN coder with different numbers of inputs.](image-url)
Table 4-1 details the processing times required to compress images taken from the experimental data set in section 3.3.

<table>
<thead>
<tr>
<th></th>
<th>1 set</th>
<th>2 sets</th>
<th>3 sets</th>
<th>4 sets</th>
<th>5 sets</th>
<th>6 sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan1</td>
<td>7.86</td>
<td>10.63</td>
<td>13.19</td>
<td>15.52</td>
<td>17.75</td>
<td>19.41</td>
</tr>
<tr>
<td>Pan2</td>
<td>7.81</td>
<td>10.66</td>
<td>13.19</td>
<td>15.42</td>
<td>17.67</td>
<td>19.42</td>
</tr>
<tr>
<td>Pan3</td>
<td>7.8</td>
<td>10.72</td>
<td>13.22</td>
<td>15.42</td>
<td>18.02</td>
<td>19.55</td>
</tr>
<tr>
<td>Pan4</td>
<td>7.81</td>
<td>10.67</td>
<td>13.3</td>
<td>15.42</td>
<td>17.02</td>
<td>19.45</td>
</tr>
<tr>
<td>Multi1</td>
<td>7.8</td>
<td>10.47</td>
<td>12.92</td>
<td>15.06</td>
<td>17.61</td>
<td>19.09</td>
</tr>
<tr>
<td>Multi2</td>
<td>7.76</td>
<td>10.42</td>
<td>12.67</td>
<td>14.69</td>
<td>16.95</td>
<td>18.73</td>
</tr>
<tr>
<td>Multi3</td>
<td>7.69</td>
<td>10.14</td>
<td>12.53</td>
<td>14.41</td>
<td>16.67</td>
<td>18.33</td>
</tr>
<tr>
<td>Multi4</td>
<td>7.67</td>
<td>11.97</td>
<td>12.005</td>
<td>14.27</td>
<td>16.66</td>
<td>18.2</td>
</tr>
<tr>
<td>Multi21</td>
<td>7.83</td>
<td>10.56</td>
<td>12.69</td>
<td>15.52</td>
<td>17.08</td>
<td>18.58</td>
</tr>
<tr>
<td>Multi22</td>
<td>7.76</td>
<td>10.16</td>
<td>12.27</td>
<td>14.5</td>
<td>16.73</td>
<td>18.34</td>
</tr>
<tr>
<td>Multi23</td>
<td>7.61</td>
<td>10.02</td>
<td>11.83</td>
<td>13.88</td>
<td>16.11</td>
<td>17.59</td>
</tr>
<tr>
<td>Multi24</td>
<td>7.59</td>
<td>9.78</td>
<td>11.59</td>
<td>13.73</td>
<td>15.91</td>
<td>17.42</td>
</tr>
<tr>
<td>Average</td>
<td>7.749167</td>
<td>10.51667</td>
<td>12.61717</td>
<td>14.82</td>
<td>17.015</td>
<td>18.67583</td>
</tr>
</tbody>
</table>

Table 4-1: Processing times required to compress some SSTL satellite images (in seconds).

It can be seen that the time taken by the neural calculations increases with the size of the neural network when the processing is carried out by a uniprocessor system.

The graph in Figure 4-2 assesses the influence of the number of input sets on the compression ratios. The results show that a neural network with three input sets, achieves a good compression ratio while having a configuration of lower complexity.

![Figure 4-2: Compression ratios of the NN coder using different numbers of inputs sets.](image)
Table 4-2 shows in detail the different compression ratios obtained on SSTL images taken from the test data set presented in section 3.3 after applying a Peano-Hilbert scan.

<table>
<thead>
<tr>
<th></th>
<th>1 set</th>
<th>2 sets</th>
<th>3 sets</th>
<th>4 sets</th>
<th>5 sets</th>
<th>6 sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan1</td>
<td>1.766</td>
<td>1.8865</td>
<td>1.9528</td>
<td>1.9762</td>
<td>1.9867</td>
<td>1.9906</td>
</tr>
<tr>
<td>Pan2</td>
<td>1.7869</td>
<td>1.8945</td>
<td>1.9573</td>
<td>1.978</td>
<td>1.9838</td>
<td>1.9826</td>
</tr>
<tr>
<td>Pan3</td>
<td>1.7377</td>
<td>1.8378</td>
<td>1.895</td>
<td>1.9152</td>
<td>1.9252</td>
<td>1.9264</td>
</tr>
<tr>
<td>Pan4</td>
<td>1.748</td>
<td>1.8476</td>
<td>1.905</td>
<td>1.927</td>
<td>1.937</td>
<td>1.9377</td>
</tr>
<tr>
<td>Multi1</td>
<td>1.91622</td>
<td>2.0044</td>
<td>2.0801</td>
<td>2.1048</td>
<td>2.1168</td>
<td>2.119</td>
</tr>
<tr>
<td>Multi2</td>
<td>2.1108</td>
<td>2.240122</td>
<td>2.353</td>
<td>2.393</td>
<td>2.4032</td>
<td>2.41</td>
</tr>
<tr>
<td>Multi3</td>
<td>2.323</td>
<td>2.526</td>
<td>2.697</td>
<td>2.7506</td>
<td>2.75995</td>
<td>2.7685</td>
</tr>
<tr>
<td>Multi4</td>
<td>2.432</td>
<td>2.6786</td>
<td>2.8855</td>
<td>2.944</td>
<td>2.9548</td>
<td>2.9619</td>
</tr>
<tr>
<td>Multi11</td>
<td>2.158</td>
<td>2.277</td>
<td>2.3795</td>
<td>2.412</td>
<td>2.417</td>
<td>2.4147</td>
</tr>
<tr>
<td>Multi22</td>
<td>2.4</td>
<td>2.555</td>
<td>2.685</td>
<td>2.728</td>
<td>2.73301</td>
<td>2.733</td>
</tr>
<tr>
<td>Multi23</td>
<td>2.97</td>
<td>3.255</td>
<td>3.4626</td>
<td>3.5185</td>
<td>3.520014</td>
<td>3.5158</td>
</tr>
<tr>
<td>Average</td>
<td>2.209993</td>
<td>2.376635</td>
<td>2.500983</td>
<td>2.538417</td>
<td>2.545854</td>
<td>2.547225</td>
</tr>
</tbody>
</table>

Table 4-2: Compression ratios obtained after compressing SSTL satellite images.

From table 4-2 we can observe that the simplest configuration (1-set configuration) achieves higher compression ratios than the Rice algorithm (1.438) and JPEG2000 (1.539) obtained in section 3.6.

The upper graph in Figure 4-3 shows the memory usage of the NN compressor at a given number of input sets.

![Figure 4-3: Memory usage of the NN coder as a function of the numbers of inputs sets.](image-url)
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

The neural predictor requires RAM memory to store the weights, the capacity of which depends on the number of neurons. It can be seen that the graph follows an exponential shape where the ratio between the RAM size and the number of neurons is 4. For 3 sets, the memory required to implement the NN is 532 KB which is 8 times less than with 6 sets. The lower graph shows the number of neurons in multiples of 1000.

The graphs in Figure 4-1, 4-2 and 4-3 show that the number of neural networks sets affects the efficiency and complexity of the neural network processor considerably. The results of the experiments point out that three sets represent a good compromise between the compression ratio, the processing time and RAM usage. Therefore three is considered an optimal number for the number of neural network sets and is retained in all subsequent work.

4.3 Implementation of the NN Compressor in Hardware using Parallel Processing

When a real-time compression scheme is implemented, software solutions can be inefficient. Their main disadvantage is their low speed since microprocessors are not customized for a specified task. Moreover, the latency of a software neural model executed on a stand-alone Von Neumann’s uniprocessors increases with the size of the network [Skrbek99] as observed in Figure 4-1. On the other hand, neural networks present a great potential in the high speed processing that can be provided through parallel VLSI implementations [Lippman87]. FPGA chips seem more suitable implementation due to their high reconfigurability, allowing introduction of changes to the hardware.
The main disadvantage of the NN method is the high processing time as it can be observed in Figure 4-1. To overcome this drawback a parallel processing scheme is proposed. Figure 4-4 shows the block-diagram of the suggested parallel configuration.

Figure 4-4: Proposed approach for a parallel processing.

The image is split into smaller blocks, referred to as tiles that are encoded in parallel by a 1-D array of identical NN processors. The design of the NN processors is detailed in section 4.3.1.

Figure 4-5 shows the processing time needed by a PIII 750MHz microprocessor to compress tiles, with sizes ranging from 16x16 pixels to 1024x1024 pixels, using 3 sets of inputs selected from 133x10^3 neurons. The value of three is used since it is the optimal number of the neural network sets as established in section 4.2.

Figure 4-5: Processing times needed to compress sub-image tiles of different sizes.
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

It is shown that tiles that are of size 16x16, 32x32 and 64x64 pixels need almost the same time of processing, which is rather small. However, tile size must be big enough such that less neural processors are used. It is shown that tiles that are of size 16x16, 32x32 and 64x64 pixels need almost the same time of processing, which is rather small. However, tile size must be big enough such that less neural processors are used.

In order to determine an optimal tiling size (whether 256x256 pixels or 512x512 pixels) of sub-images it is possible to use the formula below such that to minimize $T_T$, which is the total time required to compress an image:

$$T_T = T \times \left( \left \lfloor a \right \rfloor + \left \lceil a - \left \lfloor a \right \rfloor \right \rceil \right)$$

Where

$$a = \frac{\text{image size}}{\text{tile size} \times \text{number of sets}}$$

$T_T$ is the total time required to compress the image and $T$ is the time required by one neuron processor to process a tile. The decision of the tile size can also depend on the size of the FPGA and the buffer memory available onboard to store the sub-images.

The proposed hardware design (Figure 4-4) offers important advantages for an on-board implementation such as isolation of errors, increased reliability and real-time compression, as explained below:

- **Error Isolation:** The problem of communication and computation in the presence of errors is difficult, and general solutions can be time consuming and inflexible. The arithmetic compressor is very sensitive to bit errors and consequently synchronization of data can be lost [Grangetto03]. In fact, a single bit flip in the compressed stream that can be caused whether by single errors upset or a transmission error can lead to irreversible error propagation throughout the retrieved image. In addition, bits of corrupted encoded data while data is in storage will in general not be recoverable and may corrupt the entire decompressed file. The presented neural scheme realizes a decomposition of the image into tiles of 256x256 pixels. This will allow a partitioning of the pixels into blocks that are separately encoded with the arithmetic encoder. Consequently errors, if they occur, are isolated into a single compressed tile, and in that way, the entire image is protected from the general propagation from a local area in the image. The sizes of the satellite images are very large, and the ratio between one image tile and the whole image is around $10^{-4}$, taking into account that an Alsat1 image can reach the size of 19000x19000 pixels.
Figure 4-6 shows an example of irreversible error propagation throughout the decompressed image after using the NN-based compression when no tiling was applied. Figure 4-7 shows that the tiling of the image confines the errors into one tile with no propagation when the same bit flip was applied.

Figure 4-6: Irreversible propagation of the errors in the decompressed image.

Figure 4-7: Confinement of the errors into one block.
• **Redundancy of the neural processors:** Reliability of hardware devices on-board a spacecraft is of primary importance due to radiation effect, especially after a long flight period. Failure of subsystems can cause a malfunctioning of the whole system. The parallel configuration of the neural network compressor offers an inherent redundancy of the neural processors within the whole functioning of the scheme. Therefore failure of one neural processor will not lead to the total failure of the system, since the processing tasks can be redirected to the remaining healthy neural processors. The side effect is a slowdown in the compression speed. In case of configuration with 4 neural processors, failure of one processor will decrease the system operation by 25%.

• **Real-time compression:** Continuous streaming of data is an important requirement of on-board compression and communication. Real-time compression allows the data to flow directly to the on-board memory without using extra buffering and saving the on-board memory for other on-board applications such as attitude control, on-board image processing and data/program storage for the On-board computers. The development of our scheme on-board small satellites provides reliability in the processing and management of images in real time. It allows the images to be compressed directly from the imager and stored in the memory in a compressed format.

### 4.3.1 Neural Network Processor Design

As explained in section 4.3, the NN compressor is implemented as a set of identical NN processors. In this section we detail the design of the NN processor in hardware. The register-transfer level (RTL) representation of the NN compressor has been captured in the hardware description language VHDL and has been simulated using the Aldec's Active-HDL software package. It is aimed at implementation in the form of high-density FPGA, for example the Xilinx Virtex family of FPGA. The VHDL is a hardware description language that is commonly used by hardware designers. Other description languages such as Handel-C could have been used in this work, but with the resources available and the author’s background, the VHDL was the most appropriate hardware description language for this research.
Figure 4-8 shows the methodology used for implementing an FPGA-based application.

The following design considerations were taken into account during the translation process from the software programme to a digital hardware structure.

- The logic components used by a single NN-processor are to be minimized in order to maximize the number of NN processors in the FPGA chip;
- Integer precision calculations are preferable to double precision ones in order to make the hardware design low complex and more effective;
- The data bit-length should be optimised so that less wiring is required;
- The degree of parallelism is to be maximized via better understanding and modification of the algorithm.
The neural network processor block diagram is shown in Figure 4-9.

![Figure 4-9: The NN processor block diagram.](image.png)

The pixels are fed into a parallel-in serial-out register that will carry the inputs from an external memory to the compressor block. The input bit $Y$ is simultaneously fed into three blocks at the same time: active input selection block, weight update block and the arithmetic encoder. After the selected weights are updated they are summed and then saturated with the activation function. The result of this operation is the probability of the bit $Y$ within a context of previous bits. The arithmetic encoder will use this data and the value of the bit $Y$ to perform the subdivision, which will result in the final code. A synchronizer supervises the execution of the process following a sequential order of events; a state machine is not necessary for such a configuration. The main components are detailed in the following subsections.
4.3.1.1 Active Input Selection block

The active input selection block was designed for a six set configuration (Figure 4-10) but for a 3-set configuration only $X(0)$, $X(1)$ and $X(3)$ are fed into the weights update block. For a 1-set configuration only $X(0)$ is used.

![Figure 4-10: Active Input Selection Block.](image)

The active input selection block outputs the addresses of the weights than are selected in the RAM. The outputs are computed according to the following:

- $X(0)$: The last byte and current 0-7 bits with a leading 1, concatenated to a 16 bits number.
- $X(1)$: The last 2 bytes plus current bits and position, hashed to a 22 bits number.
- $X(2)$: The last 3 bytes plus current bits and position, hashed to a 22 bits number.

In the design, we use shift registers to store the values of previous pixel that are used with the input bit $y$ to compute the outputs of the active input selection block.

4.3.1.2 Activation Function Block

The activation function is generated by the AF block as specified by equation (3.7). It cannot be implemented in hardware without using a large area of the chip. Therefore in our design we use linear interpolations to approximate the sigmoid function.

The sigmoid values are first computed every $64k$ using Matlab and then saved in a ROM inside the FPGA chip [Atek03a]. Then any value between the sub-samples $64k$ and $64(k+1)$ is computed using a linear interpolation. This scheme can be simplified to obtain less complexity using shifts higher than 64, but this will sacrifice, in a way, accuracy.

Since the Sigmoid function is an odd function we can use this property to deduce the values of $g(x)$ when $x$ is negative. That will allow us to save only the positive values in
the ROM. Figure 4-11 shows the values of the sigmoid function that are saved in the FPGA ROM.

\[ \text{constant N\_TABLE\_SIGMA: natural;} = 175; \]
\[ 
\text{type SIGMOID is array (0 to N\_TABLE\_SIGMA) of unsigned(15 downto 0);} 
\]
\[ \text{constant TABLE: SIGMOID:=( x"0000", x"0EFF", x"7F7F", x"BEEF", x"8FEA", x"91B6", \}
\[ \x"9785", \x"B6F4", \x"9F59", \x"A114", \x"A4BF", \x"A5BE", \x"B150", \x"84AE", \}
\[ \x"E7FS", \x"EB6D", \x"BFD0", \x"C41B", \x"06F0", \x"C9B5", \x"CC5B", \x"CED9", \}
\[ \x"E14C", \x"D3A3", \x"BEED", \x"D509", \x"E112", \x"EC12", \x"EF6E", \x"EFC3", \x"E17B", \}
\[ \x"E31F", \x"E4AF", \x"E62C", \x"E797", \x"EBEF", \x"E136", \x"E604", \x"E9C4", \x"E6AC", \}
\[ \x"EE85", \x"EFE0", \x"F09E", \x"F17F", \x"F254", \x"F31D", \x"F3DB", \x"F46F", \x"F539", \}
\[ \x"F5DA", \x"F671", \x"F7D0", \x"F865", \x"F91E", \x"F95A", \x"F9B9", \x"FC66", \x"FC96", \x"FC11", \x"FD02", \x"FD5B", \x"FD83", \x"FEA9", \x"FC4F", \x"FEEF", \x"FE11", \x"FEE4", \x"FEF5", \x"FEF5", \x"FF14", \x"FF3C", \x"FF46", \x"FF53", \x"FF5D", \x"FF77", \x"FF7D", \x"FF99", \x"FF9D", \x"FFFB", \x"FFCC", \x"FFDC", \x"FFE9", \x"FFE5", \x"FFE7", \x"FFE9", \x"FFF3", \x"FFF4", \x"FFF5", \x"FFF6", \x"FFF7", \x"FFF9", \x"FFFA", \x"FFFC", \x"FFFD", \x"FFFE", \x"FFFF", \x"FFFF", \x"FFFF", \x"FFFF", \x"FFFF", \x"FFFF", \x"FFFF", \x"FFFF", \x"FFFF"; }\]

4.3.1.3 Weight Update Block

The block diagrams in Figures 4-12 and 4-13 show how the weight update is processed in the design for a 3-set and 6-set configuration respectively.

![Figure 4-11: Values of the sigmoid function.](image1)

![Figure 4-12: Block-diagram of the Weights Update module for a 3-set configuration.](image2)
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

Figure 4-13: Block-diagram of the Weights Update module for a 6-set configuration.

The first active input is selected. Then the corresponding weight is read from the RAM and directed to the weight update block via a demultiplexer. The counts of zeros and ones are performed and then the variance is calculated. The weight is updated in parallel and then stored in a single addressing mode RAM. The same process is applied to the rest of the active inputs. The signals bit lengths are indicated in section 4.4.

By comparing the hardware structures, presented in Figure 4-12 and Figure 4-13, it is possible to assess the complexity of a 6-set design against a 3-set one.

The advantage of using the proposed neural network is in the fact that only 3 virtual neurons are used in the hardware design by loading only three weights at a time out of the total number of weights 133.10^3 saved in the RAM, while conventional neural networks involve the total number of neurons such that they are all active at the same time.

The practical meaning of this is that the conventional design involving a large number of neurons has been replaced by a more effective RAM-based design occupying a smaller area on the chip.
4.3.1.4 Arithmetic Encoder Block

The output of the sigmoid function will assign a probability to the input bit $Y$, and both data are fed into the arithmetic encoder block. Once the bit probability is known, the individual bit needs, as well as its complement to one, to be assigned a range along a probability line, between 0 and 1.

The main difficulty of this operation lies in the fact that this calculation needs floating-point processing. Since the probability scale in the arithmetic encoder is determined by the difference between the high value and low value on the subdivided range it is possible to overcome the difficulty mentioned above by applying equivalent operations on a new range between two positive 32 bits integer counts 0 and $2^{32}-1$ instead of $[0,1]$.[Atek03a][Atek03b]. Figure 4-14 shows the arithmetic encoder block where the inputs are: the bit $Y$; the probability $P$, which is the output of the Activation Function block; and the Synchronizer output.

![Figure 4-14: The arithmetic encoder block.](image)

In Figure 4-15 a schematic process of the arithmetic encoder is shown.

![Figure 4-15: Example of the arithmetic encoding process.](image)
4.3.1.5 Synchronizer

The synchronizer is used to supervise the synchronised functioning of the neural processor blocks. It subdivides the total process into a cycle of operations. Below we give an example of 6 operations that are performed when the number of sets is equal to one.

1\textsuperscript{st} operation: The active inputs are selected.

2\textsuperscript{nd} operation: The weights and the counts of zeros and ones are read from the RAM.

3\textsuperscript{rd} operation: Computes the variance.

4\textsuperscript{th} operation: The weights are updated.

5\textsuperscript{th} operation: The updated weight is written into the memory as well as the new counts of zeros and ones. If the design is using more than one set, the sum of the weights are computed and the activation function calculates the probability $P$.

6\textsuperscript{th} operation: The arithmetic encoding is performed and the prediction error is then updated in the weight update block.

Figure 4-16 shows illustrates the different operations and show the number of clock cycles needed to perform each task.

![Figure 4-16: Illustration of the implementation order of each block of the neural processor.](image_url)
4.3.2 Execution Time

If the number of sets is greater than one, the rule to determine the number of cycles to process one bit is $3 \times n + 3$ where $n$ is the number of sets. Only the weight update process is affected by an increase in the number of sets and requires $3 \times n$ clock cycles, while the number of clock cycles required to process the arithmetic encoding, the activation function block and the active input selection does not change.

The number of clock cycles per bit affects the speed of our design. A further way to reduce the number of clock cycles is to use a multiple addressing mode memory instead of single addressing mode memory. For instance the use of double-addressing mode will drop the number of cycles for a 3-set configuration by three cycles. Additionally, the FPGA frequency has a major effect on the processing time. The processing time as function of the frequency is driven by the following formula:

$$time = \frac{\text{number of cycles/bit} \times \text{total number of bits}}{\text{frequency}}$$

In Figure 4-17 and Figure 4-18, we illustrate the processing times required to compress a 1024×1020 pixels image for a case of 1-set configuration and a case of 3-set configuration respectively.

![Figure 4-17: Illustration of the processing times as a function of the FPGA frequency for a 1-set configuration.](image-url)
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

4.4 Functional Simulation Results

The functionality of the neural processor design in Figure 4.9 was verified using the Aldec’s Active-HDL simulator. A testbench file was then created for each sub-block in order to apply several stimuli to assess the response of the VHDL design.
The waveforms in Figure 4-19 illustrate the Active Input block response to the testbench design.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Stimulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>clk</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>X[0]</td>
<td>0007FF</td>
<td></td>
</tr>
<tr>
<td>X[1]</td>
<td>0000FF</td>
<td></td>
</tr>
<tr>
<td>X[2]</td>
<td>0002FF</td>
<td></td>
</tr>
<tr>
<td>X[3]</td>
<td>0004FF</td>
<td></td>
</tr>
<tr>
<td>X[4]</td>
<td>0005FF</td>
<td></td>
</tr>
<tr>
<td>X[5]</td>
<td>0006FF</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-19: Waveform of the simulation of the Active Input block.

In the simulation, the vector $X$ represents the addresses of the weights than are selected in the RAM. For a 3-set configuration only $X(0), X(1)$ and $X(3)$ (32 bits signals) are fed into the weights update block. For a number of set equal to $n$, the weights addresses are represented by the signals $X(0)..<X(n-1)$. The value $y$ represents the input bit that is fed into the input selection block. In all the testbench simulations shown in this thesis the signal $clk$ correspond to the clock signal. The results of the simulations have been found to match the software results.

The waveforms in Figure 4-20 illustrate the waveforms of the Arithmetic Encoder block.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Stimulator</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLK</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>0000</td>
<td></td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>code</td>
<td>C9</td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-20: Waveform of the simulation of the Arithmetic Encoder Block.

The value $code$ shown in the simulation represents the output of the arithmetic encoding process. The 16-bit signal $P$ is the probability value derived from the activation function calculation. As mentioned in 4.3.1 the bit value $y$ is also an input of the arithmetic encoder block. The results of the hardware design of the arithmetic encoding have been found to match the results of the software simulations.
Figure 4-21 illustrates the waveforms obtained after simulating the Activation Function block.

![Figure 4-21: Waveform of the simulation of the Activation Function block.](image)

The output of the activation block is represented by the 16-bit signal \( \sigma \). The value \( x \) is the input of the function (3.7). The activation function performs its calculation after a single clock cycle as observed in Figure 4-21. The signal enable is an input signal that activates the AF block. These results have been found to match the results of the software simulation.

Figure 4-22 illustrates the waveforms obtained after simulating the Variance block, which is a sub-function in the Weight update block as seen in Figure 4-12.

![Figure 4-22: Waveform of the simulation of the Variance block.](image)

The 16-bit value \( \text{VAR} \) shown in the simulation represents the output of the Variance block. The 8-bit signals \( \text{C0} \) and \( \text{C1} \) are the counts of zeros and ones respectively. \( X \) is the output of the Active Input Selection block in the case of a 1-set configuration, which is only encoded with 12 bits.
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

The signal $VAR$ is used to update the weights as shown in the simulation of the Weight Update block in Figure 4-23.

![Waveform of the simulation of the Weight Update block.](image)

The 16-bit signal $W$ represents the previous value of the weight and $W_{UPDATE}$ the updated value. The signal $ERROR$ is the prediction error and it is encoded with 32 bits.

The results of the hardware design of the Weight Update block have been found to match the results of the software simulations.

In Figure 4-24, the simulation of the top-level design is shown. The signal $code1$ represents the compressed output in the VHDL program and the signal $code_{original}$ represents the compressed output in the software implementation.

![Simulation results of the top-level program of the NN-based technique.](image)
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

The different signals presented in the Figure 4-24 are explained in table 4-3.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Task</th>
<th>Bit Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable</td>
<td>Activates the compressor</td>
<td>1</td>
</tr>
<tr>
<td>Reset</td>
<td>Reset the compressor</td>
<td>1</td>
</tr>
<tr>
<td>clk</td>
<td>Clock signal</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>Input bit</td>
<td>1</td>
</tr>
<tr>
<td>P</td>
<td>Output of the activation function (software)</td>
<td>16</td>
</tr>
<tr>
<td>CHECK</td>
<td>Output of the activation function (hardware)</td>
<td>16</td>
</tr>
<tr>
<td>Code</td>
<td>Output of the NN compressor (hardware)</td>
<td>8</td>
</tr>
<tr>
<td>Code_original</td>
<td>Output of the NN compressor (software)</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4-3: Functionality of the design signals.

The design output is shown to match the results of the software.

The hardware design has been tested at the simulation level and was successfully compiled. However further tests would be required after the synthesis step (Figure 4-8) that can be carried out with the Synplcity software. The design is targeted at a high density FPGA. The XSV-800 board designed by XESS Corporation is among the resources available to implement the design. It uses a XCV800 Xilinx Virtex FPGA with 800,000 gates and two independent 512K x 16 bits SRAM banks. The XSV-800 board can be configured through a PC parallel port, serial port and from a 16 Mbit Flash RAM that stores the bit-stream mapping of the FPGA. In order to complete the implementation, it is essential to estimate the chip resources usage and to determine the closest estimate of the physical timing of the design using the back-annotation simulation.

4.5 Decision Support Scheme of Active Neural Sets

In section 4.2, results of the experiments carried out on the NN algorithm have shown that the number of sets could be reduced from 6 to 3, a 3-set NN configuration representing a good trade-off in terms of compression ratio, processing time and memory usage [Atek03a]. However a 2-set NN configuration brings a further increase in efficiency. For example, a NN processor requires weight update memory of 270 Kbytes with 2 sets and 532 Kbytes with 3 sets. The concept behind the adaptive scheme is to allow an individual NN processor to execute a 2-set prediction instead of a 3-set one provided the resultant loss in performance is tolerable. This is achieved by designing a decision support block.
Chapter 4: Hardware Implementation of the Neural Network Image Compressor

that performs the selection between using a 2-set or 3-set configuration. The selection criterion is defined as a tolerance value of 1%, measured by the following parameter:

$$r_{2,3} = \frac{R_3 - R_2}{R_3}$$

(4.4)

where $R_3$ is the compression ratio obtained with 3 sets and $R_2$ is the compression ratio obtained with 2 sets.

Experiments on satellite images have shown that compressing panchromatic images with 2 input sets instead of 3 does not affect the performance significantly (Table 4-4). This is due to the fact that panchromatic images are not spectrally filtered, they contain a lot of fluctuations in pixels intensity, and therefore the prediction can be very efficient when using the two previous pixels as an input.

<table>
<thead>
<tr>
<th></th>
<th>Ratio (Nset=2)</th>
<th>Ratio (Nset=3)</th>
<th>$r_{2,3}$ (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pan1</td>
<td>1.604</td>
<td>1.610</td>
<td>0.39 %</td>
</tr>
<tr>
<td>Pan2</td>
<td>1.610</td>
<td>1.613</td>
<td>0.19 %</td>
</tr>
<tr>
<td>Pan3</td>
<td>1.570</td>
<td>1.573</td>
<td>0.21 %</td>
</tr>
<tr>
<td>Pan4</td>
<td>1.574</td>
<td>1.578</td>
<td>0.24 %</td>
</tr>
<tr>
<td>Pan5</td>
<td>1.622</td>
<td>1.627</td>
<td>0.28 %</td>
</tr>
<tr>
<td>Pan6</td>
<td>1.6639</td>
<td>1.670</td>
<td>0.37 %</td>
</tr>
<tr>
<td>Pan7</td>
<td>1.804</td>
<td>1.816</td>
<td>0.64 %</td>
</tr>
<tr>
<td>Pan8</td>
<td>1.699</td>
<td>1.704</td>
<td>0.29 %</td>
</tr>
</tbody>
</table>

Table 4-4: Compression ratios of panchromatic images using 2 different neural configurations.

Multispectral images are less difficult to compress, since they represent the spectral response of a scene on a single band, the resultant image is smoother and pixels are more dependent on the neighbouring pixels of order 2 and 3. Investigation of image compression ratios with respect to a 1% value of $r_{2,3}$ shows the following. Panchromatic images contain more details which make the pixels more sensitive to their two previous pixels than three; their compression ratios $R_3$ and $R_2$ are getting closer and thus the value of $r_{2,3}$ is consistently below 1%. When the compression ratios $R_3$ of multispectral images are low ($R_3 < 1.79$), $r_{2,3}$ goes below 1% too.
Figure 4-25 shows the relationship between \( R_3 \) and \( r_{2,3} \).

![Graph showing the relationship between \( R_3 \) and \( r_{2,3} \).](image)

**Figure 4-25: Values of \( r_{2,3} \) for different compression ratios \( R_3 \) of UoSat-12 multispectral images.**

From the graph we can deduce that there is a close relationship between the compressibility of the image and the relative compression ratio \( r_{2,3} \). The value \( R_3 < 1.79 \) is carefully chosen so that all the values of \( r_{2,3} \) are less than 1% without exception.

To measure the value of \( R_3 \), we have defined a new parameter \( R_{H_0} \), which is based on the zero-order entropy \( H_0 \) [Sayood00]. The expression for calculation of \( R_{H_0} \) is as follows:

\[
R_{H_0} = \frac{\log_{2} (\text{card} X)}{\sum_{x \in X} p(x) \log_{2} \frac{1}{p(x)}}
\]

where \( p(x) \) is the probability of the 8-bit coded pixel that has a gray level intensity \( x \) and \( X = \{0, 1, 2, ..., 255\} \) is the alphabet of gray levels \( x \).
The graph in Figure 4-26 shows the relationship between $R_{H0}$ and $R_3$ across a test set of 23 satellite images.

![Figure 4-26: Compression ratios $R_3$ and $R_{H0}$ of satellite multispectral images.](image)

It can be seen that the two curves have very similar features and shape.

Figure 4-27 illustrates the plot of $R_3$ as a function of $R_{H0}$. The relationship between $R_3$ and $R_{H0}$ is a quasi-linear relationship.

![Figure 4-27: Compression ratios $R_3(R_{H0})$ of satellite multispectral images.](image)

From figure 4-25 and 4-27 we deduce the threshold value for a tolerance of 1% that is $R_{H0}=1.225$. For a tolerance of 1.5% the threshold value will be $R_{H0}=1.28$.

We have tried our scheme on the satellite images available from our test set in section 3.3. Each image has a size of 1024x1020 pixels, which on average gave 12 blocks of sub-images of 256x256 and 4 blocks of 256x252.
Table 4-5 shows the number of blocks that were compressed using a 2-set configuration over the total number of blocks. The other blocks were compressed using 3-sets.

<table>
<thead>
<tr>
<th>Image</th>
<th>$R_{\text{ho}} &lt; 1.225$</th>
<th>$R_{\text{ho}} &lt; 1.28$</th>
<th>Image</th>
<th>$R_{\text{ho}} &lt; 1.225$</th>
<th>$R_{\text{ho}} &lt; 1.28$</th>
<th>Image</th>
<th>$R_{\text{ho}} &lt; 1.225$</th>
<th>$R_{\text{ho}} &lt; 1.28$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi11</td>
<td>15/16</td>
<td>15/16</td>
<td>Multi33</td>
<td>0/16</td>
<td>0/16</td>
<td>Multi61</td>
<td>0/16</td>
<td>13/16</td>
</tr>
<tr>
<td>Multi12</td>
<td>6/16</td>
<td>11/16</td>
<td>Multi34</td>
<td>0/16</td>
<td>3/16</td>
<td>Multi62</td>
<td>1/16</td>
<td>7/16</td>
</tr>
<tr>
<td>Multi13</td>
<td>0/16</td>
<td>3/16</td>
<td>Multi41</td>
<td>0/16</td>
<td>3/16</td>
<td>Multi63</td>
<td>3/16</td>
<td>14/16</td>
</tr>
<tr>
<td>Multi14</td>
<td>1/16</td>
<td>4/16</td>
<td>Multi42</td>
<td>0/16</td>
<td>0/16</td>
<td>Multi64</td>
<td>3/16</td>
<td>9/16</td>
</tr>
<tr>
<td>Multi21</td>
<td>7/16</td>
<td>9/16</td>
<td>Multi43</td>
<td>0/16</td>
<td>0/16</td>
<td>Multi71</td>
<td>1/16</td>
<td>7/16</td>
</tr>
<tr>
<td>Multi22</td>
<td>1/16</td>
<td>2/16</td>
<td>Multi44</td>
<td>0/16</td>
<td>0/16</td>
<td>Multi72</td>
<td>0/16</td>
<td>13/16</td>
</tr>
<tr>
<td>Multi23</td>
<td>0/16</td>
<td>0/16</td>
<td>Multi51</td>
<td>10/16</td>
<td>12/16</td>
<td>Multi73</td>
<td>3/16</td>
<td>14/16</td>
</tr>
<tr>
<td>Multi24</td>
<td>0/16</td>
<td>0/16</td>
<td>Multi52</td>
<td>0/16</td>
<td>12/16</td>
<td>Multi74</td>
<td>3/16</td>
<td>9/16</td>
</tr>
<tr>
<td>Multi31</td>
<td>14/16</td>
<td>16/16</td>
<td>Multi53</td>
<td>14/16</td>
<td>16/16</td>
<td>Total</td>
<td>90/448</td>
<td>217/448</td>
</tr>
<tr>
<td>Multi32</td>
<td>3/16</td>
<td>9/16</td>
<td>Multi54</td>
<td>15/16</td>
<td>16/16</td>
<td>RAM saving</td>
<td>10%</td>
<td>24%</td>
</tr>
</tbody>
</table>

Table 4-5: Number of image tiles compressed with a 2-set configuration Adaptive Neural Processor

From table 4-4, we can have a clear idea of the amount of RAM that is saved using the adaptive scheme. A tolerance of 1% gives a performance of about 10% of the total RAM utilized. A tolerance of 1.5% allows a significant increase of RAM saving which is 24% of the RAM that could be used if a 3-set configuration was applied on all image blocks. We can see that if four neural processors were used, the overall performance of the new scheme combined with an adaptive selection mode will allow us to use only the same amount of memory storage that three neural processors with 3-set configuration can use.

To extrapolate this concept on our parallel neural compressor hardware, we can equip our system with an adaptive set selection subsystem that will vary the number of sets between 2 and 3 depending on the value of $R_{\text{ho}}$ calculated on the input image tile.
Figure 4-28 shows a schematic representation of the new model.

![Schematic of the neural network image compressor](image)

Figure 4-28: A schematic representation of the decision support scheme.

The image is fed into a zero-order entropy calculator. The output of the latter is compared with the threshold value that is set to 1% or 1.5%, which is subjectively determined by the operator. The result of the comparison will control a switch box that can be a simple multiplexer. The data will then go to a 2-set neural network processor if the $R_{HO}$ is less than the threshold value or to a 3-set neural network processor otherwise. The data is then compressed and a header stating the neural network configuration is inserted to the bit stream for decompression.

### 4.6 SSONICS: Small Satellite Onboard Neural-Based Image Compression System.

State-of-the-art image compression schemes developed for still image compression do not offer, in general, a real suitability to real time compression and parallel operations as they use the entire image or a significant segment to generate the compressed bit-stream. In addition to that, they use a high amount of memory and on-board computing resources to perform their task as they involve very complex mathematical transformations and require every pixel of the image to be processed within the scheme. The operation principles of the neural network-based compression make it possible to encode images in the scanning-line mode that makes the technique applicable to the real time image on board compression. This section describes a conceptual design of an on-board compression subsystem targeted at on-board small spacecrafts.
4.6.1 Overview of SSONICS

Small Satellite Onboard Neural-based Image Compression System (SSONICS) achieves its mission by exploiting the parallelism among image processing units and assigning computationally intensive tasks to dedicated hardware.

The FPGA that performs the processing in SSONICS comprises two major components of the compression system. The first block carries out the image splitting and the preprocessing using whether a Peano-Hilbert scan or an Integer Wavelet Transform and the second block where the neural network processors are implemented, uses a neural network prediction assisted with a the decision support scheme, and arithmetic entropy coding. The image data is received through a dedicated high-speed data link from the imager. As the image frame is split and preprocessed using a Peano-Hilbert scan or an integer wavelet transform, the output is continuously transferred to the external data memory.

The command and control flow into SSONICS is through a Controller Area Network (CAN). SSONICS is connected to a CAN bus up to two CAN buses, running in redundancy. The CAN interface is supervised by a dedicated micro-controller with an embedded CAN controller, which is connected to the Serial Port Interface of the FPGA.

The block diagram of SSONICS is given in Figure 4-29.
The data transfer between the Camera, SSONICS, the on-board memory and the RF communication subsystem can be performed using a CAN-bus physical layer with a rate up to 1 Mbps (Figure 4-30). In addition to that, a redundant pathway for all functions normally carried out by the CAN node can be used. The camera micro-controller has an interface to high-speed serial link (up to 10Mbps).

![Diagram of SSONICS within the on-board subsystems.](image)

**Figure 4-30: SSONICS within the on-board subsystems.**

### 4.6.2 Tasks of SSONICS

The main tasks of SSONICS are to provide a dedicated computing support for the on-board data handling, relieving the main On-board Computer (OBC) and managing all aspects of imaging pre-processing and compression.

The image compression unit performs the following functions:

*Image Retrieval* Images are downloaded from the cameras by issuing commands to the camera controller. Each image is downloaded line by line. This allows image cropping and removal of interlacing to be carried out by this stage. However, for the faster downlink the whole image can be requested in one go.
Image Compression/Decompression and Thumb Creation} Any further image processing can be carried out by the embedded Field Programmable Array. This actually offers an operational platform for testing any experimental image processing techniques to be implemented. Thumb images can be created using the integer wavelet transforms that will allow the user to have a preview of the captured images before complete downlink. If an additional processing of the data is required the data can be retrieved from the on-board memory and decompressed. The decompressed image or area of the image is then fed into a post-processing unit that carries out the desired processing, such as super-resolution or watermarking (Figure 4-30). The processed image is then recompressed and stored again in the on-board memory. One of the main advantages of the neural network-based compressor scheme is its ability to perform decompression without completely changing the hardware architecture of the compressor. Only the re-configuration of the program regarding the arithmetic encoding needs to be changed to allow decompression.

Sending Out Images There is a direct interface of the SSONICS system with the RF subsystem, as well as CAN node can be used to transfer data throughout the whole network. If a CAN transfer is aborted, this information is passed back to the direct interface.

4.7 Conclusions

This chapter describes the design of a new fast neural-network based lossless image compressor for use on-board small satellites. Ways of speeding up the processing time of the neural network algorithm have been investigated. It has been shown that the number of neural network input sets and the size of the image tiles affect the processing speed of the neural network predictor with three sets being a reasonable trade-off between complexity and compression ratio. A hardware implementation targeted at FPGA platform has been proposed which is based on multiple neural processors performing the compression of a certain number of image tiles in parallel. A VHDL model of the coder has been developed and simulated. Only 12 cycles per bits are required to process one bit using a 3-set configuration. The processing time simulations have shown that the design can go 6 times faster than the software implementation.

A decision support scheme has been conceptually introduced to the proposed hardware configuration. This makes the compression scheme adaptive to the image entropy.
minimizing in this way the hardware resources. Testing results show that this scheme allows the compressor to reduce the on-board RAM usage by at least 24%.

This chapter also presents a description of a hardware system referred to as Small Satellite Onboard Neural-based Image Compression System (SSONICS), which employs the proposed algorithm in chapter 4. The new hardware system is realising lossless image compression in an independent decentralized way without using the resources of the on-board computer fully. The system can easily be extended to comprise additional image-processing techniques allowing decompression and recompression without changing the internal structure of the hardware completely. This novel system can facilitate a faster way of data transmission to ground terminals and therefore can offset the data link problem.
Chapter 5

5 Conclusions and Further Work

This thesis presented a research study for a novel implementation of data compression on-board small satellite. This research is motivated by the fact that small Earth observation satellites are supporting more and more size-hungry applications while the downlink systems have limited transmission bandwidth and power budget. The solution to that problem is to use a compression scheme to reduce the amount of data stored on-board and transmitted to the ground.

The goal of the research is to achieve high compression ratios with low complexity algorithms in order to reduce the requirements for on-board memory, station contact time and data archival volume.

A literature review has been undertaken in order investigate the state-of-the-art in lossless image compression. In this thesis, lossless image compression has been classified into three distinct approaches: Entropy coding methods, Predictive methods and Transform based methods.

The major contribution of the work presented in this thesis is the design of a low complexity neural network-based lossless image compression that has been shown to achieve high compression ratios outperforming state-of-the-art methods. The main novelties of the research work are summarized in the following:

A novel lossless compression algorithm for satellite images based on a neural network scheme and Peano-Hilbert scan:

To bring a novel solution to the data link problem, an approach based on Neural Networks prediction has been studied and tested. It was found out that the average ratio over ten images was about 2.66, which is better than the CALIC algorithm. The NN method was improved by introducing a Peano Hilbert scan. The main advantage of Peano-Hilbert scan is that the entropy of the scanned pixel sequence converges asymptotically to the two-dimensional entropy of the image. The results pointed out that the ratio was about 10.5% better than the NN compression ratio with a raster scan. The
Chapter 5: Conclusions and Further Work

A novel lossless compression algorithm for satellite images based on a neural network scheme and IWT:

Since multi-resolution transmission is advantageous for satellite-compressed data, another method based on wavelets was introduced. These wavelets construct integer output for integer inputs, preventing data losses from floating point operations and non-orthogonal filters distortions.

The experiments that were carried out with a (2,2) wavelet filter (with 2 vanishing moments in the analysis and synthesis high pass filters respectively) shows that the (2,2) wavelet based method outperformed the Rice algorithm achieving an average ratio of 1.632 when it was 1.438 for the Rice algorithm over 12 satellite images.

Optimisation study of the neural network lossless compression algorithm in order to lower its complexity:

The next step of our research consisted of improving the neural network based algorithm in order to implement it using a hardware solution. Ways of speeding up the processing time of the neural network algorithm have been investigated. It has been found that a 3-set configuration represents the best trade-off in terms of performance and complexity.

A new hardware lossless compressor design of the proposed algorithm based on parallel processing:

A hardware implementation targeted at FPGA platform has been studied and simulated using a VHDL model. The concept of our proposed neural compressor is based on multiple neural processors performing the compression of a certain number of image tiles in parallel. The tile size has been set to suitably improve the processing time required to compress each tile separately with a 256x256 pixels if the number of neural processors is more than four and 512x512 pixels is the number of neural processors is less or equal than four.

The proposed compression technique described in this thesis offers a variety of advantages for satellite applications. Real-time compression is achieved, which allows a very fluid flow of the data from the camera subsystem to the on-board memory devices. It
also permits the saving of the data in a compressed format directly in the memory. The novel method offers built-in redundancy of the neural processors. Accordingly, in case of failure, all the processing power can be redirected to the remaining healthy neural processors. The independent compression of tiles realises a natural isolation of blocks of data. As a consequence, errors are stopped from spreading to the whole image when they occur within a sub-image.

**A novel hardware design of the neural network processor:**

The proposed neural network uses, in fact, only 3 virtual neurons in the hardware design by loading only three weights at a time out of the total number of weights saved in the RAM, while conventional neural networks involve the total number of neurons such that they are all active at the same time.

The advantage of this design compared to conventional design of neural network-based algorithms is that the conventional design involving a large number of neurons has been replaced by a more effective RAM-based design occupying a smaller area on the chip.

**A decision support mechanism for the selection of the neural sets:**

Further research was conducted to improve the proposed neural network scheme for the hardware implementation. Hence, a decision support scheme has been conceptually introduced to the proposed method. The main functionality of the decision support scheme is to make the compression process adaptive to the image zero-order entropy. The objective of the novel scheme is to reduce the RAM usage of the neural network compressor. Testing results show that this scheme allows the compressor to reduce the on-board RAM usage by at least 24%.

**A new hardware compression system based on our novel algorithm:**

To incorporate the novel compression technique in a satellite image system, a conceptual hardware system (SSONICS) was proposed. The idea behind such new system is to allow lossless image compression to be more readily realised in an independent decentralized way without fully using the resources of the on-board computer. The systems aims also to make the compressed data transparent to additional embarked image-processing techniques allowing decompression and recompression without fully changing the internal structure of the hardware. Basically this system can offer a faster transmission way of data to the ground terminals and therefore offset the data link problem.
5.1 Future Work

The main area of future work can be suggested as follows:

- Incorporate the idea of NN-based image compression into multispectral image compression. In order to take into account the spectral correlation many decorrelation procedures can be used. The idea of developing a 3-D integer wavelet transform for the NN-based algorithm can be suggested. In addition, research can be directed towards using predictive schemes to remove the spectral redundancies among the different mono-band images.

- Investigate a good combination between the NN-based compressor and an error protection algorithm in order to find a good trade-off between protection and compression ratio.

- Combine the IWTNN with a watermarking scheme that can insert the watermark in the frequency domain using IWTs.

- Find optimal IWT filters that best suit for satellite images.

- Investigate the optimal neighbourhood window for satellite image prediction.
References


January 23, 2004


References


References


References


References


References


M. Skrbek, ”Fast Neural Network Implementation”. In: Neural Network World. 1999, vol. 9, no. 5, p. 375-391. ISSN 1210-0552.


Space Daily Web Page:
http://www.spacedaily.com/news/marsexpress-00g.html
References


References


Appendix A

Implementation of Image Compression on-board the Prometheus Micro-satellite

A.1 Introduction

The work described in this section is part of the effort of a group of postgraduate students, at the Surrey Space Centre in response to a call for the entry to an international student competition organised by the Japan Space Forum. The Surrey team pursued this challenge into forming a small team of students with different areas of specialisation ranging from developing advanced and highly capable propulsion systems for small satellites to image compression, orbit determination and satellite formation flying. The vision of the team is to design a truly useful and challenging mission that can potentially materialise with the support of commercial and governmental institutes. The project was not carried out due to lack of funding but the concept might be realized in the near future.

A.2 Mission Description

The Prometheus mission is intended to perform a close flyby of the Near Earth Object (NEO) 4179 Toutatis and provide limited scientific data during the short duration of the encounter [Kennedy02]. Its key constraints include limitations on mass (< 50 kg), volume (50 cm x 50 cm x 50 cm stowed), and the use of Japan’s National Space Development Agency (NASDA) H2-A launch vehicle. Many series of manoeuvres would then be performed to achieve earth escape. Following escape, Prometheus will perform one or more midcourse corrections prior to entering the encounter phase.

At encounter, which is likely to be of limited duration (several minutes or less), the microsatellite and the NEO will pass each other at substantial relative velocity—as high
as 10-12 km/s. Cameras and other instrumentation will have to be pointed and slewed with precision in order to record mission data for later downloading. The final phase of the mission will involve up to two months of data recovery operations, at very low data rates, from distances of up to 13 million kilometres. Figure A-1 represents a radar image of 4179 Toutatis taken in 1992 and figure A-2 shows an optical image of Gaspra asteroid [Kennedy02].

Figure A-1: Radar image of 4179 Toutatis, taken in 1992. Figure A-2: Optical image of Gaspra, 54 m/pixel.

Figure A-3 represents the design overview of the Prometheus satellite.

Figure A-3: Prometheus satellite overview [Kennedy02].
A.3 Payload Design

Selection of specific payloads is constrained by mass and power limitations discussed in the last section. Options included high-resolution optical imaging cameras, miniaturized radar systems, and infrared or X-ray imagers/spectrometers.

Optical imagery is indisputably essential; currently available Toutatis imagery is limited to radar data at relatively low resolution. Providing 10-metre or better optical data would greatly increase our knowledge of this particular object, providing a better understanding of its complicated dynamics and structure. The selected imaging suite includes two f/1.3 multispectral cameras. The WFOV camera, with a field of view of 20° x 20°, consists of an apochromatic lens assembly and 512 x 512 three-colour detector at the focal plane. The WFOV camera’s aperture is 45 mm in diameter, and should permit target acquisition as early as seven hours prior to encounter. The NFOV camera has a somewhat more restricted field of view at 4° x 4°, with a nearly identical (but larger) lens assembly and higher-resolution detector array (1,024 x 1,024). Both detectors are amorphous silicon with three colour recognition, without requiring filters or mosaic layouts. The NFOV camera, with its 120 mm main lens, is intended to capture large surface features on the NEO during flyby. At an estimated close approach of 100 kilometres, the NFOV camera should be capable of providing 6.8-metre resolution, with Toutatis briefly filling 80% or more of the camera’s field of view. The selected NIR assembly consists of a front end lens assembly identical in design to that of the NFOV visual camera—although the lens material is necessarily different. The NIR camera will use zinc selenide or sapphire (Al₂O₃) apochromats; either option provides high transmissivity (80-90%) in the spectrum of interest. Combined with a 256-element linear PbS array, the NIR system will provide a resolution (along the flight path of the NEO) of roughly 30 metres at encounter. During encounter, the imaging system will be capable of acquiring approximately 40 seconds of high-resolution visual imagery at 30 frames per second, and perhaps one IR image per second. This equates to a significant storage requirement: 1,200 frames of 16-bit visual imagery at 1 megapixel resolutions, in addition to 40 or more IR line images. Without image compression, this level of data creation would require almost 2.5 gigabytes of storage capacity. A more manageable approach might attempt five or less NFOV frames per second, for a maximum set of 200 images. The satellite will slew at up to 6°/sec in order to maximize imaging time; all three cameras are fixed to the payload shelf, aligned with the satellite’s –Y face (opposite the single fixed solar array).
A.4 On Board Computer (OBC) and Data Handling System

A decentralized computer design could be proposed to allow the different subsystems to communicate and accomplish their tasks without a central decision-making device. This implies the use of subsystem-specific circuitry and electronic components such as microcontrollers and memory. In term of cost, weight and power consumption, this solution appears to offer less utility than a centralized system, although it is easier to implement. In Prometheus’ case, a centralized OBC will provide the computing power necessary to perform the control of payloads and subsystems and schedule various tasks. It will regulate attitude, manage RF communications links with the ground station(s), receive numerous analogue measurements from onboard sensors and instruments, compute them, and perform data compression as required. The Prometheus OBC is inspired by SSTL’s SNAP-1 spacecraft OBC. This device is a 32-bit, highly integrated RISC processor, the StrongArm SA-1100 (which retails for roughly US$45,000). The CPU core is an SA-1 equipped with an 8 KB data cache and 16 KB of instruction memory.

![OBC System Overview](Figure B-4: OBC System Overview [Kennedy02].)
A.5 Image Handling and Compression

Technical limitations of the Prometheus communications system create a bottleneck, preventing transmission of large amounts of raw data to the ground station before end of mission. The use of onboard compression schemes minimizes the total amount of data required to be downlinked. The team examined various approaches for lossless image compression using predictive neural networks and Integer Wavelet Transforms (IWTs). A block-diagram of the proposed method is shown in Figure 4-8.

First, image data is fed into a (2,2) IWT decorrelator. The mapper converts each IWT coefficient into a nonnegative integer value. The predictor, a two-layer, $4 \times 10^6$ by 1 neural network, scans the stream of the mapper output characters, and allocates a probability distribution for the next incoming character. An arithmetic encoder, starting from an initial interval $(0,1)$, subdivides this range repetitively until obtaining a binary fraction within the final subdivided range. This binary fraction is the shortest number capable of representing the data; it is known to be optimal within one bit of the data entropy. After the prediction step an arithmetic encoder begins with a range $(0,1)$, and divides the range into two sub-ranges for each input bit. The results of the method are detailed in section 4.3.1.1.

A.6 RF Communications

Radio communications with the Prometheus microsatellite will take place at distances of as little as 35,000 km (following deployment at GEO) to true “deep space” separations—up to 13.5 million kilometres. At encounter, the spacecraft (and Toutatis) will be located approximately 3 million kilometres from earth. However, within two weeks, this distance will have increased to slightly over 5 million kilometres. One month after encounter, the spacecraft will be nearly 8 million kilometres away. Insomuch as signal strength is inversely proportional to the square of this separation distance, communications link closure is highly contingent on this value. Thus, only 30 days after encounter, the spacecraft’s received power will decline by nearly 85%. Based on the requirements, the Prometheus mission must be capable of (1) receiving and translating commands, and (2) transmitting no less than 42 megabytes of data to an earthside station before retreating out of range. For purposes of this analysis, a single 3-metre uplink/downlink antenna located at the Jet Propulsion Laboratory’s Deep Space Network (DSN) site at Canberra, New
South Wales, Australia, is assumed. The team initially assumed a standard spacecraft transmitter arrangement, including a Surrey 150 mW transmitter with a 4 W second stage power amplifier, consuming 45 W while transmitting. The substitution of an experimental Class D/E amplifier (4 W RF output) for the standard amplifier represents a high-efficiency (~50%), lightweight approach, albeit at somewhat higher technical risk. In this alternate approach, which was baselined, the transmitter draws only 13 W. Maximum antenna size is fixed by spacecraft volume constraints at 50 cm x 50 cm. The selected antenna will utilize an entire spacecraft face. The additional collection area provides a 1 dB improvement in gain over a more conventional 50 cm diameter dish antenna. Given the combination of vast distances, low power, and small antenna size available to the team, only advanced modulation and coding techniques provide acceptable link margin. The space loss associated with the maximum separation distance (13.5 million kilometres on 2 December 2004) is −242 dB. Deep-space probes such as Pioneer, Voyager, and Cassini overcome these losses through reliance on complex, often bandwidth-inefficient schemes such as Binary Phase Shift Keying (BPSK) modulation with rate-$\frac{1}{2}$ Viterbi encoding—baselined for the Prometheus mission. At a bit error rate of $10^{-5}$, the required $E_b/N_0$ (energy per bit / noise spectral density) is only 4.2 dB. This compares favourably with simpler modulation schemes, such as uncoded BPSK (9.6 dB). The uplink configuration must ensure that ground-generated commands are received at the spacecraft, even at maximum separation and in contingency (loss of attitude control) situations. At encounter, a 100-W S-band transmitter at the primary ground station provides sufficient signal power to produce as much as 11 dB of margin into 5.9 x 5.9 cm enhanced microstrip antennas (located on the high gain antenna subreflector and on the nadir truss structure near the main engine nozzle). At maximum separation, this margin falls to −1 dB. This requires a lower data rate than the downlink (10 bps). Normal operations will utilize the high gain antenna for command uplink.
Appendix B

Implementation of Lossless Image Compression for Space Applications

This appendix presents a review of the literature on implementation of lossless image compression schemes in hardware and software for space applications. The purpose of this appendix is to select an efficient approach to implement image compression on-board a small satellite taking into account processing speed and suitability for a real-time implementation.

Three types of implementations are investigated. The first type uses digital signal processors (DSPs), the second is based on Field Programmable Gate Arrays (FPGA) and finally a hybrid implementation is presented where a combination of FPGA and DSP devices is shown.

The Digital Signal Processor implementation is discussed in section B.1. Section B.2 covers the FPGA-based type. The combined FPGA and DSP lossless image compression case is presented in section B.3. Section B.4 draws the conclusions.

B.1 Digital Signal Processor Implementation

A Digital Signal Processor (DSP) is a fast and powerful special purpose microprocessor that can process data in real time. This real-time capability makes a DSP perfect for applications that cannot tolerate any delays. In this section we present one example of lossless image compression, the Rice Algorithm, which is implemented for space applications.
Appendix B: Implementation of Lossless Image Compression for Space Application.

B.1.1 DSP Implementation of the Rice Algorithm

CRIL Technology has studied for the Centre National d'Etudes Spatiales (CNES) the implementation and computational performances of the RICE Algorithm on an On-board DSP card MOSAIC020. The MOSAIC020 contains the Texas instrument TSC21020 core and it has been designed by Dornier [Duffez01].

In this implementation the compression of a test image of 256x256 pixels has been carried out using an average of 152 cycles to process each pixel.

In order to obtain better performances, some considerations have been taken into account to optimize the C code of the Rice algorithm [Duffez01]:

- The 8-bits per pixel image were stored with 4 pixels per 32-bit word instead of 1 pixel per word.
- The bitstream was stored in 32 bits variables instead of 8 bits.
- Loops and tables pointers have been optimized.
- Some functions were gathered together.
- The best choice of local and global variables were taken.

The optimization led to a compression algorithm that requires only 87 cycles/pixel on average, which represents 42% of improvement. The instruction cycle rate reached 20 MHz with a data speed achieving 230 Kpixels/s. In this application, it has been found that the compression ratios were quite low (around 1.5:1), which is understandable since the Rice algorithm has proven to yield low compression ratios (section 2.10.3) but still important if we consider the sensitivity and importance of information within satellite images in remote sensing applications. Nevertheless, it is a line-based algorithm, which is well suited to real time on-board implementation since it compresses per small size of blocks (16 pixels) and as a consequence the read and write buffers do not require a lot of memory.

B.2 Field Programmable Gate Array Implementation

Field Programmable gate Arrays (FPGAs) represent a technical breakthrough in the VLSI industry. FPGAs can offer the benefits of both Programmable Logic devices (PLD) and Masked Programmed Gate Array (MPGA). FPGAs are an array of programmable logic cells, I/O cells and memory cells interconnected by a matrix of wires and programmable switches. Each logic cell performs a simple logic function defined by a user's program.
An FPGA has a large number (64 to over 20,000) of these cells available to use as building blocks in complex digital circuits [Heuser00]. Recent increases in FPGA performance and size offer a new hardware acceleration opportunity, while few years ago FPGA chips were fairly small [Hauser00].

**B.2.1 FPGA LOCO-I**

This work has been carried out by Matthew Klimesh, Valerie Stanton, and David Watola of Caltech for NASA's Jet Propulsion Laboratory.

The algorithm used in this implementation is the LOCO-I (low complexity lossless compression for images) [Klimesh02]. To reduce the complexity of the implementation on the FPGA, the LOCO-I algorithm has been modified and is now referred as FPGA LOCO. This one represents a first step towards producing a space-qualified hardware version of the LOCO-I algorithm.

The FPGA LOCO system is implemented using an APS-V240 board with a Xilinx Virtex XC\textsuperscript{5}0 FPGA. The APS-V240 uses a PC104 interface. It is connected to the PC's Industrial Standards Architecture (ISA) interface with an adaptor card. Two optional 256Kx18 zero bus turnaround (ZBT) static random access memories (SRAM) are installed on the APS-V240. One of them is installed on a daughter card. The SRAM is used to store pixel values and context data. The transmission of pixels from software to the FPGA, and bytes from the FPGA to the PC is done one byte at a time. The total SRAM needed by the algorithm is 1Kx8 for the pixel memory and 1Kx32 for the context memory.

The APS-V240 board is shown in figure B-1.

![Figure B-1: The FPGA LOCO board][1]

[1]: Figure B-1: The FPGA LOCO board [Klimesh02].
Appendix B: Implementation of Lossless Image Compression for Space Application.

Figure B-2 shows the block diagram of the FPGA LOCO device.

![FPGA LOCO block diagram](image)

Figure B-2: The FPGA LOCO block diagram [Klimesh02].

In table B-1, the hardware performance characteristics are listed.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Speed</td>
<td>12 MHz</td>
<td>1.33 M pixels/s</td>
<td>260 K pixels/s</td>
</tr>
<tr>
<td>Core Speed</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compression Speed</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table B-1: The FPGA LOCO performance characteristics [Klimesh02].

Table B-2 shows a benchmark of the compression ratios of the FPGA LOCO against JPEG-LS and the Rice algorithm [Klimesh02].

<table>
<thead>
<tr>
<th>Image</th>
<th>FPGA LOCO</th>
<th>JPEG-LS</th>
<th>RICE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lunar</td>
<td>1.786</td>
<td>1.839</td>
<td>1.471</td>
</tr>
<tr>
<td>Mars (Pathfinder)</td>
<td>1.735</td>
<td>1.705</td>
<td>1.399</td>
</tr>
<tr>
<td>Mars (Viking Orbiter)</td>
<td>2.605</td>
<td>2.712</td>
<td>2.179</td>
</tr>
<tr>
<td>Mars (Viking Lander)</td>
<td>1.670</td>
<td>1.681</td>
<td>1.504</td>
</tr>
<tr>
<td>Venus (Magellan radar)</td>
<td>1.923</td>
<td>1.918</td>
<td>1.713</td>
</tr>
<tr>
<td>Europa (Galileo)</td>
<td>1.478</td>
<td>1.471</td>
<td>1.234</td>
</tr>
<tr>
<td>Average</td>
<td>1.810</td>
<td>1.822</td>
<td>1.533</td>
</tr>
</tbody>
</table>

Table B-2: Compression ratios of the LOCO-I hardware implementation.
Appendix B: Implementation of Lossless Image Compression for Space Application.

B.3 Hybrid-Based Compression Implementations

B.3.1 PIRANHA Systematic-DSP

THALES Communications brought an innovative solution to satellite on-board hardware image compression problems by introducing "systematic-DSP" [Kajfasz01]. Systematic-DSP can be associated with usual DSPs (like TSC21020), RISC processors or ASICs. In [Kajfasz01] the on-board architecture was structured so that the algorithm’s processes are split between the different types of processing units, DSPs, FPGA and LEON.

The compression algorithm that was implemented is JPEG2000. The algorithm is split as follow:

- DSPs: Wavelets transform, scalar quantization and rate control.
- FPGA: Block based entropy coding.
- Processing Unit: Program Management, syntax and rate control.

The complexity of the implementation of JPEG2000 has been estimated. The on-board real time image compression requires an input rate of 20 Mpixels/sec with a power consumption of less than 1 W/Mpixels. It uses three levels of decompositions and scan based processing.

Table B-3 details the global distribution of the different processes of the algorithm [Kajfasz01].

<table>
<thead>
<tr>
<th>Process</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>DWT 9/7 filter</td>
<td>32 MAC/pixel</td>
</tr>
<tr>
<td>Scalar Quantization</td>
<td>3 MAC/pixel</td>
</tr>
<tr>
<td>Entropy Coding</td>
<td>4500 FPGA gates/block</td>
</tr>
<tr>
<td>Total</td>
<td>35 MAC/pixel</td>
</tr>
<tr>
<td></td>
<td>100MMACS/PIRANHA</td>
</tr>
<tr>
<td></td>
<td>2.8 MPixels/PIRANHA</td>
</tr>
</tbody>
</table>

Table B-3: Algorithmic complexity of JPEG2000 using the PIRANHA compression card.
Figure B-3 shows a block diagram of the PIRANHA system.

Figure B-3: Architecture of the PIRANHA compression card [Kajfasz01].

B.3.2 Real Time Image Processing Subsystem: GEZGIN

GEZGIN is a payload that is hosted on the microsatellite BILSAT-1 (Figure B-4), which is a Turkish satellite manufactured by SSTL [Ismailoglu02]. One of the missions of GEZGIN is to achieve real-time onboard image compression that allows an efficient use of the downlink system and the on-board memory storage.

Figure B-4: Bilsat-1 enhanced microsatellite.

The algorithm used in the mission is JPEG2000. The JPEG2000 algorithm contains many iterative blocks and parallel structures. Results of simulations on typical satellite images show that JPEG2000 compression spends 70% of processing time in wavelet transformation, which contains such iterations and parallel structures. The 30% remaining processing time is spent in entropy coding and formatting. GEZGIN achieves its mission breaking out such blocks/structures into multiple data paths among image processing.
units and implementing them in reconfigurable hardware. This has improved the overall system performance without loss of flexibility. The JPEG2000 compression task on GEZGIN is distributed as follows:

1- Wavelet transformation: A 5/3 integer wavelet filtering and signal decomposition, is implemented on Xilinx Virtex-E type FPGA.

2- Entropy coding and formatting: are implemented on TMS320C6701 DSP. JPEG2000 uses a specific version of the arithmetic coding known as MQ.

Figure B-5 shows the block diagram of the GEZGIN payload [Ismaeloglu02].

The GEZGIN compressor has a total input bandwidth of 80 Mbps and transfers the data to the SSDR over 25 Mbps in 6.5 seconds [Ismaeloglu02].
In table B-4 some compression ratios obtained when applying JPEG2000 losslessly on some image sets are presented [Ismailoglu02].

<table>
<thead>
<tr>
<th>Image</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERC_TM 512</td>
<td>1.440</td>
</tr>
<tr>
<td>DENVER512</td>
<td>1.254</td>
</tr>
<tr>
<td>LENA256</td>
<td>1.729</td>
</tr>
</tbody>
</table>

Table B-4: Performance results of the JPEG2000 algorithm when losslessly compressed with GEZGIN.

Figure 3-6 shows the test image set that has been used to assess the performances of JPEG2000, using the GEZGIN compression system.

Figure B-6: Samples of the compressed images using GEZGIN: (a) ERC_TM 512; (b) DENVER512; (c) LENA256
B.4 Conclusion

Satellite on-board processing is subject of heavy constraints on volume, weight, and power consumption. The need for computational power is always growing alongside a need for excellent reliability. For low-cost commercial small satellites, efficient utilization of the communication bandwidth is important.

System performance is a major concern while designing onboard complex schemes. Software solutions offer a very practical solution to a wide range of problems. This appendix illustrates a DSP approach used to implement the Rice algorithm. Although, many modifications and optimisations have been applied to the programs, the performances of this lossless technique have been proven to be poor in term of compression ratio (1.5:1) and compression speed (230 Kpixel/s) in comparison with the other techniques shown in the appendix.

SRAM-based Field Programmable Gate Arrays, have the property to be iteratively changed or updated while in product development. They provide greater flexibility and better price/performance ratios [Hauser00]. This is a big advantage in many applications that need multiple trial versions within development. In this appendix, an application of the LOCO-I algorithm has proven to obtain a good trade-off compression ratio and processing speed outperforming the DSP approach. The hybrid approach, where JPEG2000 was implemented, appeared very interesting when it is possible to separate the parallel processes from the sequential ones within the compression algorithm.

As a result of the investigation presented in this appendix it can be concluded that the hardware approach offers very good performance and flexibility in the design of a lossless image compression scheme. However, factors such as the nature of the algorithm, its complexity and the amount of the intrinsic parallelism play an important role in selecting the most suitable solution.