EDGE BASED STEGANOGRAPHIC TECHNIQUES FOR COVERT COMMUNICATION THROUGH OVERT CHANNELS

Submitted in partial fulfillment of the requirement for
the Award of the Degree of

DOCTOR OF PHILOSOPHY

IN

Computer Engineering

BY

AISHA CYRA FURTADO FERNANDES

(ID. NO :12PHCOMP101)

FACULTY OF ENGINEERING AND TECHNOLOGY
SAM HIGGINBOTTOM INSTITUTE OF AGRICULTURE, TECHNOLOGY AND SCIENCES
DEEMED- TO- BE UNIVERSITY
(FORMERLY ALLAHABAD AGRICULTURAL INSTITUTE)
NAINI , ALLAHABAD – 211007
2016

-John William Draper
Dedicated

With deepest love and everlasting respect

to

My father, Pedrito Fernandes

and

My husband, Austin Colaco
ACKNOWLEDGEMENT

I begin by praising the Lord Almighty, Without His immense Blessings, this work would not have been possible. During the course of this research, there have been numerous people who have helped me with guidance and inspiration for completing this thesis.

At the outset, I am indebted and would like to express my special appreciation and thanks to my advisor, Professor (Dr.) Wilson Jeberson, Head of Department, Department of Computer Science and Information Technology, SSET, SHIATS, for his continuous encouragement, guidance and support at all stages of this research.

I also owe thanks to Professor (Col.) Gurmit Singh, Department of Computer Science and Information Technology, SHIATS, for his insightful and encouraging comments and suggestions throughout this research.

I wish to acknowledge the efforts of the Chairman and members of my Student Advisory Committee: Dr. Wilson Jeberson, Head of Dept. (Computer Sc. & Information Technology, SHIATS), Dr. Amit Chattree, Head of Dept. (Chemistry) and Associate Dean (Faculty of Science, SHIATS), Dr. Nathaniel Stephen Deputy Registrar (Education, SHIATS) and Dr. Raghav Yadav, Associate Professor, Dept. of Computer Sc. & Information Technology for serving as my committee members and for all their support and suggestions, which have helped improve the quality of my thesis.

I wish to express my sincere gratitude to Professor (Dr.) Rajendra B. Lal, Honourable Vice Chancellor, SHIATS, Professor (Dr.) Newman Fernandes, Director (DDE) & Ex Pro-Vice
Chancellor, SHIATS, Professor (Dr.) P. Ramteke (Dean, P.G Studies), and Professor (Dr.) Imtiyaz (Dean, SSET) for providing adequate facilities during the course of my work.

I would like to thank the administrative and technical staff members of SHIATS who have been kind enough to help in their respective roles.

I would also like to put on record, my appreciation and gratitude to the Government of Goa for relieving me from my college duties to do my research work.

I wish to extend my sincere gratitude to Professor Vivek B. Kamat, Director, Directorate of Technical Education, Government of Goa, Prof. Pradip Kusnur, Deputy Director, Directorate of Technical Education and Prof. Vivek Bellokar, Asst. Director, Directorate of Technical Education, for their immense help.

I must also express my thanks to all my friends and colleagues at the Goa Engineering College who helped me by participating, guiding and motivating me in the course of my research.

A special thanks to my family. I would not have contemplated on this road if it were not for my parents, Ludovina and Pedrito, who instilled within me a love of creative pursuits, science and language, the art of submitting to God’s will, all of which find a place in this thesis. It was under their watchful eye that I gained so much drive and an ability to tackle challenges head on. Words cannot express how grateful I am to my parents-in-law, Carmen and Dr. Antonio, for all the sacrifices that they have made for me. Their prayers and constant encouragement was what sustained me this far.

I also thank my sisters, sister-in-law and their extended families for all their love, inseparable support and prayers.
It would not have been possible to complete my research without the immense help and support of someone who shares my passions. I express my appreciation to my husband, Austin, for his guidance, encouragement, quiet patience, unwavering love and novel ideas towards my research. He spent sleepless nights and was always my support in moments of hindrances. Without his enthusiasm, this work would not have been the same as it is.

I also thank our darling son, little Abner Mark whom God has blessed us with, for his patience throughout my preparation.

Special thanks to the SHIATS guest house management, Ruchika Daniel, Deepti Joshi and Shalini Ansari for the hospitality during my stay at Allahabad.

My thanks also go out to DIAT (Defence Institute of Advanced Technology), Pune, to CRSI (Cryptology Research Society of India) Kolkata, Department of Forensic Science, SHIATS and the Goa State Forensic Science Laboratory.

Last but not the least; I cannot forget my innumerable friends who were of great support and encouragement till date.

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of my research. I am sincerely grateful for their aspiring guidance, invaluable constructive criticism and friendly advice during the research work.

God bless you all abundantly.

Aisha Fernandes
ABSTRACT

Cross Swords between the good and the bad in the internet ground is getting tougher as more and more refined evolution takes place in this technology driven era. The focus behind data security is to ensure privacy while protecting personal or corporate data. At times, data transmission secrecy is important as more and more people are joining the cyberspace revolution. This demands the need to revamp techniques in the ancient science of steganography. Steganography is the art of concealing data in a manner that its existence itself cannot be detected. Numerous techniques for steganography are known to exist, but very few of them are found to be independent of the file formats, as well as few of them target coloured images. This thesis proposes two steganographic techniques, namely, Arithmetic Division based data embedding (ADDE) and Bit Stream based data embedding (BSDE), bearing in mind the important characteristics of an embedding algorithm: capacity, imperceptibility and robustness. Both the techniques use the least significant bits of the red, green and blue components of a 24-bit coloured image as the carrier medium and the secret data can either be in the form of text or an image. To ensure imperceptibility, the secret data is stored within the edges of the cover image. If the secret data is an image, it is compressed using a wavelet compression algorithm, before being embedded. In the ADDE method, we perform an arithmetic division operation on the secret binary data, so as to reduce its magnitude and further apply a mapping logic by analyzing the sizes of all the bytes of quotient data. In the BSDE method, we use a stream of continuous bits to store the secret data. In order to optimize the data storage, the amount of bits used by each byte of data is mapped to a value set in a flag array. The proposed methods were tested on 200 real time camera clicked digital images and on various other image databases. Various
performance metrics were used to evaluate the quality of the stego image, as well as the extracted secret data. Experimental results indicate that the proposed techniques produce stego images that are highly imperceptible, as well as resist some of the most common visual and statistical attacks. The results obtained were compared with similar works done by other researchers and our results seemed to be more promising.

Keywords : steganography, covert communication, compression, edges, least significant bits, imperceptibility.
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDE</td>
<td>Arithmetic Division based Data Embedding</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>BMP</td>
<td>Bit Map Image</td>
</tr>
<tr>
<td>BPCS</td>
<td>Bit Plane Complexity Segmentation</td>
</tr>
<tr>
<td>BPP</td>
<td>Bits Per Pixel</td>
</tr>
<tr>
<td>BSDE</td>
<td>Bit Stream based Data Embedding</td>
</tr>
<tr>
<td>BWT</td>
<td>Burrows-Wheeler Transform</td>
</tr>
<tr>
<td>CMYK</td>
<td>Cyan Magenta Yellow Key</td>
</tr>
<tr>
<td>dB</td>
<td>Decibel</td>
</tr>
<tr>
<td>DCT</td>
<td>Discrete Cosine Transform</td>
</tr>
<tr>
<td>DIIT</td>
<td>Digital Invisible Ink Toolkit</td>
</tr>
<tr>
<td>DWT</td>
<td>Discrete Wavelet Transform</td>
</tr>
<tr>
<td>EALMR</td>
<td>Edge Adaptive Least significant bit Matching Revisited</td>
</tr>
<tr>
<td>EBCOT</td>
<td>Embedded Block Coding with Optimized Truncation</td>
</tr>
<tr>
<td>et al.</td>
<td>And others</td>
</tr>
<tr>
<td>FFT</td>
<td>Fast Fourier Transform</td>
</tr>
<tr>
<td>GIF</td>
<td>Graphics Interchange Format</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HBC</td>
<td>Hiding Behind Corners</td>
</tr>
<tr>
<td>HSL</td>
<td>Hue Saturation Luminosity</td>
</tr>
<tr>
<td>HVS</td>
<td>Human Visual System</td>
</tr>
<tr>
<td>ID</td>
<td>Identification</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
</tbody>
</table>
IWT  Integer Wavelet Transform
JPEG  Joint Photographic Experts Group
KB   Kilo Byte
LSB  Least Significant Bit
LSBM Least Significant Bit Matching
LZW  Lempel-Ziv-Welch
MATLAB Matrix Laboratory
MB   Mega Byte
MSB  Most Significant Bit
MSE  Mean Squared Error
PIXEL Pictures Element
PNG  Portable Network Graphics
PSNR Peak Signal to Noise Ratio
PVD  Pixel Value Difference
QT   Quantization Table
RGB  Red Green Blue
RLE  Run length coding
ROI  Region of interest
SPAM Subtractive Pixel Adjacency Matrix
SPIHT Set Partitioning in Hierarchical Trees
SSS  Simple Steganalysis Suite
SVD  Single Value Decomposition
SVM  Support Vector Machine
TIF  Tagged Image file Format
VSL  Virtual Steganographic Laboratory
YIQ  Y – Luminance I – In Phase Q – Quadrature color space
YUV  Y – Luminance UV – Chrominance
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table No.</th>
<th>Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1</td>
<td>Technical Specifications of Nikon P600 Digital Camera</td>
<td>53</td>
</tr>
<tr>
<td>Table 3.2</td>
<td>Technical Configurations of Desktop</td>
<td>55</td>
</tr>
<tr>
<td>Table 3.3</td>
<td>Specifications of HP Laptop</td>
<td>56</td>
</tr>
<tr>
<td>Table 3.4</td>
<td>File Format Dependencies in CxImage</td>
<td>63</td>
</tr>
<tr>
<td>Table 3.5</td>
<td>Mapping Criterion used in ADDE</td>
<td>75</td>
</tr>
<tr>
<td>Table 3.6</td>
<td>Mapping Table for BSDE</td>
<td>88</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Comparison of Various Edge Detection Operators</td>
<td>115</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>Comparison of PSNR Values at 70% and 90% Compression</td>
<td>122</td>
</tr>
<tr>
<td>Table 4.3</td>
<td>File size at 90% compression quality factor</td>
<td>123</td>
</tr>
<tr>
<td>Table 4.4</td>
<td>File size at 70% compression quality factor</td>
<td>123</td>
</tr>
<tr>
<td>Table 4.5</td>
<td>Basic Statistical test data for embedding secret text using ADDE</td>
<td>128</td>
</tr>
<tr>
<td>Table 4.6</td>
<td>Basic Statistical test data for embedding secret images using ADDE</td>
<td>130</td>
</tr>
<tr>
<td>Table 4.7</td>
<td>PSNR values &amp; Embedding capacity for LenaRGB.bmp (ADDE)</td>
<td>131</td>
</tr>
<tr>
<td>Table 4.8</td>
<td>PSNR values &amp; Embedding capacity for Logo.bmp (ADDE)</td>
<td>131</td>
</tr>
<tr>
<td>Table 4.9</td>
<td>PSNR values &amp; Embedding capacity for Fieldview.bmp (ADDE)</td>
<td>132</td>
</tr>
<tr>
<td>Table 4.10</td>
<td>PSNR values &amp; image embedding capacity for Lena &amp; logo (ADDE)</td>
<td>132</td>
</tr>
<tr>
<td>Table 4.11</td>
<td>PSNR values of extracted secret images</td>
<td>134</td>
</tr>
<tr>
<td>Table 4.12</td>
<td>SSIM values of the original and stego images (ADDE)</td>
<td>135</td>
</tr>
<tr>
<td>Table 4.13</td>
<td>Basic Statistical test data for embedding secret text using BSDE</td>
<td>143</td>
</tr>
<tr>
<td>Table 4.14</td>
<td>Basic Statistical test data for embedding secret images using BSDE</td>
<td>143</td>
</tr>
<tr>
<td>Table 4.15</td>
<td>PSNR values &amp; Embedding capacity for LenaRGB.bmp (BSDE)</td>
<td>144</td>
</tr>
<tr>
<td>Table 4.16</td>
<td>PSNR values &amp; Embedding capacity for Logo.bmp (BSDE)</td>
<td>145</td>
</tr>
<tr>
<td>Table 4.17</td>
<td>PSNR values &amp; Embedding capacity for Fieldview.bmp (BSDE)</td>
<td>145</td>
</tr>
<tr>
<td>Table 4.18</td>
<td>PSNR values &amp; image embedding capacity for Lena &amp; logo (BSDE)</td>
<td>145</td>
</tr>
<tr>
<td>Table 4.19</td>
<td>PSNR values of extracted secret images (ADDE)</td>
<td>147</td>
</tr>
<tr>
<td>Table 4.20</td>
<td>SSIM values of the original and stego images (BSDE)</td>
<td>148</td>
</tr>
<tr>
<td>Table 4.21</td>
<td>SPAM accuracy for edge embedding algorithms</td>
<td>154</td>
</tr>
<tr>
<td>Table 4.22</td>
<td>PSNR values of Lena image for different steganography algorithms</td>
<td>157</td>
</tr>
<tr>
<td>Table 4.23</td>
<td>PSNR values of Logo image for different steganography algorithms</td>
<td>157</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure No.</th>
<th>Title</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>The Communication Process Diagram</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Magic Triangle Representing Characteristics of Steganography</td>
<td>5</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>Different Embodiment Disciplines of Information Hiding</td>
<td>7</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>CxImage Library architecture</td>
<td>61</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Anatomy of the Eye</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Rod and Cone Cells</td>
<td>68</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Cartesian System Representation of the RGB Model</td>
<td>70</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>ADDE method data embedding schematic diagram</td>
<td>76</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Flow chart of Text embedding using ADDE</td>
<td>77</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>ADDE Text Embedding Algorithm</td>
<td>78-79</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>ADDE Text Extraction Algorithm</td>
<td>80</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Flow chart of Image Embedding using ADDE</td>
<td>81</td>
</tr>
<tr>
<td>Figure 3.10</td>
<td>ADDE Image Embedding Algorithm</td>
<td>82-83</td>
</tr>
<tr>
<td>Figure 3.11</td>
<td>ADDE Image Extraction Algorithm</td>
<td>84</td>
</tr>
<tr>
<td>Figure 3.12</td>
<td>BSDE method data embedding Schematic diagram</td>
<td>89</td>
</tr>
<tr>
<td>Figure 3.13</td>
<td>Flow Chart for BSDE Text Embedding Technique</td>
<td>90</td>
</tr>
<tr>
<td>Figure 3.14</td>
<td>BSDE Text Embedding Algorithm</td>
<td>91-92</td>
</tr>
<tr>
<td>Figure 3.15</td>
<td>BSDE Text Extraction Algorithm</td>
<td>93</td>
</tr>
<tr>
<td>Figure 3.16</td>
<td>Flow Chart for BSDE Image Embedding Technique</td>
<td>95</td>
</tr>
<tr>
<td>Figure 3.17</td>
<td>BSDE Image Embedding Algorithm</td>
<td>96-97</td>
</tr>
<tr>
<td>Figure 3.18</td>
<td>BSDE Image Extraction Algorithm</td>
<td>98</td>
</tr>
<tr>
<td>Figure 3.19</td>
<td>Wavelet Based Image Compression</td>
<td>100</td>
</tr>
<tr>
<td>Figure 3.20</td>
<td>Wavelet Based Image De-Compression</td>
<td>100</td>
</tr>
<tr>
<td>Figure 3.21</td>
<td>Wavelet Compression Algorithm</td>
<td>102</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Figure 3.22</td>
<td>Flow Chart for Edge Detection Technique</td>
<td>105</td>
</tr>
<tr>
<td>Figure 3.23</td>
<td>Modified Canny Edge Detection Algorithm</td>
<td>106</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Test images used for study</td>
<td>114</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Edge Detection using Canny &amp; modified Canny</td>
<td>116</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Comparison of Compression algorithms at 90 % quality factor</td>
<td>118</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>Comparison of Compression algorithms at 70 % quality factor</td>
<td>120</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>Compressed images of 70 percent quality at 800 percent zoom</td>
<td>121</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>Comparison of PSNR values for different compression algorithms</td>
<td>122</td>
</tr>
<tr>
<td>Figure 4.7</td>
<td>Visual Perceptibility Test for ADDE technique</td>
<td>125</td>
</tr>
<tr>
<td>Figure 4.8</td>
<td>Comparison of zoomed LenaRGB.bmp for ADDE technique</td>
<td>127</td>
</tr>
<tr>
<td>Figure 4.9</td>
<td>Comparison of PSNR values for data embedded (ADDE)</td>
<td>134</td>
</tr>
<tr>
<td>Figure 4.10</td>
<td>Comparison of histogram of Lena for ADDE technique</td>
<td>136</td>
</tr>
<tr>
<td>Figure 4.11</td>
<td>Comparison of histogram of Fieldview for ADDE technique</td>
<td>136</td>
</tr>
<tr>
<td>Figure 4.12</td>
<td>Comparison of histogram of Logo for ADDE technique</td>
<td>137</td>
</tr>
<tr>
<td>Figure 4.13</td>
<td>Results of LSB enhancement and Chi square tests for ADDE</td>
<td>139</td>
</tr>
<tr>
<td>Figure 4.14</td>
<td>Visual Perceptibility Test for BSDE technique</td>
<td>141</td>
</tr>
<tr>
<td>Figure 4.15</td>
<td>Comparison of PSNR values for data embedded (BSDE)</td>
<td>147</td>
</tr>
<tr>
<td>Figure 4.16</td>
<td>Comparison of histogram of Lena for BSDE technique</td>
<td>149</td>
</tr>
<tr>
<td>Figure 4.17</td>
<td>Comparison of histogram of Fieldview for BSDE technique</td>
<td>149</td>
</tr>
<tr>
<td>Figure 4.18</td>
<td>Comparison of histogram of Logo for BSDE technique</td>
<td>150</td>
</tr>
<tr>
<td>Figure 4.19</td>
<td>Results of LSB enhancement and Chi square tests for BSDE</td>
<td>151</td>
</tr>
<tr>
<td>Figure 4.20</td>
<td>Comparison of PSNR values for Lena image using different algorithms</td>
<td>158</td>
</tr>
<tr>
<td>Figure 4.21</td>
<td>Comparison of PSNR values for Logo image using different algorithms</td>
<td>158</td>
</tr>
</tbody>
</table>
### Notations used in Proposed Algorithms

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>Cover Image</td>
</tr>
<tr>
<td>M</td>
<td>Secret data</td>
</tr>
<tr>
<td>S</td>
<td>Stego Image</td>
</tr>
<tr>
<td>L</td>
<td>Length of secret data</td>
</tr>
<tr>
<td>$T_h$</td>
<td>Upper threshold</td>
</tr>
<tr>
<td>$T_l$</td>
<td>Lower threshold</td>
</tr>
<tr>
<td>F</td>
<td>Flag array</td>
</tr>
<tr>
<td>E</td>
<td>Edge map</td>
</tr>
<tr>
<td>Q</td>
<td>Quotient array</td>
</tr>
<tr>
<td>R</td>
<td>Remainder array</td>
</tr>
<tr>
<td>P</td>
<td>Pixel</td>
</tr>
<tr>
<td>B</td>
<td>Bit stream</td>
</tr>
<tr>
<td>$I_1$</td>
<td>Input image (for edge detection)</td>
</tr>
<tr>
<td>$I_g$</td>
<td>Grayscale image</td>
</tr>
<tr>
<td>K</td>
<td>Gaussian Kernel</td>
</tr>
<tr>
<td>G</td>
<td>Gradient Value</td>
</tr>
<tr>
<td>$\Theta$</td>
<td>Gradient angle</td>
</tr>
</tbody>
</table>
DECLARATION

I, Aisha Cyra Furtado Fernandes declare that the work presented in this thesis entitled ‘Edge based Steganographic Techniques for Covert Communication through Overt Channels’ submitted to the Department of Computer Science and Information Technology, in the Faculty of Engineering and Technology, Sam Higginbottom Institute of Agriculture, Technology and Sciences, Deemed University, Naini, Allahabad, for the award of the Doctor of Philosophy degree in Computer Engineering, is an original work. I have neither plagiarized nor submitted the same work for the award of any other degree. In case this undertaking is found incorrect, my degree may be withdrawn unconditionally by the University.

Date :

Place : Allahabad

Aisha Cyra Furtado Fernandes

(ID No. : 12PHCOMP101)
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>TITLE PAGE</td>
<td></td>
</tr>
<tr>
<td>DECLARATION</td>
<td></td>
</tr>
<tr>
<td>CERTIFICATE OF ORIGINALWORK</td>
<td></td>
</tr>
<tr>
<td>STUDENT ADVISORY COMMITTEE CERTIFICATE</td>
<td></td>
</tr>
<tr>
<td>FINAL DEFENCE REPORT</td>
<td></td>
</tr>
<tr>
<td>ACKNOWLEDGEMENT</td>
<td>i</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>x</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xii</td>
</tr>
<tr>
<td>LIST OF ABBREVIATIONS / ACRONYMS</td>
<td>xiv</td>
</tr>
<tr>
<td>NOTATIONS USED IN PROPOSED ALGORITHMS</td>
<td>xvi</td>
</tr>
</tbody>
</table>

## I INTRODUCTION

1.1 Concepts                                      1
1.2 Classification of Covert Communication Techniques 6
1.3 History of Steganography                      7
1.4 Applications of Steganography                 9
1.5 Significance of Steganography                 11
1.6 Issues of Concern in Steganography            12
1.7 Research Questions                            13
1.8 Objectives of Research                        14
II REVIEW OF LITERATURE

2.1 Image Processing Concepts
2.2 Steganography
   2.2.1 Steganography in Spatial Domain
   2.2.2 Steganography in Temporal Domain
2.3 Edge based Steganography
2.4 Image Compression
2.5 Edge Detection
2.6 Steganalysis
2.7 Research Gap Analysis

III MATERIALS AND METHODS

3.1 Equipment / Tools Used
   3.1.1 Camera Specifications
      3.1.1.1 Overview
      3.1.1.2 Lens
      3.1.1.3 Vibration Reduction
      3.1.1.4 Sensor Technology
      3.1.1.5 Focus Technology
      3.1.1.6 Picture Processing Technology
   3.1.2 Desktop Configuration
   3.1.3 Laptop Configuration
3.2 Development Environment
   3.2.1 Processor Requirement
   3.2.2 Disk Space Requirement
   3.2.3 Memory Requirement
3.3 Software / Platform Used
   3.3.1 Microsoft Visual Studio
   3.3.2 Matlab 7.0
   3.3.3 OpenCV
   3.3.4 CxImage Library
   3.3.5 Virtual Steganographic Laboratory (VSL)
   3.3.6 Simple Steganalysis Suite
3.4 Theoretical Concepts
3.4.1 Visual Perception  
3.4.2 RGB Color Model  
3.5 Data Beds Used  
3.6 Proposed Methodology  
3.6.1 Proposed Steganography Schemes  
3.6.1.1 Arithmetic Division Based Data Embedding (ADDE)  
3.6.1.1.1 Using ADDE with Text  
3.6.1.1.2 Using ADDE with Images  
3.6.1.1.3 Mathematically understanding ADDE  
3.6.1.2 Bit Stream Based Data Embedding (BSDE)  
3.6.1.2.1 Using BSDE with Text  
3.6.1.2.2 Using BSDE with Images  
3.6.1.2.3 Mathematically understanding BSDE  
3.6.2 Wavelet Based Data Compression Algorithm  
3.6.3 Edge Detection Method  
3.6.4 Application GUI  
3.7 Performance Evaluation Measures  
3.7.1 Mean Square Error (MSE)  
3.7.2 Peak Signal to Noise Ratio (PSNR)  
3.7.3 Structural Similarity Index (SSIM)  
3.7.4 Compression Ratio (CR)  
3.7.5 Space Savings (SS)  
3.7.6 Embedding Rate (ER)  
3.7.7 Embedding Capacity (EC)  
3.7.8 Basic Statistical Measures  

IV RESULTS AND DISCUSSION  
4.1 Edge Detection in Cover Image  
4.2 Secret Image Compression  
4.3 Security Analysis of Proposed Steganography Methods  
4.3.1 Analysis of ADDE Method  
4.3.1.1 Visual Perceptibility Test  
4.3.1.2 Statistical Analysis of Stego-Images  
4.3.1.3 Payload Analysis  
4.3.1.4 Peak Signal to Noise Ratio Analysis
4.3.1.5 Structural Similarity Index Analysis 134
4.3.1.6 Histogram Analysis 135
4.3.1.7 LSB Enhancement Attack and Chi Square attack 137
4.3.2 Analysis of BSDE Method 140
  4.3.2.1 Visual Perceptibility Test 140
  4.3.2.2 Statistical Analysis of Stego-Images 142
  4.3.2.3 Payload Analysis 144
  4.3.2.4 Peak Signal to Noise Ratio Analysis 146
  4.3.2.5 Structural Similarity Index Analysis 147
  4.3.2.6 Histogram Analysis 148
  4.3.2.7 LSB Enhancement Attack and Chi Square attack 150
4.4 Steganalysis Using Feature Extractor 151
4.5 Evaluation of Proposed Algorithms 155
  4.5.1 Performance and Robustness 155
  4.5.2 Limitations and Merits 159

V SUMMARY AND CONCLUSION 161
  5.1 Summary 161
  5.2 Conclusion 164
  5.3 Future Scope 165

VI REFERENCES 167 - 180

VII APPENDICES
  A1 Application Screen Shots
  A2 Screen Shots of Steganalysis tools used
  A3 Papers Published
INTRODUCTION

1.1 Concepts

With the advent of high speed communication networks, the Internet acts as a proficient and cost effective information sharing vehicle that makes the globe at large appear like a small digital city. Due to its speed, simplicity and security, digital communication has become an integral part of everyone’s life. In today's ultra-busy society, it is very important to be mindful of the demands on people's time. Ubiquitous Networks and the sudden bloom in cloud computing technologies creates opportunities for individuals and organizations, to share, transmit, and store large amounts of data, images, audios and videos. This information is vulnerable to unauthorized access and interception, while in storage or transmission; thus providing security for this information in all its forms is a challenging issue.

The goal of any communication is to get your message across to the receiver’s clearly, unambiguously and secretly, if desired. This involves effort from both, the sender of the message and the receiver. It’s a process that can be fraught with error, with messages often misinterpreted by the recipient. A miscommunication can result in tremendous confusion, wasted effort and missed opportunity. It is a fact that communication is said to be successful only when both the sender and the receiver understand the same information as a result of the communication. Communication barriers can pop-up at every stage of the communication process (which consists of sender, message, channel, receiver, feedback and context) and have the potential to create misunderstanding and confusion. The communication process diagram has been beautifully explained in “The Mathematical Theory of Communication,” Copyright 1949, by the Board of Trustees of the University of Illinois and is shown below in figure1.1.
Messages are conveyed through channels. Different channels have different strengths and weaknesses. Wikipedia defines a communication channel as a physical transmission medium such as a wire, or a logical connection over a multiplexed medium such as a radio channel. A channel is used to convey an information signal, for example, a digital bit stream from one or several senders (or transmitters) to one or several receivers. A channel has a certain capacity for transmitting information, often measured by its bandwidth in Hz or its data rate in bits per second.

Covert communication often refers to the process of communicating data through a channel that is neither designed, nor intended to transfer information (Lampson, 1973). It is a channel that can be exploited by a process, to transfer information in a manner that violates the systems security policy. The primary use of covert channels was to allow information transfer by exploiting weaknesses in conventional communication systems like network protocols. Covert channels or the so called subliminal channels are studied as part of the science called steganography and the different steganographic methods used in telecommunication networks are referred to as network steganography. An overt
channel is a communication channel within a computer network, designed for the authorized transfer of data.

Information hiding (or data hiding) is a general term encompassing a wide range of problems beyond that of embedding messages in content. The term hiding here can refer to either making the information imperceptible (as in watermarking) or keeping the existence of the information secret. Systems for inserting messages in carriers (covers) can thus be divided into watermarking systems; in which the message is related to the cover work, and non-watermarking systems; in which the message is unrelated to the cover work. They can also be independently divided into steganographic systems, in which the very existence of the message is kept secret, and non-steganographic systems, in which the existence of the message need not be secret (Cox et al., 2007).

The tremendous growth in technology has seen more and more of people join the cyberspace revolution. Information sharing and transfer over overt channels has increased exponentially and this makes data security an inevitable task. The secrecy of digital information sent across an open communication channel like the internet is always questionable. In this digital age, it is very important to keep public information public, secret information secret, private information private and protect the copyrights of data. The need for secret communications is as old as the communication itself. To accomplish this task, new methods based on the principle of image processing are being developed and used. To protect important data from being illegally accessed, we can either encrypt it so as to make it available in a gibberish form or we can merely hide its presence. However, the encrypted data exists in a meaningless form and may attract the attention of interceptors (Schneier, 2007). Encryption or the science of cryptography only prevents adversaries from decoding the communication. Sometimes, the mere
existence of communication or even changes in communication patterns, such as an increased message frequency are enough to raise suspicion and reveal the onset of events. *Steganography* is the art and science of writing secret data in such a way, that no one, except the intended recipient, knows of the existence of the data (*Marvel, 1999*). It refers to the art of hiding and transmitting data through apparently innocuous carriers, in an effort to conceal the existence of the data. The goal of steganography, is to avoid arousing suspicion to the transmission of a hidden message. Steganography attempts to hide the very existence of communication. Steganography and cryptography are cousins in the spycraft family (*Cox et al., 2007*). Successful steganography depends upon the carrier medium not attracting attention. When the presence of stego-content is suspected, the main goal of steganography is defeated (*Arjun and Negi, 2006*). The advantage of steganography over cryptography is that the information cannot be suspected and it protects both, messages and communicating parties. Steganographic data can be embedded in a document file, image file, audio or video file. The content used to embed information is called as carrier or cover object. The cover along with the hidden sensitive information is called as stego-object (*Hemalatha et al., 2012*).

Given the high degree of redundancy present in a digital representation of multimedia content, there has been an increased interest in using multimedia content for the purpose of Steganography (*Radhakrishnan et al., 2005*). The goal of every steganographic algorithm is to embed data into a carrier format without altering too much of the original information. By maintaining most of the format’s original information, the resulting steganographic carrier has imperceptible changes for both human and computerized analysis. The strength of a steganographic algorithm is measured as being the degree of resistance against reverse engineering and detection methods(*Mare et al., 2011*).
There is a visual requirement model which is called *magic triangle* (Duric and Jajodia, 2001) in the field of information hiding, given in Figure 1.2. Imperceptibility, robustness to attacks, and the insertion capacity are in the corners of the magic triangle. This model is convenient for a visual representation of the required trade-offs between the capacity of the embedded data and the robustness to certain attacks, while keeping the perceptual quality of the stego-medium at an acceptable level. It is not possible to attain high robustness to signal modifications and high insertion capacity at the same time (Zhang and Tang, 2007).

![Figure 1.2 Magic triangle representing characteristics of Steganography](image)

The fundamental differences between steganography and watermarking have been well elaborated by Peticolas et al., (1999). The information hidden by a watermarking system is always related to the digital object to be protected or to its owner, while steganographic systems hide any information. They also differ in the robustness criteria.
wherein watermarking concerns potential removal by an adversary while steganography is concerned with the detection of the hidden data. Steganographic communications are usually point to point, while watermarking techniques are usually one towards many.

Steganalysis is the art of detecting the existence of hidden information. Steganology refers to the science of steganography and steganalysis put together. Interest in steganology increased significantly after the terrorist attacks on September 11, 2001, when it became clear that means for concealing the communication itself was likely to be used for criminal activities. Interestingly, USA Today reported on this possibility several months before the September 11, 2001 attack (Cox et al., 2007). However, there has been little evidence to substantiate these claims.

1.2 Classification of Covert Communication Techniques

Covert or Secure communication has become a major challenge in the present digital world. People are continuously striving to develop innovative methods, to aid secure and secret communication. Figure 1.3 describes the various embodiment disciplines of information hiding (Cheddad et al., 2010; Mathew, 2010). The bold face in the figure indicates the focus area in this research work.
1.3 History of Steganography

The word *steganography* was invented by Trithemius; an author of one of the early publications on cryptography. This term is derived from the Greek words *steganos*, which means “covered,” and *graphia*, which means “writing.” Steganography is the art of concealed communication. The very existence of hidden information is secretive. The first instance where steganography was being used to send messages, dates back decades ago, where *(Herodotus, 1992)* reported of a slave sent by his master, Histiaeus, to a city of Miletus with a secret message tattooed on his scalp. After tattooing, the slave was made to re-grow his hair in order to hide the message. He then journeyed to Miletus and, upon arriving, shaved his head to reveal the message to the city’s regent, Aristagoras. The message was to initiate a revolt against the Persian king. Herodotus narrates yet
another story of Demeratus, who scraped the wax off the surface of a wooden writing tablet and scribbled his warning about the planned invasion of Greece by the Persian Great King Xerxes into the wood. The tablet was then coated with a fresh layer of wax, to appear as a blank writing tablet that was safely carried to Sparta without arousing suspicion.

Aeneas too reported many data hiding methods, such as hiding messages in messenger’s soles or women’s earrings or messages carried by pigeons. He also proposed several methods for hiding within text by modifying the height of letter strokes or marking letters in a text using small holes (Tacticius, 1990). In the later years, invisible inks too have been used in the science of information hiding.

Acrostic or Linguistic steganography was one of the most noted ancient steganographic methods. Secret messages were encoded as initial letters of words or sentences or successive tercets in a poem. One of the most renowned works is Amorosa visione by Giovanni Boccacio wherein the initial of the successive tercets correspond exactly to the letters of the sonnets. Hypnerotomachia Poliphili, a puzzling and enigmatic book (Anonymous, 1499), unveiling the guilty love between a monk and a woman was another popular example of linguistic steganography. In the sixteenth century, a more advanced version of acrostic originally formulated in China and reinvented by Cardan, an Italian mathematician was Cardan’s Grille. Here the alphabets of the secret message were randomly hidden within text and a mask was then used to read the secret message. The problem was to securely deliver the mask to the receiver. The Germans and the Allies are said to have used acrostic as a means of communication during the First World
War. The World War also saw the use of microdots by the Germans for communication. However the Allies discovered the use of microdots in 1941. Gaspar Schott explained how to hide messages in music scores where each note used to correspond to a single letter. John Wilkins demonstrated how two Musicians may dialogue with one another by playing upon their instruments of music as well as by talking with their instruments of speech (Wilkins et al., 1694). He also explained how messages could be secretly hidden into geometric drawings using points, lines, and triangles. Various techniques have been used in electronic publishing projects to conceal serial numbers and copyright messages in the line spacing and other format features of documents (Brassil et al., 1994). It was found that shifting text lines up or down by one-three-hundredth of an inch to encode zeros and ones, was robust against multi-generation photocopying and could not be noticed by most people. Yet other methods included sending a message to a secret agent by marking certain letters in a newspaper using invisible ink, and adding sub-perceptible echo at certain places in an audio recording (Bender et al., 1996; Peticolas et al., 2000; Cox et al., 2007).

1.4 Applications of Steganography

The emanation of commercial espionage and the growing concerns about security due to terrorism, has intensified the nation’s interest in steganography. In the present digital era, where the entire community, banks on the internet and email for data exchange, steganography has created an atmosphere of corporate surveillance that has spawned various interesting applications. The purpose of steganography is to hide secret data in a carrier. It seeks to provide a covert communication channel between two parties. Digital media such as text files, images, audio, and video files have become the most obvious choices for data carriers. This is due to the fact that such digital media usually includes a
random noise component in which the secret message can be easily hidden (Chanu et al., 2012).

Steganography has several useful applications. However, like any other science, it can also be used with unlawful or illegitimate interests. Civilians may use it for protecting privacy while terrorists may use it for spreading terroristic information, which is potential for endangering our national as well as world security (Korhorn, 2002; Das and Tuithung, 2012). Steganography finds its application in defence related departments, police departments, detective investigation departments, medical imaging (Cheddad et al., 2008). Healthcare industry uses steganography in hiding messages in DNA sequences (Taylor et al., 1999). Anonymous communications, including anonymous remailers and Web proxies (Cheddad et al., 2010) are required by legitimate users to vote privately in online elections, make political claims, preserve online free speech, or to use digital cash. But the same techniques can be abused for defamation, blackmail, or unsolicited commercial mailing. Unobtrusive communications are required by military and intelligence agencies, even if the content is encrypted, the detection of a signal on a modern battlefield may lead rapidly to an attack on the signaller. For this reason, military communications use techniques such as spread spectrum modulation or meteor scatter transmission to make signals hard for the enemy to detect or jam (Petricolas et al., 2000).

This science is also widely used to hide data on the network in case of a breach, in peer-to-peer private communications, in posting sensitive confidential data on the Web to avoid transmission, in embedding corrective audio or image data in case corrosion occurs from a poor connection or transmission, to hide the copyrights information into the image to intact its legality to owner, to hide the descriptive elements of an image such as
name of people in the image, location in a map, or content of the image (Hussain et al., 2010). Individual’s details are also embedded in their photographs in smart IDs and identity cards (Chanu et al., 2012). The use of Steganography also has an important role in strengthening national security. It can also be regarded as secret sharing since messages can be shared secretly without being hacked or corrupted (Korhorn, 2002). Various corporations have also identified the potential of the science of steganography, in communicating trade secrets or new release information. Avoiding communication through well-known channels greatly reduces the risk of information being leaked in transit (Hemalatha et al., 2012). Hiding information in a photograph of the company anniversary celebration is less suspicious than communicating an encrypted file. This gives steganography a positive edge in secret communication.

1.5 Significance of Steganography

Electronic communication is increasingly susceptible to eavesdropping and malicious interventions. Encrypted messages are obvious, and when intercepted, it is clear that the sender and the recipient are communicating secretly. Encryption provides the means to assure the privacy of communications between various parties. If a sufficiently large encryption key is used, then the likelihood of decryption by technical means is negligible, irrespective of the resources available to a hacker. Unfortunately, the very fact that two people are exchanging encrypted messages, indicates that they have something to conceal. Any adversary may therefore decide to obtain the decryption key through forceful or even unlawful means.

Instead of encrypting messages, we can hide them in other innocuous looking objects, so that their very presence is not revealed. Steganography may therefore be a safer form of
communication. Law enforcement and counter intelligence agencies are interested in understanding the science of steganography, so that they are able to detect and trace hidden messages (Anderson and Peticolas, 1998). It is an enormous task for the law enforcement and intelligence agencies to monitor the million phone lines and million broadband connections, deciding which communication to intercept and which one to leave, so as to stop or eliminate the secret communications (Mathew, 2010).

Provos and Honeyman (2001), at the University of Michigan, scrutinized three million images from popular websites looking for any trace of steganography. They did not find any proof of infringement (Cheddad et al., 2010).

1.6 Issues of Concern in Steganography

A steganographer has the privilege to choose a cover medium of his choice. Embedding sensitive information into images freely available on the World Wide Web is not advisable, as a steganalyst might take note and opportunistically have recourse to decoding the stego image. In order to evade any Human Visual Perceptual attack, the generated stego image must not have visual artefacts. Smooth homogeneous areas such as a cloudless grey sky must be avoided; however images with natural redundant noise background and salient rigid edges should be targeted (Cheddad et al., 2008).

In image steganography, improving the capacity of hidden data into cover image, without causing any statistically significant modification is of major concern. There is always a trade-off between amount of secret data that can be hidden, and its perceptibility to attacks. The important issue is how to select the embedding regions or positions in the cover image that result in low distortion of the stego image. The
challenge is to find a way, to camouflage a secret message in an image without perceptible degrading the image quality and to provide better resistance against steganalysis process.

1.7 Research questions

Steganography is being used in order to enable secret communication. Following, are some of the research questions pertaining to the steganography domain:

- How to transfer sensitive data without arousing suspicion of a hacker?
- Is there a manner other than passwords and data encryption to safeguard confidential data?
- Which are the techniques available to transfer confidential data in a manner that it remains confidential?
- Images are often exchanged between people. Can they be used to communicate confidential data?
- What are the benefits and limitations the science of steganography brings to society?
- Can an algorithm be developed that is strong enough to withstand the many unique forms of steganalysis?
- Are the existing algorithms being used for steganography independent of file formats?
- Are terrorists the only ones with steganography tools?
1.8 Objectives of Research

➢ To investigate and analyse the existing methods used for steganography.

➢ Propose a technique for covert (secret) transmission of data with good embedding capacity and high imperceptibility, after noting the research gap analysis.

➢ Test the security/robustness of the proposed technique.
REVIEW OF LITERATURE

2.1 Image Processing Concepts

The exchange of information is essential for the development of civilization. The discovery and evolution of methods that makes transmission of information secure, attracted people since antiquity. The famous saying” What You See is What You Get” is not always true, specifically when it concerns images. Images can be more than what we identify with the Human Visual System (HVS) and hence we can say that they can convey more than what actually appears. Covert communication can be achieved using images as the cover medium.

Over the centuries, people discovered and developed techniques which gradually evolved into the sciences of Cryptography and Steganography. Cryptography disguises the message to be transmitted, so that only the intended recipient is able to read it; while Steganography hides the message by embedding it within other seemingly harmless messages or covers. Steganography dates back to the ancient Greece but only lately (late 20th century) it began research for scientific reasons. In the realm of this digital world, steganography has created an atmosphere of corporate vigilance, that has spawned various interesting applications, thus its continuing evolution is guaranteed. Today, it is widely used in Telecommunications industry, Medical imaging and in hiding strongly encrypted data (Cheddad et al., 2010; Aroukatos et al., 2012).

The drawbacks of large images are that they are cumbersome to transfer and upload, while posing a threat of drawing an adversary’s attention due to their uncommon size. Thus, compression is often recommended (Johnson et al., 1998). Wikipedia defines
Image file formats as standardized means of organizing and storing digital images. An image file may store data in uncompressed, compressed, or vector formats. Once displayed, an image becomes a grid of pixels, each of which has a number of bits to designate its color, equal to the color depth of the device displaying it. The most common image file formats used are Joint Photographic group (JPG), Tagged image file format (TIF), Portable Network Graphics (PNG), Bitmap image (BMP) and Graphics Interchange Format (GIF). Digital cameras and web pages normally use JPG files - because JPG compresses the data and heroically reduces the file size. However JPG uses lossy compression which reduces the image quality. GIF always uses lossless Lempel Ziv Welch (LZW) compression, but it is always an indexed color file (8-bits, 256 colors maximum), which is poor for 24-bit color photos. PNG and TIF files can also optionally handle the same indexed color mode that GIF uses; in addition they are more versatile too. But GIF is still very good for web graphics. PNG was invented more recently than the others, designed to bypass possible LZW compression patent issues with GIF, and since it was more modern, it offers other options too (RGB color modes, 16 bits, etc). One additional feature of PNG is transparency for 24 bit RGB images. Normally PNG files are a little smaller than LZW compression in TIF or GIF (all of these use lossless compression, of different types), but PNG is perhaps slightly slower to read or write. BMP files are uncompressed, and therefore large and lossless; their advantage is their simple structure and wide acceptance in Windows programs.

Digital images are stored in either 24-bit (true color images) or 8-bit per pixel files. Hence 8-bit color images, like GIF files, can be used to hide information. Here, each pixel is represented as a single byte, and the pixel's value is between 0 and 255 (Park et al., 2005; Juneja and Sandhu, 2013). Colour palette based steganography exploits the
smooth ramp transition in colours as indicated in the colour palette. The LSBs here are modified based on their positions in the palette index. **Johnson and Jajodia (1998)** supported using BMP (24 bit) instead of JPEG images. Their next-best choice was GIF files (256-color). BMP as well as GIF based steganography apply LSB techniques, while their resistance to statistical counter-attacks and compression are reported to be weak (**Westfeld, 2003; Mielikainen, 2006**). BMP files are bigger compared to other formats, which render them improper for network transmissions. JPEG images however, were at the beginning avoided because of their compression algorithm, which does not support a direct LSB embedding into the spatial domain. Many researchers claim that changes as small as flipping the LSB of one pixel in a JPEG image can be reliably detected (**Kharrazi et al., 2006**). The experiments using discrete cosine transform (DCT) coefficients showed promising results and redirected researchers towards the transform domain. Embedding in the transform domain using DCT makes steganography more robust and less prone to statistical attacks (**Cheddad et al., 2010**).

The best quality hidden message is normally produced using a 24-bit bitmap as a cover image (**Swain and Lenka, 2011**). These are most popular as cover media because of being simple in structure, highly standardized, extremely widespread in use and contain minimal extra information (**Juneja and Sandhu, 2013**). Each pixel is represented by three bytes, representing the red, green and blue colour values for that pixel. The higher the number, the more intense the colour is for that pixel. In a 24-bit coloured image, a bit of each of the red, green and blue colour components can be used to hide data, so, each pixel can store a minimum of 3 bits without any image degradation (**Cheddad et al., 2010**). In 24-bit bitmaps, the number of bytes per row is always end-padded with zeros to be a multiple of four. These extra bytes should not be used to hide secret data as any
alteration would be easily detectable. Thus, for an image to remain inconspicuous, only the LSBs of the actual pixel data should be altered. LSB embedding exploits the fact that the level of precision in many image formats is far greater than that perceived by the human eye. Therefore, an altered image with slight variations in its colours will be indistinguishable from the original by a human being, just by observing it (Ghasemi and Shanbehzadeh, 2010).

The peak signal to noise ratio (PSNR) metric is used to measure the distortion between the original image and a modified image (Hong et al., 2008). It is expressed on a logarithmic scale in decibels (dB). PSNR can also be used to measure the degree of imperceptibility. PSNR values falling below 30dB indicate a fairly low image quality. A high quality modified image should strive for 40dB and above (Cheddad et al., 2008).

To test computer vision and imaging algorithms, it is necessary that the original images be uncompressed. In addition the fields of steganography and image forensics, demand proper uncompressed images to ensure proper evaluation and benchmarking. The UCID10K image database comprising of 10,000 uncompressed colour images was released by Schaefer (2010). The UCID10K database is publicly available for fellow researchers and can be downloaded from: http://www-staff.lboro.ac.uk/~cogs/datasets/UCID10K/ucid10k.html. (Schaefer, 2010)

2.2 Steganography

Steganography has been defined by various authors and researchers in various ways. According to (Marvel, 1999), steganography is defined as the art and science of writing secret data in such a way that no one, except the intended receiver, knows of the
existence of the data. Radhakrishnan et al., (2005) refers to steganography as the science of invisible communication. He explains that steganographic techniques strive to hide the very presence of the message itself from an observer. Hemalatha et al., (2012) defines steganography, as the art of conveying information secretly by concealing the very existence of information in some other medium such as image, audio or video files.

The media used to embed information is called as cover object or carrier. The cover along with the hidden secret information is called as stego-object. Various digital media such as text, image, audio, video, and multimedia are utilized as carriers or covers for secret data. The art of steganography includes a vast array of techniques for hiding secret data within a wide range of cover media (Begum and Venkataramani, 2012). Successful steganography is when it conceals the secret data to be highly imperceptible. Sajedi and Jamzad, (2010) and Fridrich et al., (2001) explained that for robust image steganography, it was essential to choose an appropriate cover image. Digital images are considered a good choice for a steganography cover because of their insensitivity for the human visual system (Cole, 2003). Images with a low number of colours, computer art, and images with unique semantic content should not be used as cover images. Since the content of cover images has very little significance in steganography, the steganographer has the liberty to select a cover image from a database, to achieve higher security and satisfactory embedding capacity (Sajedi and Jamzad, 2010). They concluded that for a stego image to be undetectable, it should have a combination of smooth and non-smooth (complex) regions.

A cover selection technique for hiding a secret image in a cover image based on image texture similarity, by replacing some blocks of a cover image with similar secret image
blocks was identified and then, the indices of secret image blocks were stored in the cover image. In this cover selection method, the blocks of the secret image were compared with the blocks of a set of cover images and the image with most similar blocks to those of the secret image were selected as the best candidate to carry the secret image (Kermani and Jamzad, 2005). Kharazzi et al., (2005) studied the cover selection problem in detail by investigating three scenarios, in which the embedder had either no knowledge, partial knowledge, or complete knowledge of the steganalysis method.

Most researchers have concluded that capacity, robustness and imperceptibility are some of the crucial performance parameters of steganographic systems. Capacity is defined as the amount of data that can be hidden in the cover medium. Robustness is the ability of the stego medium to withstand manipulations. Imperceptibility is a measure of the difficulty in detecting the existence of hidden information in any media (Johnson et al., 2001; Zhang et al., 2007; Jianhong et al., 2009; Rura et al., 2011; Nag et al., 2012; Reddy et al., 2012). There is always a trade-off between the capacity and robustness characteristics of a data hiding technique. It is not possible to attain high robustness to signal modifications and high insertion capacity at the same time (Zhang and Wang, 2005).

Provos et al., (2001) explained that any classical steganographic system uses a two-step process. In the first step it identifies the redundant bits in a chosen cover medium. The second step uses all or a subset of the redundant bits to be replaced with the secret data bits. Image steganography uses two files. The first is the innocuous image that will hold the hidden information, called the cover image and the second is the secret message or
the information to be hidden. A message may be text, encrypted-text, other images, or anything that can be embedded in a bit stream. When combined, the cover image and the embedded message form a stego-image (Juneja and Sandhu, 2013).

Wayner (2002) dedicated a complete chapter in his book to explain the benefits of data embedding in noise. He clarified that such embedding is bound to be robust to cropping, compression and various image processing attacks.

A genetic algorithm based method which generates a stego-image by artificially counterfeiting statistical features of the image was proposed by Wu and Shih (2006). Time complexity, which is usually the limitation of genetic based algorithms, was not taken into consideration. They mentioned that the process was repeated until a predefined condition was satisfied or a constant number of iterations were reached. The predefined condition was the situation when they could correctly extract the desired hidden message. It was also not stated whether the process of determining such a condition was done automatically or involved a human inference.

With the September 11, 2001 World trade centre attack, the United States government authorized law enforcement agencies to monitor and intercept the electronic communications of suspected terrorists. This change in policy may have forced terrorists and their allies to search for alternative covert communication methods. Goel et al., (2007) investigated whether terrorists could be exploiting steganography and cryptography on the web for covert communication purposes. However, no definite conclusion could be drawn.
An analysis of steganography tools as carried out by Ming et al., (2006) broadly classified five methods for data hiding. The spatial domain methods hide data by embedding it in the LSB (least significant bit) plane. The secret data is embedded by either using LSB replacement or LSB matching (Chan and Cheng, 2004; Bailey and Kurran, 2006; Huang et al., 2007; Sajedi and Jamzad, 2010; Islam et al., 2014). It is seen that LSB embedding affects the relativity of the pixels in the LSB plane and thus produces some visual marks. However these methods are widely used, due to their simplicity and high data embedding capacity. The main limitation of the simple, least significant bit steganography, is that it allows limited data to be hidden (up to (1/8) of the size of the cover image). The four-LSB algorithm is an extension of the basis of the least significant bit algorithm. It uses the four LSB’s of a byte and replaces them with the payload information. However, this algorithm was prone to visual attacks and also produced a low PSNR value (Rosanne, 2010).

In the transform domain methods, the images are first transformed into the frequency domain and then the secret data is embedded in the transformed coefficients (Chan and Cheng, 2004; Sajedi and Jamzad, 2010). These methods are found to be highly imperceptible and robust but need intensive computations. Embedding data in document files using tabs or spaces is referred to as document based steganography. Inserting secret data in the unused bits of the carrier medium such as the file headers or marker segments is named structural embedding. Spread spectrum techniques spread the embedded data’s energy to a wide frequency band. This makes the hidden information difficult to detect and delete (Ming et al., 2006). Spread spectrum techniques are thus found to be robust but have a very limited data embedding capacity. Further, Cheddad (2009) carried out a comprehensive study on the various steganography techniques in
varying domains and documented his comparisons. He reported that the spatial domain techniques can handle large payloads, but offset the statistical properties of the cover medium. Besides, such techniques are not robust to compression and various image processing manipulations such as cropping, rotation and translation (Rifa et al., 2009), neither are they robust against noise. Spatial domain techniques mainly work with BMP formats. Considering the transform domain techniques, he reported that they are less prone to attacks, at the expense of providing limited embedding capacity. He also observed that they mainly work with the JPEG format and are prone to second order statistics attacks. They too were not robust to the common image processing functionalities such as cropping, rotation and translation. Cheddad (2009) then proposed an object oriented data embedding algorithm and reported that it provided a small embedding region at the cost of robustness. He also added that the proposed algorithm was robust to lossy compression and various image processing functions. He also reported that there were no known statistical vulnerabilities.

2.2.1 Steganography in Spatial domain

The most common spatial domain based steganography methods include, the Least Significant Bit (LSB) technique and the Bit Plane Complexity Steganographic (BPCS) technique. LSB steganography wherein the LSBs of the cover image are replaced by the MSBs of the secret message, is a widely used technique with low computational complexity and high insertion capacity (W. Bender et al., 1996; J Fridrich et al., 2003; Zhang et al., 2007; Cheddad et al., 2010). A steganographer embeds the secret data by modifying the $n$ least significant bits of a pixel of the cover image. The challenge is to find a maximal $n$-value to allow higher data embedding capacities, yet maintaining an acceptable imperceptibility level of the stego image. The value of an acceptable $n$ is
strongly influenced by the position and the value of the pixel in the cover image. Pixels in areas where there is a drastic change in intensity can tolerate high amount of change (Tataru et al., 2012). Cheddad et al., (2010) surveyed various steganography techniques in the spatial domain and reported how they basically involve encoding at the level of LSB’s. They experimentally demonstrated and clarified how, more the number of LSB’s that were used for embedding; more was the distortion in the embedded cover image. They concluded that there was a trade-off between the embedding payload capacity and the cover image distortion.

Many variations of basic LSB method were proposed by different researchers. The difference between LSB substitution and LSB matching was beautifully described by Cheddad et al. (2010). In simple LSB substitution the n- LSB’S of some or all of the bytes inside an image are changed to a bit of the secret message. LSB matching acts as a modified form of LSB steganography. Here, if the LSB of the cover image pixel matches with that of secret bit, then it is left unchanged, otherwise, it is added or subtracted by one, at random. BPCS steganography, hides secret data by means of block replacing. Each image plane is segmented into the same size pixel-blocks which are classified into informative and noise like blocks. The noise blocks were replaced by the secret blocks. The advantage of LSB, is its simplicity to embed the bits of the message directly into the LSB plane of cover image. Although LSB techniques have good perceptual transparency, wherein minor modifications made to the cover-image cannot be traced by the human eye they are vulnerable to steganalysis, which is based on statistical analysis. Therefore developing new LSB steganography algorithms against statistical analysis seems to have much meaning (Zhang et al., 2007; Desoky et al., 2008).
Marcal and Pereira (2005) thoroughly studied the Regular Singular pixel (RS) attack and proposed a method robust to RS steganalysis, that made the presence of secret data unnoticeable. The method was based on the application of reversible histogram transformation functions to the image, before and after embedding the secret data. The method was tested on greyscale images, with data embedding rates of 10%, 30% and 90%. The proposed method proved to be effective in eluding RS steganalysis.

Fernandes and Jeberson (2013) proposed an LSB embedding approach using 3-3-2 and 3-3-3 bit combinations in the red, green and blue planes respectively. The first approach achieved a PSNR of about 45 db, while the 3-3-3 bit combination approach helped to increase the embedding capacity, maintaining a PSNR of 35 db. Another high capacity steganographic method using division arithmetic and generalized exploiting modification direction method was presented by Kuo et al., (2014). They demonstrated how this technique was not susceptible to bitplane analysis attack.

2.2.2 Steganography in Temporal domain

The weak resistance of spatial domain methods to certain statistical attacks together with the need for enhanced security had forced researchers to search for methods in the temporal domain (Cheddad et al., 2010).

In temporal domain, at the outset, the cover image is transformed into frequency domain using transforms such as Fast Fourier Transform (FFT), Discrete Cosine Transform (DCT), Discrete Wavelet Transform (DWT) or the Integer Wavelet Transform (IWT). The LSB’s of the transformed coefficients of the cover image are then replaced by the MSB’s of the secret message. These methods are found to be highly imperceptible and
robust but need intensive computations. Different image steganography methods in the frequency domain have been proposed in literature. Embedding using various steganography methods like F5 (Westfeld, 2001), YASS (Solanki et al., 2007), and Contourlet-based steganography (Sajedi and Jamzad, 2008), is done by modifications of carefully selected coefficients in the transform domain.

Wayner (2002) argued that the coefficients in JPEG compression were normally plotted as a bell curve and the embedding of secret data within these coefficients using JSteg distorted the curve. Outguess was a better alternative to JSteg, as it used a pseudo random number generator to select the DCT coefficients for embedding. The Chi Square test could not detect data that was randomly embedded. The F5 embedding algorithm developed by Westfeld (2001), used subtraction and the matrix embedding operations. F5 embeds only into non-zero AC DCT coefficients by decreasing the absolute value of the coefficients by one. Fridrich et al., (2007) claimed that a shrinkage occurs when the same bit has to be re-embedded in case the original coefficient is either ‘1’ or ‘-1’ as at the decoding phase all zero coefficients will be skipped whether they were modified or not. Unfortunately Fridrich et al., (2003, 2007) discovered an attack that could detect the hidden contents, questioning F5’s survival. Yet another Steganography technique, named perturbed quantization, modifying the quantization table was proposed by Fridrich et al., (2005) which aimed to achieve high imperceptibility with minimal distortion. In this method each coefficient in the DCT block was assigned a scalar value that corresponded to how much impact it would make to the cover image and then a steganographer could set a selection rule to filter out the “well behaved coefficients” so as to carry out the data embedding.
According to Raja et al., (2005) steganography methods using the Fast Fourier transform (FFT) resulted in round-off errors; thus rendering them unsuitable for secure communication. However (Johnson and Jajodia, 1998) had thought differently and had explored the use of fast fourier transforms in steganography.

Cheddad et al., (2010) surveyed various steganography techniques in the frequency domain and reported how DCT is used extensively with video and image compression. They explained how each block of DCT coefficients was quantized using a specific quantization table (QT). The aim of quantization was to loosen up the tightened precision produced by DCT while retaining the valuable information descriptors. Most temporal domain techniques used JPEG images to embed data. Choosing which values in the DCT coefficient blocks were altered was very crucial, as altering any single coefficient would affect the entire 64 block pixels. The JSteg algorithm was among the first algorithms to work with JPEG images. It overcame visual attacks; however examining the statistical distribution of the DCT coefficients, indicated the existence of hidden data. JSteg was easily detected using the $X^2$ (Chi square) test (Provos and Honeyman, 2003).

Sajedi and Jamzad, (2010) argued that it is possible to increase the embedding capacity of cover images and yet decrease the detection risk. They introduced a boosted steganography scheme that pre-processed the cover image before the secret data was being embedded within. Applying contrast enhancement methods and Successive Mean Quantization Transform (SMQT) enhancement (Nilsson et al., 2005) to cover images before data embedding, helps amplify the details of an image by introducing more variation in image intensities and thus makes them more suitable to host secret data. This
was the principle behind the work proposed by Sajedi. Their experiments showed that the common steganalyzers were unable to detect such stego images.

**Mukta Goel and Rohit Goel (2013)** described how applying a discrete wavelet transform (DWT) to a two dimensional image decomposed the image into an approximation band (LL), horizontal band (HL), vertical band (LH), and a diagonal band (HH) of detailed components. They explained how the significant part of the image were seen in the approximation band while the edge and texture details were found in high frequency wavelet coefficients bands such as HH, HL, and LH (**Suresh Babu, 2008**). As for steganography in the discrete wavelet transform, a lot of work was presently being done, however it still appears to be in its infancy. **Abdel Wahab and Hassan (2008)** proposed a data hiding technique wherein both the secret image and the cover image were first decomposed using the DWT. Each of the blocks of the secret image was then compared with the cover image blocks to determine the most suitable match. Error blocks were also generated and embedded into coefficients of the best matched blocks in the HL of the cover image. However it was noted that the extracted payload was not exactly identical to the embedded payload. **Raftari et al., (2012)** in their paper proposed an image steganography technique that combined the Integer Wavelet Transform (IWT) and Discrete Cosine Transform (DCT) and embedded the secret image in the frequency domain of cover image, by using Munkres assignment algorithm. This technique was found to be robust against various image processing attacks, but computation intensive.
2.3 Edge based Steganography

The steganography domain is growing up very rapidly. A lot of mathematical papers and practical trials are published every day. Some of these papers are used in an attempt to get a new data hiding technique or to develop on existing techniques.

Hempstalk (2006) used the simple LSB substitution method and attempted to hide data within the corners of an image. However the embedding could easily be detected by statistical attacks, like the Chi-square and Sample pair value analysis (Westfeld and Pfitzmann, 1999; Fridrich et al., 2001). The concept of adaptive data hiding in edge regions of images using Spatial LSB Domain Systems was proposed by Yang et al., (2008). They observed that edges, portraying a variation in intensity level, were a good choice to conceal secret data. They concluded that depending on the size of secret data, the edge regions for embedding could be adaptively released. Li et al., (2009) proposed a colour image steganography approach based on Sobel edge detection operator in which because of the strong relevance in gradient among the Red, Green, and Blue planes, the corresponding LSB (Least Significant Bit) of pixel values to hide the secret data in other planes is modified. Finally, the stego-planes are combined and the colour image restored. Multi-times embedding was adapted to obtain high data capacity. Experimental results proved that there was no noticeable degradation after even three times use of hiding process and the average PSNR was about 45db with a data capacity of 6.3bpp.

Cheddad et al., (2010) reviewed numerous embedding techniques and documented that edge embedding is an excellent means of hiding data, maintaining good imperceptibility and robust to many attacks. A LSB matching based image steganography method along with an edge adaptive scheme which could select the embedding regions according to the
size of secret message and the difference between two consecutive pixels in the cover image was presented by Luo et al., (2010). For lower embedding rates, only sharper edge regions were used while keeping the other smoother regions intact. When the embedding rate increased, more edge regions were released adaptively for data hiding, by adjusting a few parameters. Results were good but needed human intervention as parameters had to be adjusted every time, based on the size of the secret data.

Optimal edge detection can be carried out using a combination of Canny and fuzzy edge detectors, Hussain et al., (2010). They reported that based on these edge computations, they used the LSB substitution to embed the hidden data. They also identified and recorded that modification of edge pixels of stego-image object, did not regenerate those identical edges as in cover-image, because stego-image edge pixels were drifted in context of original edges of cover-image objects. Their proposed method embeds the data around the edge boundaries of objects to consider retaining the edge boundaries identical as in cover-image. Weiqi et al., (2010) proposed an edge adaptive embedding scheme which selected the embedding regions according to the size of the secret image and the difference between two consecutive pixels in the cover image. Their method proved to be highly secure and also claimed to preserve the quality of the stego image. The limitations of various least significant bit based steganography schemes were analyzed by Sivaranjani and Sara (2011) and they proposed an edge adaptive scheme, applied to the least significant bit matching revisited (LSBMR) method. Based on the size of the secret data, the embedding regions in the greyscale cover image were being adaptively released. The proposed scheme was found to produce stego images of better visual quality when compared with the normal LSBMR method. In (2012)Teja et al., proposed an algorithm that concealed the secret data within the edge pixels of an image,
extending the least significant bit embedding algorithm. The random pixels used for data insertion were generated using the Fibonacci algorithm. However, such an embedding was easily detected.

Steganography using 2k correction method and edge detection method was described by Amanpreet Kaur and Sumeet Kaur (2012). The technique had the capability of carrying more payloads with better imperceptibility; achieved by embedding more data in edge areas as compared to smooth areas of the image, as human eye cannot detect the distortion at edges easily. The authors claim that the proposed algorithm produced better PSNR values when compared with other methods used in steganography. An image steganography technique based on the Canny edge detection algorithm was presented by Youssef Bassil (2012). It was designed to conceal secret data into a digital image within the pixels that make up the boundaries of objects detected in the image. Bits of the secret data replaced the three LSBs of every colour channel of the pixels detected by the Canny edge detection algorithm. The technique specified three crucial parameters: size of the Gaussian filter, a low threshold value, and a high threshold value, which determined the embedding capacity and the efficiency of the proposed method. The method achieved good imperceptibility.

Diverse steganographic algorithms in the spatial domain were scrutinized by Jamdar et al., (2013) and they reported that the choice of embedding positions within a cover image was mainly found to depend on a pseudorandom number generator, without considering the relationship between the image content itself and the size of the secret message. Thus the smooth regions in the cover images would inevitably be contaminated after data hiding even at a low embedding rate, and this lead to a poor visual quality and
low security. They proposed an edge adaptive scheme which selected the embedding region according to the size of the secret message and the difference between two consecutive pixels in the grayscale cover image. For lower embedding rates, only sharper edge regions were used. When the embedding rate increased, more edge regions were released adaptively for data hiding by altering a few parameters. (Geetha et al., 2013) proposed a data hiding scheme using multiple edge detection based variable embedding and the minimum error replacement method. Yet another method was proposed that had 3 main characteristics of randomness, pixel dependency and key dependency. The scheme used secret key random number generation using chaotic logistic map for random LSB substitution based on cover image pixels edges (Shahzad et al., 2014). The results obtained were satisfying.

Islam et al., (2014) proposed and elaborated on a steganography technique using edges of an 8-bit grayscale cover image to embed data. The number and type of edges to be used depended on the amount of data to be embedded, i.e., the more the amount of secret data to be hidden, the more the use of weaker edges for embedding. They altered the threshold value of the edge detection algorithm to obtain the desired number of edges. He clarified that the edge pixels, due to a change in coefficient gradient had intensities that were either higher or lower than the neighboring pixels, which made edges a good option for embedding data as a small distortion would go unnoticed. The secret message was embedded into the edges of the cover image in a random order using an edge map and a stego key. At the receiver end, to ease the extraction of the hidden message, the length of the message was prefixed to the secret message. The threshold value and the width of the Gaussian kernel used were embedded in some non-edge pixels of the cover image and sent across. Embedding was done by least two significant-bit substitutions
(2LSB). It meant that the intensity of each pixel held two bits of message. Their analysis of the experimental results has shown that the proposed technique performed well and claimed to provide high embedding capacity. However it was designed only for 8 bit images.

2.4 Image compression

Data compression has been an area of active research for decades. Most studies in image compression are aimed at achieving a good compression ratio, as well as improving the quality of the reconstructed image for better visualization. There is a strong interest in developing data encoding and decoding algorithms that can obtain higher compression ratios and faster transmission of data while keeping image quality to an acceptable level.

(Ventura et al., 2006; Tadrat and Boonjing, 2008; Begum and Venkataramani, 2012) have clarified how data compression, reduces the redundancy in data representation and thus helps in achieving a better data storage capacity. They also explained how compression helps to reduce communication costs. For efficient data transmission, the redundant information should be removed from the signal prior to transmission. Compression algorithm is what reduces the irrelevance and redundancy of data representation and decreases the data storage capacity. Compression is achieved by the removal of one or more of the three basic data redundancies: Coding Redundancy, Interpixel Redundancy and Psycho-visual Redundancy. Coding redundancy is present when less than optimal code words are used. Interpixel redundancy results from correlations between the pixels of an image. Psycho-visual redundancy is due to data that is ignored by the human visual system (i.e. visually non-essential information). Image compression techniques reduce the number of bits required to represent an image by
taking advantage of these redundancies. An inverse process called decompression (decoding) is applied to the compressed data to get the reconstructed image (Gonzalez and Woods, 2002). The two main components of a compression process were outlined as an encoding algorithm and a decoding algorithm. Compression algorithms were classified into two categories: lossless algorithms, wherein the original image could be easily reconstructed from the compressed image; and the lossy compression where it was difficult to exactly reconstruct the original image. Research in the domain of lossless data compression is gaining wide popularity specifically in the applications of medical imaging (Vikal et al., 2012). Various encoding algorithms such as such as Huffman encoding, the Lempel-Ziv encoding, arithmetic encoding, Dynamic Markov Compression (DMC), Run length coding (RLE) and Burrows-Wheeler Transform (BWT) based algorithms are in use today; however they are unable to achieve the theoretical best case compression ratio. The performance of a compression algorithm can be measured using measures such as compression ratio, bits per character and signal to noise ratio (Huffman, 1952; Francisco et al., 2010; Begum and Venkataramani, 2012).

Pattanaik et al., (2006) proposed a lossless image compression algorithm using the arithmetic modulo operation. Results when compared with lossy JPEG compression seemed to be quite good.

JPEG is an image compression standard used for storing images in a compressed format. JPEG format is quite popular and is used in a number of devices such as digital cameras and is also the format of choice, when exchanging large sized images, in a bandwidth constrained environment, such as the Internet. The remarkable quality of JPEG is that it achieves high compression ratios using the discrete cosine transformation with little loss
in quality (Youssef et al., 2012). The Inverse Discrete Cosine Transform can be used to retrieve the image from its transform representation. DCT is simple but produces noticeable blocking artifacts across the block boundaries for higher compression ratios. DCT is fast and can be quickly calculated and is best for images with smooth edges. A JPEG image may undergo further losses if it is frequently edited and then saved (Diwvedi, 2009). The operation of decompression and recompression may further degrade the quality of the image. To remedy this, the image should be edited and saved in a lossless format and only converted to JPEG format just before final transmittal to the desired medium. This ensures minimum losses due to frequent saving. It is not suited for images with many edges and sharp variations as this can lead to many artifacts in the compressed image. In these situations it is best to use lossless formats such as PNG, TIFF or GIF.

The wavelet transform has emerged as a cutting edge technology, within the field of image compression. Wavelet-based coding provides substantial improvements in picture quality at higher compression ratios. The wavelet transform has successfully been employed for signal/image compression (Kutil and Engel, 2008). The DWT has taken over DCT (Subhasis Saha) due to its ability to solve the problem of blocking artefacts introduced during DCT compression. It also reduces the correlation between the neighbouring pixels and gives multi scale sparse representation of the image. Wavelet compression provides excellent results in terms of rate-distortion compression. The problem is to find the optimal packet length for all code blocks which minimizes the overall distortion in a way that the generated target bit rate equals the demanded bit rate. The new ISO/ITU-T standard for still image coding, JPEG 2000, is a wavelet-based compression algorithm (Rehna and Jeva, 2012). Wavelets are mathematical functions
that can be used to transform one function representation into another. Wavelets as
defined by Daubechies (1992, 1993), possess three important characteristics: Multi-
resolution ability, Ability of generating lower level coefficients from the higher level
coefficients and ability of generating a wavelet system from a single scaling function or
wavelet function, just by applying the scaling and translating operations. Wavelet
transform-based applications involve discrete coefficients instead of scaling or wavelet
functions. For practical and computational reasons, discrete time filter banks are
required. Such structures decompose a signal into a coarse representation along with
added details. Discrete Wavelet Transform (DWT), transforms discrete signals from the
time domain into time frequency domain. The transformation product is a set of
coefficient organized in the way that enables not only spectrum analysis of the signal but
also spectral behaviour of the signal in time. When DWT is applied to an image, it is
decomposed into 4 sub bands: LL, HL, LH and HH. LL region contains the most
significant features. A sub-band is a set of coefficients, real numbers which represent
aspects of the image associated with a certain frequency range as well as a spatial area of
the image. After the wavelet transform, the coefficients are scalar-quantized to reduce
the amount of bits to represent them, at the expense of a loss of quality. As a result of
quantization, many of the higher frequency components are rounded to zero, and many
of the rest become small positive or negative numbers, which take many fewer bits to
store. The output is a set of integer numbers which have to be encoded bit-by-bit. The
parameter that can be changed to set the final quality is the quantization step: the greater
the step, the greater is the compression and the loss of quality. It is argued by various
Scholars that if information is hidden in the LL region, the stego image can withstand
compression or other manipulations (Mallat, 1989; Antonini et al., 1992; Khalifa, 2003). Wavelet transform is applied to entire images rather than sub-images, so
it produces no blocking artefacts. This is a major advantage of wavelet compression over other transform compression methods. Compared to DCT, DWT has less computational cost (Khalifa et al., 2005). It is notable that Wavelet compression does require more computational power than DCT-based compression.

A variant of the Discrete wavelet transform (DWT) for image compression was investigated by Yang et al., (2007). They defined the use of 2D Dual-tree Discrete Wavelet Transform (DDWT), which had direction-selective basis functions. To further sparsify DDWT coefficients, they presented an iterative projection based noise shaping method. They further analysed the statistics of DDWT coefficients as well as the inter-scale, inter-subband and intra-subband dependency among the DDWT coefficients. They also evaluated the application of Set Partitioning in Hierarchical Trees (SPIHT) and Embedded block coding with optimized Truncation (EBCOT) schemes for coding the DDWT coefficients (Said and Pearlman, 1996; Taubman, 2000). Experimental results proved that SPIHT was more effective than EBCOT for DDWT, and the DDWT-SPIHT coder outperformed JPEG2000 at low bit rates and was comparable to JPEG2000 at high bit rates.

Use of wavelets for data compression helped achieve higher compression ratios than the JPEG or GIF methods for some types of images as reviewed by Blessie et al., (2011). They reaffirmed that modifying data using a wavelet transformation, preserves good quality with little perceptual artefacts. Singh and Sharma (2012) designed an image compression method using a combination of DCT, DWT and Huffman encoding. Experiments proved that the hybrid algorithm performed better than the standalone DCT and DWT based compression algorithms.
A scheme for semantic image compression was implemented by Zhang and Zhang (2014). A compressed image was produced by embedding the calculated estimation errors into the compact image created by a compressor, using data hiding techniques.

Fernandes and Jeberson (2014) addressed the problem of use of an appropriate wavelet filter for efficient data compression. The wavelet filters tested were Haar, Antonini 7/9 and Daubechies 8. Performance measures such as Peak signal to noise ratio, Structural similarity index, Mean square error were used to evaluate the performance of the wavelet filters. Experimental results claimed that there was a minor variation in the peak signal to noise ratio values with the Haar filter having the highest value. However the Antonini filter performed better in terms of perceived visual quality.

Quantifying errors and image quality is an important task after a compression process explains Rehna et al., (2012). Two of the common error metrics used to compare image compression techniques are the mean square error (MSE) and the Peak Signal to Noise Ratio (PSNR). MSE refers to the average value of the square of the error between the original signal and the reconstructed signal, as explained in chapter 3 of this thesis. The important parameter that specifies the quality of the reconstruction is the peak signal-to-noise ratio (PSNR), expressed in decibels. Higher values of PSNR produce better image compression because it means that the ratio of Signal to Noise is higher. 'signal' refers to the original image, and 'noise' refers to the error in reconstruction. So, a compression scheme having a lower MSE (and a high PSNR), can be recognized as a better one (Babu et al., 2008; Ishaque and Sattar, 2009).
2.5 Edge detection

Edges characterize boundaries and play a significant role in image processing. Image boundary detection refers to extracting the pixels of an image where there is a high intensity variation. Edge detection techniques are classified as gradient based, laplacian based and optimal edge detection techniques. The gradient method detects the edges by looking for the maximum and minimum in the first derivative of the image. Some of the commonly used gradient operators include Roberts Operator, Sobel Operator and Prewitt Operator. The Laplacian method searches for zero crossings in the second derivative of the image to find edges. An edge has the one-dimensional shape of a ramp and calculating the derivative of the image can highlight its location. Second Order Derivative includes Laplacian, Laplacian of Gaussian, and Difference of Gaussian operators (Maini and Agarwal, 2011; Fernandes and Jeberson, 2013).

Any edge detection algorithms works by reducing noise which acts like fake edges, then carries out an edge enhancement to strengthen edges and suppress the non-edge pixels and finally edge localization by applying a threshold to decide which pixels are edges and which are not. Many classical edge operators such as Sobel, Prewitt, Laplacian and Gaussian operators are already available in literature. However they have their own limitations. Robert's mask permits a good detection of edges on diagonals. Its implementation is very easy and its computation cost is low.

Its main drawback is its extreme sensitivity to noise and poor response to the little marked edges. Prewitt and Sobel operators belong to the family of the directional edge detectors privileging the horizontal and vertical direction edges. They are not very expensive in terms of processing time and are easy to implement. However the edge smoothing carried out by
them results in thick edges. The Laplacian method is found to be more precise, providing more edge related details than the gradient based edge detectors. In addition, it is not directional and therefore edges will be detected independently of their orientations.

A computational approach to detect the edges of an image was outlined by John Canny (1986). He outlined some criteria relevant to an edge detectors performance. He argued that in edge detection, it is important that edges should not be missed and at the same time there should be no spurious responses. He also stated that the distance between the points marked by the edge detector and the center of the actual edges should be minimized. He also added a third criterion to circumvent the responses to multiple edges. To summarize, Canny defined detection and localization criteria for edges using a mathematical representation and finally used a method called feature synthesis for integration of the process. The method used adaptive thresholding with hysteresis to eliminate streaking of edge contours. The thresholds were set according to the amount of noise in the image. The Gaussian operator used was isotropic and therefore has a smoothening effect on the image in all directions, blurring sharp boundaries. Canny algorithm has the advantages of good SNR (Signal-to-Nose Ratio), location accuracy and edge thinning. However, the traditional Canny algorithm was used in grey images. There are also other problems of the traditional Canny algorithm, such as how to reach a balance between noise removal and edge preservation. Traditional Canny algorithm uses fixed upper and lower thresholds to extract image edges, so it lacks self-adaptivity to different images (Green B.). Canny operator also has the limitation of being vulnerable to several noise disturbances.
Xu et al., (2006) analysed and understood the shortcomings of the various existing edge detection techniques and proposed a new technique based on the Susan filter and embedded confidence. Experimental results indicated that this algorithm was robust to noise and accurate in the edge detection process.

An Improved Canny Edge Detection Algorithm based on Predisposal Method for Image Corrupted by Gaussian Noise was presented by Xiao et al., (2008). The two predisposal steps were gray value distance judgment and edge points correlation coefficient comparison. The result showed that the proposed method was much more reliable under the corruption of Gaussian noise environment. Wang and Fan (2009) carried out an analysis of the traditional Canny algorithm and proposed an improved version using a self-adaptive filter instead of the Gaussian filter. They also used the morphological thinning operator to obtain fine edges. Implementation indicated fair results.

In (2010), Zhang et al., proposed a novel cell based method for edge extraction using the Canny algorithm and some morphological operators. This method achieved results that were found to be more superior to the Canny algorithm. An improved version of Canny edge detection algorithm and an edge preservation filtering procedure specifically for pavement edge detection applications was introduced by Zhao et al., (2010). The proposed algorithm utilized the Mallat wavelet transform to reinforce the weak edge of input images and quadratic optimization using genetic algorithm to get a proper threshold during Canny algorithm steps. The presented algorithm eliminated noises effectively and also protected the unclear edges.
Exploring wavelets, Liang and Chen (2011) proposed an edge detection method based on the canny algorithm and wavelet space entropy. The edge detection was found to be good even in the presence of noise. Taghizadeh and Mahzoun (2011) conducted a study on global thresholding techniques and observed that most of the methods did not have an ability to exploit information of the characteristics of an image that they threshold. They proposed an approach based on edge detection algorithms for more accurate object segmentation. It was seen that the proposed method performed well in the presence of noise.

A modified version of the traditional Canny edge detection algorithm to determine edges of a coloured image was proposed by Xin et al., (2012). The proposed algorithm used a quaternion weighted average filter, vector Sobel gradient computation and non-maxima suppression based on interpolation.

Fernandes and Jeberson (2013) documented a detailed study, comparison and implementation of various edge detection algorithms such as Roberts’s operator, Laplacian of Gaussian, Canny edge detection and Shen castan. A modified Canny edge detection algorithm was proposed using a filter in the wavelet domain. The proposed canny algorithm was found to produce a more detailed edge map than the simple canny algorithm. It achieved reasonably good edge connectivity and a better feature localization. However, the modified algorithm was relatively slow and the detected edges were a little coarse.
2.6 Steganalysis

The exchange of ever-growing volumes of data through the Internet, and the widespread access to steganography software prompted the development of steganalysis tools (Marcal and Pereira, 2005).

Steganalysis refers to the art of identifying the existence of hidden data within a medium. It is also defined as the science of attacking steganography in a battle that never ends (Cheddad et al., 2010). The invasive nature of steganography leaves detectable traces within the stego image. This allows an adversary to use steganalysis techniques to reveal, if a secret communication is taking place (Chiew, 2011). There are two types of steganalysis: targeted and blind. Targeted steganalysis is designed to attack a particular embedding algorithm. Targeted steganalysis produces more accurate results, but normally fails if the embedding algorithm used is not the target. Blind steganalysis can be considered a universal technique for detecting different types of steganography. Because blind steganalysis can detect a wider class of steganographic techniques, it is generally less accurate; however, blind steganalysis can detect new steganographic techniques where there is no targeted steganalysis available. Different embedding techniques are thought to produce different changes in image characteristics. In other words, the characteristics of cover and stego images differ, and those resulting from different stego images (stego images produced by different embedding techniques) differ as well.

In steganalysis an image may be analysed by exposing it to different image processing methods such as image filtering, rotation, cropping, and translation. It is also achieved by coding a program that examines the stego-image structure and measures its statistical
properties. First order statistics measured using histograms or second order statistics measured by finding correlations between pixels, distance and direction can be used. Various steganalysis tools are also in existence (Westfeld and Pfitzmann, 1999; Fridrich et al., 2003; Ker, 2005).

Johnson and Jajodia (1998) reported that two aspects of attacks on steganography are detection and destruction of the embedded message. Any image can be manipulated with the intent of destroying some hidden information whether an embedded message exists or not. Detecting an embedded message defeats the primary goal of steganography, which is that of concealing the very existence of a hidden message (Geeta et al., 2007).

The aim of a steganalyst is to sense the existence of hidden data within a medium. Steganalyzers have taken great progress in the past few years and a number of powerful steganalysis techniques have been proposed. The core of each steganalyzer is a classifier, which given an image feature vector and decides whether the image contains any secret data. A classification task usually consists of train dataset and test dataset, which consist of some data instances. Each instance in the train set contains one class label and several features. The objective of Support Vector Machine (SVM) is to produce a model, which predicts class label of data instances in the test set (Sajedi and Jamzad, 2010).

Passive steganalysis methods assume that the copy of the original cover image is not available. Therefore, it should be difficult to find out if a pre-processing method has been applied on a cover image (Sajedi and Jamzad, 2010). Passive steganalysis attempts to destroy any trace of secret communication, without bothering to detect the hidden secret data, by using any image processing methods that could include changing the image format, flipping all LSBs or by carrying out a lossy compression. Active steganalysis
involves detecting the existence of hidden data within a cover medium (Cheddad et al., 2010)

Spatial steganography generates unusual patterns which leave traces to be picked up by steganalysis tools. LSB embedding is highly sensitive to any kind of filtering or manipulation of the stego-image. Image processing operations such as scaling, cropping, rotation, addition of noise, or lossy compression to the stego-image is very likely to destroy the hidden data. Almost any filtering process will alter the values of many of the LSBs (Anderson and Peticolas, 1998). Besides spatial domain based methods are prone to the Chi-square($\chi^2$) and the Pair-analysis attacks. Chi-square is a non-parametric statistical algorithm used to detect whether the intensity levels scatter in a uniform distribution through-out the image surface or not (Civiciooglu et al., 2004). A Chi-Square test takes into consideration the correlations between neighbouring parts of an image to compute the probability that a message is embedded for a certain assumed hidden message length (Amin et al., 2007). The classical Chi-square algorithm can be fooled by randomly embedded messages, thus Bohne and Westfeld (2005) developed a steganalysis method to detect randomly scattered hidden data in the LSB spatial domain that applies the preserving statistical properties (PSP) algorithm. Fridrich and Goljan (2001) proposed a statistical method that used higher-order statistics called RS steganalysis; it is designed to provide an estimated percentage of flipped pixels caused by embedding. Cancelli et al.,(2008) outlined that the performance of current state-of-the-art steganalysis algorithms are highly sensitive to the used training and testing databases. They therefore conclude that no single steganalysis algorithm can be termed as constantly superior.
In (2011), Chiew studied and documented the details of the chi square attack. He observed that the LSB embedding algorithm that overwrote the pixel LSBs did not change the grand total frequencies of pixel intensities. Only the frequencies of occurrences were swapped between the intensities. To summarize, when embedding occurs, the frequencies of occurrence for odd pixel intensities are transferred to the corresponding even pixel intensities and vice versa. These frequencies of odd-even pixel intensities are called pairs of values (PoV). This change involves swapping the frequencies of occurrence within each PoV and the sums of the frequencies in every PoV remain the same. If the message bits are uniformly distributed the frequencies of the intensities in each PoV will become identical after embedding. The chi-square test measures the degree of similarity between the observed sample distribution and the expected frequency distribution. The observed sample distribution is obtained from the given image distribution. The expected frequency distribution is computed from the arithmetic mean of the PoV’s. The $\chi^2$ attack can estimate the length of an embedded message as long as the message is embedded sequentially. However, the attack is unable to provide reliable detection if the message bits are randomly embedded in the image (Fridrich et al., 2003).

In the frequency domain, (Pevny and Fridrich, 2007) presented a JPEG steganalysis system that comprised of DCT features and calibrated Markov features, which were merged to produce a 274-dimensional feature vector. This vector is fed into, a Support Vector Machine multi-classifier capable of detecting the presence of model-based steganography, F5, OutGuess, Steghide and Hide&Seek. Li et al., (2008) exposed some of the weaknesses in YASS (Yet another Steganography Scheme) algorithm (Solanki et al., 2007), by noticing that it introduces extra zero coefficients into the embedded host
blocks because of the use of a quantization index modulation (QIM) method and by contrasting statistical features derived from different blocks in the stego-image.

**Westfeld (2003)** proposed a hidden data detector for colour images. It considered the count of neighbour colours. A colour pixel in a compressed cover image on an average has only 4 or 5 neighbours. A stego image achieved by LSB embedding may result in up to 7 neighbours colours for every colour in the image. Some steganographic techniques increment or decrement LSB, producing up to 26 neighbours for each modified pixel. This could be detected using a neighbourhood histogram (**Forczmański and Węgrzyn, 2009**). However, the proposed algorithm showed poor performance for grey scale images and uncompressed colour images.

A method to detect the LSB matching steganography on uncompressed grey scale images was proposed by **Huang et al., (2007)**. They explained that in LSB matching, pairs of values did not exist in the stego image, since the least two or more significant bit-planes of the cover image would be changed during the embedding. The detection rate was good, but also resulted in a large number of false positives. **Niu et al., (2009)** described a steganalysis algorithm for two least significant bits steganography. They used a local masked estimation function and estimated an approximate original image. Then they constructed a weighted stego image and finally formulated a detection equation to determine the hidden message length.

**Pevny et al., (2008; 2009; 2010)** presented a systematic method for detection of steganographic methods that embed in the spatial domain by adding a low-amplitude independent stego signal, an example of which is least significant bit (LSB) matching. First, arguments were provided for modeling the differences between adjacent pixels
using first-order and second-order Markov chains. Subsets of sample transition probability matrices were then used as features for a steganalyzer implemented by support vector machines. The major part of experiments, performed on four diverse image databases, focused on evaluation of detection of LSB matching. The comparison to prior art, revealed that the presented feature set offered superior accuracy in detecting LSB matching.

An edge adaptive steganalysis algorithm based on histograms of the absolute difference of adjacent pixels, to detect embedding done by the Edge Adaptive Least significant bit Matching Revisited (EALMR) algorithm was proposed by Edris et al., (2014). EALMR is a spatial domain information hiding technique which can choose embedding area based on the image context. The authors compared the performance of the proposed algorithm with five other steganalysis algorithms. Results demonstrated that the proposed analytical algorithm based on histogram of absolute difference of adjacent pixels could effectively detect EALMR.

Non-structural detectors use feature extractors to model cover image and to compute distortion between the cover and the stego image to detect embedding. A classifier is trained by the feature set from large number of stego and cover images. During training, the classifier learns the differences in features, and this learning is used to classify a fresh image into stego or clean image (Islam et al., 2014).

2.7 Research Gap Analysis
The process of sending messages between two parties through a public channel, in such a way that it deceives the adversary from realising the existence of the communication is
known as steganography. Many interesting steganographic techniques have been created and its continuing evolution is guaranteed by a growing need for information security. In today’s digital era, steganography has spawned a wide range of thought-provoking and beneficial applications (Cheddad et al., 2010). Inevitably, it is potentially open to abuse and can be used by criminals and terrorists (Chiew, 2011). An article from USA Today stated that steganography was used by terrorists for the world trade centre attack, although there was little evidence to substantiate this claim.

With the complexities in Steganography techniques and the simultaneous progressive strength of steganalysis algorithms, it has become a challenge to develop data hiding techniques with much better performance. Embedding capacity of an image may be different using different steganography methods. Therefore, each steganography method can embed the secret data in an image until the distortion of image features does not threaten to be detected by visual inspection or by steganalysis algorithms. A robust stego object, is one that is inconspicuous to both human and computerised analysis. It can be observed from the literature review carried out in the field of steganography that most work is being done in the area of embedding text messages into grayscale images or audio or video. It is also seen that there is a trade-off between robustness of the method and the payload that can be embedded. Steganography methods usually struggle with achieving a high embedding rate.

A few areas that need to be stressed and focussed on by researchers are:

- Embedding secret data into 24-bit coloured images.
- Embedding secret images within a cover medium.
- Considering lossless compressed file formats for data embedding.
Selecting appropriate pixels within a cover image so as to aid in the imperceptibility criteria.

Storing the secret data indirectly, using an optimized storage logic.

In the present electronic communication scenario and with the advent of cheap digital cameras, coloured images/pictures are being constantly exchanged by communicators. Keeping in mind the time and bandwidth demands, it is very much necessary to transmit images in a compressed form.

Benjamin Franklin once said “Three people can keep a secret....... only if.................... Two of them are dead !!

How very true! Can we use these compressed images to hide our personal secret data? Can we make the stego-images perceptually similar to the original cover image so as to make it undetectable, along with an increase in the data hiding capacity?

Therefore, there is a need to develop robust steganographic techniques using coloured images as the cover medium and the secret data. The proposed techniques have been designed keeping in mind the above requirements.
MATERIALS AND METHODS

This chapter elaborates on the various hardware / software tools used, experimental procedures followed, data beds used and a thorough explanation of the newly outlined algorithms for hiding sensitive data within images.

3.1 Equipment/ Tools Used

The equipment used to carry out this investigation were Nikon Coolpix P600 digital camera, HP laptop and a Pentium IV, 1.7 GHz desktop machine.

3.1.1 Camera Specifications

3.1.1.1 Overview:

The Nikon Coolpix P600 is a 16 mega pixel advanced point and shoot camera from Nikon India private limited. It has a lens which supports good zoom abilities. This camera also has vibration reduction features for better quality photography. It comes with a practical display screen which makes it easy to find the best angle for photos. The camera has the picture control options to help set preferences according to the subject and the situation.

3.1.1.2 Lens

The camera has a 60x optical zoom NIKKOR lens that delivers sharp telephoto shots, and a 120x Dynamic Fine Zoom which aids to capture minute details. The lens offers 60x optical zoom from 24 mm wide-angle to 1440 mm telephoto, and using Dynamic Fine Zoom provides amazing 120x zoom capability without compromising the high resolution.
3.1.1.3 Vibration Reduction

An enhanced lens-shift Vibration Reduction delivers unparalleled sharpness even at full zoom. This gives the flexibility of a slower shutter speed even at 60x zoom. This camera is equipped with ACTIVE mode to further minimize camera shake, when shooting from a moving vehicle or while walking and thus ensuring blur-free images.

3.1.1.4 Sensor Technology

The Nikon Coolpix P600 camera employs a CMOS image sensor. The CMOS image sensor used, has backside illumination which delivers high-quality images even when it is dark. The Nikon lens uses Super ED glass for a substantial improvement in chromatic aberration reduction.

3.1.1.5 Focus Technology

The Nikon Coolpix P600 has a fast auto focus and shutter response. This allows clicking photographs immediately when an opportunity arises. Rapid action can be clearly captured using Continuous H (in maximum image size at 7 fps, up to 7 frames) mode.

3.1.1.6 Picture Processing Technology

Nikon's exclusive Active D-Lighting function, offers the ability to preserve details in the highlights and shadowy areas of images, shot in high-contrast scenes such as those that are backlit. The camera enables customization prior to shooting by adjusting vividness of colors, contrast, and image sharpness depending on the type of subject, scene or intent. The technical specifications of the camera are as shown in table 3.1.Also available at
3.1.2 Desktop Configurations

The desktop system used was an OEM system made by Samsung. The system had a Pentium IV processor with a 22 inch LED monitor for better color experience. The system details are specified in table 3.2.

Table 3.1 Technical Specifications of Nikon P600 digital camera

<table>
<thead>
<tr>
<th>Specification</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of effective pixels</td>
<td>16.1 million</td>
</tr>
<tr>
<td>Image sensor</td>
<td>1/2.3-in. type CMOS; approx. 16.76 million total pixels</td>
</tr>
<tr>
<td>Lens</td>
<td>NIKKOR lens with 60x optical zoom</td>
</tr>
<tr>
<td>Focal length</td>
<td>4.3-258 mm (angle of view equivalent to that of 24-1440 mm lens in 35mm [135] format)</td>
</tr>
<tr>
<td>f/number</td>
<td>f/3.3-6.5</td>
</tr>
<tr>
<td>Construction</td>
<td>16 elements in 11 groups (4 ED lens elements and 1 super ED lens element)</td>
</tr>
<tr>
<td>Digital zoom magnification</td>
<td>Up to 4x (angle of view equivalent to that of approx. 5760 mm lens in 35mm [135] format)</td>
</tr>
<tr>
<td>Vibration reduction</td>
<td>Lens shift</td>
</tr>
<tr>
<td>Motion blur reduction</td>
<td>Motion detection (still pictures)</td>
</tr>
<tr>
<td>Autofocus (AF)</td>
<td>Contrast-detect AF</td>
</tr>
<tr>
<td>Focus range</td>
<td>[W]: Approx. 50 cm (1 ft 8 in.) to infinity, [T]: Approx. 2.0 m (6 ft 7 in.) to infinity Macro close-up mode: Approx. 1 cm (0.4 in.) (at a wide-angle zoom position) to infinity (All distances measured from center of front surface of lens)</td>
</tr>
<tr>
<td>Focus-area selection</td>
<td>Target finding AF, face priority, manual (spot), manual (normal), manual (wide), subject tracking</td>
</tr>
<tr>
<td>Shooting Modes</td>
<td>Auto, Scene (Night landscape, Night portrait, Landscape, Scene auto selector, Portrait, Sports, Party/indoor, Beach, Snow, Sunset, Dusk/dawn, Close-up, Food, Museum, Fireworks show, Black and white copy, Backlighting, Easy panorama, Pet portrait, Moon, Bird-watching), Special Effects, P, S, A, M, User settings</td>
</tr>
<tr>
<td>----------------</td>
<td>---------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Continuous Shooting</td>
<td>Single (default setting), Continuous H (the frame rate for continuous shooting is about 7 fps and the maximum number of continuous shots is about 7), Continuous L (the frame rate for continuous shooting is about 1 fps and the maximum number of continuous shots is about 200), Pre-shooting cache (the frame rate for continuous shooting is about 15 fps and the maximum number of continuous shots is 20, including a maximum of 5 frames captured in the pre-shooting cache), Continuous H:120 fps (the frame rate for continuous shooting is about 120 fps and the maximum number of continuous shots is 60), Continuous H:60 fps (the frame rate for continuous shooting is about 60 fps and the maximum number of continuous shots is 60), BSS (Best Shot Selector), Multi-shot 16, Intvl timer shooting</td>
</tr>
<tr>
<td>ISO sensitivity (Standard output sensitivity)</td>
<td>ISO 100-1600 ISO 3200, 6400 (available when using P, S, A or M mode) Hi 1 (equivalent to ISO 12800) (available when using High ISO monochrome in special effects mode)</td>
</tr>
<tr>
<td>Exposure Metering mode</td>
<td>Matrix, center-weighted, or spot</td>
</tr>
<tr>
<td>Exposure control</td>
<td>Programmed auto exposure with flexible program, shutter-priority auto, aperture-priority auto, manual, exposure bracketing enabled, exposure compensation (-2.0 EV - +2.0 EV in steps of 1/3 EV) enabled</td>
</tr>
<tr>
<td>Shutter</td>
<td>Mechanical and CMOS electronic shutter</td>
</tr>
<tr>
<td>Speed</td>
<td>1/4000 * -1 s 1/4000 * -15 s (when ISO sensitivity is 100 in M mode) * When the aperture value is set to f/7.6 (wide-angle end)</td>
</tr>
<tr>
<td>Aperture</td>
<td>Electronically-controlled 6-blade iris diaphragm</td>
</tr>
<tr>
<td>Range</td>
<td>8 steps of 1/3 EV (W) (A, M mode)</td>
</tr>
</tbody>
</table>
Table 3.2 Technical Configurations of Desktop

<table>
<thead>
<tr>
<th>Microprocessor</th>
<th>Intel Pentium IV 1.7 GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory</td>
<td>1 GB DDR RAM</td>
</tr>
<tr>
<td>Video</td>
<td>NVDIA GeForce AGP card</td>
</tr>
<tr>
<td>Video Memory</td>
<td>256 MB</td>
</tr>
<tr>
<td>Hard Disk</td>
<td>160 GB IDE</td>
</tr>
<tr>
<td>Multimedia drive</td>
<td>56x multi read CD-ROM drive</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows XP 32 bit with Service pack 3</td>
</tr>
<tr>
<td>Chipset</td>
<td>VIA IDE chipset</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Samsung PS2 multimedia 104 key keyboard</td>
</tr>
<tr>
<td>Mouse</td>
<td>PS 2 compatible mouse</td>
</tr>
<tr>
<td>LAN</td>
<td>VIA Rhine 10/100 fast Ethernet adaptor</td>
</tr>
<tr>
<td>Other Devices</td>
<td>Two USB 2.0 ports</td>
</tr>
<tr>
<td></td>
<td>VIA audio chipset</td>
</tr>
<tr>
<td></td>
<td>VGA port</td>
</tr>
</tbody>
</table>

3.1.3 Laptop Configuration

The laptop used was HP dv7-6178us, manufactured by Hewlett-Packard. The laptop had an I7 processor with a 17.3 inch LED monitor with ATI Radeon graphics for optimum color production and system performance. The laptop configuration details are listed in table 3.3.

Table 3.3 Specifications of HP laptop

<table>
<thead>
<tr>
<th>Product Name</th>
<th>dv7-6178us</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microprocessor</td>
<td>2nd generation Intel Core i7-2630QM Processor 2.00GHz with Turbo Boost Technology up to 2.90 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>6GB DDR3 System Memory</td>
</tr>
<tr>
<td>Video Graphics</td>
<td>Radeon HD 6490M s</td>
</tr>
<tr>
<td>Hard Drive</td>
<td>750GB (5400RPM)</td>
</tr>
<tr>
<td>Multimedia Drive</td>
<td>SuperMulti DVD Burner</td>
</tr>
<tr>
<td>Display</td>
<td>17.3” High Definition+ HP BrightView LED Display (1600 x 900)</td>
</tr>
<tr>
<td>Network Card</td>
<td>Integrated 10/100/1000 Gigabit Ethernet LAN</td>
</tr>
<tr>
<td>Wireless Connectivity</td>
<td>• 802.11b/g/n WLAN with WiDi (36)</td>
</tr>
<tr>
<td>Sound</td>
<td>• Beats Audio with the HP Triple Bass Reflex Subwoofer</td>
</tr>
<tr>
<td>Keyboard</td>
<td>Full-Size Island-Style Keyboard with Integrated Numeric Keypad</td>
</tr>
<tr>
<td>Pointing Device</td>
<td>Touchpad supporting Multi-Touch gestures. With LED border accent light and On/Off button.</td>
</tr>
<tr>
<td>External Ports</td>
<td>• Digital Media Card Reader for Secure Digital and Multimedia cards</td>
</tr>
<tr>
<td></td>
<td>• 2 SuperSpeed USB 3.0</td>
</tr>
<tr>
<td></td>
<td>• 2 Universal Serial Bus (USB) 2.0</td>
</tr>
<tr>
<td></td>
<td>• 1 HDMI</td>
</tr>
<tr>
<td></td>
<td>• 1 VGA (15-pin)</td>
</tr>
<tr>
<td></td>
<td>• 1 RJ -45 (LAN)</td>
</tr>
<tr>
<td></td>
<td>• 2 Headphone-out</td>
</tr>
<tr>
<td></td>
<td>• 1 Microphone-in</td>
</tr>
<tr>
<td>Other Devices</td>
<td>• HP TrueVision HD Webcam with integrated digital microphone</td>
</tr>
<tr>
<td></td>
<td>• HP SimplePass with integrated fingerprint reader</td>
</tr>
</tbody>
</table>
3.2 Development Environment

3.2.1 Processor Requirement
Since the application deals with a lot of complex computational requirements, it demands a processor with good computational abilities. Hence a minimum of a Pentium IV with a 1.7 GHz processor would be required.

3.2.2 Disk space Requirement
This application deals with a lot of high resolution images which in the uncompressed form would require a lot of space. A minimum requirement of 500 MB would be a very reasonable amount of disk space required.

3.2.3 Memory Requirement
This Steganography application involves a lot of computations while processing the images as well as processing the data. A requirement of 1 GB would be a good minimum requirement.

3.3 Softwares/ Platform Used

3.3.1. Microsoft Visual Studio
Visual Studio is a complete set of development tools for building ASP.NET Web applications, XML Web Services, desktop applications, and mobile applications. Visual Basic, Visual C#, and Visual C++ , all use the same integrated development environment (IDE) from Microsoft, which enables, tool sharing and eases the creation of mixed-language solutions. In addition, these languages use the functionality of the .NET Framework, which provides access to key technologies that simplify the development of ASP Web applications and XML Web Services. Support for other languages such as
Python and Ruby, among others is available via language services installed separately. It also supports XML, HTML/XHTML, JavaScript and CSS.

### 3.3.2 Matlab 7.0

Matlab (Matrix Laboratory) is a high-performance technical computing language and interactive environment, used for algorithm development, data visualization, data analysis and numeric computation. It provides tools to acquire, analyse, and visualize data, in a very easy to use friendly manner. MATLAB features a family of application-specific solutions called toolboxes. Toolboxes are comprehensive collections of MATLAB functions (M-files) that extend the MATLAB environment to solve particular classes of problems. Areas in which toolboxes are available include signal processing, control systems, neural networks, fuzzy logic, wavelets, simulation, and many others.

**Key Features**

- High-level language for numerical computation, visualization, and application development.
- Interactive environment for modelling, design, simulation and prototyping.
- Easy to use mathematical functions for linear algebra, statistics, Fourier analysis, filtering, optimization, numerical integration, and solving differential equations.
- Built-in graphics and functions for data analysis, exploration, and visualization
- Tools for building applications with graphical user interfaces
- MATLAB based algorithms can be easily integrated with external applications and languages such as C, Java, .NET, and Microsoft Excel.
Matlab’s functionality can be greatly expanded by the addition of toolboxes.

There are also some limitations:

- It uses an enormous amount of memory and on slow computers it is very hard to use.
- It sits “on top” of Windows, getting as much CPU time as Windows allows it to have. This makes real-time applications very complicated.

3.3.3 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions aimed at real-time computer vision, developed by Intel Russia research centre. It is free for use under the open source BSD license. It is a cross-platform library that focuses mainly on real-time image processing. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive, though extensive older C interface. It also supports full interfaces in Python, Java and MATLAB.

Key features:

- Open source computer vision library in C/C++.
- Optimized and intended for real-time applications.
- OS/hardware/window-manager independent.
- Generic image/video loading, saving, and acquisition.
- Has both, low and high level Application Programming Interfaces (API).
• Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids).

• Various dynamic data structures (lists, queues, sets, trees, graphs).

3.3.4 CxImage library

CxImage is a C++ class, to manage almost any kind of images. It was written by David e Pizzolato. It can load, save, display and transform images in a very simple and fast way. CxImage is open source and licensed under the zlib license. With more than 200 functions, and with comprehensive working demos, CxImage offers all the tools to build simple image processing applications on a fast learning curve. Supported file formats are: BMP, GIF, JPG, JP2, PNG, TIF, RAW and many more.

CxImage is highly portable and has been tested with Visual C++ 6 / 2008, C++ Builder 3 / 6, MinGW on Windows, and with gcc 3.3.2 on Linux. The library can be linked statically, or through a DLL or an activeX component.

There are so many outstanding graphics libraries, such as OpenIL, FreeImage, PaintLib, however they are all license agreement bound. The glue to connect all the modules and the C libraries is CxFie; a virtual class that provides the standard methods to access the data from a file on the disk or in memory.
A CxImage object is basically a bitmap, with the addition of some member variables to store useful information:

```cpp
class CxImage
{
    ...
    protected:
    void* pDib; //contains the header, the palette, the pixels
    BITMAPINFOHEADER head; //standard header
    CXIMAGEINFO info; //extended information
    BYTE* pSelection; //selected region
    BYTE* pAlpha; //alpha channel
    CxImage** pLayers; //generic layers
}
```

A CxImage object is also a set of layers. The buffers in each layer are allocated only when necessary. CxImage::pDib is the background image. CxImage::pAlpha is the transparency layer. CxImage::pSelection is the selection layer, used to create regions of interest for image processing.

Over these 3 specific planes, other generic layers can be added, stored in CxImage::pLayers. The generic layers are full CxImage objects, so complex structures of
nested layers can be built. The whole library is quite big, in the configuration header file `ximacfg.h`; switches to enable or disable a specific graphic format or feature present. Each JPG, PNG and TIFF library adds about 100KB to the final application, while the CxImage impact is about 50KB. So it is important to support and link only the formats that the application really needs.

The `CxImgLib.dsw` workspace shows the libraries required to build an application (demo.exe), including almost all the features and the formats available in CxImage. All the libraries must be compiled before you can link the final application. CxImage has a good support for memory files, new methods and file formats, and it is more portable.

**Key features**

- Supports various image formats
- Supports various types of filters and thresholding operations.
- Supports various Colorspaces: RGB, HSL, CMYK, YUV, YIQ.

The various file format dependencies are tabulated in table 3.4.
### Table 3.4 File format dependencies in CxImage

<table>
<thead>
<tr>
<th>Formats</th>
<th>#define</th>
<th>required libraries</th>
<th>size [Kbyte]</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMP</td>
<td>CXIMAGE_SUPPORT_BMP</td>
<td>built in</td>
<td>24</td>
</tr>
<tr>
<td>GIF</td>
<td>CXIMAGE_SUPPORT_GIF</td>
<td>jpeg</td>
<td>88</td>
</tr>
<tr>
<td>ICO</td>
<td>CXIMAGE_SUPPORT_ICO</td>
<td>png, zlib</td>
<td>104</td>
</tr>
<tr>
<td>TGA</td>
<td>CXIMAGE_SUPPORT_TGA</td>
<td>mng, zlib, jpeg</td>
<td>148</td>
</tr>
<tr>
<td>PCX</td>
<td>CXIMAGE_SUPPORT_PCX</td>
<td>tiff, zlib, jpeg</td>
<td>124</td>
</tr>
<tr>
<td>WBMP</td>
<td>CXIMAGE_SUPPORT_WBMP</td>
<td>jbig</td>
<td>28</td>
</tr>
<tr>
<td>WMF</td>
<td>CXIMAGE_SUPPORT_WMF</td>
<td>jasper</td>
<td>176</td>
</tr>
<tr>
<td>JPEG</td>
<td>CXIMAGE_SUPPORT_JPG</td>
<td>jpeg</td>
<td>88</td>
</tr>
<tr>
<td>PNG</td>
<td>CXIMAGE_SUPPORT_PNG</td>
<td>png, zlib</td>
<td>104</td>
</tr>
<tr>
<td>MNG</td>
<td>CXIMAGE_SUPPORT_MNG</td>
<td>mng, zlib, jpeg</td>
<td>148</td>
</tr>
<tr>
<td>TIFF</td>
<td>CXIMAGE_SUPPORT_TIF</td>
<td>tiff, zlib, jpeg</td>
<td>124</td>
</tr>
<tr>
<td>JBIG</td>
<td>CXIMAGE_SUPPORT_JBG</td>
<td>jbig</td>
<td>28</td>
</tr>
<tr>
<td>PNM, PPM, PGM RAS</td>
<td>CXIMAGE_SUPPORT_PNM CXIMAGE_SUPPORT_RAS</td>
<td>jasper</td>
<td>176</td>
</tr>
<tr>
<td>JPEG-2000</td>
<td>CXIMAGE_SUPPORT_JP2 CXIMAGE_SUPPORT_JPC CXIMAGE_SUPPORT_PGX</td>
<td>jasper</td>
<td>176</td>
</tr>
</tbody>
</table>

### 3.3.5 VSL

Virtual Steganographic laboratory (VSL) is a free image steganography and steganalysis software, in the form of a graphical block diagramming tool. It allows testing and adjusting different steganographic techniques and provides simple GUI along with modular, plug-in architecture, available under the GNU General Public License. It is written in Java, so it is cross-platform software and it can be executed on any operating system and is written and maintained by Michal Węgrzyn. Goal of the application, is hiding data in digital images, detecting its presence and testing its robustness using any number of different adjustable techniques. The pre-requisite to run VSL is Java 1.5 (5.0)
or above. To run an experiment, modules have to be arranged in processing flows by connecting them. All flows must begin with an Input module. Loop functionality can be obtained by connecting modules. There can be more than one flow at a time. VSL can write and read images that are covered by Java. JAI (Java Advanced Imaging API) can be installed in order to be able to work with more image readers and writers. To create new modules one of the interfaces from vsl-commons library must be implemented. For example; if new steganographic method is to be created, it must implement Steganographic Technique interface. When new module jar is created, two files must be edited: etc\modules.xml and MANIFEST.MF, inside vsl-app jar. VSL is available at http://vsl.sourceforge.net.

Data can be hidden with basic Least Significant Bit (LSB) method, with more advanced Karhunen-Loeve Transform (KLT) technique or by F5 algorithm, which uses DCT transformation in JPEG files. For steganalysis two advanced techniques can be used. First, RS-Analysis: efficient steganalysis for LSB methods - and the second one - Binary Similarity Measures (BSM) method with Support Vector Machines (SVMs) classifier: blind steganalysis (universal) technique, which can be used to detect any kind of steganography. VSL contains also many other modules - several distortion techniques, which can be used to test resistance of steganographic technique. Program has built-in modules, which helps with research, reports, file handling, image analysis etc.

3.3.6 Simple Steganalysis Suite

Simple Steganalysis suite (SSS) is a simple java based tool, to perform image steganalysis. It has an implementation of the most famous visual attacks, described by Westfeld and Pfitzmann in their paper titled “Attacks on steganographic systems” (Westfeld and Pfitzmann, 1999).
SSS current features include:

- LSB Enhancement
- Chi-Square test
- Neighborhood histogram
- Pixel Difference histogram
- Difference Histogram Attack
- Primary Sets

3.4 Theoretical Concepts

3.4.1 Visual Perception

Visual perception, is the ability to interpret the surrounding environment by processing information that is contained in visible light using the sense organs. The resulting perception is also known as eyesight, sight, or vision. Each sense organ is part of a sensory system, which receives sensory inputs and transmits sensory information to the brain.

A sketch of the anatomical components of the human eye is shown in Figure 3.2. The main components are the iris, lens, pupil, cornea, retina, vitreous humor, optic disk and optic nerve.
The retina is made up of two types of cells namely rod cells and cone cells. These cells were named because of their shape as viewed in the microscope and are highly responsive to light. Max Schultze (1825-1874) discovered that the retinal cones are the color receptors of the eye, and the retinal rod cells, while not sensitive to color, are very sensitive to light at low levels. It is the cone cells that allow us to see in color. It is because cone cells remain un-stimulated in low light environments that we do not see color in dimly lit places. In the human eye, there are many more rod cells in the retina, than there are cone cells. A schematic drawing of rod and cone cells is shown in Figure 3.3. The cells are divided into two sections. The bottom portion is called the inner segment. It contains the nucleus and the synaptic ending. The synaptic ending attaches to the neurons which produce signals that go to the brain. The top portion is called the outer segment. The outer segment is comprised of a membrane which is folded into several layers of disks. The disks are comprised of cells that contain the molecules that absorb
the light. Pigments are also found in cone cells. There are three types of cone cells, each of which contains a visual pigment. These pigments are called the red, blue or green visual pigment. The cone cells detect the primary colors, and the brain mixes these colors in seemingly infinitely variable proportions, so that we can perceive a wide range of colors. The choice of primary colors is related to the physiology of the human eye. Good primary colors are stimuli that maximize the difference between the responses of the cone cells of the human retina, to light of different wavelengths, and that thereby makes a large color triangle.

The normal three kinds of light-sensitive photoreceptor cells in the human eye (cone cells) respond most to yellow (long wavelength or L), green (medium or M), and violet (short or S) light. The difference in the signals received from the three kinds allows the brain to differentiate a wide gamut of different colors, while being most sensitive (overall) to yellowish-green light and to differences between hues in the green-to-orange region.

As an example, suppose that light in the orange range of wavelengths enters the eye and strikes the retina. Light of these wavelengths would activate both the medium and long wavelength cones of the retina, but not equally; the long wavelength cells will respond more. The difference in the response can be detected by the brain, and this difference is the basis of our perception of orange. Thus, the orange appearance of an object results from light from the object entering our eye and stimulating the different cones simultaneously, but to different degrees.
3.4.2 RGB Colour model

The RGB color model is a convenient color model for computer graphics, because the human visual system works in a way that is very similar to an RGB color space. The RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors.

To form a color with RGB, three colored light beams (one red, one green, and one blue) are superimposed (for example by emission from a black screen, or by reflection from a white screen). Each of the three beams is called a component of that color, and each of them can have an arbitrary intensity, from fully off to fully on, in the mixture. The RGB
color model is additive, in the sense that the three light beams are added together, and their light spectra, add wavelength for wavelength, to make the final color's spectrum.

Zero intensity for each component gives the darkest color (no light, considered the black), and full intensity of each gives a white; the quality of this white depends on the nature of the primary light sources, but if they are properly balanced, the result is a neutral white, matching the system's white point. When the intensities for all the components are the same, the result is a shade of gray, darker or lighter depending on the intensity. When the intensities are different, the result is a colorized hue, more or less saturated, depending on the difference of the strongest and weakest of the intensities of the primary colors employed.

When one of the components has the strongest intensity, the color has a hue value near this primary color (reddish, greenish, or bluish), and when two components have the same strongest intensity, then the color is a hue of a secondary color (a shade of cyan, magenta or yellow). A secondary color is formed by the sum of two primary colors, of equal intensity: cyan is green+blue, magenta is red+blue, and yellow is red+green. Every secondary color is the complement of one primary color; when a primary and its complementary secondary color are added together, the result is white: cyan complements red, magenta complements green, and yellow complements blue.

The RGB color model itself does not define what is meant by red, green, and blue colorimetrically, and so the results of mixing them are not specified as absolute, but relative to the primary colors. When the exact chromaticities of the red, green, and blue primaries are defined, the color model then becomes an absolute color space. The RGB model uses a Cartesian system of coordinates to map the available colors to the volume of a cube.
The Cartesian system representation of the RGB model has the following appearance:

![Figure 3.4 Cartesian system representation of the RGB model](image)

The RGB color model has the following characteristics:

- The major axes of the cube are assigned to the three primary colors (red, blue, green) and their complements (magenta, yellow, and cyan).
- The origin of the coordinate system (0,0,0) is black. It corresponds to a total absence of color.
- The major diagonal of the cube extends from black (0,0,0) to white (255,255,255). This diagonal represents values that mix equivalent portions of the primary colors to produce gray scales.

3.5 Data beds Used

The proposed algorithms have been tested on 200 real time camera clicked digital images and on the UCID10K(Schaefer, 2010) and nevercompressed images databases having about 15150, 24 bit RGB images and yet another database having about 200, 24 bit PNG
images. The resolution of cover images varied from 256*256, 512*512 to 1024*1024 pixels. For the secret text data we used text files of varying sizes from 6 kb to 225 kb. For the secret images various grayscale and colored images were tested.

3.6 Proposed Methodology

3.6.1 Proposed Steganography Schemes

The proposed objective of this research was to develop a steganographic technique for covert communication over overt channels. A total of two techniques, in the spatial domain have been developed for concealing secret text as well as secret images in innocuous images. They have been named as Arithmetic division based data embedding (ADDE) and Bit Stream based data embedding (BSDE).

In a 24-bit bitmap, each pixel is represented by three bytes, representing the red, green and blue colour values for that pixel. The higher the number, the more intense is the colour for that pixel. In 24-bit bitmaps, the number of bytes per row is always end-padded with zeros to be a multiple of four. However using these extra bytes to hide data would be unwise, as these bytes are supposed to contain zeros and any alteration would be easily detectable. Thus, to ensure the image is inconspicuous, only the LSBs of the actual pixel data should be altered. Both of the proposed methods work on this principle and modify the LSB’s only. LSB embedding exploits the fact that the level of precision in many image formats is far greater than that perceivable by average human vision. Therefore, a modified image with slight variations in its colors, will be indistinguishable from the original by a human eye (Johnson and Jajodia, 1998). Embedding brings about distortion in the visual and statistical properties of an image. This in turn may aid in the identification of a stego image. The goal of any steganography technique is to safeguard these properties while carrying out data embedding (Islam et al., 2014). Edges in images
are areas with strong intensity contrasts, a jump in intensity from one pixel to the next. Thus the pixels in edges are found to be good candidates for data embedding, in comparison to the other smoother areas of the image. ADDE and BSDE use edges of a coloured image for the data hiding process.

**Features of proposed algorithms**

- The secret data is hidden inside the edges of the carrier image.
- Edges identified using Canny algorithm with modified thresholds.
- The data is hidden within a color pixel using the least two significant bits of all the three colour channels of a twenty four bit image.
- We use a combination of logical operations for hiding and also for retrieving the data. The main purpose of these operations is to replace the respective carrier image data bits with the secret data bits.
- We use bit masks to erase the respective bits which are not under consideration. This is done for both the carrier image as well as the secret data.
- Wavelets based image compression algorithm is used to reduce size of secret image data.
- Optimization criteria used to minimize bit storage.

**3.6.1.1 Arithmetic Division based Data Embedding (ADDE)**

Digital images are stored in computer systems as an array of points (pixels). The size of each pixel depends on the format of the image and normally ranges from 1 byte to 3 bytes. Each unique numerical pixel value corresponds to a color; thus, an 8-bit pixel is capable of displaying 256 different colors. A 24-bit color image has three color components corresponding to Red, Green and Blue. The three components are normally
quantized using 8 bits. Each pixel is represented with three bytes to indicate the intensity of these three colors (RGB). Each byte can have a value from 0 to 255, representing the intensity of the color. The darkest color value is 0 and the brightest is 255. Transparency is controlled by the addition of information to each element of the pixel data. A 24-bit pixel value can be stored in 32 bits. The extra 8 bits is for specifying transparency. This is known as the alpha channel. The Least Significant Bit insertion technique is a common, simple approach to embed information in a cover image. The LSB (the 8th bit) of each pixel, for a specific color channel or for all color channels is replaced with a bit of the secret message. For increasing the embedding capacity, two or more LSBs in each pixel can be used to embed messages. However, there is a trade-off between the embedding payload and the quality of the stego image (Cheddad et al., 2010). Embedding in the higher bitplanes, increases the embedding capacity, but adds a larger noise component to the value of each pixel, resulting in the distortion of an image. Embedding in the 4th LSB generates more visual distortion to the cover image as the hidden information is seen as “non-natural” (Cheddad et al., 2008).

The proposed technique works on 24 bit RGB images as cover images and has a data embedding capacity of maximum 6 bits per pixel. It applies a division operation and hides the secret data in the edges of a 24 bit cover image. In order to store the data onto the image, the data is considered to be a sequence of bytes. In an effort to ensure imperceptibility and good stego image quality, we consider the least two significant bits of all the three color channels, of a 24 bit image to store the data; However since using only the least 2 significant bits could mean a very limited storage capacity, we employ an optimization technique to minimize the size of the data bytes while maximizing the data stored. The optimization technique used is a two stage
process. In the first stage, we reduce the magnitude of the data using an arithmetic division operation. Each byte of data is stored in the form of quotient and remainder. The entire array of data is then represented in the form of quotient and remainder arrays. The quotient can have a maximum size of 5 bits and the remainder can be of 3 bits. This is because we choose the divisor to be 8. The value 8 was chosen as the divisor since we could reduce the storage size of the quotient from an 8 bit value to a maximum 5 bit value. In the second stage, we further reduce the size by implementing an optimization logic. After the arithmetic division operation, sizes of all the bytes of our quotient data are analysed. Based on the sizes of the quotient values, we decide the range of the bits to be used for storage. The range would form the optimization criterion. The number of bits to be used to store the quotient data is specified by a flag array. The combination of the optimization criterion and the flag array is said to optimize and specify the number of bits to be used for data storage. Since the number of bits used to store the data varies, this technique will also be resistant to statistical attacks. The choice of the cover image can be of any file format supported by the CxImage library, such as Windows bitmap format or PNG or JPG file format. However the output stego file would be saved in either the BMP or PNG lossless data formats.

The details of the optimization criterion and the flag value combinations along with the bit sizes signified, are mentioned in table 3.5
Table 3.5 Mapping criterion used in ADDE

<table>
<thead>
<tr>
<th>Optimization Criterion</th>
<th>Flag Value</th>
<th>Bit Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>R G B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0 1 0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0 1 1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1 0 0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1 0 1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1 1 0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1 1 1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td>5</td>
</tr>
</tbody>
</table>

The ADDE method data storage schematic diagram is as shown in figure 3.5 below. The horizontal wavy line represents the identified 24-bit edge pixels. The first edge pixel always stores the optimization criteria (3 bits) as derived from table 3.5. The second edge pixel stores the length of length field, i.e. – the number of bits used to store the length field. From the third edge pixel onwards begins the length, followed by the Flag array, quotient array and finally the remainder array.
3.6.1.1.1 Using ADDE With Text

The block diagram of the arithmetic division based data embedding for embedding secret text within an image is as shown in figure 3.6. The proposed ADDE data embedding algorithm as explained in figure 3.7 would take Cover image (C), Secret text data (M) and the length of Secret data (L) as inputs.

The generated stego image is sent over an overt channel to the recipient. The recipient then runs the data extraction algorithm as shown in figure 3.8 to extract the secret data.
Figure 3.6 Flowchart of text embedding using ADDE technique
Algorithm ADUE Text Embedding Algorithm

Inputs: C - Cover Image, M - Secret Text Data, I - Length Of Secret Data
Outputs: S - Stego Image

// Initialize the Stego image with the Cover image data
S ← C

// Initialize Bit Mask for 2 LSB and 1 LSB
LSBMASK ← 1111100
LSBIMASK ← 1111110

// Erase the 2 LSB of Cover image using Mask and Logical AND operation
C ← bitAND(C, LSBMASK)

// Determine Canny edge detection thresholds depending upon image characteristics
th, tl ← getThreshold(C)

// Use Canny Edge detection to create an edge map F.
E ← CannyEdge(C, th, tl)

// Create two arrays Q and R of quotient and remainder
// from the secret data using divisor S
Q ← M/S
R ← mod(M, S)

// Scan array Q and generate a flag array F with a mapping
// of the number of bits required per byte of Q
for every element Qj of Q and F1 of F do
    B ← GetNoOfBits(Qj)
    if B ≤ 4 then
        F1 ← 0
    else
        F1 ← 1
    end if
end for

// We store the Flag array into the stego image using the edge map
for every 3 elements F1, F2, F3 of F do
    P ← GetEdgePixel(S, E)
    RVALUE ← GetRedChannel(P)
    GVALUE ← GetGreenChannel(P)
    BVALUE ← GetBlueChannel(P)
    RENC ← bitAND(RVALUE, LSBIMASK)
    GENC ← bitAND(GVALUE, LSBIMASK)
    BENC ← bitAND(BVALUE, LSBIMASK)

    // Embed the flag array data
    RENC ← RENC  F1
    GENC ← GENC  F2
    BENC ← BENC  F3
end for
Algorithm ADDE Text Embedding Algorithm

// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)

// Set the pixel P in stego image S
SetPixel(S, P)
end for

// We store the quotient array Q into the stego image S using the bit pattern specified in array F
for every element Qi of Q do
F1 ← GetFlagElement(F)
P ← GetEdgePixel(S, E)
RVALUE ← GetRedChannel(P)
GVALUE ← GetGreenChannel(F)
BVALUE ← GetBlueChannel(P)
if F1 == 0 then
RENC ← bitAND(RVALUE, LSB1MASK)
GENC ← bitAND(GVALUE, LSB1MASK)
BENC ← bitAND(BVALUE, LSB1MASK)
RENC ← RENC[(Qi >> 2)]&LSB1MASK
GENC ← GENC[(Qi >> 1)]&LSB1MASK
BENC ← BENC[(Qi)]&LSB1MASK
else
RENC ← bitAND(RVALUE, LSB2MASK)
GENC ← bitAND(GVALUE, LSB2MASK)
BENC ← bitAND(BVALUE, LSB1MASK)
RENC ← RENC[(Qi >> 3)]&LSB2MASK
GENC ← GENC[(Qi >> 1)]&LSB2MASK
BENC ← BENC[(Qi)]&LSB2MASK
end if
// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)
// Set the pixel P in stego image S
SetPixel(S, P)
end for

// We store the remainder array R into the stego image S
for every element Ri of R do
P ← GetEdgePixel(S, E)
RVALUE ← GetRedChannel(P)
GVALUE ← GetGreenChannel(F)
BVALUE ← GetBlueChannel(P)
RENC ← bitAND(RVALUE, LSB1MASK)
GENC ← bitAND(GVALUE, LSB1MASK)
BENC ← bitAND(BVALUE, LSB1MASK)
RENC ← RENC[(Ri >> 2)]&LSB1MASK
GENC ← GENC[(Ri >> 1)]&LSB1MASK
BENC ← BENC[(Ri)]&LSB1MASK
// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)
// Set the pixel P in stego image S
SetPixel(S, P)
end for

Figure 3.7 ADDE text embedding algorithm
3.6.1.1.2 Using ADDE with Images

The arithmetic division based embedding algorithm was also implemented using an image as the secret data that had to be concealed within a cover image. The proposed technique first compresses the secret data (image) using a compression method in the wavelet domain, applies a division operation on the compressed data and then hides it, in the edges of the Image. Since images require large bandwidth, compression is useful to decrease the data storage capacity and thus reduce bandwidth. Here, secret image compression is achieved using the Antonini wavelet and arithmetic coding of the transformed coefficients. The flow chart is as shown in figure 3.9 and algorithm is as explained in figure 3.10. The secret image extraction algorithm is explained in figure 3.11.

Figure 3.8 ADDE text extraction algorithm
Figure 3.9 Flow chart of Image Embedding in ADDE
Algorithm ADDE Image Embedding Algorithm:

Inputs: \( C \) - Cover Image, \( M \) - Secret Image Data, \( L \) - Length Of Secret Image Data
Outputs: \( S \) - Stego Image

// Initialize the Stego image with the Cover image data
\( S \leftarrow C \)

// Initialize Bit Mask for 2 LSB and 1 LSB
\( \text{LSB2MASK} \leftarrow 1111100 \)
\( \text{LSB1MASK} \leftarrow 1111110 \)

// Erase the 2 LSB of Cover image using Mask and Logical AND operation
\( C \leftarrow \text{bitAND}(C, \text{LSB2MASK}) \)

// Determine Canny edge detection thresholds depending upon image characteristics
\( \text{th}, \text{it} \leftarrow \text{getThreshold}(C) \)

// Use Canny Edge detection to create an edge map \( E \)
\( E \leftarrow \text{CannyEdge}(C, \text{th}, \text{it}) \)

// Compress secret image using wavelet based compression algorithm
\( M \leftarrow \text{Compress}(M) \)

// Create two arrays \( Q \) and \( R \) of quotient and remainder
// from secret data using divisor \( 8 \)
\( Q \leftarrow M/8 \)
\( R \leftarrow \text{mod}(M, 8) \)

// Scan array \( Q \) and generate a flag array \( F \) with a mapping
// for the number of bits required per byte of \( Q \)
for every element \( Ql \) of \( Q \) and \( F1 \) of \( F \) do
\( B \leftarrow \text{GetNoOfBits}(Ql) \)
if \( B < 4 \) then
\( F1 \leftarrow 0 \)
else
\( F1 \leftarrow 1 \)
end if
end for

// We store the Flag array into the stego image using the edge map
for every 3 elements \( F1, F2, F3 \) of \( F \) do
\( P \leftarrow \text{GetEdgePixel}(S, E) \)
\( \text{RVALUE} \leftarrow \text{GetRedChannel}(P) \)
\( \text{GVALUE} \leftarrow \text{GetGreenChannel}(P) \)
\( \text{BVALUE} \leftarrow \text{GetBlueChannel}(P) \)
\( \text{BENC} \leftarrow \text{bitAND}(\text{RVALUE}, \text{LSB1MASK}) \)
\( \text{GENC} \leftarrow \text{bitAND}(\text{GVALUE}, \text{LSB1MASK}) \)
\( \text{BENC} \leftarrow \text{bitAND}(\text{BVALUE}, \text{LSB1MASK}) \)

// Embed the flag array data
\( \text{BENC} \leftarrow \text{BENC}\mid F1 \)
\( \text{GENC} \leftarrow \text{GENC}\mid F2 \)
\( \text{BENC} \leftarrow \text{BENC}\mid F3 \)

Algorithm continued on next page....
Algorithm ADDE Image Embedding Algorithm

// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)

// Set the pixel P in stego image S
SetPixel(S, P)
end for

// We store the quotient array Q into the stego image S using the bit pattern specified in array P
for every element Q1 of Q do
    F1 ← GetFlagElement(P)
P ← GetEdgePixel(S, E)
RVALUE ← GetRedChannel(P)
GVALUE ← GetGreenChannel(P)
BVALUE ← GetBlueChannel(P)

if F1 === 0 then
    RENC ← bitAND(RVALUE, LSB1MASK)
    GENC ← bitAND(GVALUE, LSB1MASK)
    BENC ← bitAND(BVALUE, LSB1MASK)
    RENC ← RENC(((Q1) >> 2) & LSB1MASK)
    GENC ← GENC(((Q1) >> 1) & LSB1MASK)
    BENC ← BENC((Q1) & LSB1MASK)
else
    RENC ← bitAND(RVALUE, LSB2MASK)
    GENC ← bitAND(GVALUE, LSB2MASK)
    BENC ← bitAND(BVALUE, LSB2MASK)
    RENC ← RENC(((Q1) >> 3) & LSB2MASK)
    GENC ← GENC(((Q1) >> 1) & LSB2MASK)
    BENC ← BENC((Q1) & LSB2MASK)
end if

// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)

// Set the pixel P in stego image S
SetPixel(S, P)
end for

// We store the remainder array R into the stego image S
for every element R1 of R do
    P ← GetEdgePixel(S, E)
    RVALUE ← GetRedChannel(P)
    GVALUE ← GetGreenChannel(P)
    BVALUE ← GetBlueChannel(P)
    RENC ← bitAND(RVALUE, LSB1MASK)
    GENC ← bitAND(GVALUE, LSB1MASK)
    BENC ← bitAND(BVALUE, LSB1MASK)
    RENC ← RENC(((R1) >> 2) & LSB1MASK)
    GENC ← GENC(((R1) >> 1) & LSB1MASK)
    BENC ← BENC((R1) & LSB1MASK)

// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)

// Set the pixel P in stego image S
SetPixel(S, P)
end for

Figure 3.10 ADDE image embedding algorithm
3.6.1.1.3 Mathematically understanding ADDE with an example

Consider the image skype.bmp to be hidden within LenaRGB.bmp.

An edge map of LenaRGB.bmp would first be created using the proposed modified Canny edge detection algorithm. Skype.bmp would then be compressed using the wavelet compression algorithm. The binary data representation would then be divided to view the byte data in the form of quotient and remainder. The quotients are then scanned to determine their minimum and maximum representation bit size. Accordingly, the
optimization criterion is selected from the ADDE mapping table. In our example, the minimum bit size was seen to be 3 and the maximum bit size was found to be 5. Thus the optimization criteria to be stored in the first edge pixel was RGB =111. The second edge pixel holds the number of bits required to store the length field. The third edge pixel onwards stores the actual length. Following the flag array is being filled. The first flag value to be stored was 1 and so on. Every edge pixel stored the flag for 3 elements. Then the quotient array is being filled. Let’s consider the first quotient element to be 30. The encoding and decoding of a value into an edge pixel is as explained below:

Combinations of logical operations are used for hiding and also for retrieving the data. The main purpose of these operations is to replace the respective carrier image data bits with the secret data bits. Bit masks are used to erase the respective bits which are not under consideration. This is done for both the carrier image as well as the secret data. The bit masks used are LSB1MASK = 11111100 and LSB2MASK = 00000011.

The pixel data of a twenty four bit image will be in the form of a triplet value of Red, Green and Blue. Let us denote this as (RVALUE, GVALUE, BVALUE). The final resultant pixel too will be in the form of a triplet value of Red, Green and Blue. Let us denote this as (RENC, GENC, BENC). Let us denote our data byte value to be stored as DATA.

Consider we have an initial data value of 240 to be hidden. We divide this value and represent it in terms of quotient and remainder. Considering 8 as the divisor, we obtain quotient = 30 and remainder = 0.

Now, consider we want to store the value of 30 (quotient) in a fully white pixel of quotient array.

For a white pixel, Pixel data: RVALUE = 255, GVALUE = 255, BVALUE =255
In binary: \(RVALUE = 11111111\), \(GVALUE = 11111111\), \(BVALUE = 11111111\)

\(DATA = (quotient\ value) = 30 = 011110\).

**Step I – Clear the last 2 LSB of an edge pixel (in quotient array)**

In our example we are considering a white pixel in quotient array to be replaced:

\[
\begin{align*}
RENC &= RVALUE \& LSB1MASK = 11111111 \& 11111100 = 11111100 \\
GENC &= GVALUE \& LSB1MASK = 11111111 \& 11111100 = 11111100 \\
BENC &= BVALUE \& LSB1MASK = 11111111 \& 11111100 = 11111100
\end{align*}
\]

**Step II – Encode the secret data (quotient value) into cleared value in quotient array.**

\[
\begin{align*}
RENC &= RENC \mid ((DATA >> 4) \& LSB2MASK) \\
&= 11111100 \mid ((011110 >> 4) \& 00000011) \\
&= 11111101 = 253 \text{ (In decimal)}. \\
GENC &= GENC \mid ((DATA >> 2) \& LSB2MASK) \\
&= 11111100 \mid ((011110 >> 2) \& 00000011) \\
&= 11111111 = 255 \text{ (In decimal)}. \\
BENC &= BENC \mid (DATA \& LSB2MASK) \\
&= 11111110 \mid (011110 \& 00000011) \\
&= 11111110 = 254 \text{ (In decimal)}
\end{align*}
\]

After encoding, we now have the secret data (quotient value) encoded as \(R = 253\), \(G = 255\), \(B = 254\) in decimal values.

In order to decode the encoded pixel data we first extract the data bits.
Applying the formula:

\[
R_{\text{DEC}} = \text{REN}_c \& \text{LSB2MASK} = 11111101 \& 00000011 = 00000001
\]

\[
G_{\text{DEC}} = \text{GEN}_c \& \text{LSB2MASK} = 11111111 \& 00000011 = 00000011
\]

\[
B_{\text{DEC}} = \text{BEN}_c \& \text{LSB2MASK} = 11111110 \& 00000011 = 00000010
\]

In order to retrieve the final data (quotient value), we apply the operation:

\[
\text{DATA} = (R_{\text{DEC}} \ll 4) \| (G_{\text{DEC}} \ll 2) \| (B_{\text{DEC}})
\]

\[
\text{DATA} = (00000001 \ll 4) \| (00000011 \ll 2) \| (00000010)
\]

\[
= 00011110
\]

\[
= 30 \text{ (In decimal)}.
\]

Thus the decoded data is the same as the encoded.

### 3.6.1.2 Bit Stream based Data Embedding (BSDE)

The first data embedding technique ADDE, utilized three arrays namely the quotient array, remainder array and the flag array in its implementation. After a lot of executions, it was noticed that a lot of storage space is required by these three arrays. The second technique BSDE uses another technique, in order to reduce the number of arrays and increase the secret data embedding capacity for a given cover image. In this bit stream based data embedding technique too, we consider a 24 bit bitmap image as the cover image. We use the 2 LSB’s of each color channel of the edges of the cover image to store the secret data in the form of a bit stream.

In the BSDE technique, we employ the 24 bit image data present within the edge pixels of the image. An edge map of the edges is created using the modified Canny edge detection technique. The last two least significant bits of all the three color channels of
red, green and blue of the edge pixels are used to store the data. Thus we achieve an embedding capacity of 6 bits per pixel.

In this method we use a flag array to give us the size of each byte of data. We use a stream of continuous bits to store the data. The flag array and the bit stream are both stored into the edge pixels of the carrier image using the last two LSB. A size mapping process is carried out for each byte of data. The bit stream is constructed by appending the data bytes one after another, after the size mapping operation is carried out. In order to optimize the data, the amount of bits used by each byte of data is mapped to a value set in the flag array. Table 3.6 mentions the size mapping used to optimize the storage.

<table>
<thead>
<tr>
<th>Flag Value</th>
<th>Size Used</th>
<th>Data Size Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1 - 3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>4 - 5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>6 - 8</td>
</tr>
</tbody>
</table>

With the help of the above table, the storage space is decided for each byte of data. We use this size to store the actual data. As can be seen from the table, if the data value is zero, no data is stored in the bit stream. The bit stream is stored as a continuous series of bits, using the last two LSB of the red, green and blue channels of the edge pixels of the
carrier image. When retrieving the original data, the flag array and also the bit stream are required. The flag array is used to obtain the number of bits of storage per byte of data. Accordingly the original data has to be reconstructed.

The BSDE method data storage schematic diagram is as shown in figure 3.12 below. The horizontal wavy line represents the identified 24-bit edge pixels. The first edge pixel always stores the length of length field, i.e. – the number of bits used to store the length field. From the second edge pixel onwards begins the length of secret data, followed by the flag array stream of data. In the BSDE method 2-bit flags are used as derived from table 3.6, and thus each edge pixel can store the flags for 3 elements. For the bit stream array, the last 2 lsb’s of every channel colour of every edge pixel is used to store data. Thus every pixel stores 6 bits of secret data.

![Figure 3.12 BSDE method data embedding schematic diagram](image)

### 3.6.1.2.1 Using BSDE with Text

The flowchart for BSDE text embedding is as shown in figure 3.13. The algorithm for the same is written in figure 3.14.
Figure 3.13 Flow Chart for BSDE Text Embedding Technique
Algorithm  BSDE Text Embedding Algorithm

Inputs : C – Cover Image, M – Secret Text Data, L – Length Of Secret Data
Outputs : S – Stego Image

// Initialize the Stego image with the Cover image data
S ← C

// Initialize Bit Mask for 2 LSB
LSB2MASK ← 11111100

// Erase the 2 LSB of the Cover image using the Mask and Logical AND operation
C ← blAND(C, LSB2MASK)

// Determine Canny edge detection thresholds depending upon image characteristics
th, tl ← getThreshold(C)

// Use Canny Edge detection to create an edge map E.
E ← CannyEdge(C, th, tl)

// Scan secret data M and generate a flag array F with the number of bits required per byte of M
for every byte M1 of M and every element F1 of F do
    N ← GetNoOfBits(M1)
    if N == 0 then
        F1 ← 0
    else if 6 ≤ N ≤ 8 then
        F1 ← 3
    else if 4 ≤ N ≤ 5 then
        F1 ← 2
    else if 3 ≤ N ≤ 1 then
        F1 ← 1
    end if
end for

// We store the Flag array into the stego image using the edge map
for every 3 elements F1,F2,F3 of F do
    P ← GetEdgePixel(S, E)
    BVALUE ← GetRedChannel(P)
    GVALUE ← GetGreenChannel(P)
    BVALUE ← GetBlueChannel(P)
    RENC ← blAND(BVALUE, LSB2MASK)
    GENC ← blAND(GVALUE, LSB2MASK)
    BENC ← blAND(BVALUE, LSB2MASK)

    // Embed the flag array data
    RENC ← RENC + F1
    GENC ← GENC + F2
    BENC ← BENC + F3

    // replace the color values in the current pixel
    ReplacePixelColor(P, RENC, GENC, BENC)
end for

// Set the pixel P in stego image S
SetPixel(S, P)

Algorithm continued on next page....
Algorithm BSDE Text Embedding Algorithm

// Generate a bit stream B by concatenating all the bits M1 to Mn of M
B ← M1M2M3......Mn

// We store the bit stream B into the stego image S
for every 6 bits B1 of B do
    P ← GetEdgePixel(S, E)
    RVALUE ← GetRedChannel(P)
    GVALUE ← GetGreenChannel(P)
    BVALUE ← GetBlueChannel(P)
    RENC ← bitAND(RVALUE, LSB2MASK)
    GENC ← bitAND(GVALUE, LSB2MASK)
    BENC ← bitAND(BVALUE, LSB2MASK)
    RENC ← RENC||((B1 >> 2)&!LSB2MASK)
    GENC ← GENC||((B1 >> 1)&!LSB2MASK)
    BENC ← BENC||B1&!LSB2MASK)

// Replace the color values in the current pixel
ReplacePixelColor(P,RENC,GENC,BENC)

// Set the pixel P in stego image S
SetPixel(S,P)
end for

Figure 3.14 BSDE Text Embedding Algorithm
The BSDE text extraction algorithm is explained in figure 3.15.

Figure 3. 15 BSDE Text Extraction Algorithm
3.6.1.2.2 Using BSDE With Images

The bit stream based embedding algorithm was also implemented using an image as the secret data that had to be concealed within a cover image. This proposed technique first compresses the secret data (image) using a compression method in the wavelet domain. The compressed data is then transformed into a flag array and a stream of bits containing the data. The two arrays are then hidden in the edges of the Image. Since images require large bandwidth, compression is useful to decrease the secret data size and increase the data storage capacity and thus reduce bandwidth. Here, secret image compression is achieved using quantization and arithmetic coding of the transformed coefficients.

The steps are illustrated in the following flow chart, figure 3.16. Figure 3.17 describes the algorithm for secret image embedding using the BSDE technique. The reverse BSDE based secret image extraction process is explained in figure 3.18.
Figure 3.16 Flow Chart for BSDE Image embedding Technique
Algorithm  BSDE Image Embedding Algorithm

Inputs: C – Cover Image, M – Secret Text Data, L – Length Of Secret Data
Outputs: S – Stego Image

// Initialize the Stego image with the Cover image data
S ← C

// Initialize Bit Mask for 2 LSB
LSB2MASK ← 11111100

// Erase the 2 LSB of the Cover image using the Mask and Logical AND operation
C ← bitAND(C, LSB2MASK)

// Determine Canny edge detection thresholds depending upon image characteristics
th, tl ← getThreshold(C)

// Use Canny Edge detection to create an edge map E.
E ← CannyEdge(C, th, tl)

// Compress secret image using wavelet based compression algorithm
M ← Compress(M)

// Som secret data M and generate a flag array F with the number of bits required per byte of M
for each byte M1 of M and every element F1 of F do
    N ← GetNOfBits(M1)
    if N = 0 then
        F1 ← 0
    else if 6 ≤ N ≤ 8 then
        F1 ← 3
    else if 4 ≤ N ≤ 5 then
        F1 ← 2
    else if 3 ≤ N ≤ 1 then
        F1 ← 1
    end if
end for

// We store the flag array into the stego image using the edge map
for every 3 elements F1,F2,F3 of F do
    P ← GetEdgePixel(S, E)
    BV ALUE ← GetRedChannel(P)
    GV ALUE ← GetGreenChannel(P)
    RV ALUE ← GetRedChannel(P)
    RENC ← bitAND(BV ALUE, LSB2MASK)
    GENC ← bitAND(GV ALUE, LSB2MASK)
    BENC ← bitAND(RV ALUE, LSB2MASK)

// Embed the flag array data
RENC ← RENC|F1
GENC ← GENC|F2
BENC ← BENC|F3

// replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)

// Set the pixel P in stego image S
SetPixel(S,P)
end for

Algorithm continued on next page...
Algorithm  BSDE Image Embedding Algorithm

// Generate a bit stream B by concatenating all the bits M1 to Mn of M
B ← M1M2M3......Mn

// We store the bit stream B into the stego image S
for every 6 bits B1 of B do
    P ← GetEdgePixel(S, E)
    RVALER ← GetRedChannel(P)
    GVALER ← GetGreenChannel(P)
    BVALER ← GetBlueChannel(P)
    RENC ← bitAND(RVALER, LSB2MASK)
    GENC ← bitAND(GVALER, LSB2MASK)
    BENC ← bitAND(BVALER, LSB2MASK)
    RENC ← RENC||(B1 >> 2)&!LSB2MASK
    GENC ← GENC||(B1 >> 1)&!LSB2MASK
    BENC ← BENC||(B1&!LSB2MASK)

// Replace the color values in the current pixel
ReplacePixelColor(P, RENC, GENC, BENC)

// Set the pixel P in stego image S
SetPixel(S, P)
end for

Figure 3.17BSDE Image Embedding Algorithm
Algorithm BSDE Image Extraction Algorithm

Inputs : S - Stego Image
Outputs : M - Secret Text Data

// Initialize Bit Mask for 2 LSB
LSB2MASK ← 11111100

// Erase the 2 LSB of the Stego image using the Mask and Logical AND operation
// S ← bitAND(S, LSB2MASK)

// Determine Canny edge detection thresholds depending upon image characteristics
// th, tl ← getThreshold(S)

// Use Canny Edge detection to determine edge map E.
// E ← CannyEdge(S, th, tl)

// Read the flag array F from edge pixels of stego image
// F ← GetFlagArray(S)

// set the correct length values in F
for every element F1 of F do
    if F1 == 0 then
        F1 ← 0
    else if F1 == 1 then
        F1 ← 3
    else if F1 == 2 then
        F1 ← 5
    else if F1 == 3 then
        F1 ← 8
    end if
end for

// Read the bit stream B from edge pixels of stego image.
// B ← GetBitStream(S)
Count ← 0
for every data byte M1 of M and element F1 of F do
    M1 ← 0
    for i = 0 to F1 do
        M1 ← M1 << 1
        M1 ← M1 | B[Count + i]
    end for
    Count ← Count + F1
end for

// Uncompress the bytes of data to get the image
M ← Uncompress(M)

Figure 3.18 BSDE Image Extraction Algorithm
3.6.1.2.3 Mathematically understanding BSDE with an example

Consider the image skype.bmp to be hidden within LenaRGB.bmp using BSDE technique.

An edge map of LenaRGB.bmp would first be created using the modified canny edge detection algorithm. Skype.bmp would then be compressed using the wavelet compression algorithm. The binary data representation would then be subject to a mapping process. The size mapping process is carried out for each byte of data using the BSDE mapping table. In order to optimize the data, the amount of bits used by each byte of data is mapped to a value set in the flag array. The bit stream is constructed by appending the data bytes one after another, after the size mapping operation is carried out. The flag array and the bit stream are both stored into the edge pixels of the carrier image using the last two LSB of every edge pixel.

Consider for example the bytes of data obtained from skype.bmp are 0,7,30,100,…….

Here the data length =4 and length of length = 3 (since 4 is represented using 3 bits).

In binary, the number of bits used to represent the data would be 0,3,5,7.

On carrying out size optimization using the BSDE optimization table, the data sizes that would be used are 0,3,5,8 and the corresponding flag arrays would be set to 00, 10, 01, 11.

This data now has to be stored on the edge map. The first edge pixel always stores the length of length field. In our example – 3. The second edge pixel always stores data length. Then the flag array is stored as a stream followed by the data bytes.

In our example the flag array would have values - 00, 10, 01, 11 and the data stream would have values 111111001100100…. 

The actual process to encode and decode a pixel value is same as explained for ADDE method.
3.6.2 Wavelet based Data Compression algorithm

The proposed algorithm uses the Antonini wavelet filter proposed by Antonini et al., (1992) and the Alistair Moffat’s linear time algorithm for adaptive arithmetic coding (Alistair Moffat, 1990; Moffat et al., 1998). It takes as inputs the image, and various other coding parameters like number of subbands, quantization step size, etc.

The basic steps for wavelet based image compression are as shown in figure 3.19 below:

![Figure 3.19 Wavelet based image compression](image)

The basic steps for wavelet based image de-compression are as shown in figure 3.20 below:

![Figure 3.20 Wavelet based image de-compression](image)
A wavelet decomposition of a function $f$ defined on $\mathbb{R}^d$ is an expression (Devore et al., 1991) of the form

$$f = \sum C_{j,k} \varnothing_{j,k}$$

(3.1)

where the coefficients $C_{j,k}$ depend on $f$ and the functions

$$\varnothing_{j,k}(x) := \varnothing(2^k(x-j/2^k))$$

are the dyadic dilates (by $2^k$) and translates (by $j/2^k$) of a single function $\varnothing$ called a wavelet.

The data compression problem in the wavelet domain is seen as one of approximating $f$ by a compressed function $f_1$. In a lossless compression algorithm, the error between the original and compressed functions will be zero.

The algorithm calculates the quantized coefficients $C_{1,j,k}$ and the compressed function takes the form

$$f_1 = \sum C_{ij,k} \varnothing_{j,k}$$

(3.2)

It is very much necessary to measure the error between $f$ and $f_1$, which is assumed to be defined on some interval $I$ in $\mathbb{R}^d$. We have used the $L^p(I)$ norms error metric with $0 < p \leq \infty$. 
This norm is defined by

\[ |f - f_1|_{L^p(I)} = \left( \int |f(x) - f_1(x)|^p \, dx \right)^{1/p} \]  

(3.3)

A Discrete Wavelet transform refers to the multi-resolution decomposition of a signal. Low and high pass filters are used to extract the low frequency coefficients and the high frequency coefficients of a signal. Selected coefficients are then quantized and encoded that helps in the image compression. The proposed image compression algorithm based on the concept of wavelets is explained in figure 3.21.

![Wavelet compression Algorithm](image.png)

**Figure 3.21 Wavelet compression Algorithm**

### 3.6.3 Edge Detection Method

Edges are sudden variation in the grey level or colour of image pixels and edge detection is locating areas with strong intensity contrasts. The main principle of edge detection is
to extract the high frequency of image edge using first derivative, second derivative and high pass filtering. An important part in edge detection is the use of an appropriate filter, to handle the problems of noise and preserving the fine details of an image. Edge detection is of great importance in fields such as X-rays, computer vision, image analysis and security applications in image processing. Edge detection is a significant area of image processing and machine vision, due to the fact that edges are considered to be the important features for analysing the most essential information contained in images.

Various researches on the optimal canny edge operator have been carried out and many improvement measures have been proposed which could enhance the precision and accuracy of the edge detection. There are also some deficiencies of the dual thresholds of Canny operator. A too-high-setting threshold may miss important information, while a too-low-setting may put too much importance on the minor details. The main idea is to select thresholds, which minimize the within-class variance or maximize the between-class variance. This threshold can both suppress noise and keep the edge fine.

In the proposed edge detection method, the edgemap was obtained by a modified Canny operator using dynamically calculated high and low threshold parameters, according to the actual characteristics of the image. The entire secret image is scanned at run time and the median calculated. By experimenting on various types of images a conclusion was drawn that the thresholds could be set as 2/3 of median and 4/3 of median in order to obtain a continuous set of real edges. This enabled to get more integrated information and the continuity of the edge was strong, and positioning accurate. The block diagram of the proposed edge detection method is as shown in figure 3.22.
Steps for Edge detection using canny algorithm and dynamically calculated thresholds

1. Smoothing: Blurring of the image to remove noise.

2. Finding gradients: The edges should be marked where the gradients of the image has large magnitudes.

3. Non-maximum suppression: Only local maxima should be marked as edges.

4. Determine thresholds based on image characteristics.

5. Double thresholding: Potential edges are determined by thresholding.

6. Edge tracking by hysteresis: Final edges are determined by suppressing all edges that are not connected to a very certain (strong) edge.
Figure 3.22 Flow Chart for edge detection technique
The Canny edge algorithm with modified thresholds is given in figure 3.23.

![Algorithm: Canny Edge Detection Algorithm](image)

**Figure 3.23 Modified Canny Edge Detection Algorithm**

### 3.6.4 Application Graphical User Interface (GUI)

The application graphical user interface (GUI) is designed in a user friendly manner using Microsoft Visual Studio. It includes five main modules: Determine edges of cover image, Secret image compression, Embedding, Extraction and Controller. The GUI allows the user to choose the extract or embedding functions for the secret data. Encode module is implemented to embed the secret data (text or image) using the proposed
algorithms (ADDE or BSDE). The Decode module allows the receiver to extract the hidden data. The controller module is responsible for converting the secret data to binary for embedding purposes. If the secret data to be embedded is an image, it needs to be compressed first using a wavelet based image compression algorithm. If the secret data does not fit within the selected cover image, the user is prompted to change either the message or the image. The stego image will be saved onto the hard disk in the location specified. To hide a file inside a cover image, click on the button "Choose cover image" and on the file dialog that appears, select the file that you want to use as cover image. The next step is to determine the edges within the selected image. Then select the secret text or secret image that is to be hidden by clicking on the button “Select secret data”. If secret data is an image, click the “Compress image” button. Select the algorithm to use for encoding. Finally click the “Encode data” button.

To retrieve a file from a stego-image, select the image to retrieve the file from. Select the button marked "Choose encoded image" and from the file dialog that opens pick the file and click "Decode". Select the algorithm to use for decoding. If secret data is an image, click the “De-compress image” button. The secret data file will be saved onto the hard disk in the location specified. Sample Screenshots of the GUI are shown in Appendix A1.

### 3.7 Performance Evaluation Measures

Yalman and Erturk, (2013) identified that a fundamental task in most image processing applications is the visual evaluation of a modified image. They outlined many measures for examining image quality, such as the mean structural similarity, mean absolute error, mean square error (MSE), and peak signal-to-noise ratio (PSNR).
3.7.1 Mean Square Error (MSE)

As a performance measurement for image distortion, mean square error and the peak signal to noise ratio could be used as the key measures. MSE refers to the cumulative squared error between the modified (compressed image or stego image) and the original image (Jayachandran and Manikandan, 2010). MSE is computed by performing byte by byte comparisons of the two images. Assuming $I(i,j)$ to be the original image and $I'(i,j)$ to be the modified image, the MSE is computed as shown in equation 3.4.

$$MSE = \frac{1}{MN} \times \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} [I(i,j) - I'(i,j)]^2$$  \hspace{1cm} (3.4)

Where $i$ and $j$ are the image coordinates and $M$ and $N$ are the dimensions of the image.

For coloured images we consider the difference between the individual red, green and blue channel components.

3.7.2 Peak Signal to Noise Ratio (PSNR)

The PSNR metric, measured in decibels (dB), expresses the degree of noise introduced after embedding the secret data, by comparing the original against the modified (compressed image or stego image). The higher the PSNR value, the more the modified image resembles the original one (Mare et al., 2011). Human visual system is unable to distinguish images with PSNR more than 35 dB (Ghasemi and Shanbehzadeh, 2010).

The PSNR is a universal formula, which can be straightforwardly applied when we are dealing with gray-scale images; however, when confronting true RGB colour images, we generate the average MSE of the three colour channels and then calculate the PSNR (Cheddad et al., 2008). PSNR value is inversely proportional to MSE value. If MSE is zero, PSNR becomes infinite, means no distortion occurs after embedding.
The mathematical formula to compute PSNR is given in equation 3.5.

\[
PSNR = 10 \times \log_{10} \left( \frac{C_{\text{max}}^2}{MSE} \right)
\]  

(3.5)

Where MSE refers to the mean square error as explained in equation 3.4, \(C_{\text{max}}\) is the maximum possible pixel value of the image. It usually takes the value 1 for double precision intensity images and takes a value of 255 for 8 bit intensity images.

3.7.3 Structural Similarity Index (SSIM)

The structural similarity (SSIM) index is a method used to determine the similarity between two images. The resultant SSIM index is a decimal value between -1 and 1, and value 1 is achieved only in the case of two identical sets of data. Typically it is calculated on window sizes of 8×8. SSIM considers image degradation as perceived change in structural information. Structural information considers that the pixels have strong interdependencies especially when they are spatially close.

The SSIM metric is calculated on various windows of an image using equation 3.6. The measure between two windows and \(\hat{Y}\) of common size \(N \times N\) is (Wang et al., 2004; Yalman and Erturk, 2013):
3.7.4 Compression Ratio or Compression Power

Compression ratio quantifies the reduction in data representation size, produced by a data compression algorithm. It is the ratio between the uncompressed file size and the compressed size.

\[
SSIM(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{\mu_x^2 + \mu_y^2 + c_1\sigma_x^2 + \sigma_y^2 + c_2}
\]

where
- \(\mu_x\): Average of x
- \(\mu_y\): Average of y
- \(\sigma_x^2\): Variance of x
- \(\sigma_y^2\): Variance of y
- \(\sigma_{xy}\): Covariance of x and y
- \(c_1 = (k_1 L)^2\), \(c_2 = (k_2 L)^2\): Two variables to stabilize the division with weak denominator 
- \(L\): The dynamic range of the pixel values. Typically this is \(2^{\text{bits per pixel}} - 1\)
- \(k_1 = 0.01\) and \(k_2 = 0.03\): Values by default

\[C_r = \frac{U}{C}\]

Where:
- \(C\): Compressed File Size
- \(U\): Uncompressed File Size
- \(C_r\): Compression Ratio

3.7.5 Space Saving

Defined as the reduction in size relative to the uncompressed file size. Calculated using equation 3.8.
3.7.6 Embedding Rate

Embedding rate denotes the average number of bits that can be embedded per pixel into a given cover image.

3.7.7 Embedding Capacity

It is the size of the payload data in a cover image that can be modified without deteriorating the integrity of the cover image. The steganographic embedding operation needs to preserve the statistical properties of the cover image in addition to its perceptual quality. Capacity is represented by bits per pixel (bpp) and calculated using Equation 3.9.

Embedding capacity corresponds to the total capacity of a given cover image

\[
Embedding \, Capacity = \frac{S_{ij}}{C_{ij}}
\]

Where, \(S_{ij}\) is the size of the payload image, \(C_{ij}\) is the size of the cover image.
3.7.8 Basic Statistical Measures

Since embedding secret data in a cover image results in modification of cover, steganography inevitably leaves some traces in the statistical properties of the image. The basic statistical analysis of the cover images and the stego images was carried out using the mean and standard deviation measures. Standard deviation is the square root of the Variance. Variance is the average of the squared differences from the Mean. The mean is the average of the pixel values. The standard deviation is a measure of how far the signal fluctuates from the mean. The variance represents the power of this fluctuation. Mean is calculated as shown in equation 3.10. Standard deviation can be represented using equation 3.11.

\[
\mu = \frac{1}{N} \times \sum_{(i,j) \in R} x[i,j]
\]

\[\text{(3.10)}\]

\[
\sigma = \sqrt{\frac{1}{N-1} \times \sum_{(i,j) \in R} (x[i,j] - \mu)^2}
\]

\[\text{(3.11)}\]

Where \(N\) refers to number of pixels, \(x[i,j]\) refers to each score and \(\mu\) refers to mean or average.
RESULTS AND DISCUSSION

This chapter presents the findings along with discussions pertaining to the experiments carried out in view of this research work.

The proposed techniques were tested on 200 real time camera clicked digital images and on the UCID10K (Schaefer, 2010) and nevercompressed images databases having a total of about 15150, 24-bit RGB images and yet another database having about 200, 24 bit PNG images. The resolution of cover images varied from 256*256, 512*512 to 1024*1024 pixels to higher resolutions. For the secret text data we used files of varying sizes from 4 kb to 400 kb. The secret text data used was taken from various file formats which include the txt format, the doc format and the other formats. For the secret images various grayscale and colored images were tested.

As known and stated in the objectives of this research work, the 3 main criteria for designing a steganography algorithm are embedding capacity, imperceptibility and robustness. Both of the proposed techniques have been implemented and tested bearing in mind all these characteristics. Although exhaustive testing and analysis on the above mentioned databases has been carried out, due to space constraints, experimental results of only 6 images are demonstrated in this thesis. Two images are from the camera clicked images set and the remaining 4 are from the internet and online databases. The images whose results are presented are LenaRGB.bmp, Follow.bmp, Skype.bmp, BaboonRGB.bmp, Logo.bmp, and Fieldview.bmp. Some of the images have been used as secret data images as well as cover images. Samples of the original images analyzed for study are shown in figure 4.1.
4.1 Edge Detection in Cover images

Edges are sudden variation in the gray level or color of image pixels. Edge detection aims at identifying points in a digital image at which the image brightness changes sharply or more formally has discontinuities. An edge detection algorithm uses three steps; begins by reducing noise which acts like fake edges, then works towards enhancing the edges using a high pass filter, and finally edge localization by applying a threshold that aids in deciding which pixels form the real edges. The literature review carried out, discussed the use of various edge detection operators. However most Scholars stressed that the canny edge detection operator had proved to be the most optimal preference.
In the proposed steganography techniques, the edge map needed for secret data embedding was obtained by a modified Canny operator using dynamically calculated high and low threshold parameters according to the actual characteristics of the image. It was seen that such a variation in the threshold values of the Canny algorithm, helped to increase the embedding capacity by increasing the number of edge pixels. A comparison of various edge detection operators was carried out and the results tabulated in table 4.1

A sample of two images with the edges determined using the normal Canny algorithm and the proposed modified canny algorithm are as shown in figure 4.2 below

Table 4.1  Comparison of various edge detection operators

<table>
<thead>
<tr>
<th>Sr no</th>
<th>Criteria</th>
<th>Robert operator</th>
<th>Laplacian of Gaussian operator</th>
<th>Canny operator</th>
<th>Shen castan operator</th>
<th>Threshold Canny</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Complexity</td>
<td>Simple</td>
<td>Comparatively complex</td>
<td>Complex</td>
<td>Complex</td>
<td>Complex</td>
</tr>
<tr>
<td>2</td>
<td>Noise sensitivity</td>
<td>Highly sensitive</td>
<td>Low sensitive</td>
<td>Low sensitive</td>
<td>Highly sensitive</td>
<td>Low sensitive</td>
</tr>
<tr>
<td>3</td>
<td>False positive edge</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>False negative edge</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Corners and junction detection</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Good</td>
</tr>
<tr>
<td>6</td>
<td>Computation time</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Very High</td>
<td>Very High</td>
</tr>
<tr>
<td>7</td>
<td>Signal to noise ratio</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>Very High</td>
<td>Good</td>
</tr>
<tr>
<td>8</td>
<td>Filter used</td>
<td>N/A</td>
<td>Gaussian Filter</td>
<td>Gaussian Filter</td>
<td>ISEF Filter</td>
<td>Gaussian Filter</td>
</tr>
<tr>
<td>9</td>
<td>Localization of edge</td>
<td>Not good</td>
<td>Better than Robert</td>
<td>Good</td>
<td>Better than Canny</td>
<td>Good</td>
</tr>
</tbody>
</table>
Figure 4.2 Edge detection using Canny and modified Canny algorithm. (a) & (b) Original images  (c) & (d) Canny algorithm detected edges. (e) & (f) Modified canny algorithm edges.
It can thus be concluded that the proposed modified canny algorithm results in an edge map that is better (provides more pixels for embedding data) for concealing and retrieving the secret data.

4.2 Secret Image Compression

Image compression is very crucial in reducing the cost of data storage and transmission in relatively slow channels. A compression algorithm is what reduces the redundancy of data representation and decreases the data storage capacity (Gonzalez and Woods, 2005).

On implementation of the proposed data compression algorithm, using the Antonini wavelet filter and Alistair Moffat's linear time algorithm, for adaptive arithmetic coding of the transformed quantized coefficients, a fairly good compression ratio as well as a decompressed image of good visual quality was obtained. Results obtained were compared with normal JPEG compression with quality factors of 90 and 70. The performance of this algorithm was also compared with some more compression algorithms proposed by Pattanaik et al., (2006), Singh and Sharma (2012) and Zhang and Zhang (2014). The proposed image compression algorithm outperforms the various other algorithms as well as the JPEG algorithm in terms of image quality. Although JPEG compression helped to achieve a smaller file size, when compared with our proposed compression algorithm, the JPEG images were found to possess clear artifacts, even when zoomed by a small percentage. The clarity of our wavelet based compression algorithm images was found to be good and no artifacts were seen, even at a zoom of 700%. Figure 4.3 below displays the original images along with their compressed versions at a quality factor of 90, considering both the proposed compression algorithm and JPEG compression. Figures 4.3(d) & (f) have been zoomed for viewing purpose.
Figure 4.3 Comparison of JPEG compression and proposed wavelet based compression algorithm at 90 quality factor (a) & (b) Original images (c) & (d) Images compressed using jpeg compression. (e) & (f) Images compressed using proposed wavelet compression.
It can be noted from figure 4.3 (c) and (d) that the JPEG compression produces clear artifacts even at a quality factor of 90. For viewing and representation purpose, the skype decompressed images, 4.3 (d) and (f) has been zoomed to a size of 200%. The artifacts in the JPEG compressed image (figure 4.3(d)) are clearly visible, while the wavelet based decompressed image (figure 4.3(f)) does not report any clear distortions on visual inspection. It is the multi-resolution characteristic of wavelets which leads to superior energy compaction and in turn resulted in high quality reconstructed images.

Figure 4.4 displays the original images along with their compressed versions at a quality factor of 70, considering both the proposed wavelet based compression algorithm and JPEG compression. Figure 4.4 (c) and (d) demonstrate that JPEG compression displays obvious artifacts at a quality factor of 70.

Compression algorithms are compared and evaluated based on certain performance measures which include Compression Ratio (CR), Space saving (SS), Peak signal to noise ratio (PSNR), etc. (Rehna et al., 2012). Data compression ratio, also known as compression power, is used to quantify the reduction in data-representation size produced by data compression. Increasing PSNR means increasing fidelity of compression. The higher the PSNR; better the quality of the compressed or reconstructed image (Nilsson et al., 2005; Mathworks).
Figure 4.4 Comparison of JPEG compression and proposed wavelet based compression algorithm at 70 quality factor (a) & (b) Original images (c) & (d) Images compressed using jpeg compression. (e) & (f) Images compressed using proposed wavelet compression.
The zoomed images of 70 percent quality at 800 percent zoom are as shown in figure 4.5.

Figure 4.5 Zoomed images of 70 percent quality at 800 percent zoom. (a) & (b) Original images (c) & (d) Images compressed using jpeg compression. (e) & (f) Images compressed using proposed wavelet compression.

The PSNR results for some sample images using the proposed wavelet based compression algorithm (WCP) and some other compression algorithms are given in Table 4.2.
Table 4.2 PSNR values between original and decompressed images

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Image name</th>
<th>Quality factor</th>
<th>PSNR (Proposed)</th>
<th>PSNR (JPEG)</th>
<th>PSNR (Pattanaik et al.)</th>
<th>PSNR (Zhang &amp; Zhang)</th>
<th>PSNR (Singh &amp; Sharma)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fieldview.bmp</td>
<td>90</td>
<td>52.14 db</td>
<td>47.91 db</td>
<td>44.37 db</td>
<td>33.60 db</td>
<td>30.23 db</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp</td>
<td>90</td>
<td>80.19 db</td>
<td>64.26 db</td>
<td>48.36 db</td>
<td>37.6 db</td>
<td>34.42 db</td>
</tr>
<tr>
<td>3</td>
<td>Skype.bmp</td>
<td>90</td>
<td>75.60 db</td>
<td>54.62 db</td>
<td>43.23 db</td>
<td>35.42 db</td>
<td>31.32 db</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp</td>
<td>70</td>
<td>48.31 db</td>
<td>39.51 db</td>
<td>43.32 db</td>
<td>32.48 db</td>
<td>28.17 db</td>
</tr>
<tr>
<td>5</td>
<td>LenaRGB.bmp</td>
<td>70</td>
<td>70.78 db</td>
<td>58.23 db</td>
<td>46.71 db</td>
<td>36.2 db</td>
<td>33.76 db</td>
</tr>
<tr>
<td>6</td>
<td>Skype.bmp</td>
<td>70</td>
<td>65.09 db</td>
<td>55.38 db</td>
<td>42.78 db</td>
<td>34.58 db</td>
<td>30.79 db</td>
</tr>
</tbody>
</table>

Figure 4.6 Comparison of PSNR values of different compression algorithms

It is thus seen that images compressed - decompressed using the proposed wavelet based compression algorithm (WCP) produces consistently better images (visually) and a higher PSNR (Peak signal to noise ratio) value than the jpeg compression algorithm at the same quality factor.

Artifacts could be seen in the jpeg compressed images as compared to the proposed wavelet based compression algorithm. Jpeg compression produced a better compression
ratio; however it results in a low PSNR. Thus the saving space resulting from JPEG compression was much more than that of the proposed wavelet compression algorithm.

On compression some of the file sizes were as displayed in table 4.3 and table 4.4

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Original name &amp; filesize</th>
<th>Wcp compressed filesize</th>
<th>Decompressed filesize</th>
<th>Jpeg filesize</th>
<th>Compression ratio (WCP)</th>
<th>Compression ratio (JPEG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lena.bmp (769 kb)</td>
<td>85 kb</td>
<td>769 kb</td>
<td>68 kb</td>
<td>9:1</td>
<td>12:1</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp (450 kb)</td>
<td>64 kb</td>
<td>450 kb</td>
<td>50 kb</td>
<td>7:1</td>
<td>9:1</td>
</tr>
<tr>
<td>3</td>
<td>skype.bmp (49 kb)</td>
<td>10 kb</td>
<td>49 kb</td>
<td>8 kb</td>
<td>5:1</td>
<td>6:1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Original name &amp; filesize</th>
<th>Wcp compressed filesize</th>
<th>Decompressed filesize</th>
<th>Jpeg filesize</th>
<th>Compression ratio (WCP)</th>
<th>Compression ratio (JPEG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lena.bmp (769 kb)</td>
<td>55 kb</td>
<td>769 kb</td>
<td>34 kb</td>
<td>14:1</td>
<td>22:1</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp (450 kb)</td>
<td>45 kb</td>
<td>450 kb</td>
<td>28 kb</td>
<td>10:1</td>
<td>16:1</td>
</tr>
<tr>
<td>3</td>
<td>skype.bmp (49 kb)</td>
<td>7 kb</td>
<td>49 kb</td>
<td>5 kb</td>
<td>7:1</td>
<td>10:1</td>
</tr>
</tbody>
</table>

4.3 Security Analysis of proposed Steganography Methods

4.3.1 Analysis of the ADDE Method

4.3.1.1 Visual Perceptibility Test

The major requirement of steganographic systems is that the stego object should be perceptually indistinguishable to the degree that it does not raise suspicion. The human
visual system is a complex biological system which has three stages of perception: encoding, representation, and comprehension. There are many factors restricting the perceptibility of human's visual system. Visual sensitivity is influenced by brightness, contrast of background and various other environment related factors. The original images along with the stego images (original image with data embedded) obtained using the ADDE algorithm are as shown in figure 4.7
Figure 4.7 Visual Perceptibility Test for ADDE technique
Figure 4.7 (a, d & g) are original images chosen from the online database and the camera clicked image database. Figure 4.7 (b, c, e, f, h & i) are the stego images generated after applying the ADDE algorithm to varying secret data. Figure 4.7 (b, e & h) conceal secret text data of varying size and Figure 4.7 (c, f & i) hide secret images of varying size. Figure 4.7 (b) has 57344 bits (7 kb) embedded within while figure 4.7(c) has an image (skype.bmp) with 16384 pixels concealed. Similarly figure 4.7 (e) has 81920 bits (10 kb) embedded within while figure 4.7 (f) has an image (follow.bmp) with 16384 pixels concealed. Figure 4.7 (h) has 1474560 bits (180 kb) embedded within while figure 4.7(i) has an image (Lena.bmp) with 262144 pixels concealed.

It can be observed that despite the embedding of secret data, the file size remains constant. The original images as well as the resultant stego images were compared using visual analysis of the images. A team of ten non-technical lay people analysed the images and compared them visually. Some checked the printouts using high magnification lens, while others compared the digital form. Some even took them to a laboratory and analysed them under a microscope, while others used high end imaging software like Adobe Photoshop and zoomed the images to 800%. A few of them had doubts regarding some of the images. A lot of the suspect conclusions were based on emotional feelings regarding the images. This arose from the various moods of people and from the fact that they were conscious about trying to find something, each one trying to prove himself!! A few of them even claimed that the original images had data!! However none of them could come up with any strong logical conclusion that proved that the stego images had data embedded within them. Hence we could deduce that it is practically impossible for a human eye to visually detect any difference between the original images and the stego images. It would be very difficult to visually deduce that
the stego image contains secret data. Although the human eye can visually distinguish between various colours, as well as colour changes, the changes incorporated into the stego images are such that, the human eye is successfully deceived. This could be due to the vision properties of the eye, which has been discussed earlier. The images used for testing were of varying resolutions and sizes, with 24-bit colour depth. BMP and PNG images were mainly considered.

To determine if any distortion could be seen due to the data embedding, the LenaRGB image was zoomed at 800% at the eyes region as shown in figure 4.7. However no visible distortion could be seen.

Figure 4.8 Comparison of zoomed LenaRGB.bmp for ADDE technique
(a) Original image zoom (b) LenaRGB with secret data= 57344 bits (c) LenaRGB with secret data= 81920 bits (d) LenaRGB with secret data= 155648 bits
For the purpose of further in-depth visual analysis, the LSB plane of the original image was compared with the LSB plane of the stego image. The difference was found to be negligible. There was no visible pattern by which one could logically conclude that there was secret data present within the image.

4.3.1.2 Statistical analysis of Stego - images

Statistical steganography is the practice of detecting hidden information through applying statistical tests on image data. Literature review reports that most steganography algorithms leave a signature when embedding information that can be easily detected through statistical analysis. To be able to pass by an intruder without being detected, a steganography algorithm must not leave any mark in the image. Some basic statistical analysis was carried out with an aim to prove that the images were being tampered with. If an image is being modified, it is expected that some statistical properties of the image will deviate from the norm. The results of the statistical analysis tabulated in table 4.3 are that of stego images obtained by embedding text data using the ADDE technique; compared with the statistical properties of the original unmodified images.

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Cover Image</th>
<th>Original Image</th>
<th>Secret data=32768 bits</th>
<th>Secret data=81920 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Mean</td>
<td>Standard Deviation</td>
<td>Mean</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>128.23</td>
<td>58.98</td>
<td>128.23</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp</td>
<td>207.10</td>
<td>69.82</td>
<td>207.10</td>
</tr>
<tr>
<td>3</td>
<td>Follow.png</td>
<td>149.19</td>
<td>77.28</td>
<td>149.19</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp</td>
<td>123.66</td>
<td>94.48</td>
<td>123.66</td>
</tr>
</tbody>
</table>
Since it is stated that LSB embedding offsets the statistical properties of an image, we tested the same using multiple cover images. Due to space constraints the analysis results of only 4 images have been tabulated: LenaRGB.bmp, Logo.bmp, Follow.png, and Fieldview.bmp. Some of these are camera clicked while others are from a standard database. Varying sizes of text files including pdf formats were embedded and the mean and standard deviation of the stego (embedded) images were checked. It can be noted that on embedding secret data of size 32768 bits, there was absolutely no change in the values of the mean and standard deviation. Besides the file sizes of the original and the stego image was the same. On embedding a larger amount of secret data of 81920 bits, too no significant difference in the data values could be noticed.

Table 4.6 compares the statistical properties of stego images obtained by embedding secret images using the ADDE technique. The secret images that were used for embedding were follow.png of size 401408 bits and baboon.bmp (resized) of size 917504 bits. It can be noted that negligible change in data values were seen. After analysing the statistical data of the various stego images and comparing it with that of the original images, no significant difference in data values was observed and thus it is concluded that there is no evidence obtained that the proposed ADDE technique changed the basic statistical properties of an image while concealing the data.
Table 4.6 Basic Statistical test data for embedding secret images using ADDE

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Cover Image</th>
<th>Original Image</th>
<th>Secret data=follow.png(401 408 bits)</th>
<th>Secret data=baboon.bmp(917 504 bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Standard Deviation</td>
<td>Mean</td>
</tr>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>128.23</td>
<td>58.98</td>
<td>128.23</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp</td>
<td>207.10</td>
<td>69.82</td>
<td>207.10</td>
</tr>
<tr>
<td>3</td>
<td>Polar.png</td>
<td>135.17</td>
<td>63.24</td>
<td>135.17</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp</td>
<td>123.66</td>
<td>94.48</td>
<td>123.65</td>
</tr>
</tbody>
</table>

4.3.1.3 Payload Analysis

The payload specifies the maximum number of bits that can be hidden within a cover medium with an acceptable resultant stego-object quality. The stego object would be of no value if it can hold a large amount of data at the expense of a distorted embedded object. Usually, the high payload (or capacity) requirement, conflicts with high PSNR requirement. Generally, when the payload increases, the MSE also increases, and this affects the PSNR inversely. So, a trade-off should be made between payload (capacity) and PSNR requirements. Maximum payload refers to the maximum payload of the carrier under a certain constraint. Based on the constraint of "perceptual invisibility," maximum payload refers to the higher limit of secret data embedded into the image. If exceeding this limit, the stego image is perceived by the observer, that is, the observer discovers the change in the image quality.

Another important factor that aids in the payload analysis is the choice of the cover image. The selection is at the discretion of the person who sends the message. Images with a low number of colors, computer art, and images with unique semantic content
should be avoided as cover images. Since our proposed algorithm focuses on embedding in edge region, images which will result in a large number of edges should be used as cover images to ensure a better embedding capacity. We have calculated the embedding capacity of the proposed ADDE algorithm with respect to different cover images.

**TABLE 4.7 PSNR values of the original and stego images & Embedding capacity for LenaRGB.bmp**

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Cover image</th>
<th>Secret text (in bits)</th>
<th>MSE</th>
<th>PSNR</th>
<th>Embedding capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text1.txt (32768)</td>
<td>0.02</td>
<td>64.52 db</td>
<td>0.125</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text2.txt (57344)</td>
<td>0.05</td>
<td>61.52 db</td>
<td>0.219</td>
</tr>
<tr>
<td>3</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text3.txt (81920)</td>
<td>0.07</td>
<td>59.75 db</td>
<td>0.312</td>
</tr>
<tr>
<td>4</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text4.txt (155648)</td>
<td>0.09</td>
<td>55.36 db</td>
<td>0.594</td>
</tr>
<tr>
<td>5</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text5.txt (262144)</td>
<td>Size exceeds edge capacity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**TABLE 4.8 PSNR values of the original and stego images & Embedding capacity for Logo.bmp**

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Cover image</th>
<th>Secret text (in bits)</th>
<th>MSE</th>
<th>PSNR</th>
<th>Embedding capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Logo.bmp (450 kb)</td>
<td>Text1.txt (32768)</td>
<td>0.04</td>
<td>62.16 db</td>
<td>0.215</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp (450 kb)</td>
<td>Text2.txt (57344)</td>
<td>0.08</td>
<td>59.15 db</td>
<td>0.375</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp (450 kb)</td>
<td>Text3.txt (81920)</td>
<td>0.12</td>
<td>57.41 db</td>
<td>0.536</td>
</tr>
<tr>
<td>4</td>
<td>Logo.bmp (450 kb)</td>
<td>Text4.txt (155648)</td>
<td>Size exceeds edge capacity</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Logo.bmp (450 kb)</td>
<td>Text5.txt (262144)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE 4.9 PSNR values of the original and stego images & Embedding capacity for Fieldview.bmp

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image</th>
<th>Secret text</th>
<th>MSE</th>
<th>PSNR</th>
<th>Embedding capacity</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fieldview.bmp (11476 kb)</td>
<td>Text1.txt (32768)</td>
<td>0.002</td>
<td>76.34 db</td>
<td>0.008</td>
</tr>
<tr>
<td>2</td>
<td>Fieldview.bmp (11476 kb)</td>
<td>Text2.txt (57344)</td>
<td>0.003</td>
<td>74.89 db</td>
<td>0.014</td>
</tr>
<tr>
<td>3</td>
<td>Fieldview.bmp (11476 kb)</td>
<td>Text3.txt (81920)</td>
<td>0.003</td>
<td>73.26 db</td>
<td>0.021</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp (11476 kb)</td>
<td>Text4.txt (155648)</td>
<td>0.01</td>
<td>68.51 db</td>
<td>0.039</td>
</tr>
<tr>
<td>5</td>
<td>Fieldview.bmp (11476 kb)</td>
<td>Text5.txt (262144)</td>
<td>0.02</td>
<td>64.52 db</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>Fieldview.bmp (11476 kb)</td>
<td>Text6.txt (3063808)</td>
<td>Size exceeds edge capacity</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 4.10 PSNR values of the original and stego images & Embedding capacity for LenaRGB.bmp & Logo.bmp (ADDE)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image</th>
<th>Secret image (in pixels)</th>
<th>PSNR</th>
<th>Embedding capacity (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>Follow.bmp (16384)</td>
<td>58.37 db</td>
<td>0.062</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp</td>
<td>Skype.bmp (16384)</td>
<td>58.34 db</td>
<td>0.062</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp</td>
<td>Follow.bmp (16384)</td>
<td>50.38 db</td>
<td>0.10</td>
</tr>
<tr>
<td>4</td>
<td>Logo.bmp</td>
<td>Skype.bmp (16384)</td>
<td>48.31 db</td>
<td>0.10</td>
</tr>
</tbody>
</table>

It is apparent to an observer that tables 4.7, 4.8 and 4.9 help to conclude that there is a trade-off between the payload and the cover image distortion. The image distortion is measured using the PSNR distortion metric. It can be noted that as the payload is increased the PSNR value decreases or we can summarize and say that the image quality decreases. The embedding rate of ADDE is maximum of 6 bits per pixel and varies according to the mapping logic used. The embedding capacities too have been calculated.
and take values as shown in tables 4.7, 4.8, 4.9 and 4.10. When compared with various other data embedding techniques developed by researchers (Weiqi et al., 2010; Geetha et al., 2013, Shahzad et al., 2014; Islam et al., 2014) our proposed method provides a lower embedding capacity in some cases; however achieve a higher PSNR value. This indicates that the proposed ADDE algorithm produces highly imperceptible images.

4.3.1.4 Peak Signal to Noise Ratio Analysis

As a performance measurement for image distortion, the well known Peak-Signal-to-Noise Ratio, (PSNR), which is classified under the difference distortion metrics, is applied to the stego-images. PSNR is used more often, since it is a logarithmic measure, and human brains seem to respond logarithmically to intensity. PSNR values falling below 30dB indicate a fairly low quality, i.e., distortion caused by embedding can be obvious. A high quality stego-image should strive for a PSNR value of 40 dB and above (Cheddad, 2009).

The cover images are embedded with different size of data ranged from 32768 bits to 3145728 bits. The PSNR value of various stego color images is calculated for the visual quality. Results of PSNR values are as shown in tables 4.7, 4.8, 4.9, 4.10. The extracted stego images were also being tested for quality; and results displayed in table 4.11. Figure 4.9 shows the curve of the data hiding capacity versus the visual quality; it indicates that as capacity is increased PSNR value decreases.

The tabulated results indicate that the extracted stego images too were of good quality (PSNR > 35 db).
Figure 4.9 Comparison of PSNR values for data bits embedded (ADDE)

TABLE 4.11 PSNR values of the extracted secret data image and the original secret data for LenaRGB.bmp & Logo.bmp (ADDE)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image</th>
<th>Secret image (in pixels)</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>Follow.bmp (16384)</td>
<td>56.37 db</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp</td>
<td>Skype.bmp (16384)</td>
<td>58.34 db</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp</td>
<td>Follow.bmp (16384)</td>
<td>48.38 db</td>
</tr>
<tr>
<td>4</td>
<td>Logo.bmp</td>
<td>Skype.bmp (16384)</td>
<td>43.51 db</td>
</tr>
</tbody>
</table>

4.3.1.5 Structural Similarity Index Analysis

The Structural Similarity (SSIM) Index measure of quality works by measuring the structural similarity that compares local patterns of pixel intensities that have been normalized for luminance and contrast. This quality metric is based on the principle that the human visual system is good for extracting information based on structure.
The SSIM was calculated for various original - stego image pairs generated using the proposed ADDE algorithm and some of the results are as tabulated in table 4.12.

### TABLE 4.12 SSIM values of the original and stego images (ADDE)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image</th>
<th>Secret data</th>
<th>SSIM</th>
</tr>
</thead>
</table>
| 1       | LenaRGB.bmp       | 81920 bits           | Red Index - 0.999976  
|         |                   |                      | Green Index – 0.999909  
|         |                   |                      | Blue Index – 0.999986  |
| 2       | Baboon.bmp        | skype.bmp 16384 pixels | Red Index - 0.999997  
|         |                   |                      | Green Index – 0.999958  
|         |                   |                      | Blue Index – 0.999954  |
| 3       | Logo.bmp          | Text5.pdf (57344 bits) | Red Index - 0.999949  
|         |                   |                      | Green Index – 0.999944  
|         |                   |                      | Blue Index – 0.999980  |

The resultant SSIM index is a decimal value between -1 and 1, and value 1 is achieved only in the case of two identical sets of data. It can be concluded from the statistics in table 4.12 that the structural similarity between the original and stego image has good values.

### 4.3.1.6 Histogram Analysis

A histogram represents the graphical representation of the tonal variation in digital images. It represents the number of pixels for each tonal value. The x-axis represents the tonal variations and y-axis represents the number of pixels in particular tonal value. The left side of the horizontal axis represents the black and dark areas, the middle represents medium grey and the right hand side represents light and pure white areas. For coloured images, we generate individual histograms for the three colour channels: red, green and
blue. It is said that on embedding data within an image, its histogram tends to vary. In view of this, a histogram analysis was being carried out.

Figure 4.10  
Comparison of histogram of LenaRGB.bmp with concealed text data (32768 bits) for ADDE technique

Figure 4.11  
Comparison of histogram of Fieldview.bmp with LenaRGB.bmp embedded using ADDE technique
The histogram analysis of multiple stego images from different databases was being carried out. Screenshots of three sample tests have been documented in this thesis in figure 4.10, 4.11, 4.12. No variation between the original and the stego images could be noticed. Thus it can be concluded that stego images generated by the ADDE algorithm can withstand histogram analysis attack.

4.3.1.7 LSB Enhancement attack & Chi-Square attack

LSB enhancement is the process of manipulating the least-significant-bit (LSB) of image data in order to produce an output to discover the embedded data. LSB enhancement means setting the component value to 255 if the least-significant-bit is 1 or leave as it is, if 0. As a result of this operation, the image will have some flashy colours in the parts
where the LSBs were 1s. After performing the above operation on all pixels of the image, the resulting LSB image will reveal the parts of the image with the hidden data.

The Chi-Square attack was developed by Andreas Westfeld and Andreas Pfitzmann in 2000. It is a statistical test to measure if a given set of observed data and an expected set of data are similar or not. The idea of this attack is to compare Pair-of-Values' observed frequencies with their expected frequencies and calculate $p$ value, which will represent the probability of having some embedded data in an image. According to Westfeld & Pfitzmann, when some data like encrypted text is embedded into an image, the LSB values of the original data change in a way that the number of these pairs become nearly equal, while they differ so much when there is no embedding. An expected data set is explained by Westfeld & Pfitzmann (1999) as the average of the occurrences of the values in a pair. Both these tests were executed on the stego images generated using the ADDE algorithm, using the VSL tool and the Simple Steganalysis Suite software. Screenshots of the tools are shown in Appendix A2. Results are as shown in figure 4.13.

It can thus be observed that no artifacts could be seen on carrying out LSB enhancement attack, neither any deduction from the results of the Chi Square test.
Figure 4.13  Results of LSB enhancement and Chi square tests for ADDE.
4.3.2 Analysis of the BSDE Method

4.3.2.1 Visual Perceptibility Test

Creative methods have been devised in data hiding science, to reduce the visible detection of the embedded message. In a visual perceptibility test, we compare the quality of the stego image with the cover image as seen by the Human Visual System (HVS). The images used in our test are of varying resolutions and possess a 24-bit colour depth.

Figure 4.14(a,d,g) displays the original images, (b, e, h) are the respective stego images obtained by embedding text data and figure 4.14 (c, f, i) are the respective stego images obtained by embedding secret images. Figure 4.14 (b) has 57344 bits (7 kb) embedded within while figure 4.14 (c) has an image (skype.bmp) with 16384 pixels concealed. Similarly figure 4.14 (e) has 81920 bits (10 kb) embedded within while figure 4.14 (f) has an image (follow.bmp) with 16384 pixels concealed. Figure 4.14 (h) has 2244608 bits (274 kb) embedded within while figure 4.14 (i) has an image (Lena.bmp) with 262144 pixels concealed.
The original images as well as the resultant stego images were compared using visual inspection by a team of ten technical and non-technical people. Some used high-end imaging software like Adobe Photoshop and zoomed the images to 800% to figure out any distortions. On reporting a few of them had doubts regarding some of the images. A lot of the suspect conclusions were based on emotional feelings, moods of people,
improper illumination conditions and from the fact that they were conscious about trying
to find something. Out of the 200 images given to the volunteers, only two stego images
were correctly identified, while about 30 original images were being suspected. However
none of them could come up with any strong logical conclusion that proved that the two
suspected stego images had data embedded within them. Hence we can deduce that it is
practically impossible for a human eye to visually detect any difference between the
original images and the stego images.

4.3.2.2 Statistical analysis of Stego-images

Spatial steganography is said to generate unusual patterns, all of which leave traces to be
picked up by adversaries. In view of this a statistical analysis of stego images generated
using the BSDE algorithm were being evaluated. The results of the statistical analysis
tabulated in table 4.13 are that of stego images obtained by embedding text data using
the BSDE technique; compared with the statistical properties of the original unmodified
images.

The cover images used for testing the BSDE algorithm were LenaRGB.bmp, Logo.bmp,
Follow.png, and Fieldview.bmp. Varying sizes of files including binary files and pdf
formats were embedded and the mean and standard deviation of the stego (embedded)
images were checked. It can be noted that on embedding secret data of size 32768 bits,
there was absolutely no change in the values of the mean and standard deviation. Besides
the file sizes of the original and the stego image was the same. On embedding a larger
amount of secret data of 57344 bits, too no significant difference in the data values could
be noticed.
Table 4.13 Basic Statistical test data for embedding secret text using BSDE

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Cover Image</th>
<th>Original Image</th>
<th>Secret data= 32768 bits</th>
<th>Secret data= 57344 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Standard Deviation</td>
<td>Mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Standard Deviation</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>128.23</td>
<td>58.98</td>
<td>128.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>58.98</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp</td>
<td>207.10</td>
<td>69.82</td>
<td>207.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>69.82</td>
</tr>
<tr>
<td>3</td>
<td>Follow.png</td>
<td>149.19</td>
<td>77.28</td>
<td>149.19</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>77.28</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp</td>
<td>123.66</td>
<td>94.48</td>
<td>123.66</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>94.48</td>
</tr>
</tbody>
</table>

Table 4.14 Basic Statistical test data for embedding secret images using BSDE

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Cover Image</th>
<th>Original Image</th>
<th>Secret data= follow.png</th>
<th>Secret data= cameraman.bmp</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Standard Deviation</td>
<td>Mean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mean</td>
<td>Standard Deviation</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>128.23</td>
<td>58.98</td>
<td>128.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>58.98</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp</td>
<td>207.10</td>
<td>69.82</td>
<td>207.10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>69.82</td>
</tr>
<tr>
<td>3</td>
<td>Polar.png</td>
<td>135.17</td>
<td>63.24</td>
<td>135.17</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>63.24</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp</td>
<td>123.66</td>
<td>94.48</td>
<td>123.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>94.48</td>
</tr>
</tbody>
</table>

Table 4.14 compares the statistical properties of stego images obtained by embedding secret images using the BSDE technique.

The secret images that were used for embedding were follow.png of size 50176 bytes and cameraman.bmp of size 65536 bytes. It can be noted that negligible or no change in data values was seen. After analysing the statistical data of the various stego images and comparing it with that of the original images, no difference in data values was observed and thus it is concluded that there is no evidence obtained that the proposed BSDE technique changed the basic statistical properties of an image while concealing the data.
4.3.2.3 Payload Analysis

In the proposed techniques the embedding process affects only certain regions of interest (ROI); edges, rather than the entire image. Data Payload can be defined as the amount of information it can hide within the cover media. As with any method of storing data, this can be expressed as a number of bits, which indicates the max message size that might be inserted into an image. It can be expressed as a percentage from the full imagesize. We have calculated the embedding capacity of the proposed BSDE algorithm with respect to different cover images. Tables 4.15 to 4.18 display the results.

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image (in kb)</th>
<th>Secret text (in bits)</th>
<th>MSE</th>
<th>PSNR</th>
<th>Embedding capacity (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>Text1.txt (32768)</td>
<td>0.05</td>
<td>61.14 db</td>
<td>0.125</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp</td>
<td>Text2.txt (57344)</td>
<td>0.10</td>
<td>58.12 db</td>
<td>0.219</td>
</tr>
<tr>
<td>3</td>
<td>LenaRGB.bmp</td>
<td>Text3.txt (81920)</td>
<td>0.15</td>
<td>56.35 db</td>
<td>0.312</td>
</tr>
<tr>
<td>4</td>
<td>LenaRGB.bmp</td>
<td>Text4.txt (155648)</td>
<td>0.30</td>
<td>53.34 db</td>
<td>0.594</td>
</tr>
<tr>
<td>5</td>
<td>LenaRGB.bmp</td>
<td>Text5.txt (262144)</td>
<td>0.94</td>
<td>48.38 db</td>
<td>1</td>
</tr>
</tbody>
</table>

TABLE 4.15 PSNR values of the original and stego images & Embedding capacity for LenaRGB.bmp (BSDE)
### TABLE 4.16 PSNR values of the original and stego images & Embedding capacity for Logo.bmp (BSDE)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image (in kb)</th>
<th>Secret text (in bits)</th>
<th>MSE</th>
<th>PSNR</th>
<th>Embedding capacity (bp p)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Logo.bmp (450)</td>
<td>Text1.txt (32768)</td>
<td>0.08</td>
<td>59.07 db</td>
<td>0.215</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp (450)</td>
<td>Text2.txt (57344)</td>
<td>0.17</td>
<td>55.90 db</td>
<td>0.375</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp (450)</td>
<td>Text3.txt (81920)</td>
<td>0.25</td>
<td>54.07 db</td>
<td>0.536</td>
</tr>
<tr>
<td>4</td>
<td>Logo.bmp (450)</td>
<td>Text4.txt (155648)</td>
<td>0.37</td>
<td>51.15 db</td>
<td>1.02</td>
</tr>
<tr>
<td>5</td>
<td>Logo.bmp (450)</td>
<td>Text5.txt (262144)</td>
<td></td>
<td>Size exceeds edge capacity</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE 4.17 PSNR values of the original and stego images & Embedding capacity for Fieldview.bmp (BSDE)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image (in kb)</th>
<th>Secret text (in bits)</th>
<th>MSE</th>
<th>PSNR</th>
<th>Embedding capacity (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fieldview.bmp (11476)</td>
<td>Text1.txt (32768)</td>
<td>0.003</td>
<td>74.32 db</td>
<td>0.008</td>
</tr>
<tr>
<td>2</td>
<td>Fieldview.bmp (11476)</td>
<td>Text2.txt (57344)</td>
<td>0.003</td>
<td>72.18 db</td>
<td>0.014</td>
</tr>
<tr>
<td>3</td>
<td>Fieldview.bmp (11476)</td>
<td>Text3.txt (81920)</td>
<td>0.01</td>
<td>69.32 db</td>
<td>0.021</td>
</tr>
<tr>
<td>4</td>
<td>Fieldview.bmp (11476)</td>
<td>Text4.txt (155648)</td>
<td>0.02</td>
<td>64.52 db</td>
<td>0.039</td>
</tr>
<tr>
<td>5</td>
<td>Fieldview.bmp (11476)</td>
<td>Text5.txt (262144)</td>
<td>0.04</td>
<td>61.48 db</td>
<td>0.06</td>
</tr>
<tr>
<td>6</td>
<td>Fieldview.bmp (11476)</td>
<td>Text6.txt (3145728)</td>
<td>0.90</td>
<td>49.38 db</td>
<td>0.80</td>
</tr>
</tbody>
</table>

### TABLE 4.18 PSNR values of the original and stego images & Embedding capacity for LenaRGB.bmp & Logo.bmp (BSDE)

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image</th>
<th>Secret image (in pixels)</th>
<th>PSNR</th>
<th>Embedding capacity (bpp)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>Follow.bmp (16384)</td>
<td>55.29 db</td>
<td>0.062</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp</td>
<td>Skype.bmp (16384)</td>
<td>55.33 db</td>
<td>0.062</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp</td>
<td>Follow.bmp (16384)</td>
<td>49.35 db</td>
<td>0.10</td>
</tr>
<tr>
<td>4</td>
<td>Logo.bmp</td>
<td>Skype.bmp (16384)</td>
<td>46.23 db</td>
<td>0.10</td>
</tr>
</tbody>
</table>
It can be noted from the above statistics that the proposed technique produces stego images of good quality (PSNR > 40 db). It is also noticed that as the amount of data to be embedded is increased, the PSNR (Peak signal to noise ratio) value between the original cover image and the generated stego image decreases. The stego images are observed to be highly imperceptible. The embedding rate of BSDE is 6 bits per pixel. The embedding capacities too have been calculated and take values as shown in tables 4.16, 4.17 and 4.18. When compared with various other data embedding techniques developed by researchers (Cheddad, 2009; Raftari et al., 2012) our proposed method provides a lower embedding capacity in some cases; however achieves a higher PSNR value. This indicates that the proposed BSDE algorithm produces highly imperceptible images. When the proposed methods were compared with other edge based embedding algorithms (Weiqi et al., 2010; Geetha et al., 2013, Shahzad et al., 2014; Islam et al., 2014), it seemed to perform comparatively better.

4.3.2.4 Peak Signal to Noise Ratio Analysis

PSNR values for the stego images obtained using the BSDE algorithm are as displayed in table 4.19. The trade off between the embedding capacity and the PSNR is obvious.
4.3.2.5 Structural Similarity Index Analysis

The structural similarity (SSIM) index is a method used to determine the similarity between two images. The structural similarity index is calculated for each pixel in an image, based on its relationship to other pixels in some neighborhood.

The SSIM was calculated for various original-stego image pairs generated using the proposed BSDE algorithm and some of the results are as tabulated in Table 4.20.
### TABLE 4.20 SSIM values of the original and stego images

<table>
<thead>
<tr>
<th>Sr. No.</th>
<th>Cover image</th>
<th>Secret data</th>
<th>SSIM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp</td>
<td>81920 bits</td>
<td>Red Index - 0.999956</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Green Index – 0.999899</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Blue Index – 0.999976</td>
</tr>
<tr>
<td>2</td>
<td>Fieldview.bmp</td>
<td>Lena.bmp 262144 pixels</td>
<td>Red Index - 0.999987</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Green Index – 0.999956</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Blue Index – 0.999946</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp</td>
<td>Text5.pdf (57344 bits)</td>
<td>Red Index - 0.999947</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Green Index – 0.999948</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Blue Index – 0.999969</td>
</tr>
</tbody>
</table>

It can be concluded from the statistics in table 4.20 that the structural similarity between the original and stego image has good scores.

### 4.3.2.6 Histogram Analysis

The histogram provides a compact summarization of the distribution of data in an image. The color histogram of an image is relatively invariant with translation and rotation about the viewing axis, and varies only slowly with the angle of view. A histogram analysis of the stego images generated using the BSDE algorithm was carried out and results were as shown in figure 4.16, 4.17 and 4.18.
Figure 4.16 Comparison of histogram of LenaRGB.bmp with concealed text data for BSDE technique

Figure 4.17 Comparison of histogram of Fieldview.bmp with LenaRGB.bmp embedded using BSDE technique
Figure 4.18 Comparison of histogram of Logo.bmp with secret text data embedded using BSDE technique

The stego images were created by embedding text as well as images. Screenshots of three sample tests have been documented in this thesis in figure 4.16, 4.17, 4.18. No variation between the original and the stego images could be noticed. Thus it can be concluded that stego images generated by the BSDE algorithm, can withstand the histogram analysis attack.

4.3.2.7 LSB Enhancement attack & Chi-Square attack

These attacks were carried out using VSL and the Simple steganalysis suite softwares (SSS). VSL is a simple, easy to use, graphical block diagramming tool for steganography, steganalysis and watermarking. SSS too is a very used friendly, graphical user interface tool that aids in steganalysis. On experimenting the tools with the proposed BSDE algorithm, the results are as shown in figure 4.19
It can thus be observed that no artifacts could be seen on carrying out LSB enhancement attack, neither any deduction from the results of the Chi Square test.

4.4 Steganalysis using a feature extractor

The proposed edge based steganographic techniques were analyzed using the SPAM (Subtractive pixel adjacency matrix) feature extractor. The SPAM technique uses
Markov chains for blind steganalysis in the spatial domain. It works by adding a low-amplitude independent stego signal such as LSB embedding. The effect of embedding is equivalent to adding to the cover an independent noise-like signal called the stego noise. On providing arguments to the SPAM, subsets of sample transition probability matrices are generated which are then used as features for a steganalyzer implemented by support vector machines (SVM).

The SPAM method exploits the independence of the stego noise. It models the differences between adjacent pixels in natural images. The deviations from this model are identified and reported to be due to steganographic embedding. The steganalyzer is constructed as follows (Pevny et al., 2009); a filter suppressing the image content and exposing the stego noise is applied. Dependences between neighboring pixels of the filtered image are modelled as a higher-order Markov chain. The sample transition probability matrix is then used as a vector feature for a feature-based steganalyzer implemented using machine learning algorithms. SPAM was initially designed to work with gray scale images. It can be extended to color images by creating a specialized classifier for each color plane and fusing their outputs (Pevny and Fridrich, 2007; Pevny et al., 2009).

The construction of steganalyzers based on SPAM features relies on pattern-recognition classifiers. Feature sets generated from the original and stego images, obtained by applying the proposed edge based steganography techniques (ADDE and BSDE) were used to train the support vector machine (SVM), to learn the difference in features caused by secret data embedding. To evaluate the performance of the feature sets obtained, we subjected them to extensive tests on the images in an online database.
having about 5150 images. The images were divided into training and testing set of equal size, so that the cover image and the corresponding stego image were either in the training or in the testing set. We extracted the feature sets using SPAM for each image and used the SVM implemented by Pevny et al., (2010) to classify the class (cover or stego) of the image. For all experiments, we dedicated 80 per cent of the images to training the classifier; the remaining 20 per cent were used for testing.

Before training the SVM, the values of the penalization parameter $C$ and the kernel parameter $\gamma$ needs to be cautiously set as they balance the complexity and accuracy of the classifier. Higher values of $C$ produce classifiers more accurate on the training set but also more complex with a possibly bad generalization. On the other side, a smaller value of $C$ produces simpler classifiers with low accuracy on the training set but hopefully with better generalization. Higher values of the kernel parameter make the classifier more pliable but likely prone to over-fitting the data, while lower values have the opposite effect (Pevny et al., 2010). The standard approach is to estimate the error on unknown samples using cross-validation on the training set on a fixed grid of values and then select the value corresponding to the lowest error. High or low values of both, false positive and false negative indicate a biased relationship. The values of $C$ and $\gamma$ were adjusted to obtain minimum average values for false negatives and false positives.

Testing was done in five fold cross validation. For each technique, cover and corresponding stego sets of images were divided into five parts. SVM is trained for four sets of randomly selected sample images, and the results have been validated for the remaining set.

We used five-fold cross-validation with the multiplicative grid given by

$C \in \{0.001, 0.01, \ldots, 10000\}$,
\[ \gamma \in \{ 2^i : i \in \{-d - 3, \ldots, -d + 3\} \} \]

Where \( d \) is the number of features in the subset.

The steganalyzer performance was then evaluated using the minimal average decision error given by

\[
P_{Err} = \frac{1}{2} \times (P_{FP} + P_{FN})
\]

where \( P_{FP} \) and \( P_{FN} \) refer to the probability of false alarm or false positive (detecting cover as stego) and probability of missed detection (false negative).

Theoretically, the time needed to train the SVM classifier and to perform the classification grows with the cube of the number of training samples and linearly with the number of features (Pevny et al., 2010). The reference implementation of SPAM was downloaded from http://dde.binghamton.edu/download/spam/. The steganalyzers were implemented by SVMs with a Gaussian kernel.

**Table 4.21** SPAM accuracy for edge embedding algorithms

<table>
<thead>
<tr>
<th>Sr No.</th>
<th>Algorithm</th>
<th>Detection accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LSBM</td>
<td>93.06%</td>
</tr>
<tr>
<td>2</td>
<td>Islam <em>et al.</em></td>
<td>51.1%</td>
</tr>
<tr>
<td>3</td>
<td>ADDE</td>
<td>54.69%</td>
</tr>
<tr>
<td>4</td>
<td>BSDE</td>
<td>58.08%</td>
</tr>
</tbody>
</table>
Experimental results in Table 4.21 indicate that LSBM method is detected easily with very high accuracy of 93.06 %. The edge embedding algorithm proposed by Islam et al. for gray scale images detects with a good accuracy of 51.1 %. The proposed edge embedding algorithms detect with an accuracy of 54.69 \% (ADDE) and 58.08 \% (BSDE). On calculating the minimal average decision error it was noted that the steganalyzers employing the second-order SPAM features performed the best.

4.5 Evaluation of Proposed algorithms

4.5.1 Performance and Robustness

The majority of steganography research to date has overlooked the fact that utilizing objects within images can strengthen the embedding robustness – with few exceptions. Based on experimentation, it is found that embedding into such regions produces less distortion to the carrier image compared to embedding in a sequential order or in any other areas. Such phenomena result from the fact that the eye does not respond with equal weight of sensitivity to all visual information. This is consistent with the claim that certain information simply has less relative importance than other information in the human visual system (Gonzalez & Woods, 2002).

Both the proposed algorithms exploit the above facts and are designed to use the edges of an image to hide the data. On evaluation it has been found that the Bit Stream based Data Embedding algorithm (BSDE) outperforms the Arithmetic Division based Data Embedding algorithm (ADDE) in terms of payload capacity. In BSDE we have the embedding rate as 6 bits per pixel; however in ADDE, the embedding rate varies and in the worst case can take a maximum of 6 bits per pixel. ADDE outperforms BSDE in terms of vision perceptibility and robustness. When the proposed methods were subject
to various types of steganalysis tests such as the Chi-square test, LSB enhancement test and the histogram analysis, as documented in the result analysis, no artifacts or variations of any kind were noticed. Regarding statistical analysis, our algorithm gives no evidence of embedding due to statistical parameters.

The proposed algorithms were also compared with works done by some researchers in a similar area and results showed that the proposed techniques can enhance the security when compared, while maintaining higher visual quality of the generated stego images. The Steganography techniques developed work perfectly with minimal distortion to the image quality as can be observed from the test results. Even though the amount of data that can be hidden using these technique is very small as compared to the normal Least Significant Bit Insertion method, this technique scores over the LSB Insertion method as it is much more robust and works with a good retrieval rate. Table 4.22 and 4.23 show different PSNR values spawned by various steganography algorithms based on edge based steganographic methods compared with the two proposed methods (ADDE & BSDE). The graphs plotting the same data is as shown in figure 4.20 and 4.21.
### TABLE 4.22 PSNR values of the original and stego images for LenaRGB.bmp

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text1.txt (32768)</td>
<td>46.08 db</td>
<td>52.02 db</td>
<td>36.01 db</td>
<td>60.83 db</td>
<td>64.52 db</td>
<td>61.14 db</td>
</tr>
<tr>
<td>2</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text2.txt (57344)</td>
<td>45.46 db</td>
<td>48.72 db</td>
<td>35.34 db</td>
<td>59.31 db</td>
<td>61.52 db</td>
<td>58.12 db</td>
</tr>
<tr>
<td>3</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text3.txt (81920)</td>
<td>45.02 db</td>
<td>47.45 db</td>
<td>34.23 db</td>
<td>58.25 db</td>
<td>59.75 db</td>
<td>56.35 db</td>
</tr>
<tr>
<td>4</td>
<td>LenaRGB.bmp (769 kb)</td>
<td>Text4.txt (155648)</td>
<td>43.32 db</td>
<td>45.23 db</td>
<td>30.12 db</td>
<td>56.23 db</td>
<td>55.36 db</td>
<td>53.34 db</td>
</tr>
</tbody>
</table>

### TABLE 4.23 PSNR values of the original and stego images for Logo.bmp

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Logo.bmp (450 kb)</td>
<td>Text1.txt (32768)</td>
<td>44.11</td>
<td>49.77</td>
<td>36.0</td>
<td>59.76</td>
<td>62.16 db</td>
<td>59.07</td>
</tr>
<tr>
<td>2</td>
<td>Logo.bmp (450 kb)</td>
<td>Text2.txt (57344)</td>
<td>43.57</td>
<td>49.12 db</td>
<td>35.58</td>
<td>58.33</td>
<td>59.15 db</td>
<td>57.90</td>
</tr>
<tr>
<td>3</td>
<td>Logo.bmp (450 kb)</td>
<td>Text3.txt (81920)</td>
<td>43.0</td>
<td>48.54 db</td>
<td>33.48</td>
<td>58.05</td>
<td>58.41 db</td>
<td>57.07</td>
</tr>
<tr>
<td>4</td>
<td>Logo.bmp (450 kb)</td>
<td>Text4.txt (155648)</td>
<td>42.17</td>
<td>47.02 db</td>
<td>30.06</td>
<td>56.8</td>
<td>--</td>
<td>54.15</td>
</tr>
</tbody>
</table>
Figure 4.20 Comparison of PSNR values for Lena image using different algorithms.

Figure 4.21 Comparison of PSNR values for Logo image using different algorithms.
Proposed technique is better than many other existing techniques in comparison with PSNR, capacity and immunity to noise as shown experimentally.

4.5.2 Limitations and Merits

The drawback of the proposed methods is the computational overhead involved when the secret data to be hidden is an image. In such situations the application requires the secret image to be compressed before it is embedded in the cover image. This operation demands higher processor speed and memory (RAM). However with the fast development in the hardware manufacturing domain, these problems become trivial.

Another limitation of the method is vulnerability to alteration of the stego image. If the stego image is distorted or compressed, it becomes difficult to extract the hidden data. However, if the secret data is concealed in a manner that does not draw any suspicion and is found to be highly imperceptible, an intruder may not doubt its existence.

At the moment our application supports common file formats such as BMP and PNG. This functionality could be extended to include various other file formats. Targeting only the edges of an image due to their textural complexity for data embedding allows a limited payload, but this can be overcome by extending the functionality to video files.

The proposed techniques hold the following merits:

- Embeds secret data only into the edge regions, which aids to increase imperceptibility.
- Secret data embedded can be either text, image or any binary data.
- Uses 24-bit coloured images for embedding.
- Wavelet based compression method used for image compression.
➢ Output can be in BMP format or the lossless compressed PNG format.

➢ Ensures immunity to various visual and statistical attacks.

➢ Uncompressed and extracted data is of high quality.

➢ Optimization logic used in the data embedding.
SUMMARY AND CONCLUSION

5.1 Summary

With the drastic revolution in technologies, we find ourselves amidst vast amounts of information that needs to be accessed and shared with people all over the world. The success of the Internet facilitates communications of people and also enables illegal users to access data transmitted on the Internet. To protect the confidential data from being illegally accessed by the so called cyber terrorists, various cryptosystems can be used to encrypt the content prior to transmissions. Encryption provides secure channels for communicating entities; however the encrypted data exists in a gibberish form and may attract the attention of the interceptors to break the secret codes. This limitation in data encryption, calls the communicating entities to use steganography to achieve covertness.

The art of steganography refers to hiding confidential data within a carrier in an invisible manner. Because the secret data is integrated invisibly and covered inside other harmless carriers, it is very difficult to detect the data without knowing of its existence. It utilizes digital media such as text, image, audio, video, and multimedia as a carrier or so called cover for secret data. Steganography is an ancient science and finds its applications in various domains. Lots of work is currently being done in the field of steganography and steganalysis; however with the pace at which technology is driving the future, there yet seem to be a lot of unexplored avenues. An exhaustive study of the currently available steganography algorithms was carried out and two novel image edge based steganographic algorithms have been proposed and implemented, namely, Arithmetic Division based data embedding (ADDE) and Bit Stream based data embedding (BSDE).
In a 24-bit bitmap, each pixel is represented by three bytes, representing the red, green and blue colour values for that pixel. To ensure an image is inconspicuous, only the LSBs of the actual pixel data should be altered. Both of the proposed methods work on this principle and modify the LSB’s only, to hide the secret data, which can either be in the form of text or image. Edges in images are areas with strong intensity contrasts; thus the pixels in edges are found to be good candidates for data embedding in comparison to the other smoother areas of an image. An edge map of the cover image is created using the modified Canny edge detection algorithm. ADDE and BSDE, use the last two least significant bits of all the three color channels of red, green and blue, of the edge pixels for the data hiding process.

The proposed ADDE method applies an arithmetic division operation and hides the secret data in the form of quotient and remainder arrays in the edges of a 24 bit cover image. After the arithmetic division operation, sizes of all the bytes of quotient data are analysed. Based on the sizes of the quotient values, we decide the range of the bits to be used for storage. The range would form the optimization criterion. The number of bits to be used to store the quotient data is specified by a flag array. The combination of the optimization criterion and the flag array is said to optimize and specify the number of bits to be used for data storage. This method produces stego images that are highly imperceptible, but provides a fair embedding capacity. In order to overcome the payload limitation of ADDE method, BSDE technique was introduced. In BSDE we use a flag array to specify the size of each byte of data. A size optimization process is carried out for each byte of data. In order to optimize the data, the amount of bits used by each byte of data is mapped to a value set in the flag array. The bit stream is constructed by appending the data bytes one after another, after the size optimization operation is
carried out. The flag array and the bit stream are both stored into the edge pixels of the carrier image using the last two LSB’s.

The proposed techniques have been implemented in Visual C++, using the CxImage library. They have been tested on 200 real time camera clicked digital images and on various other online image databases. A team of ten, technical and non technical people analysed some images and compared them visually. Some checked the printouts using high magnification lens, while others used high end imaging software, like Adobe Photoshop and zoomed the images to 800%. A lot of the suspect conclusions were based on emotional feelings regarding the images. However, none of them could come up, with any strong logical conclusion that proved that the stego images had data embedded within them. The PSNR values achieved too were quite promising, when compared with some other works of similar nature. Various edge embedding and other spatial domain steganography algorithms were used for comparison. The stego images were also subject to various statistical tests, which showed no evidence of the existence of hidden data. When a payload analysis was carried out, BSDE outperformed the ADDE technique; however the ADDE method achieved a higher structural similarity index and a better PSNR value. It was also noticed that the PSNR value decreased as the embedding capacity was increased. Existing tools such as VSL and Steganalysis Suite were used for steganalysis. A steganalyser implemented using support vector machines also formed a part of the testing phase.
5.2 Conclusion

Technology is driving the future... It is up to us to do the steering.

---- Karen Korhorn,
(Computer professionals for social responsibility)

With the growing evolution of internet technology, electronic communication, science and an earnest requirement of concealing defence research work, there is a need for highly secured information exchange, which is the essence of the science of steganography. Steganography refers to the art of inconspicuously hiding data within data. It camouflages the secret data into an unsuspicious digital file. The main goal of steganography is to avoid drawing suspicion to the transmission of a hidden message. In this thesis, a comprehensive study of the currently available steganography algorithms was carried out, a research gap analysis was done and two novel image edge based steganographic algorithms have been proposed and implemented, namely, Arithmetic Division based data embedding (ADDE) and Bit Stream based data embedding (BSDE).

Edges being areas of high intensity variation, prove to be a good surface for data embedding within an image. Both of the proposed methods exploit this fact and modify the LSB’s only, to hide the secret data which can either be in the form of text or image or any other binary form.

The proposed techniques combine aspects of digital signal processing, data hiding, statistical communication theory and human perception. The algorithms have the potential for applications in privacy and security. They have been implemented in Visual C++, using the CxImage library. ADDE and BSDE use the last two least significant bits of all the three color channels of red, green and blue, of the edge pixels for the data hiding process. They have been tested on 200 real time camera clicked digital images
and on various online image databases. A team of ten, technical and non technical people analysed some generated stego images. However, none of them could come up with any strong logical conclusion that proved that the stego images had data embedded within them. The PSNR values achieved too were quite promising, when compared with some other works of similar nature. The stego images were also subject to various statistical tests, which showed no evidence of the existence of hidden data. When a payload analysis was carried out, BSDE outperformed the ADDE technique; however the ADDE method achieved a higher structural similarity index and a better PSNR value. Existing tools such as VSL and Steganalysis Suite were used for enhanced testing. A steganalyzer implemented using support vector machines also formed a part of the testing phase. To sum up, the overall thesis contribution lies in the development of steganographic techniques which are shown to be both, robust and imperceptible.

5.3 Future Scope

The present digital era is in a constant state of evolution. Steganography is considered as a technology that includes major competitive applications. Keeping this in mind, the future work could focus on increasing the robustness of the techniques and extending the functionality.

Some of the areas that could be focussed on, are:

- Use of matrix coding for data embedding.
- Cover image pre-processing for increasing the embedding capacity of images, especially for images acquired under unconstrained illumination conditions.
- Careful selection of an increased number of salient points by targeting video files as covers.
- Study and exploitation of the characteristics of the human perceptive model.
➤ Improve the algorithm to withstand JPEG compression. Look into the lossless JPEG version.
➤ Customize algorithms to function on portable devices.
➤ Optimize algorithms for newly developed file formats.
➤ Optimize algorithm towards thriving attacks, using object oriented embedding.
REFERENCES


Green, B. Canny Edge Detection Tutorial, Available at: http://dasl.mem.drexel.edu/alumni/bGreen/www.pages.drexel.edu/_weg22/can_tut.html


MathWorks, PSNR Available at: http://www.mathworks.in/help/vision/ref/psnr.html


Nevercompressed images data set. Available at: http://www.shsu.edu/~qxl005/New/Downloads


Pizzolato David, CxImage, Available at: http://www.xdp.it/cximage.htm


SourceForge. Virtual Steganographic Laboratory for Digital Images (VSL) Free tool for steganography and steganalysis Available at: http://vsl.sourceforge.net/


APPENDIX A1

SCREEN SHOTS OF APPLICATION GUI

Screenshots continued....
APPENDIX A2

SCREEN SHOTS OF VSL AND SIMPLE STEGANALYSIS SUITE

Screenshot of VSL tool

Screenshots of Simple Steganalysis Suite
APPENDIX A3

PAPERS PUBLISHED


