Error Resilient JPEG2000 Transmission over Wireless Fading Channels

A dissertation submitted by

Andrew John Clemence

In fulfilment of the requirements of

Courses ENG4111 and 4112 Research Project

Towards the degree of

Bachelor of Engineering (Computer Systems)

Submitted: October 2008
Abstract

Transferring images and video over wireless channels is becoming more of an intended use of the medium as personal media move away from the restrictions of cables and transmission standards allow higher transmission speeds. The wireless medium however is not very reliable in the way that it adds unwanted components and noise to the wireless transmission. This paper will examine the performance of the JPWL standard in protecting images from this unwanted noise.

JPEG2000 is the latest JPEG image standard created to replace the popular original JPEG standard. The new standard uses wavelet based compression rather than the block based discrete cosine transform (DCT). The JPWL extension for the wireless transmission of JPEG2000 images was finalised as a standard in 2007. As a result of the recent release of the standard, no papers could be found that utilised or tested it. Due to the time difference between the JPEG2000 standard being released in the year 2000 and the JPWL standard 7 years later, there are several papers proposing protection methods for the standard. Some of these papers show early work into what eventually became the final standard.

The ISO/IEC 15444-11 standard (JPWL) has been written to allow quite a bit of flexibility with regards to specific implementations of the standard. The only real normative (required) parts of the standard include header protection for the main header and packet headers. In the event of a bad header, the whole image will fail decoding whereas bad image data will merely reduce quality. The JPWL standard has numerous informative (optional) sections to add protection to the image codestream. There are several protections available. One of the primary requirements of the standard is that any JPWL codestream can still be read and decoded by any standard JPEG2000 decoder.

The wireless channel used for the testing phase is known as a grey channel. This is a Rayleigh Fading channel with Additive White Gaussian Noise (AWGN) added as a noise component. This channel was used over several Signal-to-Noise Ratio (SNR) to provide similar error rates as those found in some of the previous studies.

The JPWL standard was found to perform very well in protecting the images against quality degradation during transmission over wireless channels. The strength of the protection plays a large part in the protection of the image and should be chosen to suit the particular channel in use. The Default Protection also provides very strong protection for the user who does not wish to choose their own setting. The JPEG and JPEG2000 standards were also tested to show how they suffered from losses incurred in wireless channels.
Limitations of Use

The Council of the University of Southern Queensland, its Faculty of Engineering and Surveying, and the staff of the University of Southern Queensland, do not accept any responsibility for the truth, accuracy or completeness of material contained within or associated with this dissertation.

Persons using all or any part of this material do so at their own risk, and not at the risk of the Council of the University of Southern Queensland, its Faculty of Engineering and Surveying or the staff of the University of Southern Queensland.

This dissertation reports an educational exercise and has no purpose or validity beyond this exercise. The sole purpose of the course pair entitled “Research Project” is to contribute to the overall education within the student’s chosen degree program. This document, the associated hardware, software, drawings, and other material set out in the associated appendices should not be used for any other purpose: if they are so used, it is entirely at the risk of the user.

Prof Frank Bullen
Dean
Faculty of Engineering and Surveying
Certification

I certify that the ideas, designs and experimental work, results, analysis and conclusions set out in this dissertation are entirely my own effort, except where otherwise indicated and acknowledged.

I further certify that the work is original and has not been previously submitted for assessment in any other course or institution, except where specifically stated.

Andrew John Clemence
Student Number: 0050041264

________________________________________
Signature

________________________________________
Date
Acknowledgments

I would like to acknowledge the efforts of my project supervisor, Wei Xiang, for his support and direction in writing this dissertation.

My girlfriend deserves thanks as well for putting up with me talking about it and pretending to listen when I did, sometimes I think she probably did listen too.

My parents, brother and sister have also been a great support and are always available to bounce ideas off where required.
# Table of Contents

1 Introduction ........................................................................................................... 1  
  1.1 Task .................................................................................................................. 1  
  1.2 Purpose ............................................................................................................. 1  
  1.3 Objectives ......................................................................................................... 1  
2 Background ............................................................................................................. 3  
  2.1 What is JPEG? .................................................................................................. 3  
  2.2 The Original JPEG ......................................................................................... 3  
  2.3 JPEG 2000 ....................................................................................................... 4  
  2.4 JPEG2000 Part 11 .......................................................................................... 5  
  2.5 Literature Review ....................................................................................... 6  
  2.6 Looking Forward ............................................................................................ 7  
3 Wireless JPEG2000 ............................................................................................. 8  
  3.1 Overview .......................................................................................................... 8  
  3.2 Channel Model ................................................................................................ 8  
    3.2.1 Rayleigh Fading Channel ......................................................................... 8  
    3.2.2 Additive White Gaussian Noise (AWGN) ............................................ 10  
    3.2.3 Rician Fading ....................................................................................... 10  
  3.3 Modulation ....................................................................................................... 11  
  3.4 Quality Testing ................................................................................................ 11  
  3.5 Options ............................................................................................................. 12  
  3.6 Error Methods Explained ............................................................................... 13  
  3.7 Optimal Result ................................................................................................ 13  
4 Implementation ..................................................................................................... 15  
  4.1 The JPWL codec ............................................................................................ 15  
  4.2 Matlab Coding ................................................................................................ 15  
    4.2.1 Splitting Processes/Multithreading ....................................................... 16  
    4.2.2 Error Handling ....................................................................................... 16  
    4.2.3 The Code ............................................................................................... 17  
5 Results and Discussion ......................................................................................... 18  
  5.1 JPWL Performance ......................................................................................... 18
8.5 Appendix E – JPWL Images .......................................................... 81
8.5.1 Average Boat Images ............................................................... 81
8.5.2 Average Lena Images ............................................................. 115
List of Figures

Figure 1 - Code segment for generating AWGN ................................................................. 10
Figure 2 - 2nd code segment for generating AWGN .......................................................... 10
Figure 3 - SetErrorMode Code from main function ............................................................ 15
Figure 4 - Packet Protection, boat.bmp for average PSNR with Protection set to None .......... 19
Figure 5 - Packet Protection, boat.bmp for minimum PSNR with Protection set to None ....... 19
Figure 6 - Packet Protection, boat.bmp for minimum PSNR at 45 dB SNR with Protection set to None ................................................................. 19
Figure 7 - Packet Protection, boat.bmp for maximum PSNR with Protection set to None ....... 20
Figure 8 - Packet Protection, lena.bmp for average PSNR with Protection set to None .......... 20
Figure 9 - Packet Protection, lena.bmp for minimum PSNR with Protection set to None ....... 21
Figure 10 - Packet Protection, lena.bmp for maximum PSNR with Protection set to None .... 21
Figure 11 - Packet Protection, boat.bmp average PSNR with CRC-16 Protection .................. 22
Figure 12 - Packet Protection, boat.bmp minimum PSNR with CRC-16 Protection ................. 22
Figure 13 - Packet Protection, boat.bmp maximum PSNR with CRC-16 Protection ................. 22
Figure 14 - Packet Protection - lena.bmp average PSNR with CRC-16 Protection .................. 23
Figure 15 - Packet Protection, lena.bmp minimum PSNR with CRC-16 Protection (left) and lena.bmp original (right) ................................................................. 23
Figure 16 - Packet Protection, lena.bmp maximum PSNR with CRC-16 Protection ................. 24
Figure 17 - Packet Protection, boat.bmp average PSNR with CRC-32 Protection ................... 24
Figure 18 - Packet Protection, boat.bmp minimum PSNR with CRC-32 Protection ................ 25
Figure 19 - Packet Protection, boat.bmp maximum PSNR with CRC-32 Protection ................ 25
Figure 20 - Packet Protection, lena.bmp average PSNR with CRC-32 Protection ................... 26
Figure 21 - Packet Protection, lena.bmp minimum PSNR with CRC-32 Protection ................ 26
Figure 22 - Packet Protection, lena.bmp maximum PSNR with CRC-32 Protection ................ 27
Figure 23 - Packet Protection, average PSNR image for 9 dB SNR with default Protection ........ 28
Figure 24 - Average BER for each SNR from the boat.bmp run ........................................... 28
Figure 25 - Packet Protection, boat.bmp with default protection on 12 dB SNR channel (left) and boat.bmp original (right) ................................................................. 29
Figure 26 - Packet Protection, maximum PSNR of boat.bmp from 12 dB SNR channel .......... 29
Figure 27 - Packet Protection, lena.bmp average image for 9 dB SNR with default protection .... 30
Figure 28 - Packet Protection, lena.bmp average image for 12 dB with default protection ....... 30
Figure 29 - Packet Protection, lena.bmp average image for 12 dB with no protection ............ 30
Figure 30 - BER vs PSNR for boat image ............................................................................. 38
Figure 31 - BER vs PSNR for lena image ............................................................................. 39
Figure 32 - SNR vs PSNR for boat image ............................................................................. 39
Figure 33 - SNR vs PSNR for lena image ............................................................................. 40
Figure 34 - boat.jpg 36 dB SNR run 49 ................................................................................. 41
Figure 35 - boat.jpg 33 dB SNR run 2 ................................................................................... 42
Figure 36 - lena.jpg 39 dB SNR run 50 .................................................................................. 43
Figure 37 - lena.jpg 36 dB SNR run 38 .................................................................................. 43
Figure 38 - lena.jpg 33 dB SNR run 50 .................................................................................. 44
Figure 39 - BER vs PSNR for boat jpeg image ...................................................................... 45
Figure 40 - BER vs. PSNR for lena jpeg image ...................................................................... 45
Figure 41 - SNR vs. PSNR for boat jpeg image ...................................................................... 46
Figure 42 - SNR vs. PSNR for lena jpeg image ...................................................................... 46
Figure 43 - boat.j2k 33 dB SNR run 50 .................................................................................. 48
Figure 44 - boat.j2k 45 dB SNR run 49 .................................................................................. 48
Figure 45 - lena.j2k 33 dB SNR run 1 .................................................................................... 49
FIGURE 193 - LENA.J2C 12 dB SNR CHANNEL .......................................................... 115
FIGURE 194 - LENA.J2C 15 dB SNR CHANNEL .......................................................... 115
FIGURE 195 - LENA.J2C 18 dB SNR CHANNEL .......................................................... 116
FIGURE 196 - LENA.J2C 21 dB SNR CHANNEL .......................................................... 116
FIGURE 197 - LENA.J2C 24 dB SNR CHANNEL .......................................................... 116
FIGURE 198 - LENA.J2C 27 dB SNR CHANNEL .......................................................... 117
FIGURE 199 - LENA.J2C 30 dB SNR CHANNEL .......................................................... 117
FIGURE 200 - LENA.J2C 33 dB SNR CHANNEL .......................................................... 117
FIGURE 201 - LENA.J2C 36 dB SNR CHANNEL .......................................................... 118
FIGURE 202 - LENA.J2C 39 dB SNR CHANNEL .......................................................... 118
FIGURE 203 - LENA.J2C 42 dB SNR CHANNEL .......................................................... 118
FIGURE 204 - LENA.J2C 45 dB SNR CHANNEL .......................................................... 119
FIGURE 205 - LENA.J2C 9 dB SNR CHANNEL .......................................................... 119
FIGURE 206 - LENA.J2C 12 dB SNR CHANNEL .......................................................... 119
FIGURE 207 - LENA.J2C 15+ dB SNR CHANNEL ......................................................... 120
FIGURE 208 - LENA.J2C 9 dB SNR CHANNEL .......................................................... 120
FIGURE 209 - LENA.J2C 12 dB SNR CHANNEL .......................................................... 120
FIGURE 210 - LENA.J2C 15 dB SNR CHANNEL .......................................................... 121
FIGURE 211 - LENA.J2C 18 dB SNR CHANNEL .......................................................... 121
FIGURE 212 - LENA.J2C 21 dB SNR CHANNEL .......................................................... 121
FIGURE 213 - LENA.J2C 24 dB SNR CHANNEL .......................................................... 122
FIGURE 214 - LENA.J2C 27 dB SNR CHANNEL .......................................................... 122
FIGURE 215 - LENA.J2C 30 dB SNR CHANNEL .......................................................... 122
FIGURE 216 - LENA.J2C 33 dB SNR CHANNEL .......................................................... 123
FIGURE 217 - LENA.J2C 36 dB SNR CHANNEL .......................................................... 123
FIGURE 218 - LENA.J2C 39 dB SNR CHANNEL .......................................................... 123
FIGURE 219 - LENA.J2C 42 dB SNR CHANNEL .......................................................... 124
FIGURE 220 - LENA.J2C 45 dB SNR CHANNEL .......................................................... 124
FIGURE 221 - LENA.J2C 9 dB SNR CHANNEL .......................................................... 124
FIGURE 222 - LENA.J2C 12 dB SNR CHANNEL .......................................................... 125
FIGURE 223 - LENA.J2C 15 dB SNR CHANNEL .......................................................... 125
FIGURE 224 - LENA.J2C 18 dB SNR CHANNEL .......................................................... 125
FIGURE 225 - LENA.J2C 21 dB SNR CHANNEL .......................................................... 126
FIGURE 226 - LENA.J2C 24 dB SNR CHANNEL .......................................................... 126
FIGURE 227 - LENA.J2C 27 dB SNR CHANNEL .......................................................... 126
FIGURE 228 - LENA.J2C 30 dB SNR CHANNEL .......................................................... 127
FIGURE 229 - LENA.J2C 33 dB SNR CHANNEL .......................................................... 127
FIGURE 230 - LENA.J2C 36 dB SNR CHANNEL .......................................................... 127
FIGURE 231 - LENA.J2C 39 dB SNR CHANNEL .......................................................... 128
FIGURE 232 - LENA.J2C 42 dB SNR CHANNEL .......................................................... 128
FIGURE 233 - LENA.J2C 45 dB SNR CHANNEL .......................................................... 128
FIGURE 234 - LENA.J2C 9 dB SNR CHANNEL .......................................................... 129
FIGURE 235 - LENA.J2C 12 dB SNR CHANNEL .......................................................... 129
FIGURE 236 - LENA.J2C 15 dB SNR CHANNEL .......................................................... 129
FIGURE 237 - LENA.J2C 18 dB SNR CHANNEL .......................................................... 130
FIGURE 238 - LENA.J2C 21 dB SNR CHANNEL .......................................................... 130
FIGURE 239 - LENA.J2C 24 dB SNR CHANNEL .......................................................... 130
FIGURE 240 - LENA.J2C 27 dB SNR CHANNEL .......................................................... 131
FIGURE 241 - LENA.J2C 30 dB SNR CHANNEL .......................................................... 131

xii
List of Tables

TABLE 1 - REED SOLOMAN CODES ................................................................. 12
TABLE 2 - PACKET PROTECTION, boat.bmp AVERAGE PSNR'S ........................................ 32
TABLE 3 - PACKET PROTECTION, boat.bmp MINIMUM PSNRs ...................................... 33
TABLE 4 - PACKET PROTECTION, boat.bmp MAXIMUM PSNRs ....................................... 34
TABLE 5 - PACKET PROTECTION, lena.bmp AVERAGE PSNRs ...................................... 35
TABLE 6 - PACKET PROTECTION, lena.bmp MINIMUM PSNRs ........................................ 36
TABLE 7 - PACKET PROTECTION, lena.bmp MAXIMUM PSNRs ........................................ 37
TABLE 8 - boat.jpg BER AND PSNR VALUES ................................................................ 41
TABLE 9 - lena.jpg BER AND PSNR VALUES ................................................................ 43
TABLE 10 - boat.j2k BER AND PSNR VALUES .............................................................. 47
TABLE 11 - lena.j2k BER AND PSNR VALUES ................................................................ 49
**Glossary of Terms**

- JPEG: Joint Photographic Experts Group; Also the name of the imaging standard they created.
- JPEG2000: The 2\textsuperscript{nd} standard released by the JPEG group, released in the year 2000.
- JPWL: JPEG Wireless, the wireless extension to the JPEG2000 standard.
- PSNR: Peak Signal-to-Noise Ratio, a common measure of image quality, measured in dB.
- SNR: Signal-to-Noise Ratio, a ratio of a signal to the underlying noise, measured in dB.
- dB: Decibels, a measure of ratios.
- DCT: Discrete Cosine Transform, the primary element of the JPEG standard.
- DWT: Discrete Wavelet Transform, the primary element of the JPEG2000 standard.
- MSE: Mean Square Error, a statistical measure used in the PSNR calculation.
1 Introduction

1.1 Task
The aim of this project is to investigate the error resilient properties of the JPEG-2000 standard. Part 1 of the standard incorporates several error resilience tools to detect errors and resynchronise the decoder, but does not provide the ability to correct these errors (Part 11 - JPEG2000 Wireless, 2007). Part 11 of the JPEG 2000 standard (ISO/IEC 15444) is intended to be the mechanism upon which these errors are corrected.

Henceforth that task set out before us is to test the capabilities of Part 11 of the JPEG-2000 standard, also known as JPWL. The standard itself will be investigated to determine the possibilities of improvements for future implementations.

1.2 Purpose
Wireless transmission of data is not a new concept, depending of course on ones concept of new. Satellite transmission of data between continents and now wireless transmission of data in home networks are two of the more well known day to day examples. Wireless transmission however has its drawbacks including interference of other wireless devices and loss of data due to weather, physical objects and, if you go to extremes with the wrong protocols and timings, Doppler shift.

For the purpose of image transfer over channels where higher layer protocols do not look after the appropriate delivery of data through checks such as parity, CRC’s (Cyclic Redundancy Check) or the resending of lost packets (as in TCP/IP) there needs to be other means of protecting the data from loss of portions. For such scenarios as the sending of images to a projector (50 or 60 times a second depending on frame rate) there may be no room for the resending of lost data as the stream of imagery cannot be interrupted. Another very well known example could easily be television, in particular the emergence of digital and HD television.

Due to this possible loss of data without the capability of resending a correct version we need a system to protect and correct such losses. The JPWL extension of the JPEG-2000 standard was created for this explicit purpose, protecting the important portions of the code stream whilst also providing a means of recovering lost data due to transmission errors. We are going to determine how well this works throughout the project and provide ideas for beefing up the specifications if required.

1.3 Objectives
The main objectives of the project are outlined in list form below:
• Report previous error resilience studies to avoid re-inventing the wheel.
• Implement a software test bed to simulate wireless transmissions.
• Use JPWL implementation (Communications and Remote Sensing Lab, 2007) to test the standard.
• Report results of testing with and without the JPWL extensions.
• Investigate decoding JPWL encoded stream using only a Part 1 (not JPWL) decoder (this is a requirement of the JPWL standard).
• Determine best method(s) to interpret quality of images (PSNR, MSE, colour, greyscale, etc.)
2 Background

2.1 What is JPEG?
JPEG stands for Joint Photographic Experts Group, this group represents a variety of companies and academic institutions worldwide (About JPEG, 2007). It was created in 1986. The members of the group meet 3 times a year at least to discuss ideas for the creation of standards. The formal name for the JPEG and JBIG (Joint Bi-level Image experts Group) group is ‘ISO SC29/WG1’.

2.2 The Original JPEG
The original JPEG standard was passed as ISO 10918-1 in 1994 (JPEG, 2008). Work on the standard was originally started as a means to provide photo quality graphics to the predominantly text based screens of the time (1980’s) (JPEG Homepage, 2007). The standard is split up into 4 parts with the first part being the primary codec part. Part 2 sets the rules and checks for Part 1. Part 3 is extensions for the SPIFF file format. Part 4 defines methods for registering parameters used to extend JPEG.

The JPEG standard is based primarily on the Discrete Cosine Transform (DCT). This uses 8x8 blocks to perform the DCT which works on frequency components similar to the Fourier Transform. The DCT itself does not provide any compression; it merely takes the block of pixels and computes values arranged with the most significant portions in the upper left hand corner. Components are taken in Zonal Scan order so that the most important components are read first.

Prior to the DCT the colour space may be changed to the YUV colour space consisting of a luminance and two chroma (colour) components. Reducing the resolution of the chroma components by 2 also reduces data with no noticeable effect to the subjective quality of the image as the human eye is more perceptible to changes in luminance rather than chroma.

Quantization of the result from the DCT allows many of the higher frequency components (in the lower right hand corner of the matrix) to be driven to zero. This allows the images to be compressed further with a form of Huffman (entropy) coding. The dominance of zeros here allows great compression to be achieved.

The obvious disadvantages to the JPEG standard are the 8x8 DCT blocks being quantized. That is all images are created from 8x8 blocks of smaller approximations to the image. If no quantization took place then this approximation would be nearly perfect, that is, no quantization compression where the most losses are encountered.

One main advantage of the JPEG standard is the 8x8 DCT blocks themselves, by compressing the image in portions of 8x8 blocks we do not need to load the entire image into the memory of the compression device. This makes it suitable for embedded devices such as
digital cameras where fast memory comes in small quantities at high prices. Fast memory is required as the DCT, Quantization and Huffman compression all need to be performed sufficiently fast so that the consumer can look at the photo they have just taken.

2.3 JPEG 2000
The JPEG 2000 standard is formally known as ISO/IEC 15444-1; it was created to supercede the original JPEG ISO 10918 standard. There are currently 12 parts in total for the standard however Part 7 has been abandoned (JPEG 2000, 2007). The parts of the standard are as follows from (JPEG 2000, 2007):

- Part 1, Core coding system (intended as royalty and license-fee free - NB NOT patent-free)
- Part 2, Extensions (adds more features and sophistication to the core)
- Part 3, Motion JPEG 2000
- Part 4, Conformance
- Part 5, Reference software (Java and C implementations are available)
- Part 6, Compound image file format (document imaging, for pre-press and fax-like applications, etc.)
- Part 7 has been abandoned
- Part 8, JPSEC (security aspects)
- Part 9, JPIP (interactive protocols and API)
- Part 10, JP3D (volumetric imaging)
- Part 11, JPWL (wireless applications)
- Part 12, ISO Base Media File Format (common with MPEG-4)

The parts that are of most interest to this project are Part 1 for the core coding system and Part 11 where the Wireless protection mechanisms are added.

Similar to the original JPEG standard the first step in the encoding process is to change the colour space of the image from RGB. There are two forms of colour space that can be used, YUV as with the original JPEG standard and a modified version of YUV. Only the modified version is reversible and hence ideal for lossless compression. Reducing the resolution of the chrominance components is optional.

The next step is to tile the image, defining the image as a collection of several or one tiles. This allows the decoder to use less memory as each tile can be worked on individually. Ideally this is useful for embedded systems such as digital cameras but one needs to be careful how small the tiles become as the objective quality decreases and the block effects seen in the DCT may start to invade the supposedly superior wavelet transform encoding resulting in little noticeable difference in the two.

Following the tiling a wavelet transform is performed on each tile. There are two types of the wavelet transform much like the changing of colour space, one which is irreversible and a reversible method for lossless compression. The irreversible method is irreversible due to
quantisation errors. Like the Discrete Cosine Transform the Wavelet Transform does not provide any compression itself but merely makes the components more easily compressible.

Quantisation follows the Wavelet Transform process, much like the original JPEG standard this is where the compression gain is achieved and as such this is the factor that is changed for a change in compression quality or ratio.

The JPEG2000 standard requires more memory for operation as the tiles are larger than simple 8x8 blocks used in the original JPEG standard. It also requires more processor power. These two disadvantages mixed with its patent heavy core coding system (JPEG 2000, 2007) have prevented widespread take-up of the JPEG 2000. The advantages in terms of quality over both lossy and lossless communications links will be determined in this project.

2.4 JPEG2000 Part 11

Formally known as ISO/IEC 15444-11, JPEG Part 11 is commonly referred to as JPWL for JPEG Wireless (Information technology - JPEG 2000 image coding system: Wireless, 2007). This part of the JPEG2000 standard defines the wireless extensions for use of the image standard over lossy communications channels or storage mediums. One of the main requirements of the JPWL extension is that the protected code stream from it can be decoded using the Part 1 decoder. This allows decoding by decoders not implementing the JPWL extensions.

One of the main requirements for the error protection capabilities of the JPWL standard is to protect the main and tile part headers; if an error occurs in these headers then a standard decoder will be unable to decode part or the entire image. In order to correct errors during transmission protection codes, either CRC (Cyclic Redundancy Check) or RS (Reed Solomon) codes are used. These effectively embed parity data within an error protection block (EPB) that allows a certain quantity of bit errors in the headers to be recovered. This is covered in Annex B of the ISO/IEC 15444-11 standard.

The standard also provides for measures to protect the entire code stream as well so that image quality is also protected. The original ISO/IEC 15444-1 standard defines Error Resilience Tools (ERT) which are capable of detecting errors and re-synchronising the decoder, however image quality is lost as the errors are skipped rather than recovered. Such tools for protecting the data include error-resilient entropy coding, Forward Error Correcting (FEC) codes, Unequal Error Protection (UEP) and data partitioning/interleaving. These techniques are all informative, which means they are not a requirement under the JPWL standard.

Error sensitivity data is also embedded within the code stream, normally this is calculated in Part 1 of the JPEG2000 standard. Error sensitivity data is used in error protection techniques such as UEP. This is where more important parts of the code stream, typically the most significant data, is labelled as more sensitive and is given stronger protection than
that of less important portions of the code stream like the least significant data. Error sensitivity is covered in Annex D of ISO/IEC 15444-11.

Errors which cannot be corrected using the JPWL decoder can be sent to a JPWL aware Part 1 decoder for concealment or discarding. These are called residual errors and are covered briefly in Annex E of ISO/IEC 15444-11.

The capacity of this portion of the standard to protect from transmission errors is the primary purpose of this paper. Following testing recommendations may or may not be suggested to improve the standard. Error protection capabilities for the entire code stream can be added as per Annex K of the JPWL standard.

2.5 Literature Review
Several studies have been performed to investigate error concealment methods for the JPEG2000 standard. All of these studies however are pre-2007 when the JPWL standard was released; a couple however mention JPWL and their suitability as possible implementations or additions to the final JPWL standard. Several neglect the requirement to protect the main and tile part headers making them useless for real world use as an error concealment implementation. The main purpose behind analysing previous studies into error concealment and protection for wireless channels is to determine the best method of simulating the transmission over wireless channels.

(Xiangjun & Jianfei) use an improved layered unequal loss protection (UL-ULP) to layer and interleave portions of the JPEG2000 code stream so as to delay the position of the first unrecoverable error. The performance of a progressive image transmission is determined by the location of the first unrecoverable error rather than the number of errors present in the transmission (Xiangjun & Jianfei). They have used a distortion model to determine the distortion present from different levels of errors present on the transmission channel. The part of this paper that interests us the most for the performance testing of the JPWL system is the method upon which the actual transmission was simulated. The distortion model is for a packet loss system so we are not concerned with that here as we are not simulating a packet based transmission system.

The test images are greyscale rather than colour images, all 512X512 pixels. They have used a Gilbert Elliot Channel (GEC) which is a two-state Markov model for the packet loss model. Average loss rates of 5% - 20% were used with an average burst length fixed to 5 packets. Reed Solomon (RS) codes were used for channel coding and a minimum quality threshold of 25 dB was adopted for the quality requirement. 5 quality layers for the JPEG2000 encoding were used and the testing was performed over 1,000 independent simulations.

(Boulgouris, Thomos, & Strintzis, 2006) have taken on an unequal error protection study as a means of looking into the optimal allocation of protection symbols. The major problem with the solution presented here is the assumption that the main header does not need to be
transmitted and hence does not require protection. The method of protection here uses CRC’s, Turbo Codes and RS codes. This paper focus’s heavily on the theoretical side of the optimisation of distortion with respect to differing Turbo and RS code strengths.

The experimental results were performed on a flat-fading Rayleigh channel simulated using the Jakes model. This model is characterised by the average received signal-to-noise ratio and the Doppler spread. A normalised Doppler spread of $f_D = 10^{-5}$ Hz/bps and an average SNR equal to 10dB was used. Reported results were average over 50,000 simulations. The minimum required image quality was taken as 20dB.

(Munadi, Kurosaki, Nishikawa, & Kiya, A Robust Error Protection Technique for JPEG2000 Codestream and Its Evaluation in CDMA Environment, 2003) have proposed an error protection technique that is most appropriate for use as a protection method. It satisfies one of the main requirements of the JPWL specification in that it allows a standard JPEG2000 decoder to decode the code stream. The method of protection here was to implement Forward Error Correcting (FEC) codes to create parity data to protect the main header and Most Significant Layers (MSL) of the code stream. The parity data is then stored in the Least Significant Layers (LSL). It was found that 10% of the total bit rate can be used to hide the parity data. As such an image encoded at 1 bit per pixel (BPP) will have 0.1 BPP parity data and 0.9 BPP image data. 10% allowed an insignificant reduction in quality of the final image. RS codes were used for the parity data.

The method was simulated on a BPSK-CDMA system over 100 independent trials, which is considerably less than the previous papers. Images which cannot be decoded are replaced by images of all zero value pixels which have a PSNR of 5.66 dB. The BPSK-CDMA channel was comparatively described as an Additive White Gaussian Noise (AWGN) channel.

(Munadi, Kurosaki, Nishikawa, & Kiya, ERROR PROTECTION FOR JPEG2000 ENCODED IMAGES AND ITS EVALUATION OVER OFDM CHANNEL, 2003) is similar to their other paper mentioned previously in (Munadi, Kurosaki, Nishikawa, & Kiya, A Robust Error Protection Technique for JPEG2000 Codestream and Its Evaluation in CDMA Environment, 2003). This study, instead of protecting all layers not used for data hiding, seeks to protect the main header and the MSL. The header and MSL are duplicated and placed in the LSL with FEC parity data. As a result, only the MSL and main header are protected at the cost of the LSL. Once again there were 100 independent trials however this time using an Orthogonal Frequency Division Multiplexing (OFDM) system.

2.6 Looking Forward
We have covered the basics to set up the design and research component of the project. From these building blocks it should be a simple task of designing the model, coding it, then reporting the results and findings. In the event that the protection of the JPWL extension is found to be lacking then suggestions of future extensions to the model will be investigated but not implemented as time is likely to be a tight constraint.
3 Wireless JPEG2000

3.1 Overview
Thus far the basics of the project have been covered, that is the background material required to know what is going on. The task before us is simple and, once designed, largely iterative with regards to the testing. The entire testing process will follow a main directional flow as follows:

Image Coding -> Modulation -> Channel Model (errors) -> Demodulation -> Image Decoding

The next chapter on design will cover primarily the modulation technique(s) used as well as the channel model. It is expected that the channel used will be a Rayleigh channel to simulate non line of sight with multiple scatterers (urban environment) and a modulation technique has not been decided but will more than likely be chosen for simplicity. This is because the strength of the modulation technique is not what we are seeking to test here, but rather the effect of creating errors on a channel.

Several tests will be run through this model. We are testing the performance of the original JPEG DCT based standard as well as the JPEG2000 DWT based standard. Within the JPEG2000 standard we can test it with the ERT (Error Resilient Tools) of Part 1 turned on, without them turned on, JPWL protection from Part 11 of the standard, as well as optimising the correction coding of the JPWL section.

3.2 Channel Model

3.2.1 Rayleigh Fading Channel
As mentioned above the Rayleigh model is a good simulation of wireless communication in a non line of sight built up urban environment where there are many scatterers. As such we will use this model for the purpose of our testing as you will find signals from television stations often don’t have line of sight to your house.

The model we will be using is Jakes model, which is based upon a sum of sinusoids. The idea of the model is that N equal strength rays arrive at a moving receiver with uniformly distributed arrival angles $\alpha_n$ (Dent, Bottomley, & Croft, 1993). Each ray has a Doppler Shift which can be calculated from the vehicle speed, carrier frequency and speed of light. However in literature the frequency of the Doppler Shift can be used rather than calculations using undefined carrier frequencies and vehicle speeds (Boulgouris, Thomos, & Strintzis, 2006). The following equation is used for Doppler Shift is:

$$\omega_n = \omega_M \cos \alpha_n$$
Where:

\[ \omega_M = \frac{2\pi f v}{c} \]

This is where we encounter the carrier frequency (f), the vehicle speed (v) and the speed of light (c). To use the Doppler shift \( f_D \) so as to remove the requirement for a vehicle and carrier frequency, we adapt \( \omega_M \) to be (Smith, 2002):

\[ \omega_M = 2\pi f_D \]

Now all we need to calculate \( \omega_n \) is work out \( \alpha_n \), which is defined as:

\[ \alpha_n = \frac{2\pi n}{N} \]

We have \( N_0 \) complex oscillators where \( N_0 \) is:

\[ N_0 = \frac{(N/2 - 1)}{N} \]

Now we can get to the big scary equation:

\[
T(t) = K \left\{ \frac{1}{\sqrt{2}} \left[ \cos \alpha + i \sin \alpha \right] \cos(\omega_M t + \theta_0) + \sum_{n=1}^{N_0} \left[ \cos(\beta_n) + i \sin(\beta_n) \right] \cos(\omega_n t + \theta_n) \right\}
\]

\( K \) is a normalisation constant, (Rayleigh Fading, 2008) uses \( 2\sqrt{2} \) for this value. The coefficients \( \alpha \) and \( \beta_n \) are phases with \( \alpha \) usually set to zero and \( \beta_n \) usually set to \( \frac{\pi n}{(N_0+1)} \) to give a zero cross-correlation between the real and imaginary components. \( \theta_n \) are the initial phases and normally set to zero.

A modification to the model for a single waveform has been proposed by (Dent, Bottomley, & Croft, 1993), this modification seeks to remove the deterministic nature of the model but primarily the cross-correlation between multiple waveforms. For our model we will be looking at a single waveform rather than multiple waveforms. The value of \( \alpha_n \) is changed to \( \frac{2\pi(n-0.5)}{N} \). This leads to the model:

\[
T(t) = \sqrt{2} \sum_{n=1}^{N_0} \left[ \cos(\beta_n) + i \sin(\beta_n) \right] \cos(\omega_n t + \theta_n)
\]

Where \( N_0 \) is \( N/4 \) and \( \beta_n \) is given by \( \frac{\pi n}{N_0} \). Randomising \( \theta_n \) then provides us with different waveform realisations. This is the model we intend to use for our channel model.
3.2.2 Additive White Gaussian Noise (AWGN)

Additive White Gaussian Noise is added to the transmitted signal, hence the term additive, to simulate the addition of noise within the channel. Thermal noise of the antennas is one source of noise within the channel that can be modelled with AWGN. White noise refers to the fact that the noise signal is generated independently from the transmitted signal and its power spectral density is flat, which means the autocorrelation of the signal with itself at any lag other than zero time is zero. The noise samples take on a Gaussian distribution. Together these terms describe the reasoning for labelling this channel model as Additive White Gaussian Noise (Additive White Gaussian Noise Channels, 1995).

Knowing what AWGN is, we now need to know how to calculate it. The website (Donadio) presents two methods of generating White Gaussian Noise, the simplest of which is the Central Limit Theorem. That is the sum of N random numbers will approach a normal distribution as N approaches infinity. The following code segment is directly off the website to demonstrate the code which will do this, where the function uniform provides a true random number between 0 and 1.

```
x=0
for i = 1 to N
    U = uniform()
    X = X + U
end

/* for uniform randoms in [0,1], mu = 0.5 and var = 1/12 */
/* adjust X so mu = 0 and var = 1 */
X = X - N/2
X = X * sqrt(12 / N)
```

Figure 1 - Code segment for generating AWGN

(Donadio) goes on to describe that X will be a unit normal random, to generate the noise with a specified mean and variance we use the code:

```
X' = mean + sqrt(variance) * X
```

Figure 2 - 2nd Code segment for generating AWGN

It is recommended to use values of N greater than 20 for good results, as stated earlier in the Central Limit Theorem, the larger the value of N, the greater the approximation of a Gaussian distribution. The performance of these code segments will not impact this project as a script from Matlab Central for the channel model encompasses the AWGN requirements.

3.2.3 Rician Fading

Rician Fading is another model that could be used. This model is similar to the Rayleigh model in that it involves more than one signal arriving at the receiver. Just like the Rayleigh
model has a Rayleigh distribution, the Rician model has a Rician distribution. It is commonly used for line of sight applications (Rician Fading, 2008) due to a strong dominant component; this would make it ideal for computer to projector simulation but not so for mass media transmission systems. Computer to projector transmission can take on a communications based protocol with retransmission request and hence will not be the focus of our research into the error concealment and correcting capabilities of the JPEG standards.

3.3 Modulation
There are 3 main kinds of modulation we can choose from. They are Frequency Shift Keying (FSK), Amplitude Shift Keying (ASK) and Phase Shift Keying (PSK) (Langton, 2005). Data is modulated from a digital form onto an analogue carrier wave for transmission over wireless mediums. This is due to the nature of wireless mediums allowing the transmission of Radio Frequency (RF) waves which are analogue in nature and not digital. The method that will be used here is QAM or Quadrature Amplitude Modulation.

This method involves modulation two carrier waves by changing the amplitude of these waves. The waves are 90° out of phase with each other making them quadrature carriers (Quadrature amplitude modulation, 2008).

3.4 Quality Testing
There are several methods which can be used to test the quality of an image. The common few, Mean Squared Error (MSE), Mean Absolute Error (MAE) and Peak Signal-to-Noise Ratio (PSNR) are very similar; they all rely on the difference between the pixel value of the original image and the reconstructed image. For the purpose of our quality testing we will be using the PSNR as used previously in (Boulgouris, Thomos, & Strintzis, 2006) and (Xiangjun & Jianfei). The PSNR is defined at (Peak signal-to-noise ratio, 2008) as:

$$\text{PSNR (dB)} = 10 \log_{10} \frac{\text{MAX}_i^2}{\text{MSE}}$$

MAX_i is the maximum value a pixel can take in the image. A standard greyscale will have an 8 bit resolution with a maximum value of 255. The MSE is defined as:

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} ||I(i,j) - K(i,j)||^2$$

It can be seen from the equations that a larger PSNR value indicates a better image quality. This is due to the division by the MSE in the PSNR equation. The MSE decreases with smaller differences between the original image pixels and the final image pixels. Division with a denominator approaching zero yields a result approaching infinity, thus a replication of the original image produces a result of infinity, but is in fact undefined.
3.5 Options

The formats we are testing here are the original JPEG standard, the JPEG2000 standard and the JPWL standard. The JPEG and JPEG2000 standards are relatively simple; however the JPEG2000 standard can be tested both with and without its Error Resilience Tools.

The JPWL standard lists several protection codes which are available in the OpenJPEG JPWL codec. There are two types of CRC protections currently available, CRC-CCITT (X.25) 16 bits CRC and Ethernet CRC 32 bits. The available Reed-Solomon codes are shown in Table 1.

<table>
<thead>
<tr>
<th>Available Reed Soloman Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS(37,32)</td>
</tr>
<tr>
<td>RS(38,32)</td>
</tr>
<tr>
<td>RS(40,32)</td>
</tr>
<tr>
<td>RS(43,32)</td>
</tr>
<tr>
<td>RS(45,32)</td>
</tr>
<tr>
<td>RS(48,32)</td>
</tr>
<tr>
<td>RS(51,32)</td>
</tr>
<tr>
<td>RS(53,32)</td>
</tr>
<tr>
<td>RS(56,32)</td>
</tr>
<tr>
<td>RS(64,32)</td>
</tr>
<tr>
<td>RS(75,32)</td>
</tr>
<tr>
<td>RS(80,32)</td>
</tr>
<tr>
<td>RS(85,32)</td>
</tr>
<tr>
<td>RS(96,32)</td>
</tr>
<tr>
<td>RS(112,32)</td>
</tr>
<tr>
<td>RS(128,32)</td>
</tr>
</tbody>
</table>

There are also predefined codes that can be used in the event that one does not wish to define their own protection codes. These are summarised as:

- Reed Soloman RS(160,64) to be used for the first EPB marker segment of the main header
- Reed Soloman RS(80,25) to be used for the first EPB marker segment of a tile-part header
- Reed Soloman RS(40,13) to be used for the other EPB marker segments of both the main header and the tile-part header

The EPB is the Error Protection Block of the JPWL codestream. This is the location within the codestream that specifies the error protection to be used, its primary purpose is to protect the main header and tile-part headers. Protecting the bitstream (image information, not headers) is an optional extra. The error protection data is also stored here but this does not concern us for our testing purposes.

The Error Sensitivity Descriptor (ESD) marker contains sensitivity information for the codestream. This marker holds the remaining parameters for the OpenJPEG JPWL codec. It defines both the addressing mode and the data sensitivity method. The addressing mode can be either packet mode, byte-range mode or packet-range mode. The data sensitivity method can be:

- Relative Error Sensitivity
- Mean Square Error (MSE)
- MSE Reduction
- Peak Signal-to-Noise Ratio (PSNR)
- PSNR Increase
- MAXERR (Absolute Peak Error)
- TSE (Total Squared Error)
The error sensitivity information is stored as two bytes for packet mode, however due to the large number of packets byte-range or packet-range is also possible to spread the sensitivity information more efficiently. A value of zero is reserved for the headers as they are the most important and a value of $2^{16}$-1 is reserved for dummy sensitivity records. The above methods are used to determine the level of importance of the remaining data segments, which are then assigned values in the range $1 - 2^{16}$-2 with 1 being the most important and $2^{16}$-2 the least important (Grangetto, Magli, & Olmo, 2003).

### 3.6 Error Methods Explained

The first of the error methods is not tied to any specific error metric (Grangetto, Magli, & Olmo, 2003). Relative error is merely the ratio between the absolute error and the original value where the absolute error is the difference between the approximation and the original value (Harder, 2005). The other methods are all methods of metrically calculating the error.

MSE stands for Mean Squared Error; this is the method of squaring the difference between the original and reconstructed signals (pixels). The idea of squaring the difference is to provide a larger weighting to larger differences as well as to make all values positive. The formula for MSE was previously shown in the Quality Testing section. MSE Reduction is a method that seeks to reduce the MSE.

Peak Signal-to-Noise Ratio (PSNR) has been shown previously in the Quality Testing section. It effectively takes the MSE, throws it in a ratio with the maximum pixel value and turns the result into a decibel (dB) value with the log function. As was shown previously the idea is that a larger PSNR value has a better approximation to the original. As a result the next method is PSNR Increase, also known as PSNR Increment; this is basically to increase the PSNR of the image.

MAXERR or Absolute Peak Error is the method of calculation the Maximum Absolute Error. The absolute error was described previously; this method merely takes the maximum of this rather than the other methods which prefer to take means (averages).

The Total Squared Error (TSE) is effectively the sum of the squared errors. This is similar to the Absolute Peak Error but sums and squares the errors as well. Once again a mean or average error is not taken but rather a total error.

### 3.7 Optimal Result

There are several levels that need to be independently tested. These are as follows:

- Header Protection
- Packet Protection
- Data Sensitivity
- Addressing Mode
To determine the optimal encoding we are looking for the best quality to size ratio. Size has to be taken into account, as the best quality would logically result from the largest codestream. As a result the ratio that will be computed to determine the optimal method from each category will be to divide the PSNR by the size in bytes of the codestream.

Each category will be tested by maintaining the remaining categories at a constant value. The header and packet protection will be maintained as predefined with relative error sensitivity and a packet addressing mode.

Unfortunately due to time constraints only the packet protection was able to be tested.
4 Implementation

The code will be written in Matlab for the sake of coding speed rather than efficiency. It will be done by a modular approach where possible; this will allow an easy implementation of the code and easy debugging where required. The full code can be found in the Appendix.

4.1 The JPWL codec

The JPWL codec proposed many problems when working with code streams that had been destroyed by a noisy channel (our channel model). A bug in the codec required an index file to be specified where this should not be necessary. Obtaining the SVN repository of the codec and compiling the latest code fixed this issue.

The noisy code streams had a habit of regularly providing general protection faults which required the user of the windows computer running the simulation to press a button to close the program. A workaround for this was to insert the Windows API call “SetErrorMode” to turn off the general protection fault messages. This line of code is shown in Figure 3 and was placed in the ‘main’ function of the JPWL_j2k_to_image executable.

```matlab
/* Disable Error Message Dialogues */
SetErrorMode(0x0002);
```

Figure 3 - SetErrorMode Code from main function

Changing the header protection appeared to cause the codec to often enter an endless loop of bad markers; this was only fixed by manually ending the process via the Windows Task Manager.

4.2 Matlab Coding

Matlab provides a coding environment designed for rapid development. The ease of use of the available functions and coding constructs places it in a very good position for fast prototyping. This has allowed a reasonably speedy code to be constructed to perform the required simulations. The speed of Matlab running its own scripts however is much slower than that of a compiled program, this is a qualitative observation and no actual figures were recorded to verify this. Compiling Matlab scripts using the Matlab Compiler helps to speed them up and also proved useful for a multi process approach for running on a multi-core CPU.

The general Matlab program will test the individual parts of the codec (effectively the standard); within each of these tests will be several loops to perform the test through several channel models. Only the packet protection was tested due to time constraints on the project.
The simulation ran through several loops to perform an array of the appropriate tests. These appear as follows:

- For each image
- For each independent simulation
- For each protection method
- For each SNR

For the JPEG and JPEG2000 simulations, the protection method loop was removed.

### 4.2.1 Splitting Processes/Multithreading

In an ideal world Matlab would be perfect at choosing where to separate the program into several functional threads. However Matlab is still largely single threaded even when the multithreading option is turned on in the Matlab preferences. This means that each of the above loops would take considerable time to run sequentially even on a multi-core system. To take advantage of a multi-core system we need to split the code up manually into separate processes, each can then be compiled using the Matlab compiler.

It was decided that the easiest way to take advantage of a dual core CPU was to split the two images into their own processes. As a result the ‘for each image’ loop was altered to first work with image number one and compiled as such, then to work with image number two and compiled as such. This results in two executables which can be run side by side on the same machine or separate machines to approximately half the total run time. Should more cores become available then further splitting would be possible, each of the ‘for’ loops can all be split into multiple processes. In the C programming language this would be simple with a ‘fork’ command to spawn child processes; however Matlab is not believed to support such multi-threading commands.

### 4.2.2 Error Handling

The JPWL codec, as mentioned previously, had a very bad habit of falling over consistently with bad code streams. As a result the code needed to be structured so that the program could simply skip over a bad result and continue operation. To do this the system call calling the codec needed to capture the output status, a test on this status then informed the program whether to continue attempting to decode and continue with the current iteration or whether to give up and skip to the next iteration.

Even when the JPWL codec returns a successful status it was found that some of the decoded images were corrupt. That is, Matlab was not able to read them successfully without encountering its own errors. To overcome this, the imread function to read in the bmp decoded image was placed in a try-catch statement, which attempted to execute imread and upon an error the code in the catch section would be executed. Using this to signal a variable flag allowed a further test to determine if it was safe or not to try to use
this image data, logically it would not be safe if the image data could not be read into the program.

A try-catch statement was also required for the PSNR calculation as this would fail if the image had faulty dimensions. That is if the codestream had been destroyed to the point that its dimensions had been lost, even if only by a few pixels, the PSNR calculation would be undefined and return an error.

4.2.3 The Code
The full code can be found in the appendices. Commenting within the code will guide anyone who wishes to read it as to how it works. The overall approach is quite simple with the program primarily pasting together a channel model onto the codec then seeing what happens.
5 Results and Discussion

The results from the testing will tell us approximately what is going on with regards to the performance of the codec. The coding unfortunately ignored failed decodes rather than recording them as a failed decode and as such the decoding rate cannot be commented on. This was done due to the slightly unstable nature of the codec used and the fact that a failed decode could be a result of either the codec failing or the codestream failing, the two cannot be distinguished.

5.1 JPWL Performance
The packet protection of the JPWL standard will provide us with the best indication of the performance of the standard. In the event that the header fails to be properly protected, the image will fail to decode and as such the packet protection be completely irrelevant. The first set of results that will be investigated are those which left the data sensitivity and addressing mode as the defaults for the OpenJPEG codec.

It will be assumed that 20dB is considered the minimum acceptable image quality.

5.1.1 No Protection
For the boat image with no packet protection mechanism, the minimum SNR required to achieve the 20 dB average PSNR mark is 33 dB, this image can be seen in Figure 4. The minimum PSNR of all iterations was able to achieve 20 dB at 42 dB SNR however 45 dB SNR had a considerably lower PSNR. The 42 dB minimum PSNR is shown in Figure 5 whereas the 45 dB SNR minimum PSNR image is shown in Figure 6. The 45 dB SNR image has a PSNR of 14.84 dB. This demonstrates the reasoning behind running multiple iterations of each condition to allow for the random nature of wireless channels. 27 dB SNR was able to reach 20 dB PSNR when the maximum PSNR’s of each run are taken into account, this image can be seen in Figure 7.
Figure 4 - Packet Protection, boat.bmp for Average PSNR with Protection set to none

Figure 5 - Packet Protection, boat.bmp for Minimum PSNR with Protection set to none

Figure 6 - Packet Protection, boat.bmp for Minimum PSNR at 45 dB SNR with Protection set to none
It can be seen from these images that even from similar PSNR values the degradation due to fading and noise can vary significantly. These images show the effect of transmission with no protection to the image bitstream. The headers were protected but this merely ensures the decidability of the images and has little effect on the data itself.

The lena images had similar results to those of the boat images. The average PSNR values reached 20 dB PSNR when the SNR was 30 dB and this image can be seen in Figure 8. The minimum PSNR’s reached 20 dB PSNR when the SNR was at 42 dB and this image is also available in Figure 9. Once again the 45 dB SNR test managed to achieve an even lower PSNR coming in at a little over 3 dB less. This image has not been shown here. The maximum PSNR’s reached the 20 dB mark at an SNR of 24 dB and can be seen in Figure 10.

The JPEG2000 codestream image size for no packet protection came out to be 142 kB according to Windows Vista.
Standard Global Systems for Mobile communications (GSM) networks have a Signal-to-Noise Ratio of approximately 9 dB (Carvalho & Rebelo, 2004). As can be seen from the results of both images with no protection, we are a long way of achieving a quality image over such a network with no protection enabled. Next we will have a look at the CRC protections provided by the JPWL standard and codec. The two protections offered are CRC-16 and CRC-32.

5.1.2 CRC Protection
Cyclic Redundancy Checks are an error detection mechanism with no real means of correcting such errors. The purpose of protecting images using CRC’s is merely to ensure that packets with errors are not decoded so that they may not diminish the overall quality of the image, in theory.

The first lot of results we shall look at are the images for the CRC-16. The average PSNR value to get to 20 dB PSNR required an SNR of 33 dB as seen in Figure 11. Figure 12 shows us the image for minimum PSNR which required an SNR of 45 dB to reach 20 dB SNR. This
was lucky as this is the highest SNR used for our tests. The maximum PSNR reached 20 dB at an SNR of 24 dB. This image can be seen in Figure 13.
From the boat image alone it is obvious that the results of CRC protection are not that great when compared with no protection at all.

Figure 14 - Packet Protection - lena.bmp Average PSNR with CRC-16 Protection

![Average PSNR with CRC-16 Protection](lena.bmp)

The Average PSNR for the lena image with CRC-16 protection reached 20 dB PSNR at an SNR value of 30 dB and the image closest to this average value can be seen in Figure 14. The minimum PSNR reached 20 dB at an SNR of 42 dB as seen in Figure 15. The maximum PSNR reach 20 dB at an SNR of 27 dB.

Figure 15 - Packet Protection, lena.bmp Minimum PSNR with CRC-16 Protection (left) and lena.bmp Original (right)

![Minimum PSNR with CRC-16 Protection](lena.bmp)

The image with the Minimum PSNR’s that reached 20 dB quite surprisingly stands out from the other images we have seen. The image appears to be of quite decent quality even though it only has a PSNR of 20.914 dB. However the PSNR equation as seen previously works on the difference of each pixel from the original value. The image upon close inspection is darker in parts than the originals dark areas and lighter in some of the bright areas. This accentuation, so to speak, can be said to have changed the DC offsets of each section from that of the original thus increasing the overall differences. This would
lead us into the argument as to whether metric measures are capable of accurately defining image quality but that argument is beyond the scope of this study.

![Figure 16 - Packet Protection, lena.bmp Maximum PSNR with CRC-16 Protection](image)

It can be seen from the results that the protections of the CRC-16 protected images appear to be comparable to that of no protection. In some cases no protection yields better performance than that of the CRC-16 protection. This is notable with the minimum PSNR for 42 dB SNR. The minimum PSNR for no protection on this channel was 24.784 dB whereas for the CRC-16 protection it was 20.914 dB. This could be considered an expected result due to the CRC mechanism not providing any error correction capabilities.

The images for the CRC-32 protections will now be looked at to determine if they have any better results. The SNR required for the average PSNR’s to reach 20 dB PSNR was found to be 33 dB and this image can be seen in Figure 17. For the minimum PSNR’s the SNR was 45 dB. The maximum PSNR values required an SNR of 27 dB to reach the required 20 dB.

![Figure 17 - Packet Protection, boat.bmp Average PSNR with CRC-32 Protection](image)
The results are still looking very similar to those of no protection at all as was expected from a protection method that performs no data recovery. For the sake of completeness we will have a look at the lena images for the CRC-32 protection to see if they tell a different story.

The average PSNR values reached the 20 dB PSNR mark at an SNR of 33 dB as can be seen in Figure 20. The minimum PSNR values reached this point at an SNR of 39 dB as per Figure 21 and the maximum PSNR value was reached at an SNR of 24 dB.
Once again we have seen poor quality images that have suffered greatly due to the conditions of the wireless channel model. The CRC protections have proven to be highly ineffective in protection the bitstream whatsoever with results very similar to those of no protection mechanism at all. The minimum PSNR group showed that the CRC-16 images required a 45 dB SNR channel to achieve a PSNR image of more than 20 dB whereas no protection at all required on 42 dB for the same PSNR cut-off. This shows that the performance of the CRC cannot be considered even marginally better due to the fact that in some cases, the CRC performed worse than that of no protection. This merely proves that a cyclic redundancy check is not a viable option of protecting data where a retransmission request is not available.
5.1.3 Default Protection

The OpenJPEG codec allows the user to choose an option for the default protection. This does however create an image about 435 kB in size making it 3 times the size of the original image. With such extra padding the size of 2 more entire identical images we would expect some very good protection from this method.

The average PSNR for all channels of 12 dB SNR and above is in the order of infinity. This is surprising as we would have expected the compression algorithm to take some quality loss of its own but the compression method chosen with the number of quality layers must have prevented such a loss occurring. We cannot argue with the raw data however.

As mentioned earlier, the standard SNR for a GSM channel is 9 dB. As such the results of the default protection, on average, are sufficient for such a network. This image can be seen in Figure 23. The quality of this image appears similar to that of the previous images we have encountered, however this image comes from a much noisier channel. The difference in Bit Error Rate between this channel and those encountered in the no protection and CRC protection images is evident in Figure 24 when you consider how different the spacing between the 9 dB and over 20 dB points are.
The effect of a 12 dB channel shows the considerable protection offered by the default mode in Figure 25. This figure offers a visual representation of infinite PSNR by placing the image beside the original, there is metrically no difference between the two images. Figure 26 presents the same image from the same 12 dB channel with no protection. This shows the true performance of the enacted protection mechanisms.
The lena image also yielded similar results for the case of default protection. Figure 27 shows the lena image from the 9 dB channel with the average PSNR for the images from that channel. Whilst not as healthy as the 12 dB channel image shown in Figure 28 it is still very good compared to no protection at all. A lena image with no protection from the 12 dB range with the average PSNR for this run is shown in Figure 29.
Figure 27 - Packet Protection, lena.bmp average image for 9 dB SNR with Default Protection

Figure 28 - Packet Protection, lena.bmp average image for 12 dB with Default Protection

Figure 29 - Packet Protection, lena.bmp average image for 12 dB with No Protection
### 5.1.4 Reed Solomon Protections

The Reed Solomon codes provide several different layers of protection. The larger the protection capability the larger the file size becomes. This is evident from the default protection providing us with an image more than 3 times the size of the original image. Due to the sheer number of images and results provided from the Reed Solomon testing the results are best presented in a table. These tables consist of the CRC, no protection and default protection values as well because it is a manipulation of all the data output.

For the boat image the following results were noted. On average, at a 9 dB SNR channel, the minimum strength required to reach 20 dB PSNR is a RS(75,32) code. For the maximum PSNRs this can be reached with the RS(64,32) code and for the minimum PSNRs even the RS(128,32) code managed a PSNR of between 5 and 6 dB. A 9 dB channel is perfectly capable of handling images with infinite PSNR with RS(96,32) providing the infinite values for both the average and maximum PSNRs. The RS(75,32) code provides a file size of 331 kB which is quite smaller than that of the default protection size of 435 kB. This is 76% the size of the Default Protection for a similar level of protection.

Channels of 27 dB SNR and above, on average, feature infinite PSNRs for all Reed Solomon protection strengths. This is when compared to an average PSNR of 14.97 dB for no protection. The size of a RS(37,32) code is also much smaller than that of the Default Protection which provides infinite PSNR on average for all channels of 12 dB SNR and greater. This size comes in at 164 kB which is only 38% the size of a RS(128,32) code and 1.15 times the size of the no protection file. An SNR of 21 dB is all that is required for all Reed Solomon codes to have a PSNR of more than 20 dB. The size advantage of low strength Reed Soloman codes is quickly eroded as the channel gets noisier.

The lena image provided similar results with the Reed Solomon code of RS(75,32) being the minimum average for the PSNR to reach 20 dB on the 9 dB SNR channel. The maximum PSNRs and minimum PSNRs also provide similar results. Once again the RS(96,32) is also capable of providing images with no error on a 9 dB SNR channel.

The lena image with no protection comes in at 139 kB. The RS(37,32) image is 161 kB which comes in at 1.16 times the no protection image. The RS(75,32) lena image has a size of 325 kB which is also 76% the size of the default protection which comes in at 426 kB. The no protection size comes in at 33% the size of the image with default protection.

The results of the lena image are very similar to those of the boat image which assists in the verification of these results.
<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>Rs(37,32)</th>
<th>Rs(38,32)</th>
<th>Rs(40,32)</th>
<th>Rs(43,32)</th>
<th>Rs(45,32)</th>
<th>Rs(48,32)</th>
<th>Rs(51,32)</th>
<th>Rs(53,32)</th>
<th>Rs(56,32)</th>
<th>Rs(64,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>5.680</td>
<td>20.264</td>
<td>5.758</td>
<td>5.755</td>
<td>5.659</td>
<td>5.796</td>
<td>5.597</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>5.641</td>
<td>Inf</td>
<td>5.549</td>
<td>5.809</td>
<td>5.646</td>
<td>5.531</td>
<td>5.592</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>7.73E-03</td>
<td>5.445</td>
<td>Inf</td>
<td>5.513</td>
<td>5.564</td>
<td>5.593</td>
<td>8.245</td>
<td>13.171</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>3.91E-03</td>
<td>5.605</td>
<td>Inf</td>
<td>5.537</td>
<td>5.396</td>
<td>11.379</td>
<td>22.301</td>
<td>31.141</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>1.98E-03</td>
<td>5.965</td>
<td>Inf</td>
<td>5.807</td>
<td>6.723</td>
<td>25.526</td>
<td>37.934</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>2.50E-04</td>
<td>19.572</td>
<td>Inf</td>
<td>18.098</td>
<td>19.046</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>24.716</td>
<td>Inf</td>
<td>24.161</td>
<td>22.171</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>6.27E-05</td>
<td>27.345</td>
<td>Inf</td>
<td>25.491</td>
<td>26.669</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>3.14E-05</td>
<td>31.636</td>
<td>Inf</td>
<td>31.133</td>
<td>31.119</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1.58E-05</td>
<td>37.105</td>
<td>Inf</td>
<td>35.489</td>
<td>36.017</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>7.97E-06</td>
<td>41.779</td>
<td>Inf</td>
<td>41.356</td>
<td>43.109</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>Rs(75,32)</th>
<th>Rs(80,32)</th>
<th>Rs(85,32)</th>
<th>Rs(96,32)</th>
<th>Rs(112,32)</th>
<th>Rs(128,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>21.271</td>
<td>30.075</td>
<td>30.676</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.73E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.91E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.98E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>9.92E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.99E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.50E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>6.27E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>3.14E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>1.58E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>7.97E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>SNR</td>
<td>BER</td>
<td>No Protection</td>
<td>Default Protection</td>
<td>CRC-16</td>
<td>CRC-32</td>
<td>RS(37,32)</td>
<td>RS(38,32)</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>---------------</td>
<td>--------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>9</td>
<td>2.82E-02</td>
<td>5.119</td>
<td>4.769</td>
<td>5.165</td>
<td>5.141</td>
<td>5.125</td>
<td>5.051</td>
</tr>
<tr>
<td>12</td>
<td>1.48E-02</td>
<td>5.219</td>
<td>41.646</td>
<td>4.328</td>
<td>5.174</td>
<td>3.962</td>
<td>4.444</td>
</tr>
<tr>
<td>18</td>
<td>3.74E-03</td>
<td>3.81</td>
<td>Inf</td>
<td>4.442</td>
<td>4.103</td>
<td>4.011</td>
<td>5.17</td>
</tr>
<tr>
<td>21</td>
<td>1.87E-03</td>
<td>4.985</td>
<td>Inf</td>
<td>4.608</td>
<td>3.983</td>
<td>4.466</td>
<td>18.753</td>
</tr>
<tr>
<td>24</td>
<td>8.94E-04</td>
<td>4.197</td>
<td>Inf</td>
<td>3.93</td>
<td>4.449</td>
<td>3.865</td>
<td>27.066</td>
</tr>
<tr>
<td>27</td>
<td>4.25E-04</td>
<td>4.183</td>
<td>Inf</td>
<td>4.598</td>
<td>3.851</td>
<td>34.916</td>
<td>47.251</td>
</tr>
<tr>
<td>30</td>
<td>1.99E-04</td>
<td>4.146</td>
<td>Inf</td>
<td>4.993</td>
<td>5.214</td>
<td>51.607</td>
<td>60.143</td>
</tr>
<tr>
<td>33</td>
<td>9.23E-05</td>
<td>4.804</td>
<td>Inf</td>
<td>4.314</td>
<td>3.672</td>
<td>61.794</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>4.37E-05</td>
<td>4.104</td>
<td>Inf</td>
<td>5.1</td>
<td>5.09</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>1.73E-05</td>
<td>5.118</td>
<td>Inf</td>
<td>4.233</td>
<td>5.189</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>6.54E-06</td>
<td>22.361</td>
<td>Inf</td>
<td>4.119</td>
<td>6.435</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>5.76E-07</td>
<td>14.843</td>
<td>Inf</td>
<td>27.577</td>
<td>26.921</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(43,32)</th>
<th>RS(45,32)</th>
<th>RS(48,32)</th>
<th>RS(51,32)</th>
<th>RS(53,32)</th>
<th>RS(56,32)</th>
<th>RS(64,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.82E-02</td>
<td>5.059</td>
<td>4.381</td>
<td>4.052</td>
<td>3.907</td>
<td>3.836</td>
<td>4.393</td>
<td>3.809</td>
</tr>
<tr>
<td>12</td>
<td>1.48E-02</td>
<td>3.763</td>
<td>4.131</td>
<td>4.991</td>
<td>4.316</td>
<td>4.496</td>
<td>6.902</td>
<td>33.998</td>
</tr>
<tr>
<td>15</td>
<td>7.48E-03</td>
<td>3.885</td>
<td>13.444</td>
<td>4.278</td>
<td>23.12</td>
<td>18.865</td>
<td>57.76</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.74E-03</td>
<td>16.908</td>
<td>24.773</td>
<td>55.877</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.87E-03</td>
<td>46.711</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>8.94E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.25E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>1.99E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>9.23E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>4.37E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>1.73E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>6.54E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>5.76E-07</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(75,32)</th>
<th>RS(80,32)</th>
<th>RS(85,32)</th>
<th>RS(96,32)</th>
<th>RS(112,32)</th>
<th>RS(128,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.82E-02</td>
<td>5.114</td>
<td>5.966</td>
<td>4.976</td>
<td>5.661</td>
<td>6.044</td>
<td>5.868</td>
</tr>
<tr>
<td>12</td>
<td>1.48E-02</td>
<td>48.909</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.48E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.74E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.87E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>8.94E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.25E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>1.99E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>9.23E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>4.37E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>1.73E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>6.54E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>5.76E-07</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>SNR</td>
<td>BER</td>
<td>No Protection</td>
<td>Default Protection</td>
<td>CRC-16</td>
<td>CRC-32</td>
<td>RS(37,32)</td>
<td>RS(38,32)</td>
</tr>
<tr>
<td>-----</td>
<td>---------</td>
<td>---------------</td>
<td>-------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>12</td>
<td>1.54E-02</td>
<td>6.249</td>
<td>Inf</td>
<td>6.568</td>
<td>6.78</td>
<td>7.342</td>
<td>6.887</td>
</tr>
<tr>
<td>18</td>
<td>4.11E-03</td>
<td>11.68</td>
<td>Inf</td>
<td>7.112</td>
<td>6.702</td>
<td>22.756</td>
<td>30.551</td>
</tr>
<tr>
<td>21</td>
<td>2.11E-03</td>
<td>14.764</td>
<td>Inf</td>
<td>16.069</td>
<td>17.697</td>
<td>34.524</td>
<td>49.399</td>
</tr>
<tr>
<td>24</td>
<td>1.07E-03</td>
<td>19.823</td>
<td>Inf</td>
<td>20.293</td>
<td>18.947</td>
<td>52.281</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>5.61E-04</td>
<td>22.261</td>
<td>Inf</td>
<td>22.592</td>
<td>20.912</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.95E-04</td>
<td>26.152</td>
<td>Inf</td>
<td>26.552</td>
<td>26.012</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.53E-04</td>
<td>31.1</td>
<td>Inf</td>
<td>31.404</td>
<td>31.081</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>8.54E-05</td>
<td>35.204</td>
<td>Inf</td>
<td>35.742</td>
<td>37.085</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>4.77E-05</td>
<td>41.882</td>
<td>Inf</td>
<td>39.229</td>
<td>41.505</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>2.91E-05</td>
<td>48.867</td>
<td>Inf</td>
<td>45.126</td>
<td>46.309</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>1.81E-05</td>
<td>54.239</td>
<td>Inf</td>
<td>52.53</td>
<td>59.467</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(43,32)</th>
<th>RS(45,32)</th>
<th>RS(48,32)</th>
<th>RS(51,32)</th>
<th>RS(53,32)</th>
<th>RS(56,32)</th>
<th>RS(64,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>1.54E-02</td>
<td>15.543</td>
<td>17.883</td>
<td>25.122</td>
<td>28.765</td>
<td>31.877</td>
<td>42.491</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.97E-03</td>
<td>28.599</td>
<td>38.75</td>
<td>63.904</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>4.11E-03</td>
<td>59.954</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>2.11E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>1.07E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>5.61E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.95E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.53E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>8.54E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>4.77E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>2.91E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>1.81E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(75,32)</th>
<th>RS(80,32)</th>
<th>RS(85,32)</th>
<th>RS(96,32)</th>
<th>RS(112,32)</th>
<th>RS(128,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.92E-02</td>
<td>30.632</td>
<td>43.675</td>
<td>50.777</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>12</td>
<td>1.54E-02</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.97E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>4.11E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>2.11E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>1.07E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>5.61E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.95E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.53E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>8.54E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>4.77E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>2.91E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>1.81E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>SNR</td>
<td>BER</td>
<td>No Protection</td>
<td>Default Protection</td>
<td>CRC-16</td>
<td>CRC-32</td>
<td>RS(37,32)</td>
<td>RS(38,32)</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>---------------</td>
<td>--------------------</td>
<td>--------</td>
<td>--------</td>
<td>-----------</td>
<td>-----------</td>
</tr>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>5.808</td>
<td>21.204</td>
<td>5.906</td>
<td>5.897</td>
<td>5.816</td>
<td>5.802</td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>5.717</td>
<td>Inf</td>
<td>5.615</td>
<td>5.837</td>
<td>5.848</td>
<td>5.710</td>
</tr>
<tr>
<td>15</td>
<td>7.72E-03</td>
<td>5.788</td>
<td>Inf</td>
<td>5.882</td>
<td>5.685</td>
<td>5.676</td>
<td>7.172</td>
</tr>
<tr>
<td>18</td>
<td>3.92E-03</td>
<td>5.515</td>
<td>Inf</td>
<td>5.672</td>
<td>5.620</td>
<td>12.057</td>
<td>23.848</td>
</tr>
<tr>
<td>21</td>
<td>1.97E-03</td>
<td>6.062</td>
<td>Inf</td>
<td>6.498</td>
<td>6.335</td>
<td>26.527</td>
<td>42.058</td>
</tr>
<tr>
<td>24</td>
<td>9.94E-04</td>
<td>10.974</td>
<td>Inf</td>
<td>8.720</td>
<td>9.325</td>
<td>41.055</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.98E-04</td>
<td>14.437</td>
<td>Inf</td>
<td>13.269</td>
<td>14.004</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.49E-04</td>
<td>20.101</td>
<td>Inf</td>
<td>21.283</td>
<td>19.216</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>24.149</td>
<td>Inf</td>
<td>24.493</td>
<td>24.928</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>6.28E-05</td>
<td>29.320</td>
<td>Inf</td>
<td>29.353</td>
<td>29.552</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>3.13E-05</td>
<td>31.970</td>
<td>Inf</td>
<td>33.602</td>
<td>34.040</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>1.56E-05</td>
<td>39.668</td>
<td>Inf</td>
<td>38.111</td>
<td>37.356</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>7.88E-06</td>
<td>42.788</td>
<td>Inf</td>
<td>42.365</td>
<td>42.836</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(43,32)</th>
<th>RS(45,32)</th>
<th>RS(48,32)</th>
<th>RS(51,32)</th>
<th>RS(53,32)</th>
<th>RS(56,32)</th>
<th>RS(64,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>5.819</td>
<td>5.913</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.72E-03</td>
<td>22.483</td>
<td>27.385</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.92E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.97E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>9.94E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.98E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.49E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>6.28E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>3.13E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>1.56E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>7.88E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(75,32)</th>
<th>RS(80,32)</th>
<th>RS(85,32)</th>
<th>RS(96,32)</th>
<th>RS(112,32)</th>
<th>RS(128,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>20.936</td>
<td>31.420</td>
<td>35.323</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.72E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.92E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.97E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>9.94E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.98E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>2.49E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>6.28E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>3.13E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>1.56E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>7.88E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>
Table 6 - Packet Protection, lena.bmp Minimum PSNRs

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>No Protection</th>
<th>Default Protection</th>
<th>CRC-16</th>
<th>CRC-32</th>
<th>RS(37,32)</th>
<th>RS(38,32)</th>
<th>RS(40,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.83E-02</td>
<td>5.416</td>
<td>5.005</td>
<td>5.426</td>
<td>5.46</td>
<td>5.154</td>
<td>5.367</td>
<td>5.177</td>
</tr>
<tr>
<td>12</td>
<td>1.47E-02</td>
<td>5.057</td>
<td>41.773</td>
<td>5.253</td>
<td>5.264</td>
<td>5.013</td>
<td>4.551</td>
<td>4.298</td>
</tr>
<tr>
<td>18</td>
<td>3.75E-03</td>
<td>4.253</td>
<td>Inf</td>
<td>4.88</td>
<td>4.443</td>
<td>4.53</td>
<td>4.66</td>
<td>4.469</td>
</tr>
<tr>
<td>21</td>
<td>1.86E-03</td>
<td>4.304</td>
<td>Inf</td>
<td>4.308</td>
<td>4.161</td>
<td>4.709</td>
<td>22.703</td>
<td>40.558</td>
</tr>
<tr>
<td>27</td>
<td>4.36E-04</td>
<td>4.559</td>
<td>Inf</td>
<td>4.351</td>
<td>4.687</td>
<td>30.454</td>
<td>51.823</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>1.95E-04</td>
<td>4.196</td>
<td>Inf</td>
<td>5.362</td>
<td>4.653</td>
<td>33.19</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>9.24E-05</td>
<td>5.077</td>
<td>Inf</td>
<td>4.897</td>
<td>4.838</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>3.82E-05</td>
<td>6.598</td>
<td>Inf</td>
<td>5.44</td>
<td>4.249</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>1.60E-05</td>
<td>4.479</td>
<td>Inf</td>
<td>15.465</td>
<td>23.823</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>5.28E-06</td>
<td>24.784</td>
<td>Inf</td>
<td>20.914</td>
<td>25.759</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>1.48E-06</td>
<td>21.046</td>
<td>Inf</td>
<td>5.386</td>
<td>24.385</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(43,32)</th>
<th>RS(45,32)</th>
<th>RS(48,32)</th>
<th>RS(51,32)</th>
<th>RS(53,32)</th>
<th>RS(56,32)</th>
<th>RS(64,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.83E-02</td>
<td>5.246</td>
<td>4.453</td>
<td>4.533</td>
<td>4.261</td>
<td>4.581</td>
<td>4.349</td>
<td>4.78</td>
</tr>
<tr>
<td>12</td>
<td>1.47E-02</td>
<td>5.028</td>
<td>4.405</td>
<td>4.244</td>
<td>4.731</td>
<td>4.675</td>
<td>5.429</td>
<td>33.703</td>
</tr>
<tr>
<td>15</td>
<td>7.44E-03</td>
<td>4.132</td>
<td>5.227</td>
<td>6.065</td>
<td>5.219</td>
<td>24.446</td>
<td>56.338</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.75E-03</td>
<td>29.844</td>
<td>36.305</td>
<td>51.705</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.86E-03</td>
<td>43.385</td>
<td>59.487</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>9.20E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.36E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>1.95E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>9.24E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>3.82E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>1.60E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>5.28E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>1.48E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>RS(75,32)</th>
<th>RS(80,32)</th>
<th>RS(85,32)</th>
<th>RS(96,32)</th>
<th>RS(112,32)</th>
<th>RS(128,32)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.83E-02</td>
<td>4.448</td>
<td>5.271</td>
<td>4.7</td>
<td>5.256</td>
<td>6.341</td>
<td>Inf</td>
</tr>
<tr>
<td>12</td>
<td>1.47E-02</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>15</td>
<td>7.44E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>18</td>
<td>3.75E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>21</td>
<td>1.86E-03</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>24</td>
<td>9.20E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>27</td>
<td>4.36E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>30</td>
<td>1.95E-04</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>33</td>
<td>9.24E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>36</td>
<td>3.82E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>39</td>
<td>1.60E-05</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>42</td>
<td>5.28E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>1.48E-06</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
<td>Inf</td>
</tr>
</tbody>
</table>
Weak RS codes on noisy channels provide no extra signal quality to that of no protection. Such combinations would be RS(37,32) on 15 db SNR and less channels, RS(38,32) on 12 dB SNR and less channels and finally, up to RS(56,32) on 9 dB and less SNR channels. This would be a direct result of those channels not providing enough parity bits to accommodate for the number of error bits present on the channel resulting in significant data loss.
5.1.5 Performance Plots

Several plots of the performance of the JPWL standard are able to show us quickly and simply its performance.

Figure 30 shows the effect of increasing PSNR for a decreasing bit error rate. The 75 dB PSNR value has been chosen to represent an exact reconstruction for the graph as using infinity could cause a bit of an issue with the lower values. Some protections are seen to provide very good protection in very noisy channels as can be seen from the couple of points in the upper right hand corner of the images. Other protections (or lack thereof) are seen to require very quiet channels before any improvement in PSNR is noticed, this is seen in the lower left of the graph. Once the PSNR starts to improve it does so quite quickly before it reaches an exact reconstruction. Using linear distances between SNRs in the channel model could have caused this. In hindsight, logarithmic spacing between the chosen SNRs should have been used.

Figure 31 shows the same graph for the lena image. The graph is extremely similar to that of the boat image in Figure 30 but differences are noticeable. This backs up that the images tell us the same story, but differences between images performance using any codec is to be expected. As a result it would have been preferred to use many images in the simulation, available computing time was a hindering factor in taking on such a task.
Figure 31 tells us the same story but in a slightly different light. The logarithmic relationship between our SNR values and BER values allows the same data to be seen differently. It can be seen that all the Reed-Solomon codes increase in protection very quickly up to the exact reconstruction value. The No Protection and CRC Protection values poor performance can be seen to only increase slowly with an increase in SNR whilst never reaching an exact reconstruction.

Figure 32 - BER vs PSNR for lena image

Figure 32 tells us the same story but in a slightly different light. The logarithmic relationship between our SNR values and BER values allows the same data to be seen differently. It can be seen that all the Reed-Solomon codes increase in protection very quickly up to the exact reconstruction value. The No Protection and CRC Protection values poor performance can be seen to only increase slowly with an increase in SNR whilst never reaching an exact reconstruction.

Figure 32 - SNR vs PSNR for boat image
Figure 33 backs up the interpretations for the boat image by providing a similar graph for the lena image. As a result of these findings it can be seen that no bitstream protection requires a very quiet channel to be of any use whilst even a weak Reed-Solomon code can provide a significant improvement to bitstream protection than that of No Protection.

Figure 33 - SNR vs PSNR for lena image

5.2 JPEG Performance
Testing the straight JPEG images was much simpler than testing the various JPWL settings. This is because there are no special settings for a JPEG image.

5.2.1 Boat Image
First we will look at the performance of the boat image. Table 8 shows us the average bit error rate and PSNR for the differing SNRs used in our channel model simulations. As seen with the JPWL performance the higher SNR channels at 42 dB and 45 dB provided very good results with some infinite PSNRs present. This performs better than JPWL images with no protection on the bitstream.

This performance is short lived when we look at the other end of the scale. Channels of 18 dB SNR or less, provided complete decoding failure; this has been replaced with a PSNR of 0 for this run so that we could obtain some figures to see what is going on. The JPWL images, whilst useless with values around 6 dB PSNR, were still able to be decoded.

Of the images with an error that could be decoded, that is images from 21 dB SNR to 39 dB SNR, none had reached more than 20 dB PSNR. For the maximum PSNRs a channel of 36 dB SNR was required to obtain a PSNR more than 20 dB whereas a 27 dB SNR channel was required for the boat JPWL image with no bitstream protection.
It should also be noted that from a run of 50 iterations, all channels of 42 dB SNR and lower will provide at least one run with a failed decode. The failed decoding rate has not been calculated for this project.

Table 8 - boat.jpg BER and PSNR values

<table>
<thead>
<tr>
<th>Average SNR</th>
<th>Average BER</th>
<th>Average PSNR</th>
<th>Minimum SNR</th>
<th>Minimum BER</th>
<th>Minimum PSNR</th>
<th>Maximum SNR</th>
<th>Maximum BER</th>
<th>Maximum PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.87E-02</td>
<td>0</td>
<td>9</td>
<td>2.81E-02</td>
<td>0</td>
<td>9</td>
<td>2.94E-02</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>0</td>
<td>12</td>
<td>1.46E-02</td>
<td>0</td>
<td>12</td>
<td>1.55E-02</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>7.73E-03</td>
<td>0</td>
<td>15</td>
<td>7.39E-03</td>
<td>0</td>
<td>15</td>
<td>8.02E-03</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>3.92E-03</td>
<td>0</td>
<td>18</td>
<td>3.65E-03</td>
<td>0</td>
<td>18</td>
<td>4.17E-03</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1.97E-03</td>
<td>0.27486</td>
<td>21</td>
<td>1.74E-03</td>
<td>0</td>
<td>21</td>
<td>2.20E-03</td>
<td>13.743</td>
</tr>
<tr>
<td>24</td>
<td>1.00E-03</td>
<td>0.52636</td>
<td>24</td>
<td>8.82E-04</td>
<td>0</td>
<td>24</td>
<td>1.15E-03</td>
<td>10.357</td>
</tr>
<tr>
<td>27</td>
<td>4.94E-04</td>
<td>3.26434</td>
<td>27</td>
<td>3.99E-04</td>
<td>0</td>
<td>27</td>
<td>5.74E-04</td>
<td>11.555</td>
</tr>
<tr>
<td>30</td>
<td>2.50E-04</td>
<td>4.78784</td>
<td>30</td>
<td>1.87E-04</td>
<td>0</td>
<td>30</td>
<td>3.14E-04</td>
<td>14.029</td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>9.08004</td>
<td>33</td>
<td>8.76E-05</td>
<td>0</td>
<td>33</td>
<td>1.78E-04</td>
<td>18.137</td>
</tr>
<tr>
<td>36</td>
<td>6.45E-05</td>
<td>11.77216</td>
<td>36</td>
<td>3.32E-05</td>
<td>0</td>
<td>36</td>
<td>9.37E-05</td>
<td>23.223</td>
</tr>
<tr>
<td>39</td>
<td>3.41E-05</td>
<td>18.59882</td>
<td>39</td>
<td>1.21E-05</td>
<td>0</td>
<td>39</td>
<td>5.74E-05</td>
<td>59.099</td>
</tr>
<tr>
<td>42</td>
<td>1.67E-05</td>
<td>Inf</td>
<td>42</td>
<td>0.00E+00</td>
<td>0</td>
<td>42</td>
<td>3.63E-05</td>
<td>Inf</td>
</tr>
<tr>
<td>45</td>
<td>7.37E-06</td>
<td>Inf</td>
<td>45</td>
<td>0.00E+00</td>
<td>8.136</td>
<td>45</td>
<td>2.12E-05</td>
<td>Inf</td>
</tr>
</tbody>
</table>

Figure 34 - boat.jpg 36 dB SNR run 49

The maximum PSNR image to reach 20 dB PSNR, that is from the 36 dB channel, is shown in Figure 34. The nature of image quality loss within the JPEG image format can be seen in its block like nature. This is evident where the shading changes around the centre of the image. The overall subjective quality of this image could be considered better than that of the JPEG2000 images with the same PSNR values. This is due to the nature of the PSNR calculation; the contrast in this image would have as much an effect on the PSNR as the noise in the JPEG2000 images which have the right contrast.
A 9 dB PSNR image can be seen in Figure 35. Compared to the previous JPEG2000 images of this range of quality you would first say that, subjectively, this image portrays its original quite well. It can be seen that here is a boat, there are clouds in the sky, and there is a lighthouse and also a man standing beside the boat. The quality degradation once again can be seen to be caused by the block like DCT. The bottom of the image starts to become skewed when related to the top half of the image making the measuring of distances between items useless in a noisy image. Also the contrast at the lower end of the image has brightened somewhat. It is argued on the Wikipedia talk page for the JPEG2000 standard as to which JPEG image standard suffers from noise and quality degradation better. This is a subjective argument and may not be solved any time soon, if ever.

The quality of a GSM channel image cannot be compared to that of the JPWL formats as all JPEG images for the 9 dB channel failed to decode. As a result it can be seen outright that the JPWL protections are required for such noisy channels. There is no extension for the original JPEG standard regarding protection for wireless channels.

5.2.2 Lena Image
Table 9 shows us the results from the run on the Lena image. Like the boat image, all images sent over channels of 18 dB SNR or less failed to decode. For average PSNRs the lena image was able to reach 20 dB PSNR at a 39 dB SNR channel as seen in Figure 36. The maximum PSNR images managed to do so at 36 dB SNR as per Figure 37.
### Table 9 - lena.jpg BER and PSNR values

<table>
<thead>
<tr>
<th>SNR</th>
<th>BER</th>
<th>PSNR</th>
<th>SNR</th>
<th>BER</th>
<th>PSNR</th>
<th>SNR</th>
<th>BER</th>
<th>PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>0</td>
<td>9</td>
<td>2.82E-02</td>
<td>0</td>
<td>9</td>
<td>2.97E-02</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>0</td>
<td>12</td>
<td>1.47E-02</td>
<td>0</td>
<td>12</td>
<td>1.57E-02</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>7.72E-03</td>
<td>0</td>
<td>15</td>
<td>7.40E-03</td>
<td>0</td>
<td>15</td>
<td>8.19E-03</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>3.94E-03</td>
<td>0</td>
<td>18</td>
<td>3.69E-03</td>
<td>0</td>
<td>18</td>
<td>4.23E-03</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1.99E-03</td>
<td>0.142</td>
<td>21</td>
<td>1.75E-03</td>
<td>0</td>
<td>21</td>
<td>2.19E-03</td>
<td>7.108</td>
</tr>
<tr>
<td>24</td>
<td>1.01E-03</td>
<td>1.26328</td>
<td>24</td>
<td>8.98E-04</td>
<td>0</td>
<td>24</td>
<td>1.17E-03</td>
<td>14.543</td>
</tr>
<tr>
<td>27</td>
<td>5.02E-04</td>
<td>3.40124</td>
<td>27</td>
<td>4.24E-04</td>
<td>0</td>
<td>27</td>
<td>5.94E-04</td>
<td>13.534</td>
</tr>
<tr>
<td>30</td>
<td>2.49E-04</td>
<td>5.24774</td>
<td>30</td>
<td>1.87E-04</td>
<td>0</td>
<td>30</td>
<td>3.10E-04</td>
<td>13.018</td>
</tr>
<tr>
<td>33</td>
<td>1.24E-04</td>
<td>8.45526</td>
<td>33</td>
<td>6.34E-05</td>
<td>0</td>
<td>33</td>
<td>1.74E-04</td>
<td>17.358</td>
</tr>
<tr>
<td>36</td>
<td>6.05E-05</td>
<td>12.60754</td>
<td>36</td>
<td>2.67E-05</td>
<td>0</td>
<td>36</td>
<td>1.03E-04</td>
<td>33.981</td>
</tr>
<tr>
<td>39</td>
<td>3.05E-05</td>
<td>22.12738</td>
<td>39</td>
<td>3.34E-06</td>
<td>0</td>
<td>39</td>
<td>6.01E-05</td>
<td>67.676</td>
</tr>
<tr>
<td>42</td>
<td>1.82E-05</td>
<td>24.26754</td>
<td>42</td>
<td>3.34E-06</td>
<td>0</td>
<td>42</td>
<td>4.34E-05</td>
<td>71.504</td>
</tr>
<tr>
<td>45</td>
<td>1.00E-05</td>
<td>Inf</td>
<td>45</td>
<td>0.00E+00</td>
<td>0</td>
<td>45</td>
<td>2.67E-05</td>
<td>Inf</td>
</tr>
</tbody>
</table>

- **Figure 36 - lena.jpg 39 dB SNR run 50**

- **Figure 37 - lena.jpg 36 dB SNR run 38**
The quality degradation in these images, the maximum PSNR image on the 36 dB SNR channel in particular, is barely noticeable by subjective means. Figure 36 has an obvious shift in pixels part way down the image which provides a means for the PSNR function to derive its final result. Figure 37 displays an artefact in the top left hand corner which is typical of the block like degradation in the original JPEG standard.

![Figure 38 - lena.jpg 33 dB SNR run 50](image)

An example of a lena image with a PSNR of 8.45 dB is shown in Figure 38. It can be seen here that the quality degradation for the JPEG standard is not noise on the image, but rather contrast and alignment issues. Whether this is better or worse than that of the noise issues of the JPEG2000 images is subjective and will be a decision left up to the reader.

Whilst the JPEG images provide decent quality across many of the not so noisy channels, it should be kept in mind that the JPWL images with default protection provide very good protection on the noise levels which render JPEG images undecodable.

### 5.2.3 Performance Plots

Figure 39 shows the BER vs PSNR plot for the boat image. It can be seen here that the PSNR follows the BER much like that of the JPWL images with no bitstream protection. This would be a logical interpretation as here we also have no protection. It should be noted however that these images have a much lower PSNR for high error rate channels. The image quality is barely seen to be much over 20 dB PSNR except for a very quiet channel.

As expected, the graph for the lena image in Figure 40 provided similar results. One noticeable difference however is that some channels obtained a higher PSNR with values just over 20 appearing on the graph.
Figure 41 shows a very interesting plot of the SNR vs. the PSNR for the boat image. The zero values are known to be undecodable images and as such they can be largely ignored with regards to plot shape. The 75 dB values are known to be exact reconstructions and as such can also be largely ignored with regards to overall plot shape. This however leaves a nice little dip at 12 dB, the JPEG standard would appear to perform better on a 9 dB SNR channel than that of a 12 dB SNR channel which goes against everything we have seen previously. In any regard, the quality of the images at this point is below 20 dB PSNR and as such
considered too noisy for all practical purposes and intents anyway. A channel around 35 dB is seen to be required to obtain that 20 dB PSNR value.

Figure 41 - SNR vs. PSNR for boat JPEG image

The lena image plot has the same odd shapes with a few more curves as seen in Figure 42. A usable image is still seen to require a channel of at least 35 dB SNR and channels up at around 45 dB SNR are seen to provide very good reconstructions. These plots show that the effect of noise on the JPEG standard is not uniform as the noise levels increase.

Figure 42 - SNR vs. PSNR for lena JPEG image
5.3 JPEG2000 Performance

The performance of the JPEG2000 standard without the JPWL extension will also be investigated. This will help to determine the effect and performance of the JPWL extensions on top of the base JPEG2000 standard.

5.3.1 Boat Image

Table 10 shows us the results from the testing performed on the JPEG2000 images. It can be seen here that the JPEG2000 standard even without the JPWL extensions performed much better than the original JPEG standard with regards to how well it can be decoded. No JPEG2000 images were able to be decoded on the 9 dB PSNR channel whereas the JPEG images failed on channels up to 18 dB.

Table 10 - boat.j2k BER and PSNR values

<table>
<thead>
<tr>
<th>Average</th>
<th>Minimum</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR</td>
<td>BER</td>
<td>PSNR</td>
</tr>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>1.50E-02</td>
<td>0.2285</td>
</tr>
<tr>
<td>15</td>
<td>7.69E-03</td>
<td>0.82384</td>
</tr>
<tr>
<td>18</td>
<td>3.90E-03</td>
<td>1.63436</td>
</tr>
<tr>
<td>21</td>
<td>1.97E-03</td>
<td>5.51158</td>
</tr>
<tr>
<td>24</td>
<td>9.92E-04</td>
<td>8.95056</td>
</tr>
<tr>
<td>27</td>
<td>4.99E-04</td>
<td>12.7934</td>
</tr>
<tr>
<td>30</td>
<td>2.52E-04</td>
<td>15.9692</td>
</tr>
<tr>
<td>33</td>
<td>1.25E-04</td>
<td>22.36348</td>
</tr>
<tr>
<td>36</td>
<td>6.14E-05</td>
<td>26.6788</td>
</tr>
<tr>
<td>39</td>
<td>3.28E-05</td>
<td>31.50194</td>
</tr>
<tr>
<td>42</td>
<td>1.54E-05</td>
<td>35.76176</td>
</tr>
<tr>
<td>45</td>
<td>7.48E-06</td>
<td>39.64852</td>
</tr>
</tbody>
</table>

For average PSNRs the 20 dB mark is reached at an SNR value of 33 dB. For this mark to be reached for the JPEG channels, the images needed a 42 dB channel. However for a 42+ dB SNR channel the JPEG standard suffered from no quality degradation whereas the JPEG2000 standard suffered some degradation on all channels.

Figure 43 shows the boat image which reaches an average PSNR of just over 22 dB for the 33 dB SNR channel. As can be seen in the image the JPEG2000 standard does not suffer from the block like degradation like that of JPEG standard. The image and location of all items within the image are not distorted and the contrast remains relatively accurate throughout. The noise however approximates that of an old photograph that has aged with time.

Overall the JPEG2000 standard without the JPWL extensions has performed better than that of the JPEG standard, at least, for the boat image.
Figure 44 shows the boat image for the 45 dB SNR channel. It can be seen here that the image results in a very good quality reproduction of the original but it is not perfect. The JPEG standard provided an exact reproduction for this channel but the small bit error rate is still enough to have an effect on the wavelet transform of the JPEG 2000 standard image. As a result, channels with a very small BER should consider the original JPEG standard, whilst keeping in mind that even at 42 dB SNR, this original standard can have a failed decoding out of 50 runs. However the JPEG2000 standard without the JPWL extensions also managed a failed decode on a 45 dB SNR channel.

5.3.2 Lena Image
The lena image produced similar results to that of the boat image. One noticeable difference however is that the lena image had no decodable code streams for the 12 dB SNR channel as well as the 9 dB channel. The PSNR values however increase at a similar rate with the 20 dB average PSNR mark being reached at the 33 dB SNR channel, which is the same as the boat image. The 45 dB SNR channel also does not have an exact reconstruction
of the original which adds weight to the argument that the DCT based standard has a better
resilience to a relatively quiet channel.

Table 11 - lena.j2k BER and PSNR values

<table>
<thead>
<tr>
<th>SNR</th>
<th>Average BER</th>
<th>Average PSNR</th>
<th>Minimum BER</th>
<th>Minimum PSNR</th>
<th>Maximum BER</th>
<th>Maximum PSNR</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>2.88E-02</td>
<td>0</td>
<td>9</td>
<td>2.84E-02</td>
<td>0</td>
<td>2.91E-02</td>
</tr>
<tr>
<td>12</td>
<td>1.51E-02</td>
<td>0</td>
<td>12</td>
<td>1.47E-02</td>
<td>0</td>
<td>1.53E-02</td>
</tr>
<tr>
<td>15</td>
<td>7.70E-03</td>
<td>0.44914</td>
<td>15</td>
<td>7.44E-03</td>
<td>0</td>
<td>7.91E-03</td>
</tr>
<tr>
<td>18</td>
<td>3.93E-03</td>
<td>3.00164</td>
<td>18</td>
<td>3.79E-03</td>
<td>0</td>
<td>4.08E-03</td>
</tr>
<tr>
<td>21</td>
<td>1.97E-03</td>
<td>4.96508</td>
<td>21</td>
<td>1.88E-03</td>
<td>0</td>
<td>2.07E-03</td>
</tr>
<tr>
<td>24</td>
<td>9.90E-04</td>
<td>8.06712</td>
<td>24</td>
<td>9.12E-04</td>
<td>0</td>
<td>1.07E-03</td>
</tr>
<tr>
<td>27</td>
<td>4.98E-04</td>
<td>14.14726</td>
<td>27</td>
<td>4.59E-04</td>
<td>0</td>
<td>5.45E-04</td>
</tr>
<tr>
<td>30</td>
<td>2.46E-04</td>
<td>19.29074</td>
<td>30</td>
<td>2.15E-04</td>
<td>0</td>
<td>2.82E-04</td>
</tr>
<tr>
<td>33</td>
<td>1.24E-04</td>
<td>21.70398</td>
<td>33</td>
<td>9.11E-05</td>
<td>0</td>
<td>1.45E-04</td>
</tr>
<tr>
<td>36</td>
<td>6.17E-05</td>
<td>26.83124</td>
<td>36</td>
<td>4.42E-05</td>
<td>0</td>
<td>8.05E-05</td>
</tr>
<tr>
<td>39</td>
<td>3.28E-05</td>
<td>30.92504</td>
<td>39</td>
<td>2.30E-05</td>
<td>9.15</td>
<td>4.42E-05</td>
</tr>
<tr>
<td>42</td>
<td>1.61E-05</td>
<td>37.15968</td>
<td>42</td>
<td>5.31E-06</td>
<td>15.025</td>
<td>2.56E-05</td>
</tr>
<tr>
<td>45</td>
<td>8.28E-06</td>
<td>40.43092</td>
<td>45</td>
<td>2.65E-06</td>
<td>0</td>
<td>1.68E-05</td>
</tr>
</tbody>
</table>

Figure 45 shows us the image where the average PSNRs reach 20 dB. Once again this
confirms the difference in the effect of noise on the JPEG2000 image when compared to
that of the JPEG images. As a result, the underlying strength of the JPEG2000 standard is
that it can be further degraded and still maintain a decodable state. Also when it is decoded
the contrast is maintained and all elements of the picture remain aligned. The noise on the
image of the JPEG2000 standard is, once again, quite degrading for any imaging requiring a
decent quality where as the JPEG standard maintains a crisp clear image.
5.3.3 Performance Plots

Figure 46 provides a bit of a cross between what we have seen previously with our other standards. Like the JPWL images with no bitstream protection the PSNR increases evenly along the scale upwards through 20 dB and beyond. The PSNR on the noisier channels however approaches 0 dB whereas for the JPWL images it was not able to get below 5 dB.

Figure 46 - BER vs. PSNR for boat JPEG2000 image

Figure 47 - BER vs. PSNR for lena JPEG2000 image
Figure 47 re-enforces what we have just seen for the boat image. There is little difference between the two graphs aside from the lena image having a smoother quality improvement.

Figure 48 shows us the SNR vs. PSNR for the boat image. Like the JPEG standard we have a bit of an unexpected dip part way through the graph. This was not expected for the JPEG2000 standard as the graphs for the JPWL images do not have such a dip. It can be seen here that the JPEG2000 standard recovers from a complete decode failure at 9 dB SNR to a very good (but not exact) reconstruction up at 45 dB SNR. It is not known what would cause the dip in quality between the 15 dB and 18 dB SNR values, this is either the standard or the program malfunctioning.

![Graph of SNR vs. PSNR for boat JPEG2000 image](image)

Figure 49 shows the plot for the lena image. A complete decode failure occurs for the 12 dB channel as well as the 9 dB channel here. A dip in quality also occurs at 18 dB SNR before recovering to a good but still not exact reconstruction up at a 45 dB channel. In both plots it can be seen that a useful image can be obtained from a 25 dB SNR channel.
Figure 49 - SNR vs. PSNR for lena JPEG2000 image
6 Conclusions

The wireless protection extension for the JPEG2000 standard has been tested across several of its protection mechanisms. The standard performed extremely well across most of the protections available and provides a clear indication of a working protection mechanism for wireless channels. The original JPEG and base JPEG2000 standard were also tested and found to work in channels which are not high in errors. The JPEG2000 standard worked better in noisier channels. The JPEG standard could be considered to work better in the not so noisy channels; its quality could also be considered subjectively far more resilient to errors than that of the wavelet transform properties of the JPEG2000 standard.

The protection of the codestream headers in the JPWL standard was evident by seeing that many of the images were decodable in some form even in very noisy channels where only black and white meaningless images from the bitstream could be recovered. This proves that the normative header protection component of the standard works effectively for the OpenJPEG implementation.

It was also seen that the CRC protection mechanisms provided no extra protection at all and as a result are of no use to protecting the bitstream itself. However it was not expected that these codes with no error correcting component would be capable of doing so.

The Default Protection for the bitstream as defined by the OpenJPEG codec provided the best protection with regards to image quality. It did, however, also have a significant size increase on that of the original compressed JPEG2000 file. The Reed Solomon codes also provided excellent protection for the bitstream when the appropriate strength code was used. These codes were also able to provide a better protection than the Default Protection at a smaller file size which would result in a smaller quantity of bits required for transmission.

It would be good to see the encoder for an implementation of the JPWL standard measure the noise of the channel upon which it is transmitting so that it may be able to know exactly which strength of RS code to use for the existing channel conditions. This would provide the benefit of reducing the transmission size to that required rather than a standard size regardless of the channel conditions. However this cannot be implemented as the standard is geared for implementations where no signal can be received from the receiver. This is the point of using strong Forward Error Correcting codes to protect the data so that it does not need to be re-transmitted. The transmitter does not find out the condition of the received data at the other end.

The JPWL standard accommodates for the differing data sensitivities of each part of the bitstream. The more important parts of the bitstream can be protected by stronger codes.
and the less important parts of the bitstream can be protected by weaker codes, if at all. The result of taking into account this data sensitivity method of differing protections has not yet been analysed.

The JPEG standard failed to decode on several of the noisier channels. On the channels where it did decode it suffered contrast changes through the image as well as parts of the image having an out of synch horizontal shift. The quality of a noisy JPEG image however is much better than that of a noisy JPEG2000 image when considered subjectively.

The JPEG2000 standard without the JPWL extensions can take more noise than that of the JPEG standard and still be decoded. The effect of the noise on images that can be decoded is that of an ageing photograph or noisy television channel. This reduces the subjective appeal of the image when compared to similar JPEG standard images in the same noise channel. The wavelet transform suffers quality loss from bit errors greater than that of the cosine transform.

Finally, the JPWL standard performed very well at protecting the JPEG2000 codestream from a large array of error levels. It would be useful for anyone wanting to implement a wireless stream of JPEG2000 imagery.
7 Bibliography


8 Appendices

8.1 Appendix A – Project Specification

University of Southern Queensland

FACULTY OF ENGINEERING AND SURVEYING

ENG4111/4112 Research Project

PROJECT SPECIFICATION

FOR: Wei Xiang

TOPIC: Error Resilience of JPEG 2000 format over Wireless Fading Channels

SUPERVISOR: Wei Xiang

PROJECT AIM: To investigate the error resilient properties of the JPEG-2000 standard. It will also be compared to the error resilient capabilities of the original JPEG standard. A software platform will simulate and analyse these errors.

PROGRAMME: (Issue C: 28 April 2008)

1. Analyse previous error correction studies (studies performed for the purpose of gathering ideas for the JPEG-2000 Part 11 standard) for hints and direction.
2. Draft a set of tests to determine image quality loss, detail loss, etc.
3. Draft an initial design approach for the software test bed to implement (2).
4. Implement test bed for transmission errors (add errors via software intervention).
5. Run test bed for several error types and magnitudes to test JPWL extension using the available OpenJPEG JPWL implementation.
6. Run test bed for several error types and magnitudes to test the original JPEG standard.
7. Report on the quality of the reconstructed images both subjectively and objectively.

As time permits:

8. Investigate improvements to the error protection (Part 11) and/or underlying code layer (Part 1) of the JPEG-2000 standard.
9. Test available codec’s (Part 1 of the standard) for the measure of their performance.

AGREED: _____________________ (student) _____________________ (supervisor)

Date: / /2008 Date: / /2008

Examiner/Co-examiner: ____________________________________

Andrew Clemence

Page 57
8.2 Appendix B - Code

8.2.1 Main.m

%Main Operating File for Project
%Written By Andrew Clemence
%06 July 2008

%Clear screen and variables...
clc; clear; close all;
tic;

%addpath('D:\Documents\ENG4111-2 - Research Project\Code');
%addpath('I:\Documents\ENG4111-2 - Research Project\Code');
%addpath('C:\Documents and Settings\Andrew.ANDREWLAPTOP\My Documents\ENG4111-2 - Research Project\Code');
%addpath('Z:\ENG4111-2 - Research Project\Code');

%Read in Images
[Names Images cmaps] = readimg(); %http://links.uwaterloo.ca/greyset2.base.html

% Determine optimal packet protection JPWL
cd Output
optimalpacket = packetprotect(Images, Names);
cd ..

% Determine optimal packet protection JPEG
cd Output
optimalpacket = packetprotectJPEG(Images, Names);
cd ..

% Determine optimal packet protection JPEG2000
cd Output
optimalpacket = packetprotectJPEG2000(Images, Names);
cd ..

timetaken = toc;
fprintf('Total time taken is %.3fn', timetaken);
%end

8.2.2 Reading.m

%Read in the 4 test Images from the Images Folder...
%Andrew Clemence
%18 March 2008

%Modified for Project 21 July 2008

function [Names Images cmaps] = readimg()
cd ..
cd Images
Files = dir;

%Open and Display Image
for i = 1:length(Files)-2
  %read the image
  [Images{i} cmaps{i}] = imread(Files(i+2).name);
  Names{i} = Files(i+2).name;
end
cd ..
cd Code
end

8.2.3 Packetprotect.m

%Determine optimal header protection
%By Andrew Clemence
%3 August 2008

%The header and packet protection will be maintained as predefined with
%relative error sensitivity and a packet addressing mode.

function optimal = packetprotect(Images, Names)
%Produce array of possible packet protections
protections = [0, 1, 16, 32, ... %none, predefined, CRC-16, CRC-32
37, 38, 40, 43, 45, 48, 51, 53, 56, 64, 75, 80, 85, 96, 112, 128]; % RS Codes
iterations = 50;

%%%% FOR EACH IMAGE %%%%%
for count = 1:1 %2:2 for 2nd image (this is the multi-threading part)

%%%% FOR EACH SIMULATION %%%%%
for x = 1:iterations
Image = Images{count};
fileName = Names{count};
resultsFile = sprintf('ResultsPacket%s.txt', fileName);
results = fopen(resultsFile, 'a');
%Open Results file for appending
if results == -1
while results == -1
pause(1);
results = fopen(resultsFile, 'a');
if count == 10
fprintf('Failed to open results file for run %d on image %s!
', x, fileName);
end
end
fprintf('Testing Image %s for run number %d
', fileName, x);

%%%% FOR EACH PROTECTION %%%%%
for i = 1:length(protections)
% Create CoDEStream %%%%%
outFile = sprintf('Packet_%s_%d.j2k', fileName(1:length(fileName)-4), protections(i));
formatString = sprintf('JPWL_image_to_j2k -r 50,40,30,20,10,5 -i %s -o %s -OutFor j2k -W h=1,p=%d', ...
fileName, outFile, protections(i));
% Create Codestream
[status, result] = system(formatString);
if status ~= 0
fprintf('File did not encode properly, critical error with protection %d for packet
', protections(i));
else
%%%% RUN CHANNEL MODEL %%%%%
% Run Channel Model
[SNR, BER] = ChannelModel(outFile);

%%%% EACH SNR %%%%%
for k = 1:length(SNR)
% Decode files %%%%%
decodedFilename = sprintf('Packet_Decoded_%s_%d_%d_%d.bmp', fileName(1:length(fileName)-4), protections(i), SNR(k), x);
formatString = sprintf('JPWL_j2k_to_image -i %s -o %s -OutFor bmp -W c=1', ...
decodedFilename);
status = system(formatString);
if status ~= 0
fprintf('File did not decode properly, critical failure with protection %d for packet
', protections(i));
else
%%%% PSNR CALC AND WRITE RESULTS%%%%
% Read in decoded file
imageError = 0;
try
decodedFile = imread(decodedFilename);
catch
end
end
end
end
end
end
end
end
end
end
end
imageError = 1;
end

if imageError ~= 1
%Calculate PSNR
try
filePSNR = PSNR(Image, decodedFile);
catch
filePSNR = 0;
end

%Calculate PSNR/size ratio
currentFile = dir(sprintf('b_SNR-%d_%s', SNR(k), outFile));
fileSize = currentFile.bytes;
ratio = filePSNR/fileSize;
%Write Results for each run
fprintf(fresults, '%s %03d %03d %02d %3.03E %2.3f %07d %3.3E %s', datestr(now), x, protections(i), ...
SNR(k), BER(k), filePSNR, fileSize, ratio, fileName);
end
end
end
fclose(fresults);
end
fclose('all');
optimal = 0; %Not actually used
end

8.2.4 packetprotectJPEG.m
%Determine optimal header protection
%By Andrew Clemence
%3 August 2008

%The header and packet protection will be maintained as predefined with %relative error sensitivity and a packet addressing mode.

function optimal = packetprotectJPEG(Images, Names)
%Produce array of possible packet protections
iterations = 50;

%%%%%%%%%%%%%%%%%%%%% FOR EACH IMAGE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for count = 1:length(Images)
%%%%%%%%%%%%%%%%%% FOR EACH SIMULATION %%%%%%%%%%%%%%%%%%%%%%%%%%%%
%For x independant simulations
for x = 1:iterations
Image = Images{count};
fileName = Names{count};
resultsFile = sprintf('ResultsPacket%s.txt', fileName);
results = fopen(resultsFile, 'a');

%Open Results file for appending
if results == -1
while results == -1
pause(1);
results = fopen(resultsFile, 'a');
if count == 10
error('Failed to open results file for run %d on image %s!', x, ...
fileName);
end
end
end

fprintf('Testing Image %s for run number %d', fileName, x);


%%%%%%%%%%%%%%%%%%%%%% EACH SNR %%%%%%%%%%%%%%%%%%%%%%%
for k = 1:length(SNR)
decodedFilename = sprintf('b_SNR-%d_%s', SNR(k), fileName);


% Read in decoded file
imageError = 0;
try
decodedFile = imread(decodedFilename, 'jpeg');
catch
imageError = 1;
end

currentFile = dir(sprintf('b_SNR-%d_%s', SNR(k), fileName));
fileSize = currentFile.bytes;
if imageError ~= 1
    % Save file with current run number for use in reporting...
    try
        imwrite(decodedFile, sprintf('Decoded_%s_SNR_%d_%d.jpg', fileName, SNR(k), x), 'jpeg');
catch
        fprintf('Failed to write image for run: %d and SNR: %d.
', x, SNR(k));
    end
end
% Calculate PSNR
try
    filePSNR = PSNR(Image, decodedFile);
catch
    filePSNR = 0;
end
% Calculate PSNR/size ratio
ratio = filePSNR/fileSize;
else
    filePSNR = 0;
    ratio = 0;
end
% Write Results for each run
fprintf(fresults, '%s %03d %02d %3.03E %2.3f %07d %3.3E %s
', datestr(now), x, SNR(k), BER(k), filePSNR, fileSize, ratio, fileName);
end
end
fclose(fresults);
end
fclose('all');
optimal = 0;
end

8.2.5 packetprotectJPEG2000
% Determine optimal header protection
% By Andrew Clemence
% 3 August 2008

% The header and packet protection will be maintained as predefined with relative error sensitivity and a packet addressing mode.

function optimal = packetprotectJPEG2000(Images, Names)
% Produce array of possible packet protections
iterations = 50;

% FOR EACH IMAGE
for count = 1:length(Images)
    % FOR EACH SIMULATION
    for x = 1:iterations
        Image = Images{count};
        fileName = Names{count};
        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
        fresults = fopen(resultsFile, 'a');

        % Open Results file for appending
        if fresults == -1
            while fresults == -1
                pause(1);
                fresults = fopen(resultsFile, 'a');
            end

            % FOR x independant simulations
            for x = 1:iterations
                Image = Images{count};
                fileName = Names{count};
                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                fresults = fopen(resultsFile, 'a');

                % Open Results file for appending
                if fresults == -1
                    while fresults == -1
                        pause(1);
                        fresults = fopen(resultsFile, 'a');
                    end

                    % FOR x independant simulations
                    for x = 1:iterations
                        Image = Images{count};
                        fileName = Names{count};
                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                        fresults = fopen(resultsFile, 'a');

                        % Open Results file for appending
                        if fresults == -1
                            while fresults == -1
                                pause(1);
                                fresults = fopen(resultsFile, 'a');
                            end

                            % FOR x independant simulations
                            for x = 1:iterations
                                Image = Images{count};
                                fileName = Names{count};
                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                fresults = fopen(resultsFile, 'a');

                                % Open Results file for appending
                                if fresults == -1
                                    while fresults == -1
                                        pause(1);
                                        fresults = fopen(resultsFile, 'a');
                                    end

                                    % FOR x independant simulations
                                    for x = 1:iterations
                                        Image = Images{count};
                                        fileName = Names{count};
                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                        fresults = fopen(resultsFile, 'a');

                                        % Open Results file for appending
                                        if fresults == -1
                                            while fresults == -1
                                                pause(1);
                                                fresults = fopen(resultsFile, 'a');
                                            end

                                            % FOR x independant simulations
                                            for x = 1:iterations
                                                Image = Images{count};
                                                fileName = Names{count};
                                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                fresults = fopen(resultsFile, 'a');

                                                % Open Results file for appending
                                                if fresults == -1
                                                    while fresults == -1
                                                        pause(1);
                                                        fresults = fopen(resultsFile, 'a');
                                                    end

                                                    % FOR x independant simulations
                                                    for x = 1:iterations
                                                        Image = Images{count};
                                                        fileName = Names{count};
                                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                        fresults = fopen(resultsFile, 'a');

                                                        % Open Results file for appending
                                                        if fresults == -1
                                                            while fresults == -1
                                                                pause(1);
                                                                fresults = fopen(resultsFile, 'a');
                                                            end

                                                            % FOR x independant simulations
                                                            for x = 1:iterations
                                                                Image = Images{count};
                                                                fileName = Names{count};
                                                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                fresults = fopen(resultsFile, 'a');

                                                                % Open Results file for appending
                                                                if fresults == -1
                                                                    while fresults == -1
                                                                        pause(1);
                                                                        fresults = fopen(resultsFile, 'a');
                                                                    end

                                                                    % FOR x independant simulations
                                                                    for x = 1:iterations
                                                                        Image = Images{count};
                                                                        fileName = Names{count};
                                                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                        fresults = fopen(resultsFile, 'a');

                                                                        % Open Results file for appending
                                                                        if fresults == -1
                                                                            while fresults == -1
                                                                                pause(1);
                                                                                fresults = fopen(resultsFile, 'a');
                                                                            end

                                                                            % FOR x independant simulations
                                                                            for x = 1:iterations
                                                                                Image = Images{count};
                                                                                fileName = Names{count};
                                                                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                fresults = fopen(resultsFile, 'a');

                                                                                % Open Results file for appending
                                                                                if fresults == -1
                                                                                    while fresults == -1
                                                                                        pause(1);
                                                                                        fresults = fopen(resultsFile, 'a');
                                                                                    end

                                                                                    % FOR x independant simulations
                                                                                    for x = 1:iterations
                                                                                        Image = Images{count};
                                                                                        fileName = Names{count};
                                                                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                        fresults = fopen(resultsFile, 'a');

                                                                                        % Open Results file for appending
                                                                                        if fresults == -1
                                                                                            while fresults == -1
                                                                                                pause(1);
                                                                                                fresults = fopen(resultsFile, 'a');
                                                                                            end

                                                                                            % FOR x independant simulations
                                                                                            for x = 1:iterations
                                                                                                Image = Images{count};
                                                                                                fileName = Names{count};
                                                                                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                                fresults = fopen(resultsFile, 'a');

                                                                                                % Open Results file for appending
                                                                                                if fresults == -1
                                                                                                    while fresults == -1
                                                                                                        pause(1);
                                                                                                        fresults = fopen(resultsFile, 'a');
                                                                                                    end

                                                                                                    % FOR x independant simulations
                                                                                                    for x = 1:iterations
                                                                                                        Image = Images{count};
                                                                                                        fileName = Names{count};
                                                                                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                                        fresults = fopen(resultsFile, 'a');

                                                                                                        % Open Results file for appending
                                                                                                        if fresults == -1
                                                                                                            while fresults == -1
                                                                                                                pause(1);
                                                                                                                fresults = fopen(resultsFile, 'a');
                                                                                                            end

                                                                                                            % FOR x independant simulations
                                                                                                            for x = 1:iterations
                                                                                                                Image = Images{count};
                                                                                                                fileName = Names{count};
                                                                                                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                                                fresults = fopen(resultsFile, 'a');

                                                                                                                % Open Results file for appending
                                                                                                                if fresults == -1
                                                                                                                    while fresults == -1
                                                                                                                        pause(1);
                                                                                                                        fresults = fopen(resultsFile, 'a');
                                                                                                                    end

                                                                                                                    % FOR x independant simulations
                                                                                                                    for x = 1:iterations
                                                                                                                        Image = Images{count};
                                                                                                                        fileName = Names{count};
                                                                                                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                                                        fresults = fopen(resultsFile, 'a');

                                                                                                                        % Open Results file for appending
                                                                                                                        if fresults == -1
                                                                                                                            while fresults == -1
                                                                                                                                pause(1);
                                                                                                                                fresults = fopen(resultsFile, 'a');
                                                                                                                            end

                                                                                                                            % FOR x independant simulations
                                                                                                                            for x = 1:iterations
                                                                                                                                Image = Images{count};
                                                                                                                                fileName = Names{count};
                                                                                                                                resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                                                                fresults = fopen(resultsFile, 'a');

                                                                                                                                % Open Results file for appending
                                                                                                                                if fresults == -1
                                                                                                                                    while fresults == -1
                                                                                                                                        pause(1);
                                                                                                                                        fresults = fopen(resultsFile, 'a');
                                                                                                                                    end

                                                                                                                                    % FOR x independant simulations
                                                                                                                                    for x = 1:iterations
                                                                                                                                        Image = Images{count};
                                                                                                                                        fileName = Names{count};
                                                                                                                                        resultsFile = sprintf('ResultsPacket%s.txt', fileName);
                                                                                                                                        fresults = fopen(resultsFile, 'a');

                                                                                                                                        % Open Results file for appending
                                                                        end
                                                                    end
                                                                end
                                                            end
                                                        end
                                                    end
                                                end
                                            end
                                        end
                                    end
                                end
                            end
                        end
                    end
                end
            end
        end
    end
end
if count == 10
    error('Failed to open results file for run %d on image %s!\n\n', x, fileName);
end
end
end
fprintf('Testing Image %s for run number %d\n\n', fileName, x);

%%%%%%%%%%%%%%%%%%%%%CREATE JPEG2000 IMAGE%%%%%%%%%%%%%%%%%%%%%
j2kFileName = sprintf('%s.j2k', fileName);
formatString = sprintf('JPWL_image_to_j2k -i %s -o %s', fileName, j2kFileName);
status = system(formatString);
if (status ~= 0)
    fprintf('Failed to create Bitmap file for Run: %d Image: %s\n\n', x, fileName);
else
    %%%%%%%%%%%%%%%%%% RUN CHANNEL MODEL %%%%%%%%%%%%%%%%%%%%%
    %Run Channel Model
    [SNR, BER] = ChannelModel(j2kFileName);
    for k = 1:length(SNR)
        %%%%% CREATE BMP IMAGE OF J2K IMAGE %%%%%%%%%%%%%%%%%%%%
        channelFilename = sprintf('b_SNR-\d_%s', SNR(k), j2kFileName);
decodedFilename = sprintf('Decoded_%s_%d_%d.bmp', fileName, SNR(k), x);
        formatString = sprintf('JPWL_j2k_to_image -i %s -o %s', channelFilename, decodedFilename);
        status = system(formatString);
        if (status ~= 0)
            filePSNR = 0;
            ratio = 0;
            fileSize = 0;
        else
            % PSNR CALCS AND WRITE RESULTS%%%%%
            try
                decodedFile = imread(decodedFilename, 'bmp');
            catch
                imageError = 1;
            end
            fileSize = 0;
            if imageError ~= 1
                %Save file with current run number for use
                %in reporting...
                try
                    writeln(decodedFile, sprintf('Decoded_%s_SNR_%d_%d.bmp', fileName, SNR(k), x), 'jpeg');
                catch
                    fprintf('Failed to write image for run: %d and SNR: %d\n\n', x, SNR(k));
                end
            %Calculate PSNR
            try
                filePSNR = PSNR(Image, decodedFile);
            catch
                filePSNR = 0;
            end
            %Calculate PSNR/size ratio
            ratio = filePSNR/fileSize;
        else
            filePSNR = 0;
            ratio = 0;
        end
        %Write Results for each run
        fprintf(fresults, '%s %03d %02d %3.03E %2.3f %07d %3.3E %s\n\n', datestr(now), x, ...
        SNR(k), BER(k), filePSNR, fileSize, ratio, fileName);
8.2.6 ChannelModel.m

%Test sim_sqskgray_multipath
%By Andrew Clemence
%29 July 2008

function [SNRlist, BER1] = ChannelModel(File)
%Open the file...
    fid = fopen(File, 'rb');
    if fid == -1
        error('Could not open test tile!
');
    else
        %Read in test file as binary string...
        binarystring = fread(fid, 'ubit1');
        %Close the file
        fclose(fid);
        %Variables
        SNRlist = 9:3:45;
        %Pre-allocate for speed...
        dataout = cell(length(SNRlist),1);
        BER1 = zeros(length(SNRlist), 1);
        %Channel
        for i = 1:length(SNRlist)
            [dataout{i}, BER1(i)] = sim_qpskgray_test(binarystring', SNRlist(i));
            %Write back to file
            fopen(sprintf('b_SNR-%d_%s', SNRlist(i), File), 'wb');
            if fid == -1
                fprintf('Could not open output file
');
            else
                fwrite(fid, dataout{i}, 'ubit1');
                fclose(fid);
            end
        end
end

8.2.7 sim_qpskgray_test.m (Gray Channel)

This is from Matlab Central with minor modifications to make it suitable for our needs.

%QPSK simulation with Gray coding and simple Rayleigh (no LOS) multipath and AWGN added
%Run from editor debug(F5)
%JC-7/1/08
%The purpose of this m-file is to show a baseband simulated version of QPSK with
%Gray coding (Rayleigh multipath and AWGN added) which may give valid results
%(still trying to figure out if this program is correct - multipath so subjective)
%when compared to theoretical/simulated AWGN MPSK analysis SER and BER.
%The simulation assumes a single channel (no diversity or FEC codes other than Gray)
%perfect system with perfect sync and no intersymbol interference. The program contains
%no Root Raised Cosine or Raised Cosine filters as they would just add delay. I hope
%it will be useful to others to play with and give a basic understanding of the problems
%encountered in the channel with various types of multipath.
%I have provided comments, notes and references for review. You can also
%download the file sim_qpskgray.m under JC file for BER and SER simulation
%only in AWGN channel. What this all proves is that you need at least 17 dB
%of fade margin at 10^-3 BER with Rayleigh multipath when comparing only with AWGN
%at SNR of 7 to 8dB. Of course you can lower this with antenna diversity, FEC codes, etc
%or possibly with DSSS with pseudo random codes. If you have the communications toolbox
%you can make comparisons with what it gives in it's plots (see references)

function [dataout, BER1] = sim_qpskgray_test(data, SNR)
clear
randn('state',0);%keeps bits the same on reruns
nr_data_bits = length(data);% 0's and 1's, keep even number-Takes ~1 minute for a run of 1
million
%64000 allows bits and complex values to be shown in array editor
nr_symbols=nr_data_bits/2;

b = (randn(1, nr_data_bits) > .5);%random 0's and 1's

% Map the bits to be transmitted into QPSK symbols using Gray coding. The
% resulting QPSK symbol is complex-valued, where one of the two bits in each
% QPSK symbol affects the real part (I channel) of the symbol and the other
% bit the imaginary part (Q channel). Each part is subsequently
% modulated to form the complex-valued QPSK symbol.
% The Gray mapping resulting from the two branches are shown where
% one symbol error corresponds to one bit error going counterclockwise.
% imaginary part (Q channel)
%         ^
%         |
%  10 x   |   x 00   (odd bit, even bit)
%         |
%         ---------> real part (I channel)
%         |
%  11 x   |   x 01
%         |
% Input:
%   b = bits {0, 1} to be mapped into QPSK symbols
% Output:
%   d = complex-valued QPSK symbols 0.70711 + 0.70711i, etc

d=zeros(1,length(b)/2);
definition of the QPSK symbols using Gray coding.
for n=1:length(b)/2
    p=b(2*n);
    imp=b(2*n-1);
    if (imp==0)&&(p==0)
        d(n)=exp(j*pi/4);%45 degrees
    end
    if (imp==1)&&(p==0)
        d(n)=exp(j*3*pi/4);%135 degrees
    end
    if (imp==1)&&(p==1)
        d(n)=exp(j*5*pi/4);%225 degrees
    end
    if (imp==0)&&(p==1)
        d(n)=exp(j*7*pi/4);%315 degrees
    end
end
qpsk=d;

% SNR=0:30;%change SNR values
BER1=[];
% SNR=[ ];
% SER=[];
% signal=[];

Rayleigh multipath/AWGN(Additive White Gaussian Noise)

for SNR=0:length(SNR);%loop over SNR-change SNR values {0,5,10 etc dB}
sigma = sqrt(10.0^(-SNR/10.0));

sigma=sigma/2;%Required a division by 2 to get close to exact solutions(Notes)-WHY?
%Is dividing by two(2) legitimate?
%signal=[signal sigma];
%add Rayleigh multipath(no LOS) to signal(qpsk)
x=randn(1,nr_symbols);
y=randn(1,nr_symbols);
ray=ray+ray(0.5*(x.^2+y.^2));%%variance=0.5-Tracks theoretical PDF closely
mpqpsk=qpsk.*ray;
%add noise to QPSK Gray coded signals with multipath
mpsnpqsk=(real(mpqpsk)+sigma.*randn(size(mpqpsk)))+i.*(imag(mpqpsk)+sigma.*randn(size(mpqpsk)));
%Receiver
%received signal plus noise and multipath
r=mpsnqpsk;
%Detector
%When Gray coding is configured as shown, the detection process
%becomes fairly simple as shown. A system without Gray coding requires a much
%more complex algorithm detection method
bhat=[real(r)<0;imag(r)<0];
bhat=bhat(:)';
%0's and 1's
ne=sum(b~=bhat1);
%number of errors
BER=ne/nr_data_bits;
%SER=ne/nr_symbols;%consider this to be Ps=log2(4)*Pb=2*Pb
%SNR=[SNR1 SNR];
BER1=[BER1 BER];
end
dataout = bhat1;

%Notes: Theoretical QPSK EXACT SOLUTION for several SNR=Eb/No points on BER/SER plot
%Assuming Gray coding and AWGN
%Pb=Q(sqrt(2SNRbit))
%Ps=2Q(sqrt(2SNRbit))[1-.5Q(sqrt(2SNRbit))]
%SNR=7dB
SNRB=10^((7/10)-5.0118) get ratio
Ps=2Q(sqrt(2SNRbit))=Q(sqrt(10.0237))=7.7116e-4 (bit error rate)
%SNR=9dB
SNR=10^((9/10)-7.943 get ratio
Ps=2Q(sqrt(2SNRbit))=Q(sqrt(15.866))=3.37e-5 (bit error rate)
%0,1,2,3,4,5,6,8,10,11,12 You can do the rest of these with a loop and hold
for hand plot on "Simulation of BER/SER for QPSK with Gray coding
%(Rayleigh multipath and AWGN) graph 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Plots
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure(1);
plot(d,'o');%plot constellation without noise
axis([-2 2 -2 2]);
grid on;
xlabel('real'); ylabel('imag');
title('QPSK constellation');

figure(2);
semilogy(SNR1,BER1,'*',SNR1,SER1,'o');
grid on;
xlabel('SNR=Eb/No(dB)'); ylabel('BER/SER');
title('Simulation of BER/SER for QPSK with Gray coding (Rayleigh multipath and AWGN)');
legend('BER-simulated','SER-simulated');

figure(3)
plot(real(qpsk));
grid on;
axis([1 200 -2 2]);
title('QPSK symbols');
xlabel('symbols'); ylabel('Amplitude');

figure(4)
plot(20*log10(abs(ray)));
grid on;
axis([1 200 -30 10]);
title('Rayleigh Fading Envelope(variance=0.5)');
xlabel('symbols'); ylabel('Amplitude/RMS(dB)');

%References
%This website shows Matlab code for various fading channels that may be helpful
%search the web for "Mobile Radio Channels Modeling in Matlab-Nikolay KOSTOV". If
%you go through and understand this paper, I'm sure it will be helpful. I even learned
%how to use -inf.
%PROAKIS, Digital Communications 4th ed. (chapter 14) 14.4.2 Fading Multiphase Signals

end

8.2.8 PSNR.m

%Calculate the PSNR of an image
%By Andrew Clemence
%23 April 2008

function PSNR = PSNR(Original, Image)
%MSE = 1/(mn) sum (i=0 to m-1) sum (j=0 to n-1) abs(I(i,j) - K(i,j))^2
%where I = Original Image and K = Changed Image...
%m and n
Original = double(Original);
Image = double(Image);
[m n] = size(Original);
if size(Original) == size(Image)
    runningsum = 0;
    for i = 1:m
        for j = 1:n
            runningsum = runningsum + abs(Original(i,j) - Image(i,j))^2;
        end
    end
    MSE = runningsum/(m*n);
    PSNR = 10*log10(256^2/MSE);
else
    %Return failure if image sizes are not equal
    PSNR = 0;
end
end
8.3 Appendix C – JPEG Images

8.3.1 Average Boat Images

Figure 50 - boat.jpg 21 dB SNR channel

Figure 51 - boat.jpg 24 dB SNR channel

Figure 52 - boat.jpg 27 dB SNR channel
Figure 56 - boat.jpg 39 dB SNR channel

Figure 57 - boat.jpg 42 dB SNR channel

Figure 58 - boat.jpg 45 dB SNR channel
8.3.2 Average Lena Images

Figure 59 - lena.jpg 21 dB SNR channel

Figure 60 - lena.jpg 24 dB SNR channel

Figure 61 - lena.jpg 27 dB SNR channel
Figure 62 - lena.jpg 30 dB SNR channel

Figure 63 - lena.jpg 33 dB SNR channel

Figure 64 - lena.jpg 36 dB SNR channel
Figure 65 - lena.jpg 39 dB SNR channel

Figure 66 - lena.jpg 42 dB SNR channel

Figure 67 - lena.jpg 45 dB SNR channel
8.4 Appendix D – JPEG2000 Images

8.4.1 Average Boat Images

Figure 68 - boat.j2k 12 dB SNR channel

Figure 69 - boat.j2k 15 dB SNR channel

Figure 70 - boat.j2k 18 dB SNR channel
Figure 71 - boat.j2k 21 dB SNR channel

Figure 72 - boat.j2k 24 dB SNR channel

Figure 73 - boat.j2k 27 dB SNR channel
Figure 74 - boat.j2k 30 dB SNR channel

Figure 75 - boat.j2k 33 dB SNR channel

Figure 76 - boat.j2k 36 dB SNR channel
Figure 77 - boat.j2k 39 dB SNR channel

Figure 78 - boat.j2k 42 dB SNR channel

Figure 79 - boat.j2k 45 dB SNR channel
8.4.2 Average Lena Images

Figure 80 - lena.j2k 15 dB SNR channel

Figure 81 - lena.j2k 18 dB SNR channel

Figure 82 - lena.j2k 21 dB SNR channel
Figure 83 - lena.j2k 24 dB SNR channel

Figure 84 - lena.j2k 27 dB SNR channel

Figure 85 - lena.j2k 30 dB SNR channel
Figure 86 - lena.j2k 33 dB SNR channel

Figure 87 - lena.j2k 36 dB SNR channel

Figure 88 - lena.j2k 39 dB SNR channel
Figure 89 - lena.j2k 42 dB SNR channel

Figure 90 - lena.j2k 45 dB SNR channel
8.5 Appendix E – JPWL Images

8.5.1 Average Boat Images

8.5.1.1 No Protection

Figure 91 - boat.j2c 9 dB SNR channel

Figure 92 - boat.j2c 12 dB SNR channel
Figure 93 - boat.j2c 15 dB SNR channel

Figure 94 - boat.j2c 18 dB SNR channel

Figure 95 - boat.j2c 21 dB SNR channel
Figure 96 - boat.j2c 24 dB SNR channel

Figure 97 - boat.j2c 27 dB SNR channel

Figure 98 - boat.j2c 30 dB SNR channel
Figure 99 - boat.j2c 33 dB SNR channel

Figure 100 - boat.j2c 36 dB SNR channel

Figure 101 - boat.j2c 39 dB SNR channel
Figure 102 - boat.j2c 42 dB SNR channel

Figure 103 - boat.j2c 45 dB SNR channel

8.5.1.2 Default Protection

Figure 104 - boat.j2c 9 dB SNR channel
8.5.1.3 **CRC-16 Protection**

![Figure 105 - boat.j2c 12 dB SNR channel](image)

![Figure 106 - boat.j2c 15+ dB SNR channel](image)

![Figure 107 - boat.j2c 9 dB SNR channel](image)
Figure 108 - boat.j2c 12 dB SNR channel

Figure 109 - boat.j2c 15 dB SNR channel

Figure 110 - boat.j2c 18 dB SNR channel
Figure 111 - boat.j2c 21 dB SNR channel

Figure 112 - boat.j2c 24 dB SNR channel

Figure 113 - boat.j2c 27 dB SNR channel
Figure 114 - boat.j2c 30 dB SNR channel

Figure 115 - boat.j2c 33 dB SNR channel

Figure 116 - boat.j2c 36 dB SNR channel
Figure 117 - boat.j2c 39 dB SNR channel

Figure 118 - boat.j2c 42 dB SNR channel

Figure 119 - boat.j2c 45 dB SNR channel
8.5.1.4 CRC-32 Protection

Figure 120 - boat.j2c 9 dB SNR channel

Figure 121 - boat.j2c 12 dB SNR channel

Figure 122 - boat.j2c 15 dB SNR channel
Figure 123 - boat.j2c 18 dB SNR channel

Figure 124 - boat.j2c 21 dB SNR channel

Figure 125 - boat.j2c 24 dB SNR channel
Figure 126 - boat.j2c 27 dB SNR channel

Figure 127 - boat.j2c 30 dB SNR channel

Figure 128 - boat.j2c 33 dB SNR channel
Figure 129 - boat.j2c 36 dB SNR channel

Figure 130 - boat.j2c 39 dB SNR channel

Figure 131 - boat.j2c 42 dB SNR channel
8.5.1.5 *RS*(37,32) Protection

![Figure 132 - boat.j2c 45 dB SNR channel](image)

![Figure 133 - boat.j2c 9 dB SNR channel](image)

![Figure 134 - boat.j2c 12 dB SNR channel](image)
Figure 135 - boat.j2c 15 dB SNR channel

Figure 136 - boat.j2c 18 dB SNR channel

Figure 137 - boat.j2c 21 dB SNR channel
8.5.1.6 RS(38,32) Protection
Figure 141 - boat.j2c 12 dB SNR channel

Figure 142 - boat.j2c 15 dB SNR channel

Figure 143 - boat.j2c 18 dB SNR channel
8.5.1.7  RS(40,32) Protection

Figure 144 - boat.j2c 21 dB SNR channel

Figure 145 - boat.j2c 24+ dB SNR channel

Figure 146 - boat.j2c 9 dB SNR channel
Figure 147 - boat.j2c 12 dB SNR channel

Figure 148 - boat.j2c 15 dB SNR channel

Figure 149 - boat.j2c 18 dB SNR channel
8.5.1.8 *RS(43,32) Protection*

Figure 150 - boat.j2c 21+ dB SNR channel

Figure 151 - boat.j2c 9 dB SNR channel

Figure 152 - boat.j2c 12 dB SNR channel
Figure 153 - boat.j2c 15 dB SNR channel

Figure 154 - boat.j2c 18 dB SNR channel

Figure 155 - boat.j2c 21+ dB SNR channel
8.5.1.9 RS(45,32) Protection

Figure 156 - boat.j2c 9 dB SNR channel

Figure 157 - boat.j2c 12 dB SNR channel

Figure 158 - boat.j2c 15 dB SNR channel
8.5.1.10 RS(48,32) Protection

Figure 159 - boat.j2c 18+ dB SNR channel

Figure 160 - boat.j2c 9 dB SNR channel

Figure 161 - boat.j2c 12 dB SNR channel
Figure 162 - boat.j2c 15 dB SNR channel

Figure 163 - boat.j2c 18 dB SNR channel

Figure 164 - boat.j2c 21+ dB SNR channel
8.5.1.11 RS(51,32) Protection

Figure 165 - boat.j2c 9 dB SNR channel

Figure 166 - boat.j2c 12 dB SNR channel

Figure 167 - boat.j2c 15 dB SNR channel
8.5.1.12 RS(53,32) Protection

Figure 168 - boat.j2c 18+ dB SNR channel

Figure 169 - boat.j2c 9 dB SNR channel

Figure 170 - boat.j2c 12 dB SNR channel
Figure 171 - boat.j2c 15 dB SNR channel

Figure 172 - boat.j2c 18+ dB SNR channel

8.5.1.13 RS(56,32) Protection

Figure 173 - boat.j2c 9 dB SNR channel
Figure 174 - boat.j2c 12 dB SNR channel

Figure 175 - boat.j2c 15 dB SNR channel

Figure 176 - boat.j2c 18+ dB SNR
8.5.1.14 RS(64,32) Protection

Figure 177 - boat.j2c 9 dB SNR channel

Figure 178 - boat.j2c 12 dB SNR channel

Figure 179 - boat.j2c 15+ dB SNR channel
8.5.1.15 RS(75,32) Protection

Figure 180 - boat.j2c 9 dB SNR channel

Figure 181 - boat.j2c 12 dB SNR channel

Figure 182 - boat.j2c 15+ dB SNR channel
8.5.1.16 RS(80,32) Protection

Figure 183 - boat.j2c 9 dB SNR channel

8.5.1.17 RS(85,32) Protection

Figure 184 - boat.j2c 12+ dB SNR channel

Figure 185 - boat.j2c 9 dB SNR channel
**8.5.1.18 RS(96,32) Protection**

Figure 186 - boat.j2c 12+ dB SNR channel

Figure 187 - boat.j2c 9 dB SNR channel

Figure 188 - boat.j2c 12+ dB SNR channel
8.5.1.19 RS(112,32) Protection

Figure 189 – boat.j2c 9 dB SNR channel

8.5.1.20 RS(128,32) Protection

Figure 190 - boat.j2c 12+ dB SNR channel

Figure 191 - boat.j2c 9+ dB SNR channel
8.5.2 Average Lena Images

8.5.2.1 No Protection

Figure 192 - lena.j2c 9 dB SNR channel

Figure 193 - lena.j2c 12 dB SNR channel

Figure 194 - lena.j2c 15 dB SNR channel
Figure 195 - lena.j2c 18 dB SNR channel

Figure 196 - lena.j2c 21 dB SNR channel

Figure 197 - lena.j2c 24 dB SNR channel
Figure 198 - lena.j2c 27 dB SNR channel

Figure 199 - lena.j2c 30 dB SNR channel

Figure 200 - lena.j2c 33 dB SNR channel
Figure 201 - lena.j2c 36 dB SNR channel

Figure 202 - lena.j2c 39 dB SNR channel

Figure 203 - lena.j2c 42 dB SNR channel
8.5.2.2 Default Protection

Figure 204 - lena.j2c 45 dB SNR channel

Figure 205 - lena.j2c 9 dB SNR channel

Figure 206 - lena.j2c 12 dB SNR channel
8.5.2.3 CRC-16 Protection

Figure 207 - lena.j2c 15+ dB SNR channel

Figure 208 - lena.j2c 9 dB SNR channel

Figure 209 - lena.j2c 12 dB SNR channel
Figure 210 - lena.j2c 15 dB SNR channel

Figure 211 - lena.j2c 18 dB SNR channel

Figure 212 - lena.j2c 21 dB SNR channel
Figure 213 - lena.j2c 24 dB SNR channel

Figure 214 - lena.j2c 27 dB SNR channel

Figure 215 - lena.j2c 30 dB SNR channel
Figure 216 - lena.j2c 33 dB SNR channel

Figure 217 - lena.j2c 36 dB SNR channel

Figure 218 - lena.j2c 39 dB SNR channel
8.5.2.4 CRC-32 Protection
Figure 222 - lena.j2c 12 dB SNR channel

Figure 223 - lena.j2c 15 dB SNR channel

Figure 224 - lena.j2c 18 dB SNR channel
Figure 225 - lena.j2c 21 dB SNR channel

Figure 226 - lena.j2c 24 dB SNR channel

Figure 227 - lena.j2c 27 dB SNR channel
Figure 228 - lena.j2c 30 dB SNR channel

Figure 229 - lena.j2c 33 dB SNR channel

Figure 230 - lena.j2c 36 dB SNR channel
Figure 231 - lena.j2c 39 dB SNR channel

Figure 232 - lena.j2c 42 dB SNR channel

Figure 233 - lena.j2c 45 dB SNR channel
8.5.2.5 RS(37,32) Protection

Figure 234 - lena.j2c 9 dB SNR channel

Figure 235 - lena.j2c 12 dB SNR channel

Figure 236 - lena.j2c 15 dB SNR channel
Figure 237 - lena.j2c 18 dB SNR channel

Figure 238 - lena.j2c 21 dB SNR channel

Figure 239 - lena.j2c 24 dB SNR channel
Figure 240 - lena.j2c 27 dB SNR channel

Figure 241 - lena.j2c 30 dB SNR channel

Figure 242 - lena.j2c 33+ dB SNR channel
8.5.2.6 RS(38,32) Protections

Figure 243 - lena.j2c 9 dB SNR channel

Figure 244 - lena.j2c 12 dB SNR channel

Figure 245 - lena.j2c 15 dB SNR channel
Figure 246 - lena.j2c 18 dB SNR channel

Figure 247 - lena.j2c 21 dB SNR channel

Figure 248 - lena.j2c 24 dB SNR channel
8.5.2.7  **RS(40,32) Protections**
Figure 252 - lena.j2c 12 dB SNR channel

Figure 253 - lena.j2c 15 dB SNR channel

Figure 254 - lena.j2c 18 dB SNR channel
8.5.2.8  \textit{RS(43,32)} Protections
Figure 258 - lena.j2c 12 dB SNR channel

Figure 259 - lena.j2c 15 dB SNR channel

Figure 260 - lena.j2c 18 dB SNR channel
8.5.2.9 RS(45,32) Protections
Figure 264 - lena.j2c 12 dB SNR channel

Figure 265 - lena.j2c 15 dB SNR channel

Figure 266 - lena.j2c 18 dB SNR channel
8.5.2.10 RS(48,32) Protections
Figure 270 - lena.j2c 12 dB SNR channel

Figure 271 - lena.j2c 15 dB SNR channel

Figure 272 - lena.j2c 18 dB SNR channel
8.5.2.11 RS(51,32) Protections

Figure 273 - lena.j2c 21+ dB SNR channel

Figure 274 - lena.j2c 9 dB SNR channel

Figure 275 - lena.j2c 12 dB SNR channel
8.5.2.12 RS(53,32) Protections
Figure 279 - lena.j2c 12 dB SNR channel

Figure 280 - lena.j2c 15 dB SNR channel

Figure 281 - lena.j2c 18+ dB SNR channel
8.5.2.13 RS(56,32) Protections

Figure 282 - lena.j2c 9 dB SNR channel

Figure 283 - lena.j2c 12 dB SNR channel

Figure 284 - lena.j2c 15 dB SNR channel
8.5.2.14 RS(64,32) Protections

Figure 285 - lena.j2c 18+ dB SNR channel

Figure 286 - lena.j2c 9 dB SNR channel

Figure 287 - lena.j2c 12 dB SNR channel
8.5.2.15 RS(75,32) Protections

Figure 288 - lena.j2c 15+ dB SNR channel

Figure 289 - lena.j2c 9 dB SNR channel

Figure 290 - lena.j2c 12+ dB SNR channel
8.5.2.16 RS(80,32) Protections

Figure 291 - lena.j2c 9 dB SNR channel

8.5.2.17 RS(85,32) Protections

Figure 292 - lena.j2c 12+ dB SNR channel

Figure 293 - lena.j2c 9 dB SNR channel
8.5.2.18 RS(96,32) Protections

Figure 294 - lena.j2c 12+ dB SNR channel

Figure 295 - lena.j2c 9 dB SNR channel

Figure 296 - lena.j2c 12+ dB SNR channel
8.5.2.19 RS(112,32) Protections

Figure 297 - lena.j2c 9 dB SNR channel

Figure 298 - lena.j2c 12+ dB SNR channel

8.5.2.20 RS(128,32) Protections

Figure 299 - lena.j2c 9+ dB SNR channel