Implementation of an Application Specific Low Bit Rate Video Compression Scheme

IAN JAMES MCINTOSH

Submitted in fulfilment of the academic requirements for the degree of Master of Science in Engineering in the School of Electrical & Electronic Engineering, University of Natal, Durban.

December 2001

Preface

The author carried out the work described in this dissertation during the period February 2000 to December 2001 under the supervision of Professor R. Peplow in the School of Electrical and Electronic Engineering, University of Natal, Durban.

The author declares that this is his own work, except where specifically indicated to the contrary, and that the work has not been submitted to any other university for degree purposes.

As the candidate's supervisor I have approved this dissertation for submission.

Signed:_____Name:_____Date:_____

Acknowledgements

Firstly I would like to thank my supervisor, Prof. Peplow, for his help and guidance over the last two years.

Thanks go to ARMSCOR for their sponsorship of the work and crucial financial assistance, without which the degree would not be possible.

For their extra guidance and encouragement I would like to thank the rest of the postgraduate group who have made the period an enjoyable time.

Finally I would like to thank my family for their support and encouragement throughout.

Abstract

The trend towards digital video has created huge demands on the link bandwidth required to carry the digital stream, giving rise to the growing research into video compression schemes. General video compression standards, which focus on providing the best compression for any type of video scene, have been shown to perform badly at low bit rates and thus are not often used for such applications. A suitable low bit rate scheme would be one that achieves a reasonable degree of quality over a range of compression ratios, while perhaps being limited to a small set of specific applications. One such application specific scheme, as presented in this thesis, is to provide a differentiated image quality, allowing a user-defined region of interest to be reproduced at a higher quality than the rest of the image.

The thesis begins by introducing some important concepts that are used for video compression followed by a survey of relevant literature concerning the latest developments in video compression research. A video compression scheme, based on the Wavelet transform, and using an application specific idea, is proposed and implemented on a digital signal processor (DSP), the Philips Trimedia TM-1300. The scheme is able to capture and compress the video stream and transmit the compressed data via a low bit-rate serial link to be decompressed and displayed on a video monitor. A wide range of flexibility is supported, with the ability to change various compression parameters *'on-the-fly'*. The compression algorithm is controlled by a PC application that displays the decompressed video and the original video for comparison, while displaying useful rate metrics such as Peak Signal to Noise Ratio (PSNR).

Details of implementation and practicality are discussed. The thesis then presents examples and results from both implementation and testing before concluding with suggestions for further improvement.

Contents

List of Figures

List of Tables

List of Abbreviations

2G	Second Generation
3-D	Three Dimensional
3G	Third Generation
CCD	Charged Coupled Device
CIF	Common Image Format
CMY	Cyan-Magenta-Yellow
DCT	Discrete Cosine Transform
EZW	Embedded Zerotree Wavelet
fps	Frames per Second
HVS	Human Visual System
ISO	International Standards Organisation
ITU	International Telecommunications Union
JPEG	Joint Pictures Expert Group
JPEG2000	Joint Pictures Experts Group 2000
kbps	Kilobits per second
KLT	Karhunen-Loeve Transform
LZW	Lempel-Ziv-Welch
Mbps	Megabits per second
MJPEG	Motion Joint Pictures Experts Group
MJPEG2000	Motion Joint Pictures Experts Group 2000
MPEG	Motion Pictures Expert Group
NTSC	National Television System Committee
PACC	Partitioning, Aggregation and Conditional Coding
PAL	Phase Alternate Line
PSNR	Peak Signal to Noise Ratio
QCIF	Quarter-Common Image Format
RGB	Red-Green-Blue
RLE	Run-Length Encoding
SFQ	Space-Frequency Quantisation
SIF	Source Input Format
SNR	Signal to Noise Ratio
SPIHT	Set Partitioning in Hierarchical Trees
ZTE	Zerotree Entropy Encoding

Chapter 1 Introduction

The modern digital era is coming of age with a myriad of digital devices now available for many applications. In this digital era communication between people has become increasingly easier with a person being able to communicate with another in just about any location on the globe. This ease of communication prompts vendors to deliver more and better services than before while trying to maintain similar usage of available resources. One such service is the provision of better quality video for a variety of video based applications from video conferencing to home video editing.

To achieve this goal manufacturers have moved to digital video as it has become increasingly popular over its analogue predecessor for two main reasons. Firstly, with the readily available digital storage media, a video stream, once digitised, is more easily stored than the analogue version and, secondly, once captured, replicating the video or image is a lossless process whereas, with analogue, every copied version adds noise due to the copying process. Digitised video allows for a multitude of applications, each of which is becoming an essential part of the digital era. In the medical profession, using a video stream allows doctors to make remote consultations without unnecessary travel to the location, while with 3-D imagery they are able to make a diagnosis on the computer screen, where before surgery would have been required. Satellites are constantly transmitting images over communications channels for weather, Digital TV and many other uses. In the business world, digital video has become indispensable with applications in video conferencing, allowing the businessman to converse face-to-face with clients without having to travel great distances to meet them. Perhaps the most widely used medium for digital video transmission is the Internet. With this global network users are able to download video sequences and communicate with friends and family abroad through videophones, all of which are taken for granted and seen as vital tools for modern living. Today's digital consumer expects to have all these applications delivered at real-time, all the time, and on top of existing applications which places a huge strain on available resources.

But what is video? How is it represented? Why can't you watch a movie quality video over the Internet? These are a few issues this chapter aims to cover. Firstly a brief description of how video is represented digitally is given; this is vital for a better understanding of the requirements of employing video in any application. Then a case for video compression is presented followed by some video compression fundamentals.

1.1 What is Video?

Anyone who has a television set has been exposed to video. Most have probably not thought too much about the generation of the video and accepted that it is a carbon copy of events as they transpired. This is, in actual fact, not strictly true.

Consider a photograph of a scene. The photograph captures the scene as it is at the moment in time when the photograph was taken. If the light changes or an object is placed in the scene, the photograph is no longer an exact replica of the scene. In video, the goal is to replicate a scene over a period of time, which could be anything from a few hours to a few days. In this case, multiple photographs taken at regular instants in time are needed to properly represent a scene, especially if the scene is always changing. The captured video sequence can then be viewed by displaying each photograph of the scene at the same regular rate. If an object was moving in the scene then a still image of that object would be captured

every time a photograph was taken. When the photographs, or video frames, are played back quickly enough an illusion of the object moving in the scene is created.

Video, as seen on TV and other such applications, is represented as described, by displaying a series of still pictures fast enough to fool the brain into thinking there is motion. This process is an exploitation of the Human Visual System [Bhas97] (HVS) which is able to interpolate between successive frames and convince the brain that continuous motion is being viewed. The number of still images displayed in a specific time period, to create the illusion of motion, is called the *temporal resolution*. Even though the HVS is able to interpolate frames to create the illusion of continuous motion, there still exists a minimum number of still images to be shown per second before the motion appears to become 'jerky'. Typically in broadcast TV the temporal resolution is 25-30 frames per second, which provides a continuous motion video sequence.

In digital video each frame is an N×M matrix of dots, called *pixels*, which are of varying intensities and colours to represent the image. As with motion, the HVS interpolates between neighbouring pixels to create the illusion of a continuous image. The more pixels present, i.e. the higher the *spatial resolution*, the better the quality of the image [Hoan01]. Figure 1.1 highlights this fact by showing a magnified view of a portion of a 512×512 digital image. The image itself looks continuous but as the 8×8 magnified area shows there are step variations between pixels and thus the image is not actually continuous.

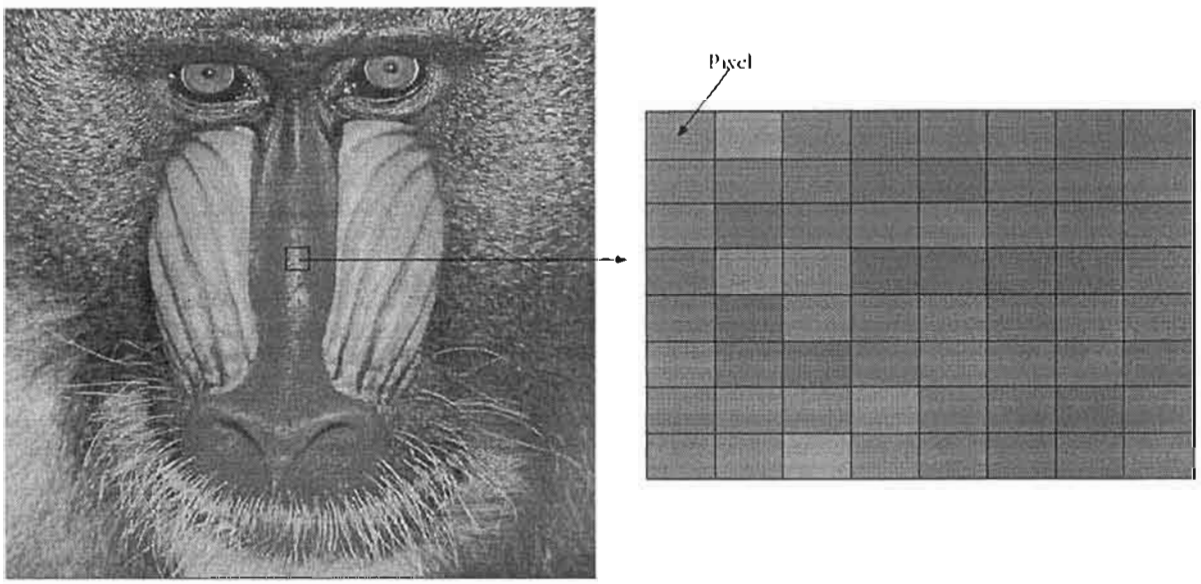


Figure 1.1: Example of pixels that make up an image

1.2 Why the Need to Compress?

The representation of video as a series of still images is very cumbersome; a little arithmetic and a look at current communications systems and storage devices reveals that there is a problem of storing and transmitting this video.

Table 1.1 shows uncompressed bit rates required to transmit some typical data sources. The table highlights the enormous amount of bandwidth required to transmit uncompressed video sources at their required rates. To fully appreciate the strain that video can put on a communications network it is useful to look at typical transfer rates for commonly available communication channels. Table 1.2 lists a few of

the most common communications links used today. Looking at both tables and considering that there are many users of communications devices at any one time, it is plain to see that the inclusion of video traffic puts a great deal of strain on available communications links. For an alternate viewpoint, using Table 1.1, we can compute the amount of storage space needed to store a two-hour broadcast quality video, which works out to be ≈106 gigabytes. Considering that the capacity of a single-sided DVD is about 4.7 gigabytes, it will only be able to hold less than 6 minutes of video. Clearly some form of significant compression is required to make video a viable communications tool.

Application	Bandwidth Required
Video Conference (15fps) *Framesize 352×288, 24 bits/pixel*	34.8 Mbps
Broadcast video (25fps) *Framesize 720×576, 24 bits/pixel*	237.3 Mbps

Table 1.1 : Uncompressed data sizes

Connection Type	Transfer Rate
Analog Modem	56 kbps
ISDN	128 kbps
Cable Modems	2-5 Mbps
Inficom Wireless Modem	10 Mbps
Point-to-Point Wireless	100 Mbps
Fiberless Optics (Laser)	1 Gbps
Current 2G Mobile	9.6 kbps – 14.4 kbps
Proposed 3G Mobile	382 kbps – 2 Mbps

Table 1.2 : Typical Transfer Rates for communication links

Even if communications channels and storage devices were able to cope with the large amounts of data required for video, compression would be valuable as it translates to more money. The driving force in research for better compression schemes is, like in so many things, rooted in economics. The best compression scheme translates to more images or video media stored with the same storage capacity and thus more money. This point is well made in an article from Business Week that highlights the need for compression.

> *"The biggest obstacle to the vaunted multimedia revolution is digital obesity. That's the bloat that occurs when pictures, sound and video are converted from their natural analog form into computer language for manipulation or transmission...Compression, a rapidly developing branch of mathematics, is putting digital on a diet...Its popularity is rooted in economics: compression lowers the cost of storage and transmission by packing data into a smaller space. Many new electronic products and services simply couldn't exist without it."*
>
> Business Week, Feb. 14, 1994

1.3 How is Compression Achieved?

Compression is the art of representing information with as few data bits as possible [Taub86]. It uses the fact that real world data is usually highly correlated. Simply put, correlation is the relation between two

or more data points such that changes in the one are accompanied by changes in the other. An example of this is the English language where if a 'q' is used then there is a large chance that the next letter used will be 'u', meaning that 'q' and 'u' are correlated. So the need to represent the 'u' after the 'q' is redundant and often unnecessary. A compression algorithm could be designed to insert a 'u' after every 'q' and thus represent information more efficiently[1]. This type of compression scheme is called *predictive* as it predicts future information values ('u') from previous values ('q'). The information could also be represented more sparsely by decorrelation, a process that effectively removes the relationship between information values, making them independent of each other, and as such representing the information as sparsely as possible [Taub86]. The principle of decorrelating information can be easily extended to natural images (i.e. those that exist in the real-world and not generated) as they contain a high degree of correlation. For this reason, many successful image and video compression schemes employ some form of decorrelation or prediction.

Another method of compressing data is to take into account the probabilities of occurrence of certain values over others. Again the English language provides a good example of this. Work done on the language has found that certain letters are more likely to occur than others [Syme01]. For example the letter 'e' is the most widely used letter in the English alphabet and the letter 'y' much less frequently used. Using this fact a compression scheme could be devised to assign a short data value to the letter 'e' and a longer one to the less probable 'y'. In this case, less data bits are required to represent the English language as, now the letter 'e' occupies less storage space and, even though the letter 'y' may occupy more space, it is uncommonly used, so the average effect is compression. Morse code provides a good example of this code assignment in practice. In Morse code the letter 'e' is assigned the shortest code whereas 'y' has a code 4 times greater than that of 'e'. Thus the average Morse code message sent is shorter than it would be if all letters in the alphabet were assigned the same length. In video and image compression, this principle is often applied by first obtaining a probability distribution of the image data and then using this to more efficiently represent the data.

A further, simple, technique that can be applied to images is to encode runs of the same value more efficiently, often termed *run-length* encoding (RLE). The basic idea behind RLE is to encode 'runs' of the same byte. For example, a data stream 'aaaaaaaa' could be run-length encoded as '8a', meaning repeat the character 'a' 8 times to get the original data. This technique is often used in conjunction with other techniques to improve on compression.

The techniques already described are termed *lossless* as, the result of decompression is an exact reconstruction of the original data that was compressed. In the case of images, characteristics of the Human Visual System (HVS) allow for information deemed irrelevant to be thrown away without perceptible loss in quality [Bhas97]. This type of compression is termed *lossy* and is used to compress data a great deal more than lossless schemes. An example of a property of the HVS (which is often used in video) is colour sensitivity. The HVS is particularly adept at detecting changes in brightness but rather poor at distinguishing colour changes. A colour compression scheme throws away colour information

[1] Cleary there does exist cases when 'u' does not follow 'q' and the algorithm may fail. The example is only used to illustrate the principle.

that is imperceptible to the HVS and thus irrelevant, allowing for a more compact representation of the colour image.

The removal of irrelevant data introduces the difficult problem of choosing which data is relevant and which is not as if too much data is removed the output video may become intolerably degraded. This also presents a unique problem where the effects of the degradation may make the video appear of visually better quality than before, making the effective removal of irrelevant data very difficult to predict and control. Lossy compression schemes tend to affect images in two distinctly different ways, things that are lost from the image and things that are added that shouldn't be there (artefacts). Losses occur in the spatial domain with luminance and colour aberrations and in the temporal domain with lost frames. Spatial artefacts include blockiness, quantization noise, ringing, stepping of greyscales, all of which may make the subjective quality of the image better. A successful lossy compression scheme, should thus effectively marry the acceptable subjective loss with the desired compression ratio.

One solution to classifying irrelevant data is to use information about the target application to decide which information can be thrown away. Consider an application that monitors access to a specific room, as is the case for many security applications. In such a system, the information of relevance could be the identity of the person entering the room and the items the person is carrying. The identification of the person requires only the face, rendering all other information about the person irrelevant for identification purposes. Including the case of carried items, there are only two areas of relevancy in the entire video sequence: the persons face and the items carried. The user of such an application would accept large amounts of degradation in the video quality, which may be unacceptable to other users, as long as there is sufficient quality in areas that are required for identification, such as the face. In such circumstances, a video compression scheme can be devised that is particularly suited to a certain application. Thus once the areas of relevance are defined all other areas can be highly compressed while keeping the area of interest at a desired quality level. This application specific compression allows for higher compression ratios than normally possible using general compression schemes, yet still results in a usable video sequence, albeit limited in application.

1.3.1 Temporal Compression

The compression techniques introduced, thus far, have focussed on the spatial domain, but video also has a temporal domain that can be compressed. In a typical video sequence there can be as many as 30 still images needed per second to achieve a reasonable degree of perceived motion. In such cases a few minutes of video quickly uses up large amounts of storage space or communications bandwidth. However, a video sequence often contains little or no motion meaning that successive images are the same. In such cases, storing the same image twice is redundant and so compression can be achieved by storing a single image and telling the decompression algorithm to display the same image for the time required.

Movement in a video scene can manifest itself as objects moving over a static background, or a panning camera with the background changing and objects that are moving. In this type of scene, there are still areas of temporal redundancy but the spatial positioning between each frame has changed. In this case, an intelligent compression scheme would be to try to recognise temporally redundant information even if the information is spatially displaced between frames. Figure 1.2 gives a simple example of such temporal redundancy. In this case, the background is static and the object, the stick figure, has moved

position. Most of the information to construct the new frame, called *current* frame from the *reference* frame resides in the reference frame. A temporal compression scheme could represent the new frame by taking the object from the old image and displace it spatially to recreate the new image. Thus only the information needed to displace the object is required and not the entire frame.

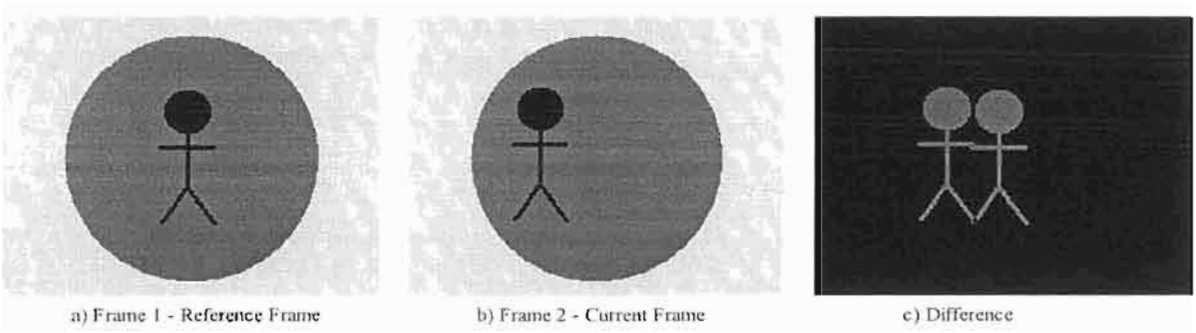


Figure 1.2: Example of Temporal Redundancy

Temporal compression of video tends to result in large compression ratios because the amount of data per frame needed to represent a second of full motion video can be drastically reduced. However, the techniques used are not perfect and hence temporal compression is a lossy process.

1.4 Roadmap to Dissertation

This chapter has presented an overview of the need for digital video and its usefulness in modern society. The requirement for compression of the video is explained along with some basic concepts and terminology of digital video. The chapter then explains the basic principles of compression as it applies to digital video. These fundamentals serve to prepare the reader for a more specific discussion in the next chapter.

Chapter 2 gives a brief explanation into the generation of digital video that serves to help in understanding the techniques used in the compression of the video. The chapter then proceeds by presenting the video compression fundamentals with a bias towards those topics most relevant to this dissertation. Some of the popular standard compression schemes are then explained before summarising the current technology available in video compression research.

Chapter 3 presents the system as implemented on the hardware. A systems overview is given and choices and justifications explained. Some extra theoretic information is provided where necessary to explain certain choices in the design.

Chapter 4 goes on to explain implementation considerations of the design. The chapter begins by giving an overview of the hardware used, highlighting parts that are relevant to the implementation. An explanation of software coding design is given and problems and solutions discussed.

The performance of the system is evaluated in chapter 5. This chapter begins by examining initial performance through the use of performance measures such as PSNR and subjective results for a set video sequence. The system is then compared to existing inter-frame based schemes and some intra-frame based schemes using standard video test sequences. The chapter concludes with some discussion of the results obtained highlighting the value of the implemented system.

Chapter 6 provides some suggestion for further improvement on the current work before leading onto chapter 7 which concludes the dissertation with an overview of important results and the contribution made.

Chapter 2 Video Compression Theory

This chapter introduces some of the theory of video compression, gives an overview of the evolution of the field and discusses some of the major research milestones. It starts by focussing on the still-image compression components present in most video compression schemes to date. Each component of the still-image compression scheme is discussed with further detail given to areas of special interest, i.e. areas that were implemented in the codec. A common block diagram of video compression scheme is given and the affects of temporal compression briefly discussed. As the focus of this dissertation is a video compression scheme without temporal compression, this component will not be discussed in depth but it still warrants mentioning as it is used in many video compression algorithms. Common video compression standards and relevant still image compression standards are presented to give an overview of existing internationally accepted schemes. The chapter concludes with a discussion of recent developments in compression schemes highlighting advantages and disadvantages of each.

2.1 Digital Video Representation

A video capture process consists of two basic parts: firstly the capture of a single frame and secondly, the speed at which successive frames are captured in order to create the illusion of motion. A block diagram of a video capture system is given Figure 2.1.

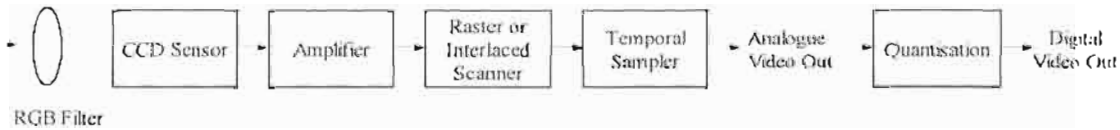


Figure 2.1: Basic Block Diagram of Video Capture

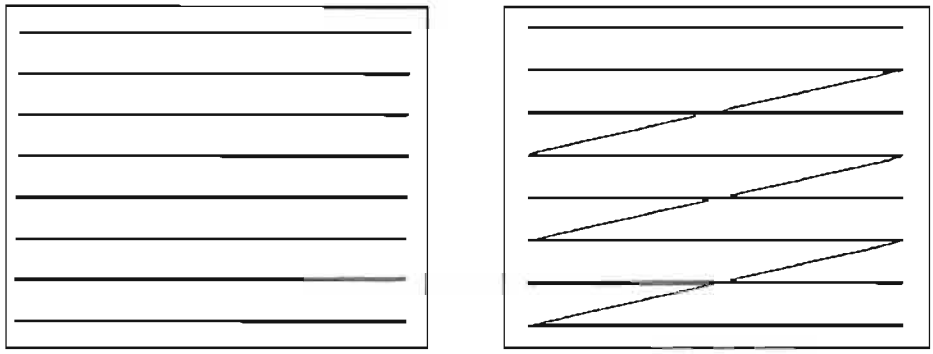
The first concern in capturing an image is representing the colour information. Classically it is known that the eye perceives colour as varying wavelengths of light, however it is also known that any colour can be represented by a weighted combination of a specific set of colours, known as *primary colours* [Hech87]. There are two commonly used sets of primary colours; namely the Red-Green-Blue (RGB), an *additive* set as the presence of these colours at their maximum intensities equates to white, and Cyan-Magenta-Yellow (CMY), a *subtractive* set as the absence of these colours equates to white [Hech87]. Thus in a video capture system, the light intensities representing the scene are first filtered into the contributions of each primary colour toward the scene and then captured by a photosensitive device.

The next step in capture is the sensing of the RGB information and the sampling of this information to represent the image. The sensor used in such applications is commonly a Charged-Coupled Device[2] (CCD) [Hoan01] arranged in an N×M lattice that is sampled according to a specific scanning pattern, often termed *raster* scanning [Hoan01], to produce a continuous analogue signal representing the image.

---

[2]In recent times there has been a great deal of interest shown in the use of CMOS devices for image capture. These devices offer superior integration, power dissipation and system size while sacrificing some image quality (particularly in low-light areas) and flexibility. However with the advances made in technology, the image quality produced by CMOS devices is still suitable for many applications and is slowly beginning to rival that of it CCD counterparts.

When subsequently displaying the captured image, the display device follows the same scanning order to display the image from the analogue waveform. Two types of scanning are commonly used, *progressive* and *interlaced*. In a progressive system a frame is represented in one complete scan from top to bottom left to right as shown in Figure 2.2a). An interlaced scan Figure 2.2b) scans the odd lines of the images first and then the even lines to produce two interlaced fields representing an image frame. Progressive scanning is typically used in computer displays while interlaced scanning is used in TV. The interlaced scanning for TV allows for reasonable temporal resolution in scenes of motion and acceptable vertical resolution in static scenes while maintaining modest bandwidth requirements. Once the information has been captured in an analogue format, transmission of the video signal can be achieved through similar modulation techniques used in radio.



a) Progressive Scan b) Interlaced Scan

Figure 2.2: Scanning Patterns used in Video Capture

For video, the images are sampled in the time domain, which is essentially sampling a number of still images in a certain period to meet the desired temporal resolution.

Quantisation converts the continuous analogue signal into a discrete set of digital values. Digital data cannot represent data with infinite precision as this requires infinite storage space; hence digital data must be of finite precision, where increasing accuracy requires increasing storage space. The quantiser meets this trade-off by mapping the continuous signal onto a digital set of values to a precision acceptable to the user, while keeping the required storage space to a minimum. The quantiser shown in Figure 2.3 has seven discrete levels, called *bins*, each having a binary code assigned to it. Each bin, except for the extreme values and 0, has the same step size which classifies this quantiser as a *uniform* quantiser. The number of quantisation steps is often a power of two so that a binary number can be assigned to each bin. Standard video schemes generally use 8-bit quantisation for digitising video frames [Bhas97][Sola97] as this precision results in no perceivable loss of image quality.

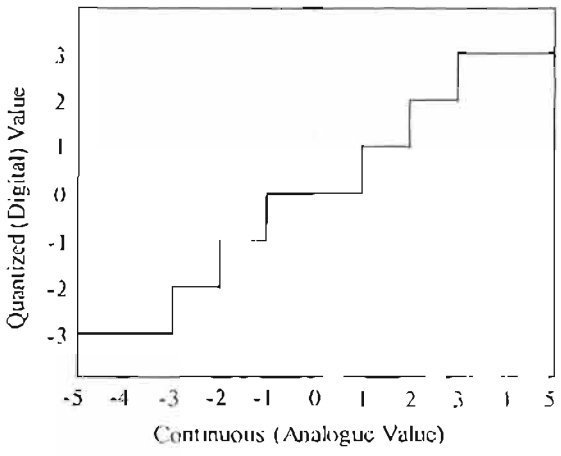


Figure 2.3: A 7-Level Uniform Quantizer

Thus the result of the capture process is a digitised series of N × M matrices of pixels representing images which when played back quickly creates the impression of motion. (see Figure 2.4.)

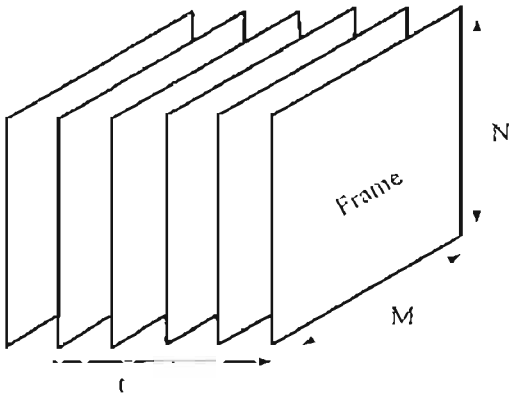


Figure 2.4: Digital Video Representation

2.1.1 Colour Spaces

In colour digital video, each frame consists of three matrices, representing the contributions of the three primary colours (usually RGB). This representation is very cumbersome as a colour video stream is three times the size of a monochrome video stream. However, the HVS's poor colour sensitivity can be exploited by separating those components that contribute solely to colour from those that contribute to image brightness [Week96, Bhas97]. Generally this is achieved through transforming colour into a brightness component, which is effectively the monochrome image, plus two colour components, which when combined with the brightness produce the correct colour information.

There are three commonly used colour spaces which separate colour from brightness [Bhas97]. These are YC_BC_R, YUV and CIELAB. These three spaces represent a colour by its' brightness, or *luminance*, component and two colour, or *chrominance*, components. Each space can be generated through linear matrix operations on the RGB space as is shown in Appendix A.

Once separated the colour components can now be sub-sampled (reduced spatial resolution) with minimal perceivable visual loss. The video-compression scheme chosen for this implementation uses the YUV space and so examples of sub-sampling are given in the YUV space. In the YUV space, the colour

components (U and V matrices) are sub-sampled according to three commonly used sub-sampling standards, 4:4:4, 4:2:2, and 4:2:0[3] [Poyn95]. 4:4:4 represents no subsampling, where the Y, U and V matrices are all the same size. 4:2:2, (shown in Figure 2.5) sub-samples the U and V components by a factor of two horizontally while 4:2:0, (see Figure 2.6) sub-samples the U and V components by a factor of 2 in both directions. The origin and significance of the numbers used in representing the subsampling is given in Appendix A.

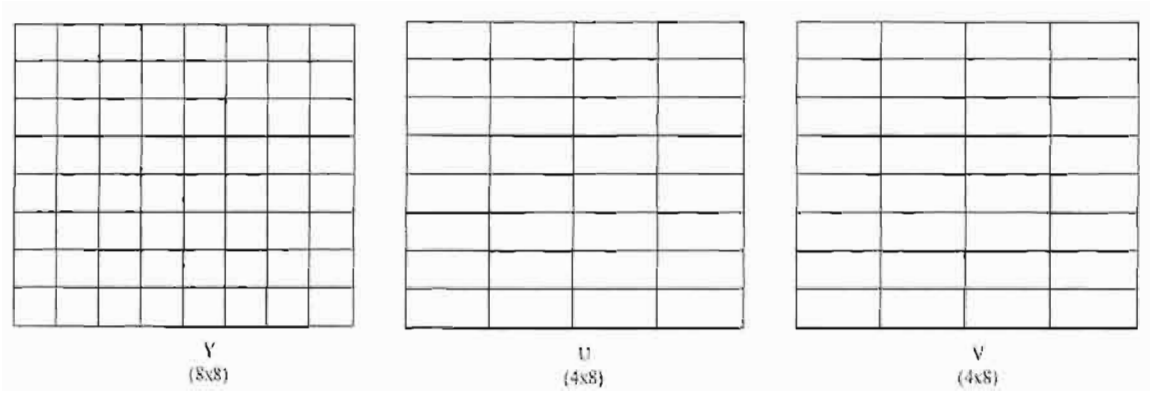


Figure 2.5 : The 4:2:2 format

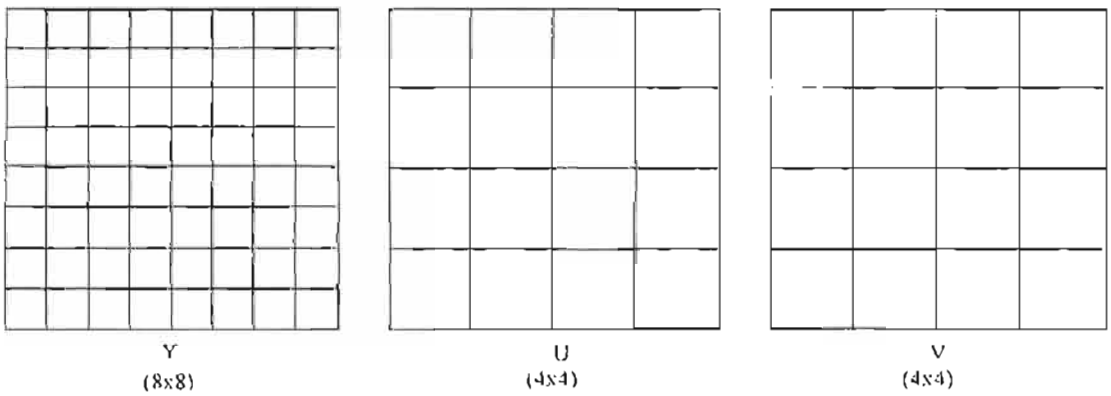


Figure 2.6 : The 4:2:0 format

In terms of digital television, the ITU-R BT.601-4 [ITU601] specifies video formats for standardisation. For National Television System Committee (NTSC), the recommendation specifies a spatial resolution of 720×480 and temporal resolution of 30 frames per second and for Phase Alternate Line (PAL) and Sequential Couleur Avec Memoire (SECAM[4]) it specifies a spatial resolution of 720×576 at 25 frames per second. Colour for both standards is represented as luminance and chrominance with 4:2:2 subsampling used. A commonly used extension of these standards is the Source Input Format (SIF) [Bhas97] which has a luminance frame of 360×240 at 30 frames per second for NTSC and 360×288 at 25 frames per second for PAL. The obvious drawback is the compatibility between the standards, for

[3] 4:2:0 should not be confused with 4:1:1 which is colour subsampling by a factor of 4 in only the horizontal direction.

[4] The difference between PAL and SECAM is found in broadcast television where colour is transmitted slightly differently. However in terms of spatial and temporal resolutions both standards are equivalent.

example, SIF as used in NTSC, is not compatible with SIF as used in PAL. Therefore, the Common Intermediate Format (CIF) has been introduced which, as with any good compromise, matches neither of the two existing standards but both being able to adapt to fit the format. CIF has a spatial resolution of 352×288, giving it half the linear resolution of the TV standards, and a temporal resolution of 30 frames/sec with colour sub-sampled at 4:2:0. Hence, NTSC systems require spatial re-sampling whereas PAL systems require temporal re-sampling. For videoconferencing and low bit-rate applications the Quarter-CIF (QCIF) format is offered which is half the linear resolution of the CIF format (176×144). For such applications, the frame rate is often reduced from 30 frames/sec to 5 frames/sec. This drastically reduces the amount of storage space needed for the video but results in a 'jerky' motion video.

2.2 Lossless Compression Techniques

Lossless compression techniques are used in all image and video compression schemes [Syme01]. They can be used on their own or in conjunction with a lossy based scheme, however used on their own they suffer from low compression ratios with compression being largely dependent on the input data. Certain applications, such as medical imaging, require zero quality loss so there exists a great deal of lossless compression research. Since lossless compression is used in conjunction with most lossy schemes, a discussion on popular lossless schemes is relevant here. The major classes of lossless encoding are those that use predictive techniques and those that use some model-based schemes to compress the source. The rest of this section describes the more popular lossless encoders used.

2.2.1 Predictive Encoding

Predictive encoding is a branch of compression, generally not used in conjunction with lossy schemes, that has proven very effective for lossless compression [Sola97]. The predictive encoder, as its name suggests, makes use of prediction of pixels in an image. These predictive algorithms use the fact that natural images are often Markov sources, meaning that the probability of a particular pixel value, in an image, is dependent on the pixel values surrounding that pixel (i.e. natural images are quite highly correlated) [Syme01]. So, instead of encoding the intensity value of a certain pixel, the predictive encoder uses known surrounding pixel intensities to predict the intensity of the current pixel, relying on the decoder following the same prediction process. The prediction is then compared to the actual value and the error term transmitted. The decoder follows the same prediction process and uses the received error term to correct its prediction. Compression is achieved in that the prediction is likely to be very close to the actual value and thus the error term will be small, meaning fewer bits will be needed to represent it.

The accuracy of the prediction and the complexity of the algorithm increase with the number of surrounding pixels used to make the prediction. Hence, a successful predictive encoder tries to achieve maximum accuracy with minimum encoding complexity. Habibi [Syme01] showed that there is a substantial increase in predictor accuracy when going from predictions based on 1 surrounding pixel to those based on 2 surrounding pixels and from 2 to 3 but little significant increase thereafter, so most practical prediction schemes use 2-3 surrounding pixels to make a prediction allowing for modest computation complexity with acceptable accuracy.

An advance on the fixed predictive scheme is adaptive prediction. In this scheme, the encoder uses a predefined series of predictors to perform the prediction. It then chooses the best prediction and sends its

corresponding error along with the predictor used. This produces an efficient prediction error but there is the increased overhead needed to tell the decoder which predictor to use. A more efficient adaptive scheme is able to determine the choice of predictor from the data values so that the decoder can make the same choice and thus eliminate the overhead.

Although there have been advances in the performance of predictive techniques [Wu97] [Robi97], these techniques still remain in the lossless domain and are not suited for low bit rate compression.

2.2.2 Lempel-Ziv-Welch

The Lempel-Ziv-Welch (LZW) scheme is an adaptation of the LZ77 [Ziv77] and LZ78 [Ziv78] encoders developed by Lempel and Ziv. These encoders are dictionary based, meaning that they build up a dictionary of previously used strings of characters and assign references to each dictionary entry. The resulting encoded output is a series of dictionary references which is used by the decoder to reconstruct the original. Coding gain is achieved by the re-usability of dictionary entries as the source data can often be represented by a combination of existing dictionary entries.

In LZW [Welc84] there are some fundamental changes that make it slightly more efficient. The LZW uses a table of entries with an index field and a substitution-string field, which is pre-loaded with every possible dictionary entry. In practice, this is generally the ASCII character set from 0-255. The encoder then assigns codes (which are references to entries in the dictionary) to input data, adding any new strings to the dictionary. Compression is achieved when large strings of symbols are represented by a single dictionary reference. The algorithm is particularly effective in data streams that have a great deal of repetition such as graphic images. The advantage of the algorithm is speed of compression and decompression; however, it is unable to meet the compression potential of the model-based schemes [Witt87]. Common image formats like the UNIX "compress" utility and CompuServe's Graphics Interchange Format (GIF) are the most well-known implementations of the LZW scheme.

2.2.3 Entropy Encoding

Entropy Encoding is a lossless encoding scheme that makes use of the frequency distribution of symbols appearing in the uncoded data stream. While the concept of Entropy may be a complex one, the principle of entropy encoding can be explained easily with some careful definition of terms.

If the source data is represented as an 8-bit value having any number between 0 and 255 then the entropy encoder will use a symbol to represent each possible value in the source data stream, and transmit the sequence of symbols. The choice of the symbol is governed by a set of predefined rules and as long as the decoder follows the same rules, the original value can be recovered from the symbol. The obvious way to code the symbols is to just send the value itself, but then there is no evident coding gain. An important factor in entropy encoding is that the symbols need not all be the same length and as long as there is some way of determining where one symbol ends and another begins the decoder is able to recover the data.

This variable symbol length is what allows the entropy encoder to achieve compression. By assigning short symbols to frequently appearing values and longer symbols to less frequent values less data is needed to represent the entire source data set and thus compression is achieved. Using information theory this process can be proven to provide lossless compression [Taub86]. While the entropy encoder can provide good compression, the encoder must have prior knowledge of the frequency of values in order to

determine the best possible symbol mapping. Thus the choice of the model for determining the frequency of values has a huge impact on the efficiency of the encoder. There exist many entropy encoders (Fano, Gilbert etc.) but for most popular video compression schemes, the Huffman encoder is the most popular choice [Bhas97]. The entropy encoder is usually the last step in any image and video compression algorithm.

Huffman Encoding

Huffman encoding is an algorithm used to produce size-efficient codes to express data for which there are known probabilities of occurrence. The algorithm was first introduced by David Huffman in 1952 [Huff52]. The best way to explain the algorithm is through example.

We first define the source data to be constructed from an *alphabet* of fixed length symbols where the probability of occurrence of each symbol is known and tabulated as shown in Table 2.1(a). These alphabetic symbols could be anything, such as pixel values of an image, for example. Table 2.1(a) shows a sample alphabet of six symbols indicating that, for example, the symbol 'e' is 4 times more likely to be transmitted than the symbol 'd'. The Huffman algorithm starts by arranging the probabilities in decreasing order as shown in Figure 2.7. The two lowest probabilities are assigned the values '1' and '0' and then combined to form a new probability. This, in conjunction with the next lowest probability symbol are combined in a similar fashion and the process repeated until all probabilities have been accounted for and the tree completed, Figure 2.7. Once represented in the tree structure, the assignment of codes is achieved by traversing the tree in reverse order. To illustrate this Table 2.1(b) shows the Huffman codes assigned to the relevant symbols. As can be readily seen the letters of highest probability ('e','c') have the shortest codes as they appear more frequently than other letters. Huffman [Huff52] proved that by assigning codes to symbols in this manner the entropy could be reduced if an efficient model for the probability distribution for the data was used.

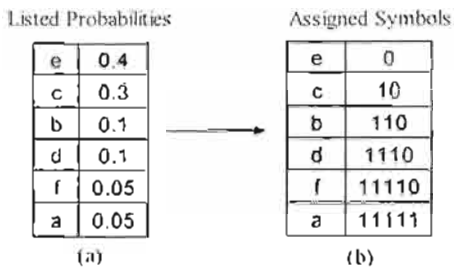


Table 2.1 : Example of Huffman Encoding

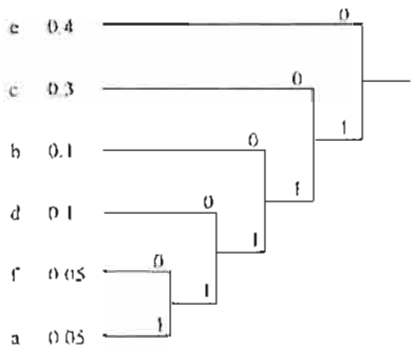


Figure 2.7 : Huffman Code Tree

Adaptive Huffman Encoding

An improvement to Huffman encoding, conceived independently by Faller [Fall73] and Gallager [Gall78] and then further improved upon by Knuth [Knut85] is Adaptive Huffman encoding or the FGK algorithm. An alternative adaptive scheme was introduced by Vitter [Vitt87] in 1987. All these methods function, as in normal Huffman encoding, by mapping source data into codewords to achieve an optimal representation of information. The difference in the adaptive scheme is that the encoder is continually updating the source probabilities to provide a more accurate statistical representation of the source. Thus, the encoder is effectively learning the characteristics of the source. This means that the decoder must do the same, continually updating the Huffman tree and maintaining synchronization with the encoder.

Of course, if there exists a known statistical representation of a source and this is not expected to change, then there is no advantage to using the adaptive Huffman over the normal static Huffman. This means that the performance of the adaptive method over the static method relies on the type of source data being encoded. For ever-changing source data, like a stream of video, adaptive Huffman encoding provides useful results with increased complexity [Vitt87].

2.2.4 Arithmetic Encoding

Arithmetic encoding is a very efficient lossless compression scheme, first presented by Witten et al in 1987 [Witt87]. The arithmetic encoder takes a stream of input symbols and replaces this with a single, real number between 0 and 1. This is in contrast to Huffman encoding which replaces an input symbol with a code whose size is determined by a probability distribution of the input stream. The longer the input stream the longer (i.e. more precision needed) the real number needed to represent it.

Symbol	Probability	Range
a	0.2	[0. 0.2)
b	0.2	[0.2, 0.4)
c	0.1	[0 4, 0.5)
d	0.2	[0.5, 0.7)
e	0.3	[0.7, 1)

a) Symbols in alphabet and respective probabilities and ranges

Transmit Sequence "a b c d e"

Symbol	Low	High
a	0	0.2
b	0.04	0 08
c	0.056	0.06
d	0.058	0.0588
e	0.05856	0.0588

0.04 = 0 + (0.2 x 0.2)
0.08 = 0 + (0.2 x 0.4)

0.056 = 0.04 + (0.04 x 0.4)
0.060 = 0.04 + (0.04 x 0.5)

b) Transmitted symbols and adjusted ranges due to coding

Transmit any number between 0.5856 inclusive and 0.0588 exclusive

Figure 2.8: Example of Arithmetic encoding

An example of arithmetic encoding is given for clarification. Consider an arbitrary alphabet used for transmission with probabilities of occurrence as given in Figure 2.8a). An arithmetic encoder would then divide the alphabet into ranges between 0 and 1, according to the symbol probabilities as in the Figure 2.8a). If the transmission sequence of "a b c d e" were to be transmitted, the encoder would start with the initial symbol "a" and list its low range value and high range value. The next symbol to be transmitted is "b" and so the encoder now calculates the new range of "b" with respect to the range of "a" as seen in Figure 2.8b). This process is repeated until the message has finished and a final low and high range value left, 0.05856 and 0.0588 in this example. Now the encoder need only transmit any value between the low inclusive and the high exclusive and the decoder would be able to reconstruct the message. Decoding is achieved by reversing the range calculation of the encoder with the constraint that the decoder must have the same probability map as the encoder for correct reconstruction. In the example, if the encoder were to transmit 0.0587 then the decoder immediately knows that the first symbol is "a" as the transmitted value lies within and only within the probability range of "a". To calculate the next symbol the decoder subtracts the low range value of "a" (which is 0) and then divides by the probability of the symbol "a" to obtain 0.2935, which lies in the range of the symbol "b". This process is continued to decode the entire sequence. An immediately noticeable problem with arithmetic encoding is the termination of the decoding process. This is usually achieved by sending an EOF character or carrying the stream length along with the encoded data.

In this example, the message can be represented as 4 8-bit values (0.5,8,7) as the decoder can assume the decimal point at the left. This is opposed to sending the 5 values for each symbol. This is perhaps not the greatest compression achieved but the example is given only to show the principle of the system. In practice and with good source modelling, very high compression ratios can be achieved. Although this example is a base 10 case (for easier understanding) the arithmetic encoder can be adapted to the binary case, which is desirable for most compression schemes.

2.3 Measurement of Image Quality

Image quality is a very subjective attribute and hence difficult to measure [Syme01]. Nevertheless, to allow for some formal comparison of compression techniques, some form of objective quality measure is desired. In the case of image and video compression the accepted norm for quality measure is the Peak Signal to Noise Ratio (PSNR) as given in (2.1).

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right)dB \qquad (2.1)$$

Generally, a good quality image is seen to be one with a high PSNR, however a low PSNR does not, necessarily, mean an image is of bad quality [Bhas97]. An example of such a case is a decompressed image which is slightly brighter than the original image, but in all other aspects, identical. In this case, the resulting PSNR calculation would be low and thus objectively the image is deemed of poorer quality, while, in fact, it may look more pleasing to the eye. However, although the measure is flawed, a better comparison technique has yet to be adopted and thus the PSNR has remained the standard objective measure of quality. To make up for the flaws in the objective measure, image and video compression results are often quoted in both subjective and objective terms in order to gain a fairer measure. Such a subjective technique is achieved by providing visual results along with the PSNR results.

2.4 Still Image Compression Techniques

In section 2.2, lossless compression techniques were reviewed. Since the compression algorithm implemented in this dissertation is a lossy scheme, from now on, compression is treated as lossy unless specifically stated. Video is essentially a collection of still images and so compression of video and still images is very closely related as all the principles used in still image compression are also used in video compression [Syme01]. Figure 2.9 shows a basic systems view of a still image compression process.

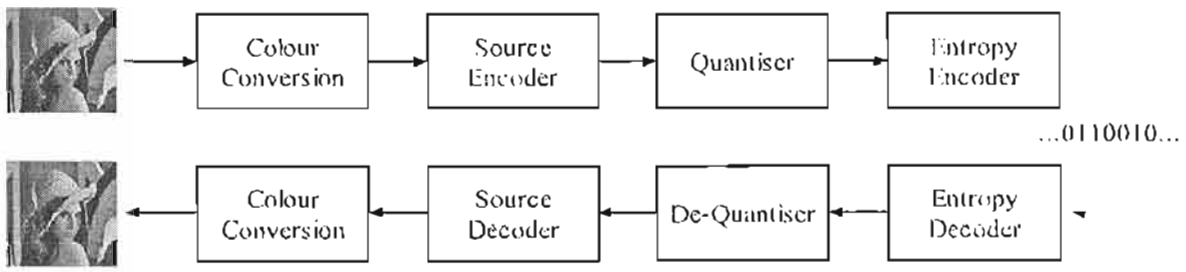


Figure 2.9: Block Diagram of Common Still Image Compression Scheme

The remainder of this section provides a brief overview of each of the subsystems that comprise Figure 2.9.

2.4.1 Colour Conversion

The first concern of any image compression / decompression scheme (often termed a *codec*) is the treatment of the image's colour. Successful codecs must be able to convert an image's colour to a desired colour space. Typical colour spaces used are YUV, YC_RC_B and CIELAB [Bhas97] as these result in good compression performance, as discussed in Section 2.1.

2.4.2 Source Encoder/Decoder

The source encoder/decoder transforms the image data into a less correlated domain. Since the compression achieved is directly related to the extent at which the data is decorrelated [Taub86], the transform that is able to maximally decorrelate the image data is desired. In an implementation based system the computational complexity is of vital import, so a transform must also be computationally inexpensive. Two of the most popular image compression related transforms are presented below, with the advantages and disadvantages of each highlighted.

Karhunen-Loeve Transform (KLT)

The KLT is considered the optimum transform for image coding applications as it is able to completely decorrelate the image data, allowing individual processing of transform coefficients without affecting the others [Elli82]. The KLT's basis functions are the eigenvectors of the covariance matrix of an input signal and therefore the calculation of the KLT requires determining the covariance matrix for the input and diagonalising the matrix. In terms of video processing, this means that for each image frame the covariance matrix needs to be calculated in order to calculate the eigenvectors and eigenvalues for the transform. This calculation is very computationally expensive and thus the KLT is seldom used in image and video processing [Bhas97]. However, for comparison purposes the KLT provides a good benchmark for other transforms. Elliot et al give a comparison of the performance of various transforms as compared to the KLT [Elli82]. The results show that the DCT and the DFT provide the closest match to the KLT and these have gained popularity in signal processing. In image-processing the DCT has gained popularity as it orders transformed coefficients in an easily compressible manner and there exist fast algorithms which allow for fast implementations.

Discrete Cosine Transform (DCT)

Since it was first introduced by Ahmed, Natarjan and Rao in 1974 [Elli82] followed by the subsequent fast implementation algorithms of Kamanga [Kama82] and Cho [Cho91], the 1-D and 2-D DCT has been the mainstay in image and video compression schemes. This transform algorithm neatly decorrelates the image data and allows quantization and entropy techniques to efficiently compress the transformed coefficients while maintaining acceptable computational cost. The advantage of this transform, over the KLT, is its image independence meaning the image does not need to be pre-processed to determine the optimal set of basis functions. This feature comes at the expense of efficiency of decorrelation and the DCT is therefore a sub-optimal solution.

Images are two-dimensional and hence the 2-D variation of the DCT (Equation 2.2) is used. The 2D-DCT decomposes an image into a series of cosine functions varying in magnitude and spatial frequency. The output of the transform is a two dimensional matrix representing the contributions of each frequency component starting from DC (situated at the top-right corner) with increasing horizontal frequency to the

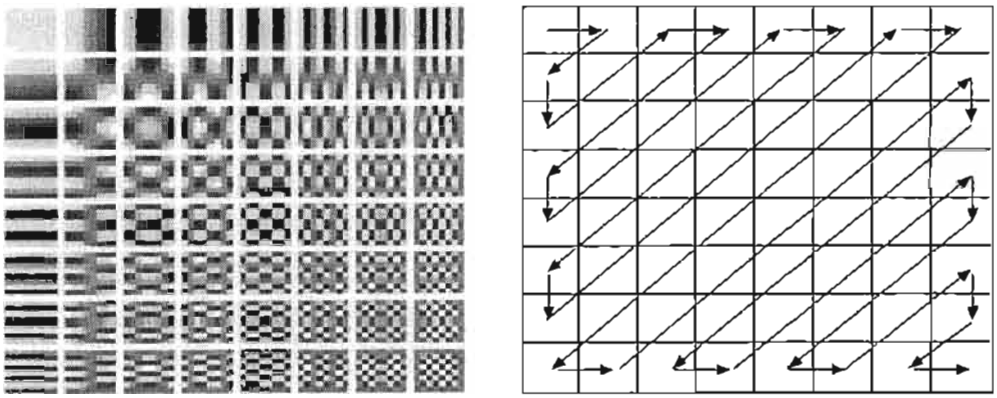
left, and increasing vertical frequency down Figure 2.10(a). The DCT uses the fact that natural images contain a great deal of energy in the lower frequencies (representing the average brightness of the image) and low energy in the higher frequencies (representing the details of the image) [Week96]; hence the most significant transform coefficients representing the lower frequencies reside in the upper left triangle of the transformed matrix. Compression is achieved by quantising and thresholding those coefficients deemed irrelevant and then entropy encoding the transformed data. This process must be carefully controlled as too much removal of data can result in an unacceptably degraded result. As the coefficients in the transformed matrix are ordered in a diagonalised manner an efficient compression algorithm scans the matrix in a diagonal manner, Figure 2.10(b), to maximise the groups of insignificant data which results in optimal performance from encoders such as RLE.

$$C(n,m) = k_1(n)k_2(m)\sum_{y=0}^{M-1}\sum_{x=0}^{N-1} f(x,y).\cos\left(\pi n\frac{x+1/2}{N}\right).\cos\left(\pi m\frac{y+1/2}{M}\right) \qquad (2.2)$$

Where

$0 \le n \le N-1$

$0 \le m \le M-1$



(a) Frequency Map (b) Scanning Order

Figure 2.10 : Illustration of Frequency Distribution and Scanning Order in DCT

Practical implementations of the DCT divide the video frame into N×N blocks to reduce computational complexity. The choice of blocksize is important as the DCT must not use unreasonable memory resources and should not be too computationally complex. Typical DCT based compression schemes use an 8×8 blocksize, which has been shown to result in fast implementation and modest memory requirements [Bhas97].

Image and video compression schemes using the 2D-DCT have the advantage of using a well established transform algorithm that allows for good compression results over a wide range of compression ratios. However, these algorithms suffer at high compression ratios, producing images and video streams that are 'blocky'. This effect is a direct result of dividing the image into N×N blocks, as the higher the compression ratio, the more significant coefficients get discarded and thus the N×N block cannot be acceptably recovered, causing discontinuities at the edges of each block. Figure 2.11 shows a example of a highly compressed JPEG image highlighting the 'blocky' effect.

Figure 2.11 : Example of 'blockyness' with the DCT

The 'blockiness' associated with the DCT can be overcome and techniques such as filtering along the edges of each block to reduce the affect of 'blockiness' have proven reasonably successful [Bhas97]. Another technique used, is to vary the blocksize of the macroblocks according to the activity in an image, for example, assigning larger blocksizes to areas in the image with less activity (less entropy). The variable block size DCT has also proven successful but still is unable to totally remove the 'blocky' affects [Chen89].

2.4.3 Quantisation

The transformed image often results in a matrix of real coefficients with a large dynamic range and therefore some quantisation is needed to reduce the range by removing insignificant coefficients and reducing the precision of significant coefficients. A well-designed quantiser is able to reduce the precision of transformed coefficients without perceivable loss in image quality; the choice of which coefficients are insignificant making the largest impact on image quality loss. Generally, there exist two types of quantisers used in image and video compression, scalar and vector quantisation.

Scalar Quantization [Sola97]

An example of a uniform scalar quantiser has already been explained in Section 2.1 but a scalar quantiser need not be uniform. In fact, in some cases a non-uniform quantisation process is often more advantageous. For example if input data has a range of 0-5 and it is known that most of the input data points will occur between 1-3 then a quantizer can be designed to be more accurate in the region between 1-3 and less accurate between 0-1 and 3-5. This will minimize the quantisation error for most data points. A further advancement is an *adaptive* quantizer where the quantisation step size is varied over time according to the variance of the input signal. This form allows for a better and more compact quantised representation of the input signal at the expense of increased computation, as the variances of the input need to be constantly recalculated.

Vector Quantization (VQ)

Vector quantisation, unlike scalar quantisation, operates on multiple data values, called *vectors*, making use of codebooks to assign codewords to vectors of input data. Figure 2.12 shows a VQ system. The input vector is compared to all vectors listed in the codebook and a 'best fit' is found whereupon the binary code for this vector is output. The strength of VQ lies in the fact that the output binary code consists of far less bits than the input vector and so compression is achieved. Obviously, this quantization scheme relies heavily on the size and the accuracy of the codebook, as the larger the codebook the better the accuracy of the quantisation but, the larger a code book, the longer the binary code becomes, decreasing compression performance. A larger codebook also increases the processing power required to search it and so there exists a trade-off between codebook size, processing power and compression efficiency; hence there are many algorithms that aim to provide an optimal codebook with minimal complexity [Bhas97]. As in scalar quantisation, there are adaptive VQ schemes where the codebook can change according to the input data. In these cases the generated codebooks needs to be transmitted along with the coded data for effective decoding.

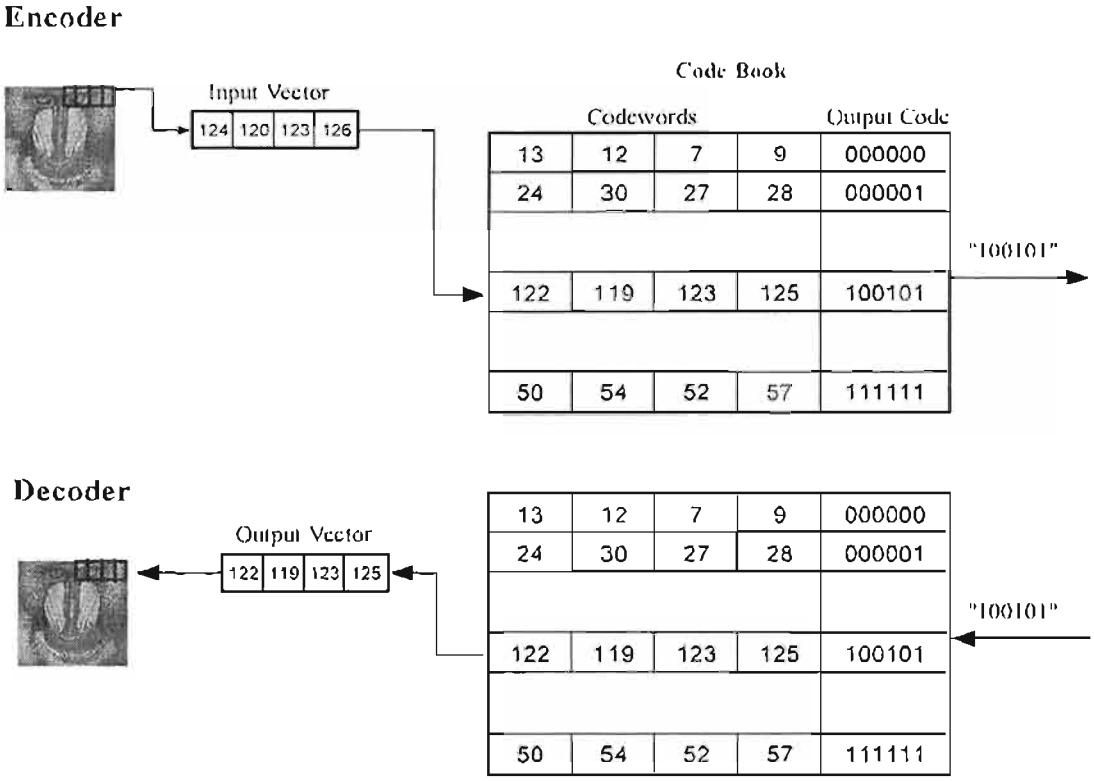


Figure 2.12 : Example of a Simple Vector Quantization Process

Known video compression standards that take advantage of VQ are Cinepak from Radius and Indeo 3.23 from Intel [Brya95]. A Video compression scheme using an advanced VQ method was implemented on a VAMPIRE processor and results reported by Fowler et. al. [Fowl95]. In this scheme the use of VQ is shown to provide useful real-time video compression; however compression ratios reported were not suitable for low bit rate applications and the scheme lacks scalability as the compression was dependent on the source video stream.

2.4.4 Entropy Encoder/Decoder

The final stage of common, still-image compression techniques is usually a lossless encoding scheme. This is generally an entropy-based scheme with Huffman or Arithmetic encoding being the more popular choices [Wall91].

2.5 Fractal Compression

Fractal image compression is a relatively new and promising compression technique. Fractals are signals that recursively contain themselves [Jacq97]. A zoomed fractal signal looks as though it has infinite resolution but is, in fact, the signal reappearing in itself. A fractal image compression scheme tries to find fractals in the image by searching for a combination of transforms that best represents that image. The advantage of fractal compression is a very fast decoding time and the ability to decompress an image to any resolution. However the process of finding the fractals for compression is a very time consuming task and, as such, the encoding of fractal compression is slow. By way of example Uys [Uys00] reports an implemented fractal image encoding system using parallel Texas Instruments TMS320C80's with an encode time of approximately 11 seconds and a decode time of about 0.8 seconds. Thus, although it has been shown that fractals can provide good compression results [Papa92][Nich96], for practical, real-time DSP video implementations the use of fractals is severely limited.

2.6 Wavelet Compression

The Wavelet transform has gained enormous popularity in video compression circles as it provides better compression results than the DCT counterparts at similar computational cost [Marp99] for applications that require large amounts of compression. This section aims to introduce the wavelet concept but, as the field is very large, only parts that are deemed relevant to the dissertation are presented and a rigorous theoretical approach is omitted. An interested reader is referred to [Chan95] and [Chui92] for an excellent theoretical introduction.

2.6.1 Theory

The purpose of a wavelet transform is to decompose a signal $f(t)$ by iteratively convolving the signal with an infinite set of wavelets continually varying in scale and translation, all derived from a *prototype* or *mother* wavelet [Chan95]. Mathematically this is represented as the Continuous Wavelet Transform (CWT) as in (2.5) with $\psi_{a,b}(t)$ being the mother wavelet as described in (2.6). The parameters a and b correspond to the varying scale and time respectively and the factor $|a|^{-1/2}$ ensuring that there is energy normalization across the different scales [Riou92]. An important factor is that the actual wavelet basis functions are not defined but rather a framework is presented which highlights the general properties of a wavelet transform. Thus any function meeting the requirements of a wavelet can be used to perform the wavelet transform.

$$\begin{aligned} &Foward: \quad CWT(a,b) = \int_{-\infty}^{\infty} f(t)\psi^{*}_{a,b}(t)\,dt \\ &\quad * = complex\ conjugate \\ &\quad (a,\ b)\ vary\ continuously \end{aligned} \tag{2.5}$$

$$Inverse: \quad f(t) = \int_{-\infty}^{\infty}\int_{a>0}^{\infty} CWT(a,b)\psi_{a,b}(t)\,dbda$$

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right) \tag{2.6}$$

For purposes of explaining how the transform is implemented, some important properties need to be highlighted. The most important properties of the wavelets are the *admissibility* and the *regularity* conditions [Riou92]. These are the properties that give the wavelet its name. It can be shown that square integrable functions $\psi(t)$ satisfying the admissibility condition (2.7) ($\Psi(\omega)$ is the Fourier Transform of $\psi(t)$) can be used to analyse and then reconstruct a signal without loss of information [Riou92]. The admissibility condition implies that $\Psi(\omega)$ vanishes at the zero frequency (DC), which means that the wavelet cannot have a low-pass characteristic. As it turns out the wavelet has a band-pass characteristic [Chan95]. This observation is important as it means adjustments must be made in order to represent an input signals' DC components. The null at the zero frequency also implies that the average value of the wavelet in the time domain must be zero (2.8). Therefore the wavelet must be oscillatory, i.e. a wave.

$$\int_{0}^{\infty}\frac{|\Psi(\omega)|^{2}}{|\omega|}d\omega < +\infty \tag{2.7}$$

$$\int_{-\infty}^{\infty}\psi(t)dt = 0 \tag{2.8}$$

A complete mathematical proof of the regularity conditions of the wavelet can be found in [Cohe92], but for the context of this dissertation only the implications are relevant. The regularity condition of the wavelet states that the wavelet must be smooth and concentrated in both time and frequency. This condition leads to a fast implementation, which is desirable for practical use of the transform.

2.6.2 Discrete Wavelet Transform

There are two significant problems that make the CWT impractical for image processing purposes. The CWT is calculated by continuously shifting a continuously scalable function over a signal and calculating the correlation between the two. However the wavelets used are often not orthogonal, meaning the transformed data will contain redundancy and thus not properly decorrelate the signal [Riou92]. As stated earlier the more decorrelation a transform achieves the better for compression, so it is desirable to have an orthogonal wavelet transform to achieve higher decorrelation. The second significant issue is that the CWT, as its name implies, is used for continuous signals and as such is not suitable for the discrete image environment. Thus the CWT needs to be discretised in order to effectively transform image data.

The CWT can be discretised as in (2.9). This means that the wavelet functions are now not continuously scalable and translatable but can only be scaled and translated in fixed steps [Chan95]. Usually $a_0 = 2$ and $b_0 = 1$ is chosen which corresponds to dyadic sampling on the scale and time axes, a natural choice for implementation purposes on computers, and also leads to special choices of wavelets that are orthonormal, translating to good decorrelation [Anto92].

$$\begin{aligned} DWT(m,n) &= a_0^{\frac{-m}{2}} \sum_k f(k)\psi\left(a_0^{-m}k - nb_0\right) \\ a &= a_0^m \\ b &= nb_0a \end{aligned} \tag{2.9}$$

Even with the DWT an infinite number of scaling and translations are required to represent a signal adequately. This is obviously quite unsuitable for practical implementations and so some method of realistically performing the transform is required. The solution to the problem is presented by Mallat [Mall89] using a technique called multi-resolutional analysis. In multi-resolutional analysis the wavelet transform is thought of as a filter bank with a signal being analysed at different resolutions (hence the term multi-resolutional). In this analysis a *scaling* function ϕ (2.10) is introduced along with the mother wavelet ψ (2.11). To understand the reason for the scaling function some explanation is required.

In the frequency space (Fourier domain) a wavelet compressed in time by a factor of 2 (resulting from $a_0 = 2$, $b_0 = 1$) is stretched in frequency and shifted upwards by a factor of 2 [Taub86] as illustrated by Figure 2.13. So in terms of Fourier space each time the wavelet is stretched in time by the factor of 2 its spectrum is halved, meaning it only covers half the remaining spectrum. Remembering that the wavelet transform is iterative, as the iterations continue the wavelet is continually scaled and can only represent half of the remaining spectrum. It is obvious that it would require infinite iterations of the transform to represent the entire spectrum of the input signal and the zero frequency would still not be represented. The observation supports what was implied in the admissibly property of the wavelet with the band-pass nature. Thus Mallat introduced the low pass scaling function, subject to the same dilation and translation of the wavelet, to represent the DC and low frequency parts of the signal.

$$\phi_{m,n}(x) = 2^{\frac{-m}{2}}\phi\left(2^{-m} - n\right) \tag{2.10}$$

$$\psi_{m,n}(x) = 2^{\frac{-m}{2}} \psi\left(2^{-m} - n\right) \tag{2.11}$$

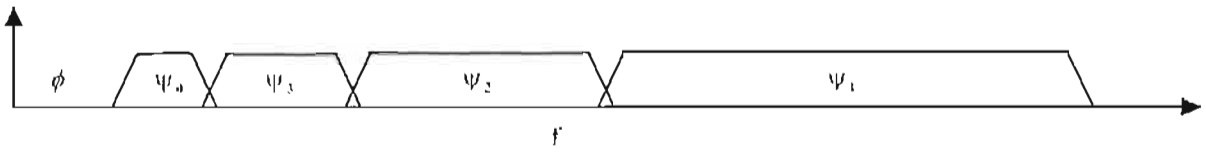


Figure 2.13: Band-Pass Nature of Wavelet

Further work [Riou92][Daub88][Cohe92][Esta77] showed that the DWT can be implemented using FIR filters, Figure 2.14, and more specifically the Quadrature Mirror Filters. Since the theory of FIR filters was well-known and the Quadrature Mirror Filter was known to engineers and widely used, this connection[5] is very significant as realistic implementations were now possible. In these implementations the scaling filter is often termed the *h* filter and the wavelet filter the *g* filter[6]. The wavelet transform can now be seen as dividing the input image into a set of spatially decorrelated frequency sub-bands, very similar in implementation to existing sub-band encoders [Bhas97].

Sub-band encoding schemes are well-known and have been used extensively, mainly in voice and audio encoding but also in image encoding [Bhas97]. However, the sub-band wavelet encoder has certain implications and conditions for the filter design that restrict the type of wavelets that can be used. Images are generally smooth and so it is appropriate that the exact reconstruction wavelet scheme should correspond to an orthonormal basis with a smooth mother wavelet. However for implementation purposes the filter needs to be short to reduce the computational load. This however leads to lower smoothness and so the filters cannot be too short. Also the filters need to be linear phase as they are required to be cascaded in a pyramidal structure without phase compensation [Vett92]. For this reason a special set of bi-orthogonal wavelet bases has been constructed to provide reasonable smoothness and short filter design with the exact reconstruction property [Anto92].

[5] This was not really a discovery as the QMF theory had stumbled onto the subject already without making the vital connection to wavelets

[6] This notation is from the multi-resolution analysis presented by Mallat and for consistency with the literature it will be used in this dissertation.

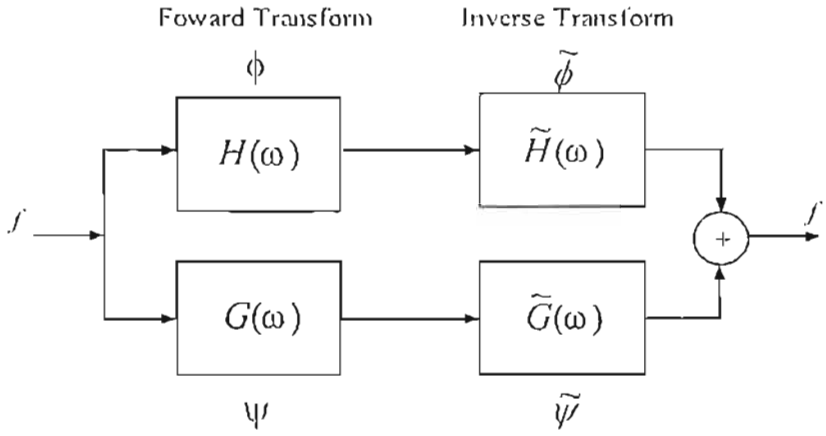


Figure 2.14 : Filter Bank Structure for Wavelet Transform

2.6.3 Boundary Handling

The issue of the treatment of signal boundaries in the wavelet transform filter bank implementation leads to some problems that require explanation. Applying the filter implementation to an image incorrectly will yield some discontinuities at the borders of the image [Bris96] as shown in Figure 2.15. Brislawn [Bris96] showed that this problem is easily solved by extending the image signal at the boundaries either symmetrically or periodically. Therefore in designing a wavelet transform implementation care must be taken in handling the borders of the image to achieve correct reconstruction.

Figure 2.15: Results of Incorrect Border Handling of Image

2.6.4 The 2-D Wavelet Transform as Applied to Images

As in the DCT an image is two dimensional and so a two dimensional wavelet transform needs to be applied. This is achieved by applying the 1-D wavelet transform on the rows and then on the columns with dyadic down-sampling to achieve spectral coverage as shown in Figure 2.16. The process is inversed to reproduce the image losslessly.

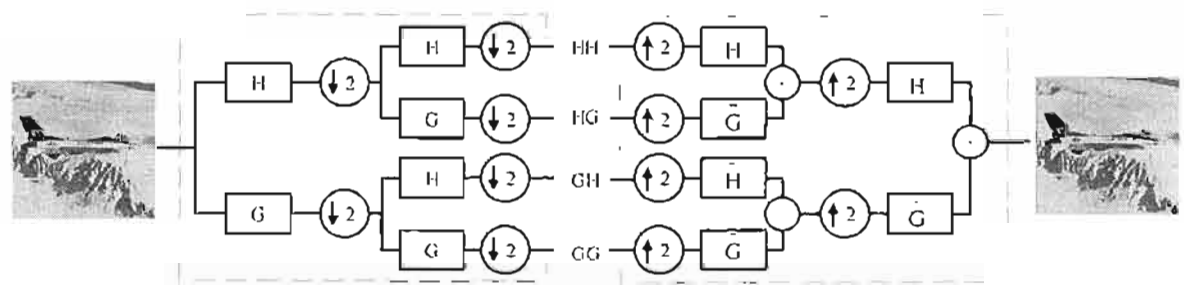
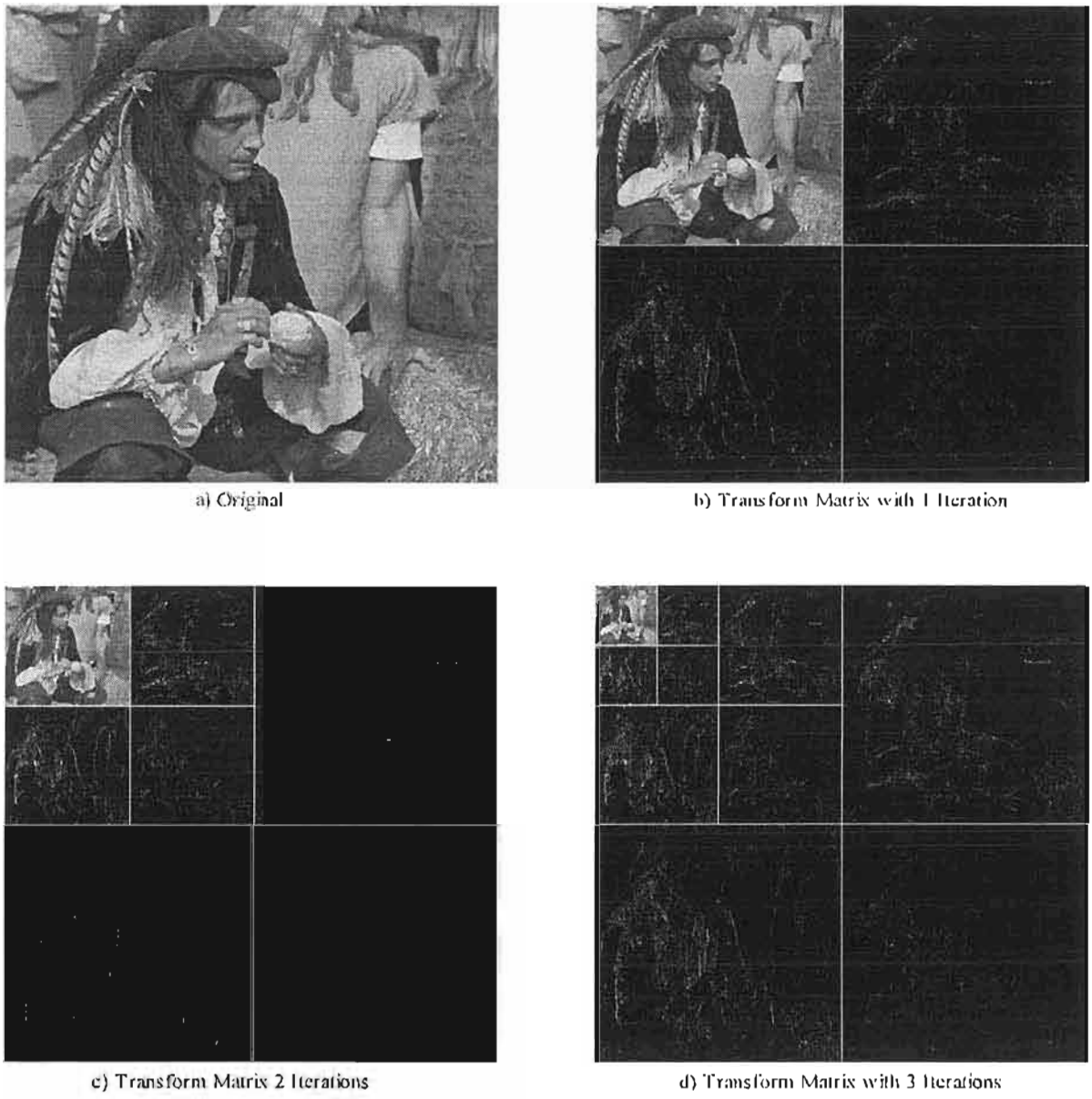


Figure 2.16: Filter Implementation of the 2-D Wavelet Transform

In the filter implementation as shown in Figure 2.16 the result of a single iteration divides the image into four sub-bands representing the low frequencies of the image, or the average (HH), and the high frequencies in each direction; horizontal (HG), vertical (GH) and diagonal (GG). In order to get better decorrelation, the process can be iterated on the low-frequency (HH) result, until either the average of the

image is represented by a single value or a desired point of decorrelation has been reached. Figure 2.17 shows the result of a 2-D wavelet transform iterated 3 times with the transformed results displayed at each iteration. This figure gives an engineering perspective on the wavelet transform as it shows graphically exactly what is happening. To match up with the notation introduced in Figure 2.16 and referring to Figure 2.17b the top left matrix is the result of applying the H filter in both directions (HH), the top right is the H filter on the rows and the G filter on the columns (HG), the bottom left is the result of the G filter on the rows and the H filter on the columns (GH) and the last matrix is the result of the G filter applied in both directions (GG). The transform is iterated on the HH result as can be seen in the figure.

To support what has been discussed, it is clear that the transform is in fact dividing the image into frequency components, with the entire image being visible throughout, at decreasing resolution. The other parts of the transformed matrix are those areas of detail needed to reproduce the image at its original resolution. Another noticeable feature is that the resulting transformed matrix is of the same size as the original image and so there has been no compression as yet. But visually there appears to be more and more areas of blackness as the transform iterates, which translates to better entropy encoder performance. Intuitively this suggests higher potential for compression as the transform iterates, which supports the theory that as the wavelet transform iterates, so the data becomes more decorrelated yielding better compression performance.



a) Original

b) Transform Matrix with 1 Iteration

c) Transform Matrix 2 Iterations

d) Transform Matrix with 3 Iterations

Note: The detail coefficients have been multiplied by a factor of 2 in order to more clearly show these areas
The lighter lines are added to help differentiate between the various subbands

Figure 2.17: A Wavelet Transform of 'Man' with 3 iterations

2.6.5 The Advantages of the Wavelet Transform

The first and major advantage of the transform is that it operates on the entire image rather than dividing it into several blocks. Thus when the image has been highly compressed and becomes degraded, there are no blocky artefacts as associated with DCT-based schemes. In fact, when degradation does occur it is often a more graceful form of degradation, where the image becomes 'smoothed' over as detail components are suppressed. This is often more pleasing to the human eye than the blocky characteristic of DCT based degradation [Bhas97]. However although the transform provides better visual results than the DCT-based scheme its fast implementation is still slower than that of the fast DCT implementations, so the classic processing power versus performance trade-off still exists. This disadvantage has, in recent times, become negligible with the increasing speed and availability of digital signal processors.

2.6.6 Wavelet Based Compression Schemes

The wavelet transform in video compression has yielded modified approaches to quantising and compressing the transform coefficients. Using a useful property of the wavelet transform researchers have developed compression schemes, unique to the transform, that provide very favourable results [Said96][Shap93]. This section will start by introducing the properties of the wavelet transform used to achieve the high compression ratios before highlighting the techniques used.

For the discussion some clarification on notation is required. As the transform iterates, it is effectively representing the image in the next coarser scale. In terms of this document, the highest scale represents the sub-bands in the final iteration of the transform whereas the lowest scale is the original image to be transformed. Figure 2.18 shows the notation used to refer to specific sub-bands in different scales. In this figure HG2 is the scale immediately higher than HG1.

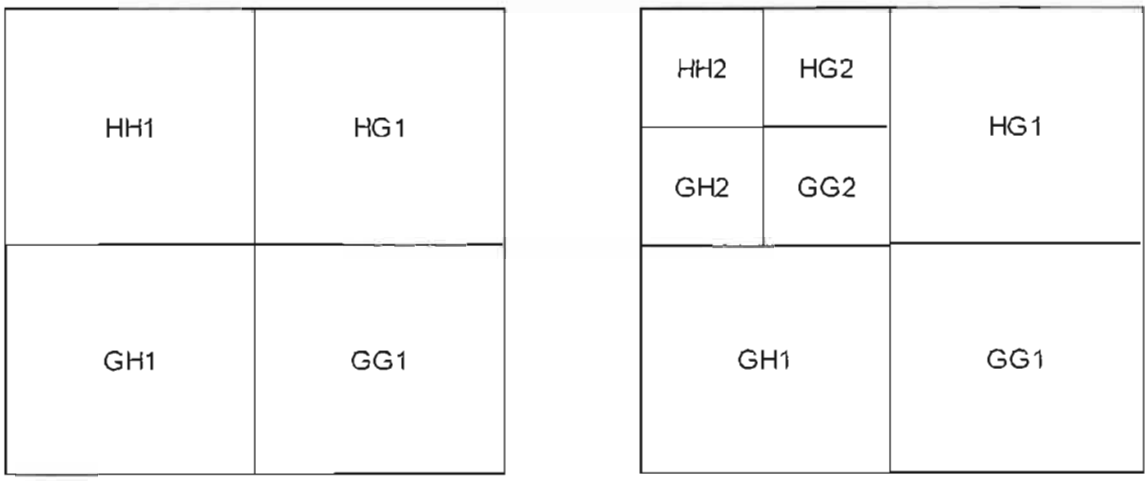


Figure 2.18: Notation for Various Subbands

An important property of the transform, that has been alluded to, but never mentioned is the hierarchical nature of the wavelet transform. Due to the iterative nature of the transform, coefficients in the higher scales are dependant on coefficients in the lower scales of similar spatial orientation [Shap93]. The dyadic down-sampling in the transform means that a coefficient in one scale is dependant on the 4 coefficients in the scale directly below it with similar spatial coordinates, as illustrated in Figure 2.19.

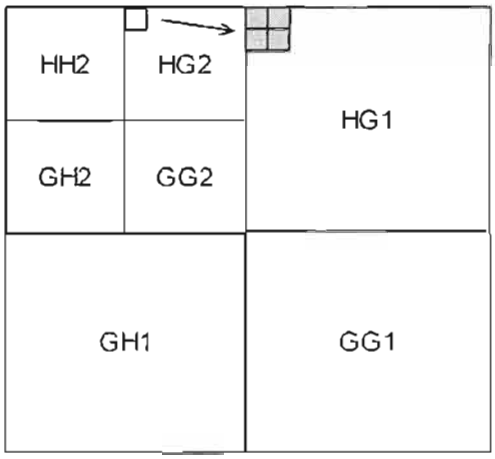


Figure 2.19: Illustration of Hierarchical Nature of Wavelet Transform

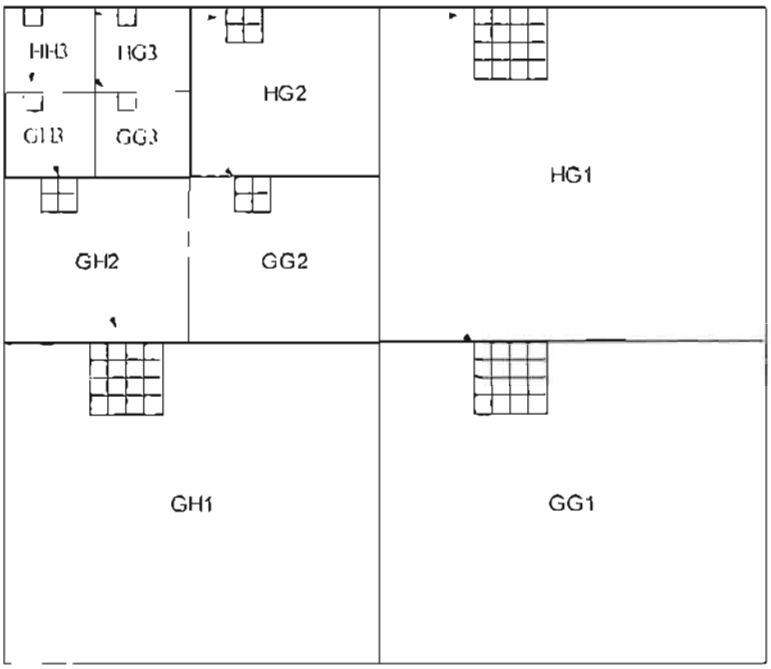


Figure 2.20: Example showing the Tree-like Structure within the Transform

Other terms that require clarification are the *tree-like* structure of the transform and the *parent-child* dependencies. In Figure 2.20 a three level wavelet transform is shown with the hierarchical coefficients highlighted and linked to define a tree of related coefficients, with the coefficient in HH3 being the root of the tree. The coefficient in HG3 is the parent to the coefficients in HG2 whose children are the coefficients in HG1.

Embedded Zerotree Wavelet (EZW)

The hierarchical nature of the wavelet transform immediately lends itself to compression. Consider a coefficient in the HH3 band that is deemed insignificant by a compression algorithm, then, since it is dependent on all the children coefficients of lower scales, there is a good chance that the children coefficients are also insignificant. An efficient compression algorithm could use this information to represent an entire tree of insignificant coefficients with very few codewords. Shapiro [Shap93] is widely

credited as being the first to develop an algorithm using the hierarchical nature of the wavelet transform, called the Embedded Zerotree Wavelet (EZW).

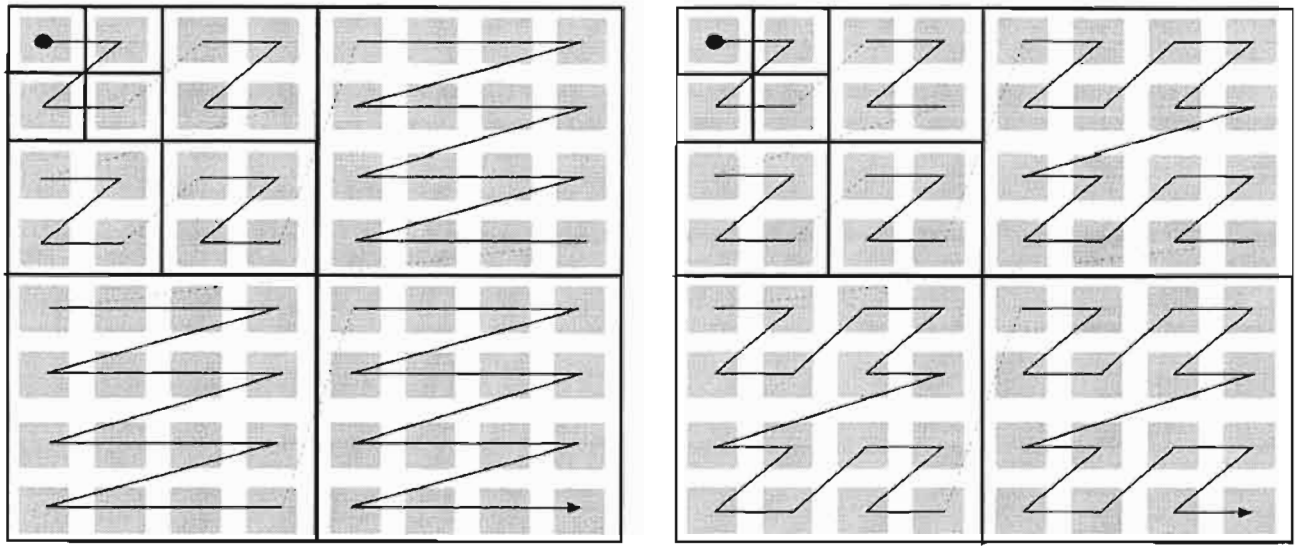
The EZW is a progressive encoder meaning that it encodes in increasing accuracy, similar to JPEG in progressive mode. The progressive (also called embedded) system is advantageous as it allows an image to be reconstructed immediately at a very coarse level with subsequent information refining the overall quality of the image. The encoding algorithm uses the fact that natural images have a low pass spectrum [Bhas97] and as such most of the image energy resides in the lower frequencies, or the HH sub-band. An obvious but still mentionable property of the wavelet transform is that large wavelet coefficients are more important than smaller ones. The compression algorithm uses the concept of *zerotrees* (trees of insignificant coefficients) to effectively encode large volumes of insignificant data.

The EZW starts with a threshold and scans the coefficients of the transformed matrix in two passes, a *dominant* and a *subordinate*. The dominant pass assigns a codeword for each coefficient. If the coefficient is larger than the threshold then a **P** (positive) is assigned; if a coefficient is smaller than the negative of the threshold then a **N** (negative) is assigned; if a coefficient is smaller than the threshold and the root of a zerotree then a **T** is assigned and finally if the coefficient is smaller than the threshold and not the root of a zerotree then a **Z** (isolated zero) is assigned. The subordinate pass then scans all **P** and **N** coded coefficients and assigns a refinement bit to each. The threshold is decreased and the process repeated. Encoding can be terminated at any time or when the threshold reaches zero.

In this scheme there are a few factors that are important for correct encoding and decoding. The first is the order that the coefficients of the transformed matrix are scanned. This is important as the encoder and decoder must follow the same scanning order or the coded data will be incorrectly decoded. In the EZW the two common scanning orders used are the raster or Morton scan as shown in Figure 2.21 [Shap93], although, any scanning order can be used as long as the encoder and decoder follow the same pattern and parent coefficients are scanned before children. The determination of the initial threshold is also important and, in the EZW, bitplane-encoding is commonly used [Shap93] and so the threshold is calculated as in (2.12) and then halved for each successive scan.

$$Threshold = 2^{\left\lfloor \log_2\left(MAX\left(\left|C(x,y)\right|\right)\right)\right\rfloor} \qquad (2.12)$$

$$C(x,y) = Matrix\ Coefficient$$



a) Raster Scan

b) Morton Scan

Figure 2.21: Common Scanning orders used by EZW. In this case the wavelet transform is 3 levels

Results published in Shapiro's paper proved that the EZW significantly outperformed the existing JPEG standard. For example, a test image 'Barbara' was compressed a rate of 20.37:1 for both JPEG and EZW and the resulting PSNR's for each was 26.99dB and 29.39dB respectively. This objective measure shows a significant improvement over JPEG. Shapiro also quotes subjective quality tests that outperform JPEG. An example of this can be seen in Figure 2.22 and Figure 2.23. If looked at carefully there is degradation in both images, however the JPEG degrades into a 'blocky' image whereas the EZW degradation is represented at the edges in the image where ringing is seen to occur. This is an example of the graceful degradation that is associated with the wavelet transform.

Figure 2.22: Visual Comparison of Compression of "Lena" at 32:1 for JPEG

Figure 2.23: Visual Comparison of 'Lena' at 32:1 for EZW

The EZW produced good compression results and outperformed JPEG quite significantly at higher compression ratios. A further feature of the EZW that makes it desirable, is the ability to choose an exact bit rate for compression whereas JPEG only allows the user to choose a desired quality factor which does not give the user exact bit rate control.

Set Partitioning In Hierarchical Trees

An algorithm that resembles the EZW but is different in many areas is Set Partitioning in Hierarchical Trees (SPIHT), first presented by Said and Pearlman [Said96]. The resemblance is found in the algorithm's similar use of the hierarchical nature of the wavelet transform as EZW, but the method in which this is done is different.

The SPIHT algorithm traverses the trees of the transformed image matrix using a predetermined threshold to order trees and pixels into three lists, outputting code bits indicate in which list a coefficient resides; these the decoder can use to decode. As in the EZW, the order that the algorithm traverses the transformed matrix must be the same for the encoder and decoder. The threshold is then reduced and the process repeated until a desired rate is achieved or the threshold reaches 0. The lists that the algorithm uses are called the List of Insignificant Pixels (LIP), the List of Insignificant Sets (LIS) and the List of Significant Pixels (LSP). The LIP contains coordinates to all the pixels scanned that have been deemed insignificant, the LIS contains the coordinate to the roots of insignificant trees in the matrix and the LSP contains the coordinates of coefficients deemed significant. The algorithm divides the trees in the transformed image into two types; the D-Type and L-Type. If a tree is D-Type then all descendants of the tree are checked for significance and if L-Type then all the descendants with exception of the immediate children are checked. Two passes are performed, similar to the EZW, for each threshold value used. The dominant pass orders the pixels into the various lists while Subordinate pass checks the LSP and encodes the next significant bit for each coefficient in the LSP, thus refining the accuracy of significant coefficients.

Results reported were shown to outperform those of the JPEG standard. Table 2.2 shows a table highlighting the increased performance of the SPIHT over JPEG for the standard test images of 'Lena' and 'Goldhill'. The results for JPEG were generated using a freely available software package [JPEG]. It is immediately evident that at all compression ratios the SPIHT is able to outperform JPEG in terms of PSNR. Visually the results show an improvement as can be seen in Figure 2.24 and Figure 2.25.

	JPEG		SPIHT	
Rate (bpp)	Lena (PSNR)(dB)	Goldhill (PSNR)(dB)	Lena (PSNR)(dB)	Goldhill (PSNR)(dB)
1	39.02	34.41	40.41	36.55
0.5	35.53	31.31	37.21	33.13
0.25	31.18	28.66	34.11	30.56
0.2	29.88	27.44	33.15	29.85

Table 2.2: Comparison of JPEG and SPIHT [Said96]

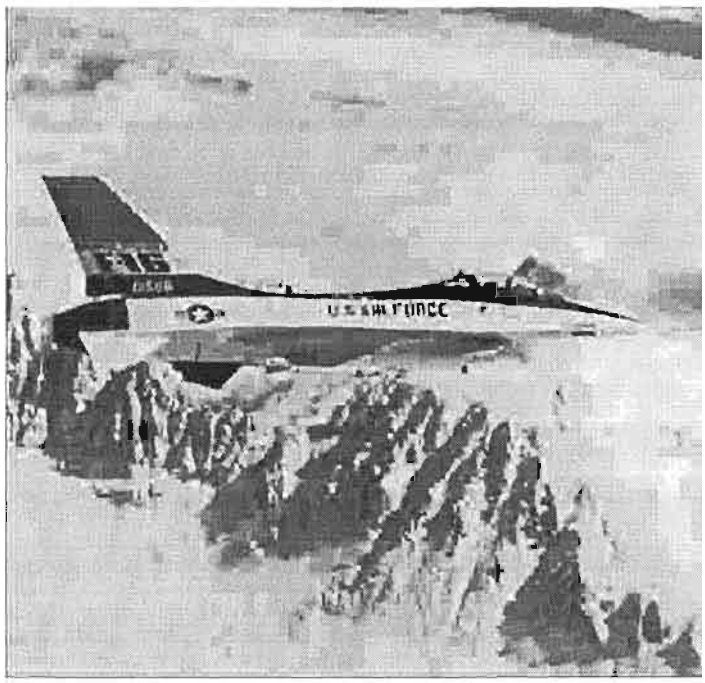


Figure 2.24: JPEG Image of 'plane' at 40:1

Figure 2.25: SPIHT Image of 'plane' at 40:1

The SPIHT algorithm has similar advantages to the EZW in that, it is an embedded scheme that degrades gracefully at higher compression ratios [Said96]. However, the scheme is more complex than JPEG and thus requires extra processing demands.

Space-Frequency Quantisation (SFQ)

The SFQ [Xion97] algorithm for wavelet image encoding is a slightly different approach to using the zerotree structure as shown in the previous schemes. SFQ uses two simple quantisation modes to exploit both the frequency and spatial compaction properties of the wavelet transform. Simply put, the algorithm first quantises trees of zeros in a similar fashion to the zerotree method of Shapiro [Shap93]; it then uses a single uniform scalar quantiser to quantise the remaining wavelet coefficients. The algorithm achieves impressive results, trying to match the minimum image distortion at a given rate, by adjusting the spatial subset of coefficients that should be thrown away, and manipulating the uniform quantiser step size that is used for quantising the remaining coefficients. This process is extremely complex as the two criteria are interdependent making computation of the optimal solution very complex and computationally intensive.

The results reported in [Xion97] show that the algorithm is able to compete with other state-of-the-art wavelet compression algorithms and in some cases perform better. Objective results shown in Table 2.3 highlight the performance of SFQ against JPEG. The results for SFQ were obtained from [Xion97] while the JPEG results were generated as in Table 2.2. In the case of 'Lena' and 'Goldhill' the algorithm is seen to significantly outperform JPEG. However the cost of using this scheme is the large computational time required to calculate the optimum quantisation technique for each image.

	JPEG		SFQ	
Rate (bpp)	Lena (PSNR)(dB)	Goldhill (PSNR)(dB)	Lena (PSNR)(dB)	Goldhill (PSNR)(dB)
1	39.02	34.41	40.52	36.70
0.5	35.53	31.31	37.36	33.37
0.25	31.18	28.66	34.33	30.71
0.2	29.88	27.44	33.32	29 86

Table 2.3: Comparison between JPEG and SFQ

Figure 2.26: JPEG Image of 'Lena' at 32:1

Figure 2.27: SFQ Image of 'Lena' at 32:1

2.7 Temporal Compression Techniques

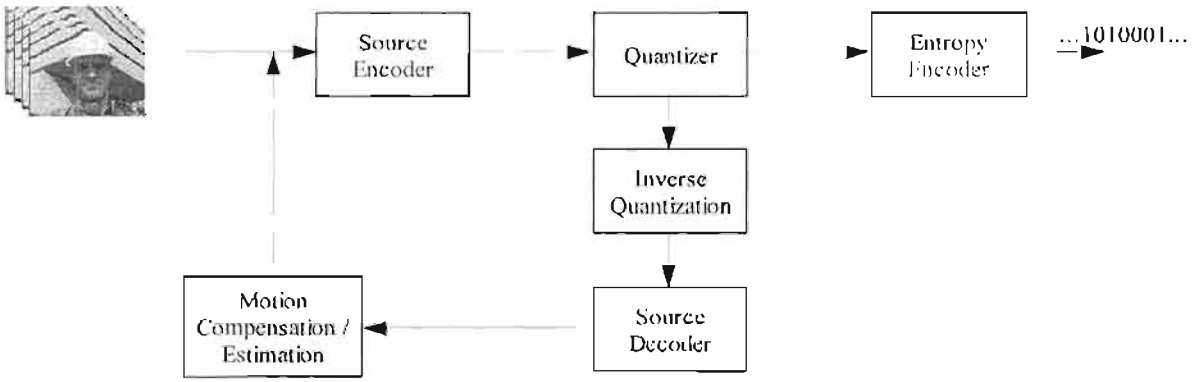


Figure 2.28 : Basic Block Diagram of a Video Encoder

Figure 2.28 shows a basic block diagram of the typical video compression encoder. It is plainly evident that the video encoder is very similar to that of the still image encoder. The only difference being the Motion Compensation/Estimation block which performs temporal compression.

As mentioned in the introduction, a video compression scheme is able to take advantage of the temporal redundancy between successive video frames. This is achieved through a technique known as motion compensation and estimation[7] [Sola97]. Motion compensation/estimation tries to predict motion between frames and use *motion vectors* to represent this motion. It then uses these motion vectors to reconstruct

[7] The process of obtaining the motion vector is called *motion estimation* and the process of using the motion vector to reduce the temporal redundancy of the video sequence is called *motion compensation*.

the current frame from the reference frame and compute a difference between the reconstructed frame and the actual current frame to obtain an *error frame*. This error frame now has a great deal of spatial correlation and thus can be highly compressed.

An Example

An example is used to clarify some conceptual issues. A generated image sequence is given in Figure 2.29, although unrealistic it serves the purpose of illustration.

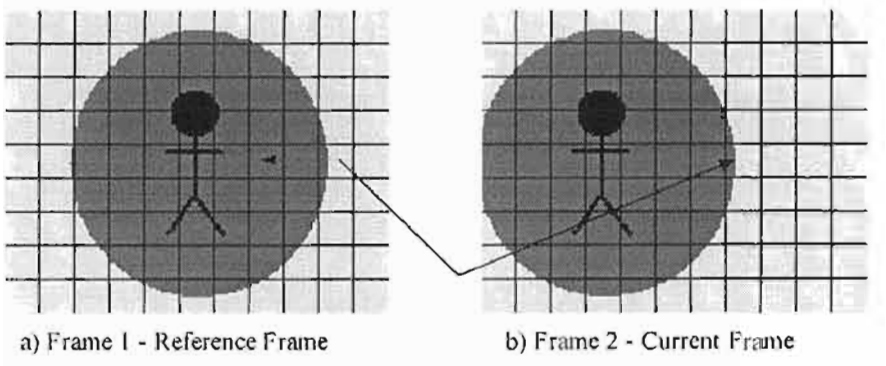


Figure 2.29 : Illustration of Motion Compensation/Estimation

A typical motion compensation/estimation scheme would divide the frames into blocks, called *macro-blocks*, and find displacements of the blocks rather than single pixels. This simplification is valid because motion in a video sequence affects groups of pixels and not just single pixels. Further, performing motion compensation on a pixel-by-pixel basis, while yielding the best motion estimate, is too computationally intensive. The size of the macro-block is an important issue, as too large a macro-block and the motion estimation does not remove the redundancy effectively and too small a macro-block and the computational time becomes considerable while the number of motion vectors needed to represent the motion becomes too large for worthwhile compression. In practice the macro-blocks are generally of size 16×16 pixels, because it suits DCT based video compression standards and for a better motion estimate 8×8 block sizes are used [Syme01].

Once the frames have been divided into blocks, a search begins which tries to match a block in the current frame with one in the reference frame. The commonly used matching criterion is either the Minimum Mean Square Error (2.3) or the Mean Absolute Error (2.4). A matching block has the least MMSE or MAE and once a match is found, the motion vectors are then encoded for each block. Figure 2.29 shows an example of a matching block, the dotted line representing the motion of the block on the reference frame and the solid line indicating the matching block across the frames. The search algorithm requires a great deal of computation and thus a fast algorithm that finds the best block match is required. In practice there are many search algorithms used that provide varying motion estimation accuracy at varying computational costs. The scheme implemented in this thesis does not use temporal compression, so an in-depth discussion of each search algorithm is irrelevant, however for the sake of completeness Appendix B highlights some of the more popular search techniques used.

$$MMSE(i,j) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left(C(x+k,y+l) - R(x+i+k,y+j+l)\right)^2 \qquad (2.3)$$

$$MAE(i,j) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1}\left|C(x+k,y+l) - R(x+i+k,y+j+l)\right| \tag{2.4}$$

The example given describes motion prediction using only two frames. Frames coded in such a manner are often referred to as *P-frames* when describing a video compression technique. Further compression is achieved by not only using the previous frame for prediction but also the future frame. Such bi-directional prediction allows for an even more highly correlated difference image at the expense of more computational time. In video compression terms a frame encoded using bi-directional prediction is often called a *B-frame*. Many of the current video compression schemes use a mixture of predictive and bi-directional predictive encoding to achieve good compression results.

The advantage of the motion compensation/estimation is a very efficient representation of the video stream. However the prediction scheme means that video editing becomes a very computationally intensive task, as the editing software needs to look at past and future frames to construct the current frame. Furthermore this dependence on other frames makes random access to various frames a near impossible task [Hoan01]. A further and major drawback of the system is the blocky artefacts (similar to those in DCT still image compression) that appear in fast motion sequences, translating to a displeasing video sequence. Techniques such as overlapped block motion estimation [Wata91], that apply smoothing filters over the error frame have proven quite successful in reducing the blocky artefacts. Other solutions propose a slightly more complex scheme of dividing the video stream into arbitrarily shaped regions rather than blocks [Dang95]. The choice of how to split into regions is based upon which regions are in motion. Results shown in this field have proven to reduce the blockiness of motion at the expense of greater processing power, as now the image requires segmentation into useful moving objects and extra data is required to specify the various, arbitrarily shaped regions.

2.8 Compression Standards

This section introduces some of the existing standards, giving the major features of each standard and which basic compression techniques each uses. Every aspect of each standard is not covered and references are quoted to guide an interested reader.

2.8.1 Joint Pictures Expert Group (JPEG)

In 1991 a joint ISO/CCITT committee, called the Joint Pictures Expert Group, officially published the first continuous tone image compression standard, now commonly called JPEG [Wall91]. The goal was to create a generic standard to cover all aspects of compression while also being universally used and accepted. The standard is a fast implementation allowing compression and decompression to be achieved without an appreciable delay, thus being useful for real-time applications. Some of the requirements of the standard were that it should be near to the state-of-the-art in compression technology, while allowing a large degree of adjustability and scalability. The codec should be robust meaning suitability for any image size and colour space and content. Implementation of the scheme should be efficient allowing for software implementations to be easily developed. Part of the scalability option means that the JPEG standard has been divided into 4 modes, Sequential, Progressive, Lossless and hierarchical.

Sequential Mode

This is the most common form of operation. The JPEG encoder first decides whether to perform a colour space conversion and then performs an 8×8 DCT as described earlier. The resulting DCT coefficients are then quantised using an 8×8 quantisation table according to a user defined rate metric. The quantised coefficients are then ordered in increasing frequency as described in Figure 2.10. This ordering maximises the chance of successive zero coefficients being grouped, as the higher frequencies of the image will tend to have zero coefficients. The final step is to entropy encode the resulting data using a run-length encoder followed by either a Huffman or arithmetic encoder.

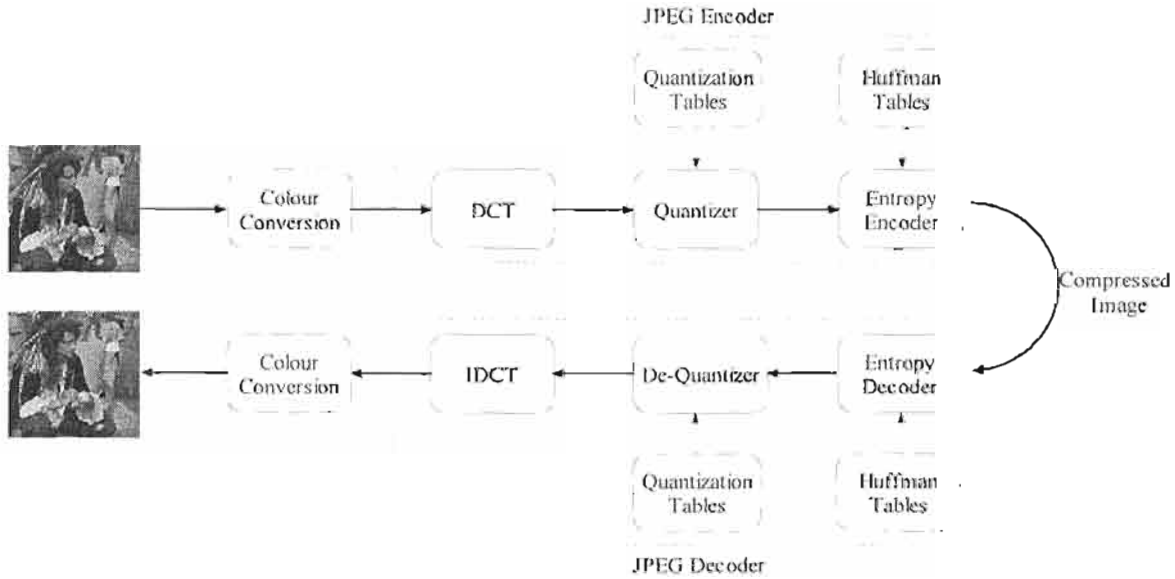


Figure 2.30: Block Diagram of JPEG Encoder/Decoder

Progressive Mode

Progressive mode allows the user to transmit a picture over a number of scans thus allowing for accessible low bit rate transmission. The first scan yields the entire image at low resolution and each successive scan increases the resolution of the image. Effectively this is an sorting of coefficients in order of importance. For example the first scan would encode all the DC coefficients for all macro blocks first, as they contribute most to the image energy, and then encode the AC coefficients in order of importance. Thus when decoding, every DC coefficient for all macro blocks would be decoded first and thus, a low resolution image would immediately be visible and with each subsequent scan the resolution would increase.

Lossless Mode

In this mode the JPEG codec does not use the DCT but rather a predictive technique that determines a pixel value based on surrounding pixel values. It then entropy encodes the result to give a compressed output.

Hierarchical Mode

The Hierarchical mode allows a user to decode a low-resolution version of the picture without first decoding to the maximum resolution. The drawback of this system is the need for extra picture buffering and the increased implementation complexity, which translates to a slower runtime.

2.8.2 MJPEG

In terms of video compression, the initial success of the JPEG system yielded a video format called Motion-JPEG (MJPEG)[Bhas97]. Although this scheme is not a standard it has become a *de facto* standard as many vendors have developed an MJPEG codec [Main]. MJPEG encodes every video frame in a sequence as a single JPEG image. The advantage of the MJPEG is fast execution time, as it does not implement any temporal compression. However the lack of temporal compression means that compression ratios are generally modest compared to schemes that do use temporal compression. The fact that every frame is treated independently means that any frame in the sequence can be randomly and quickly accessed. This feature makes MJPEG a very useful tool for video editing purposes.

2.8.3 JPEG2000

The new JPEG2000 standard [Ogun00][Skod00] for still picture compression is set to replace the JPEG standard and is in the final stages of standardisation. JPEG2000 uses wavelet-based techniques to provide better compression results than its JPEG predecessor. The codec is based on the DWT, scalar quantising, context-modelling, arithmetic encoding and post-compression rate allocation. The codec caters for lossless and lossy modes by allowing various different wavelet filters configurations to be used. The boundaries of the image are periodically extended to allow for efficient wavelet boundary handling. JPEG2000 aims to provide the same robustness and scalability that JPEG provides but also has added features like random access to areas in a picture and extra error-resilience. An added feature that is used in this video codec is the region of interest, which allows certain regions in the image to be encoded at higher quality than other regions. In December 2000 JPEG2000 Part 1 was standardized by the ISO [JPEG2000], which specifies the image encoding system, and soon after 'Image Power' [Image] announced its JPEG2000 software implementation. Currently there are 5 other parts of the standard, which, to the knowledge of the author, have yet to be standardised. These include extensions to the current core, motion JPEG2000, conformance, reference software (currently only Java and C implementations are scoped) and a compound image format for applications such as fax.

2.8.4 MPEG-1,2,4

The Motion Pictures Expert Group (MPEG) has defined a series of video compression standards, each designed to meet separate video bit rate requirements. Since these all have a direct bearing on the design of a video compression scheme these standards need to be outlined and discussed.

MPEG-1

MPEG-1 [Gall91] is a layered, DCT-based video compression standard that results in VHS quality compressed video at a target bit rate of about 1.5Mbs. It supports both PAL and NTSC formats and their respective SIF offshoots. Colour video is converted to YC_RC_B space and downsampled at 4:2:0. The MPEG-1 stream consists of three different types of frames each employing a slightly different encoding

to decrease compressed output size. The Inter-Frame, or I-frame, is encoded as a separate image as in lossy JPEG. The predictive coded, or P-frame, is a block-based motion compensated frame between the current frame and the previous frame. The Bi-directional predictive encoding frame, or B-frame, is the result of motion compensation of the current frame with the previous and future frame to get a more compact representation. MPEG-1 video defines a group of pictures (GOP) scheme that contains one I frame, a few P frames and lots of B frames. The I frame is necessary for error resilience as, since P and B frames are dependant on one another, any error occurring in these frames will propagate through the video. The I frame, however is independent and therefore stops any error propagation. The independence of the I frame from other frames also makes access to various parts of the video possible but true random access to every frame requires a large buffer to store the past and future frames in order to construct the current frame. The size of the GOP is very important as too large a GOP and the error resilience of the video drops and too small a GOP and the compression ratio increases. MPEG-1 systems use a GOP size of 15 as this provides both acceptable compression and good error resilience as an error can only propagate for 16 frames which is less than a second of PAL video [Bhas97].

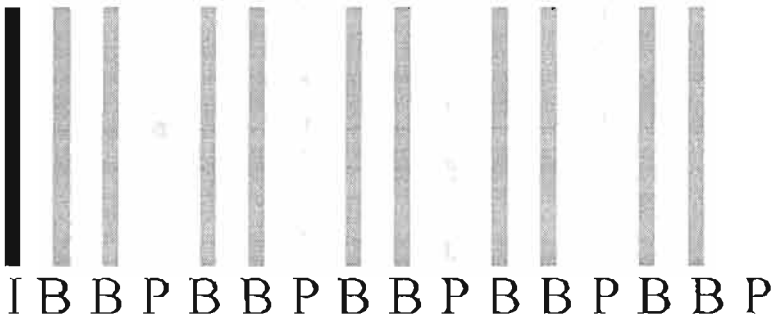


Figure 2.31: MPEG-1 Group of Pictures

MPEG-1 has been designed for bit rates of 1.5Mbs and CD-ROM applications and generally performs well under these conditions [Sola97]. However the codec is not designed for low bit rate applications and as such suffers severely from 'blocky' artefacts when the bit rate is substantially reduced.

MPEG-2

MPEG-1's limitation of the bit rate means that it is unsuitable for broadcast applications; MPEG-2 was developed for high-end video applications and filled the void left by MPEG-1 [Okub95]. The MPEG-2 codec is very similar to MPEG-1 but caters for extra scalability to include features necessary for broadcast applications. The codec is aimed at bit rates of 4Mbps up to 100Mbps and supports resolutions of 352×288 up to 1920×1152. As in MPEG-1, the MPEG-2 treats colour in the YC_RC_B domain, however, it permits various types of colour down-sampling such as the 4:2:0 or 4:2:2 format. It also uses optional B frames in order to increase quality at the expense of compression. The standard uses a more accurate form of motion compensation, which means that the resulting reconstructed motion frame is a closer approximation to the actual frame and thus the error frame is more sparse, allowing for better compression results. MPEG-2 broadcast TV is already in use in many countries as digital satellite television has started to gain prominence, [UEC] but is unsuitable for low bit rate applications [Syme01].

MPEG-4

MPEG's answer to low bit rate video transmission is found in the newly released and constantly updated MPEG-4 [Koen99][Wen99][MPEG4]. The codec is designed to allow maximum interactivity between multimedia applications and users and also bring the video to low bit rate channels, namely the Internet. The standard provides for a great deal of multimedia issues as pertains to the Internet but the focus here is going to be on its description of video. There are a multitude of features in MPEG-4 and therefore, in this section, detailed examinations of the exact workings of each feature of the codec are not given; rather a basic description of relevant features is presented. The quoted references each provide a myriad of information on the codec.

For video, MPEG-4 redefines the way video is represented and thought of. Typically a video stream is thought of as a series of images or frames representing the movement of a real-scene. This video stream is treated on a frame basis and compression techniques like MPEG-2 compress the video on this frame basis. MPEG-4, however, redefines the video stream as a series of *video objects*. For example consider a scene with a news reader; there is typically one person and the background. An MPEG-4 description of the scene could be to separate the foreground person video object from the background video object. The image of the person is now called a *texture* and there also exists information describing the *shape* of the various video objects.

MPEG-4 divides the scene into layers. The first layer is the *Video Object Plane (VOP)* which is a complete spatial representation of the object at a given time period. Thus every object in the video scene is represented by a series of VOP's. Each VOP contains texture data describing the texture of the object and shape information describing the shape of the object at that time. The next plane is the *Group of Video Object Planes(GOV)* which groups a series of VOP's into a single plane, much like MPEG-2's GOP, however data is organised so that each VOP can be separately encoded. The *Video Object Layer (VOL)* caters for scalable encoding of a video sequence of VOP's or GOV's. This scaling can be done temporally or spatially and thus allows for variable encoding to be achieved within limited computational power or bandwidth. The *Video Session (VS)* is the final layer and includes all the information about the particular video sequence.

Some of the aims of the standard that apply to natural-image video compression are to provide an efficient compression of images and video and random access to all types of visual objects. This random access means that a video object from one video scene can be seamlessly placed in another video scene. As in all MPEG standards scalability is provided for, with the user being able to compress a video stream to their exact requirements.

The basic MPEG-4 encoder is shown in Figure 2.32 and highlights some aspects of the MPEG-4 codec. MPEG-4 uses the 8×8 DCT for texture encoding as in the MPEG-2 standard and also uses 16×16 block-based motion compensation if needed. However with the encoding of arbitrarily shaped objects it provides alternative algorithms such as shaped-adaptive DCT and the use of motion compensation using "sprites", similar to the sprite concept in graphics programming.

Basically MPEG-4 divides the compression of video into three categories; Very Low Bitrate Video (VLBV), High Bitrate Video (HBV) and Content based functionalities.

The VLBV layer is aimed at bit rates from 5...64kbits/s and low spatial resolution (typically QCIF) with frame rates between 5-15 fps. The VLBV provides for error robustness, low latency and low complexity

for real-time multimedia applications. It also proposes algorithms for random access of frames and fast forward and fast reverse features.

The content-based functionalities allow for separate encoding and decoding of content. For example separating VO's and allowing VO's to be encoded and decoded separately. This part of MPEG-4 is where it differs substantially from the previous standards as it now provides for increased interactivity and manipulation of VO's within a video scene.

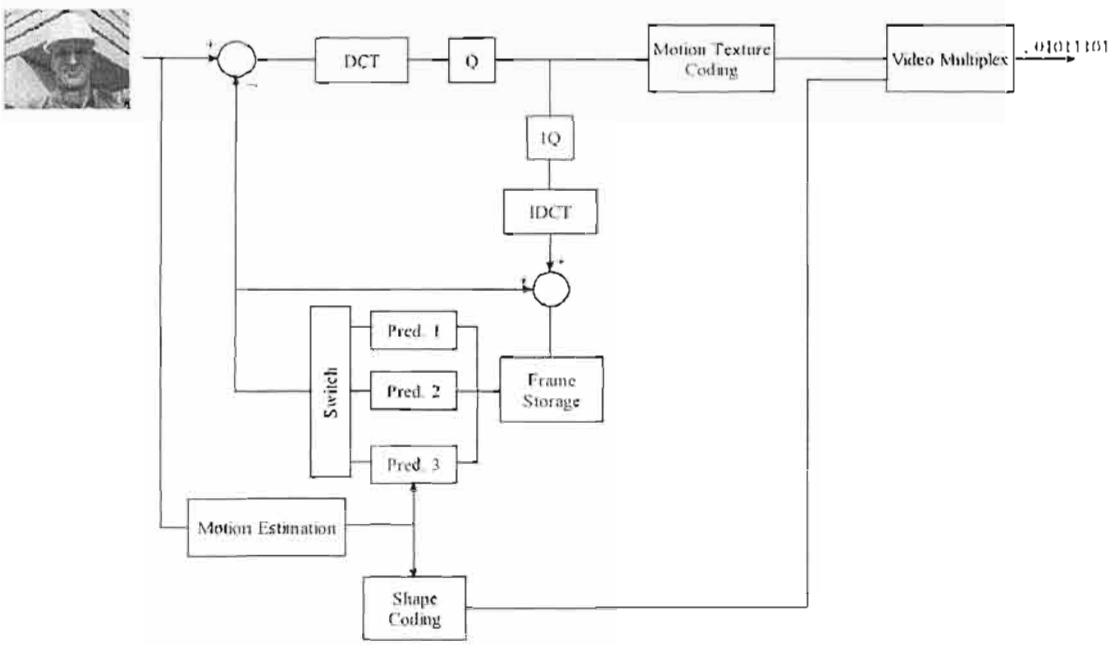


Figure 2.32: Basic Block Diagram of MPEG-4 Encoder [MPEG4]

For low bit rates an important feature of the MPEG-4 codec its scalability. MPEG-4 allows for parts of the bit stream to be decoded with reduced quality and reduced spatial and temporal resolution, thus allowing for a wide range of bit rate targets to be met. To cater for the error prone mobile environment MPEG-4 provides for error resilient features like resynchronisation, where the decoder is able to resynchronise the video stream after an error event. The codec uses techniques for error correcting and concealment of known errors, by copying of adjacent blocks.

MPEG-4 provides excellent video results and describes the future in multimedia applications where the end-user is allowed almost total control of the media downloaded from the Internet. However for the low bit rate applications the high compression ratios required still produce some 'blocky' artefacts associated with the block-based DCT and block motion compensation. A further drawback is that the arbitrary object shapes require extra encoding and introduce complex issues such as handling the boundaries of the object. The description of the video object is of greatest importance, as this requires the object to be efficiently segmented from the scene, a process that has proven to be a very complex issue [CKim00] and for which, MPEG-4 provides no formal algorithms.

2.8.5 H.261

The ITU [Turl93] recommended the H.261 standard for video conferencing over ISDN networks. It is fundamentally the same as a generic MPEG coded using the DCT, block-based motion compensation, quantization and variable length encoding. MPEG codecs support image resolutions of CIF, SIF or

higher while H.261 only supports CIF and QCIF. H.261 does not use the GOP concept of MPEG and does not allow for Bi-directional prediction. Furthermore it uses a limited motion compensation algorithm where block motion is only allowed to be a maximum of 15 pixels away from the reference pixel. This allows a faster computational time but the accuracy of the estimation suffers. The codec is sufficient for ISDN networks but suffers at bit rates below 64kbps, which results in a blocky video signal.

2.8.6 H.263

The ITU-T H.263 [H.263] and its extension H.263' is a improved version of H.261 and is designed for videoconferencing over the telephone system where bit rates are typically <64kbps. This codec basically unlocks modes not supported by H.261, supporting video resolutions of sub-QCIF (128×96), QCIF, CIF, 4CIF (704×576) and 16CIF(1408×1152) with the 4:2:0 YC_RC_B colour space. It also allows for unrestricted motion compensation, which can give a better motion estimate than the restricted mode while increasing complexity. Another feature is the use of advanced motion compensation techniques, where instead of using the 16×16 macroblock it is divided into 4 8×8 macroblocks with extra motion vectors describing them. This increases the number of bits needed to describe the motion while yielding a better motion estimate to allow for higher compression. H.263 takes advantage of the extra compression achieved by bi-directional prediction by introducing a concept of PB-frames, which, as the name suggests, is a P frame and a B frame that are amalgamated to make a single frame. Upon decoding the single frame is split into two and the B frame is decoded first followed by the P frame. Texas Instruments reports a real-time hardware implemented H.263 encoder using a TMS320C6201 DSP (1600 MIPS) [Miya00]. This encoder meets the minimal requirements of H.263 with none of the annexes of the ITU-T specification implemented. The system requires a video capture card to capture the frames, which are then presented to the DSP for encoding. The DSP then presents a coded output that is decodable by any H.263 decoder.

The H.263$^+$ extension allows for SNR, spatial and temporal scalability. It tries to reduce the 'blockiness' affect of the DCT and block motion compensation by using filters to smooth over the edges of the blocks to make a continuous image and maintains a high frame-rate in sequences with high motion by encoding a low-resolution update of the higher resolution image while keeping stationary areas at high resolution. Extra error resilience is added and a better packetisation scheme is used to allow for better performance on error-prone communications channels. These improvements create a better performing codec at the expense of increased processing power.

2.9 *Current Video Research*

2.9.1 A Zero Tree Wavelet Video Encoder

Martucci et al. [Mart97] proposed a zerotree based video encoder in 1997 that was designed for low bit rate systems. The proposed encoder incorporated the EZW and proposed an improved version called zerotree entropy encoding (ZTE). The ZTE sacrifices the embedded feature of the EZW to allow for more flexibility, adaptability and improved encoding efficiency. The algorithm quantises transformed data, explicitly allowing for optimised quantisation according to a specific scene. The scanning of zerotree coefficients is done in a manner that exploits the connections between coefficients and what they mean in a frame, allowing for objects in a scene to be treated differently. These modifications are done in

order to conform to the specifications put forward by MPEG-4. Block motion estimation and advanced overlapped, block motion compensation is used to take advantage of temporal redundancy in a video scene. The affect of the overlapped block motion compensation is to reduce 'blocky' artefacts in low bit rate scenes.

Figure 2.33 shows some results of the wavelet video encoder with the EZW and ZTE scheme against those of the MPEG-4 VM. Figure 2.33a) shows still image results for the video sequence 'Akiyo'. It is evident that the objective PSNR results for the I-frames outperform those of MPEG-4 VM with the video sequences reporting very similar PSNR readings. However, the authors claim improved subjective image quality due to the lack of 'blockiness'.

Sequence	Bits (kb)	Y C	MPEG-4 VM (PSNR)	EZW (PSNR)	ZTE (PSNR)
Akiyo	14	Y C	33.06 36.31	33.77 35.71	34.62 36.19
Akiyo	28	Y C	38.42 40.81	40.18 39.50	40.18 40.81

a) I-frame coding PSNR Results

Sequence	Rate (kbps)	Y C	MPEG-4 VM (PSNR)	ZTE (PSNR)
Akiyo @ 5 frames/sec	10	Y C	34.61 39.48	34.61 39.86
Akiyo @10 frames/sec	24	Y C	37.46 42.15	36.64 44.02

b) Entire Sequences PSNR Results

Figure 2.33: Results of the Zerotree Wavelet Encoder versus MPEG-4 VM. Results from [Mart97]

2.9.2 Partitioning, Aggregation and Conditional Coding (PACC)

The PACC algorithm published in 1999 [Marp99] presented a video compression algorithm that outperformed both the MPEG-4 VM (Version 5.1) and the ZTE based video encoder. In this algorithm, motion is treated as with H.263 but an overlapped motion compensation scheme is applied, which reduces the false high frequency components present at block boundaries of the predictive frame, and thus is more suited to the wavelets. The algorithm divides source data into separate sets, by separating the produced wavelet sub-bands, in an attempt to decrease the entropy of the overall signal. The partitioning is continued by dividing the source data into three maps; a *significance map*, where all significant pixels appear, a *magnitude map* where the magnitude of the significant pixels appear and a *sign map* holding relevant sign information. The significance map is then zerotree encoded using a similar zerotree scheme as in [Shap93]. However it differs from the Shapiro scheme by confining the zerotrees to the lower sub-bands as, the experimental evidence of the authors suggests that traversing the trees further does not appreciably increase quality. The algorithm then uses conditional probabilities where the conditioning "context" is obtained from neighbouring pixels. This conditioning allows a more compact representation of the various maps. The final step is to arithmetically encode the resulting binary data along with the motion vectors from the motion compensation.

Results reported (Table 2.4) are very impressive with PACC outperforming the MPEG-4 VM in all the sequences tested and in some cases by more than 1dB. Visually with the use of the Wavelet Transform and overlapped motion compensation, image degradation is more pleasing to the eye.

Sequence	Rate (kbps)	Y/C	MPEG-4 VM	PACC
Akiyo	20.5	Y	37.81	39.75
		C	40.65	41.76
	31.5	Y	41.49	43.72
		C	43.58	45.34
Hall Monitor	20.9	Y	35.09	36.55
		C	39.78	40.23
	28.5	Y	37.82	39.53
		C	41.16	42.14
Foreman	19.3	Y	33.95	35.16
		C	40.56	40.86
	28.7	Y	37.11	38.57
		C	42.08	43.57

Table 2.4: Results of PACC versus MPEG-4 VM [Marp97]

2.9.3 Texas Instruments Wavelet Coder

Texas Instruments published an application note describing the implementation of a real-time video encoder using four TMS320C40s [Farv97]. The objective was to design a video compression scheme that could capture and encode QCIF video at bit rates of 64–384 kbps, and then decode and display the result on a monitor in real-time. Encoding was done on two DSP's, by performing the DWT using FIR filters and then compressing the transformed coefficients using quantisation and Huffman encoding. The decoding process was performed on another pair of DSP's before the output video was displayed.

Although the simplistic compression techniques used resulted in modest results, the codec was the first published implementation of a wavelet based video codec on DSP's [Farv97] and proved that a fast hardware implementation of the DWT was achievable. Motion estimation and compensation was not implemented, as the increased processing power associated with this technique would result in the codec not being able to compress and decompress in real-time. Although the system was scalable in both spatial and temporal resolutions, as the spatial resolution increased so the time to encode and decode increased and the system lost the ability of real-time compression. The authors proposed using a faster DSP for such systems or increased optimisation of the DSP code. Other areas of improvement proposed by the authors, were improved quantisation and encoding algorithms to achieve better compression and motion compensation schemes are proposed for improvement on the scheme.

2.9.4 3-D Video Compression

A relatively new approach to video compression is that of 3-D video processing, by applying 3-D transforms on the video signal. This form of video compression has received some attention over recent years and some current algorithms are highlighted in this section.

3-D Wavelet

The 3-D wavelet transform is an extension of the 2-D version and is applied by selecting a group of frames, usually 16 for computational ease, and performing the 2-D transform spatially on each frame followed by a 1-D transform temporally as is seen in Figure 2.34. Since the temporal direction usually contains a great deal of redundancy the wavelet transform in this direction is able to represent this data very sparsely and thus the potential for compression is high.

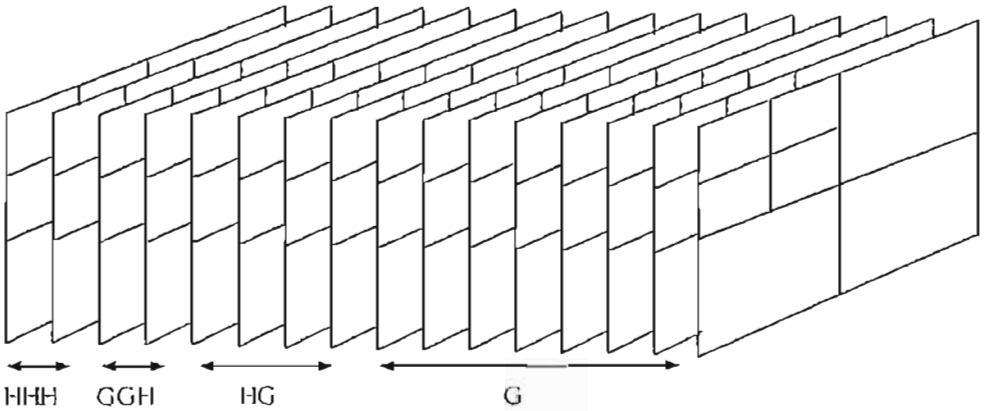


Figure 2.34: 3-D Wavelet Transform on a Group of Frames

3-D SPIHT

In December 2000 Kim et al. [CKim00] published a paper detailing a low bit rate video compression scheme using 3-D SPIHT. Conceptually the 3-D version of the SPIHT is a simple extension of the 2-D version. In the paper Kim describes the SPIHT as it applies to 3-D video compression and gives some results compared to H.263. The implemented codec applies the 3-D wavelet transform followed by the 3-D SPIHT to get a coded binary output stream that is then decoded by performing the inverse of encoding. An added option to the codec is the inclusion of motion compensation. The motion compensation further increases temporal redundancy and allows the 3-D wavelet to even more sparsely represent the group of frames.

Results reported are impressive in comparison to H.263 as it visually outperforms and objectively outperforms the H.263 codec. Figure 2.35 shows some quantitative results, which highlight the effectiveness of the codec. These results were obtained for the QCIF resolution 'Carphone' video sequence at 30kbps and 10fps. An immediate feature to notice is the degradation difference between the two sequences. In Figure 2.35b) the degradation is 'blocky' in nature which is unpleasant to the eye, whereas Figure 2.35c) shows a more acceptable form of image degradation. Another noticeable feature is the 'blocky' artefacts present in the MC 3D-SPIHT Figure 2.35d) result. This is due to the block-based motion compensation which when highly compressed results in the artefacts shown.

However the drawback to such a scheme is the implementation becomes very time consuming. The paper reports an encode time for a GOP is 28.48 seconds with the 3-D wavelet consuming 56.95% of that time. This amount of processing time makes the 3-D compression schemes unsuited for a real-time DSP application even though the reported results provide very good quality at low bit rates.

a) Top-Left, Original Frame
b) Top-Right, H.263 Decoded Frame
c) Bottom-Left, 3D-SPIHT Decoded Frame
d) Bottom-Right, MC 3D-SPIHT Decoded Frame

Figure 2.35: Results as Reported by Kim et al [Kim00] of 3D-SPIHT versus H.263. The video is the standard 'Carphone' sequence at 30kbs and a frame rate of 10fps.

2.9.5 Motion JPEG2000

With JPEG2000 recently being standardised the video extension of the standard is the next logical step. A Motion JPEG2000 codec would seek to replace the existing MJPEG for applications including video streaming, digital video editing and digital cameras, as JPEG2000 is to replace JPEG. In this vein, Image Power, the same company that released the first software JPEG2000 codec, announced on the 27 of June 2001 the first motion JPEG2000 software implementation to be released later in the year [Image]. This is to coincide with the expected announcement of the standardisation of motion JPEG2000 in December of 2001.

2.10 Summary

This chapter introduced some important video compression theory needed for better understanding of the dissertation. In order to complete the picture, Figure 2.36 shows a taxonomy of the techniques covered. Compression techniques presented have highlighted the advancements made in compression technology in recent times. Such advanced schemes as Fractal and Wavelet compression have been explained and the compression gain of each shown. Importantly, with the increase in complexity of the compression

techniques comes the requirement for more processing power. Some of the methods explained in this chapter have shown impressive compression results but are complex and computationally intensive. With the advances in technology, manufacturers are producing devices with increasing processing capabilities making the implementation of more complex compression schemes more and more viable, justifying the research into these techniques. Recent developments in video compression have shown a great deal of promise for the future of video with advanced 3-D techniques being explored to allow for better quality at lower bit rates. The increasing number of video applications that are appearing as a result of the global trend towards the digital era mean that such developments and ongoing research bodes well for the future of video.

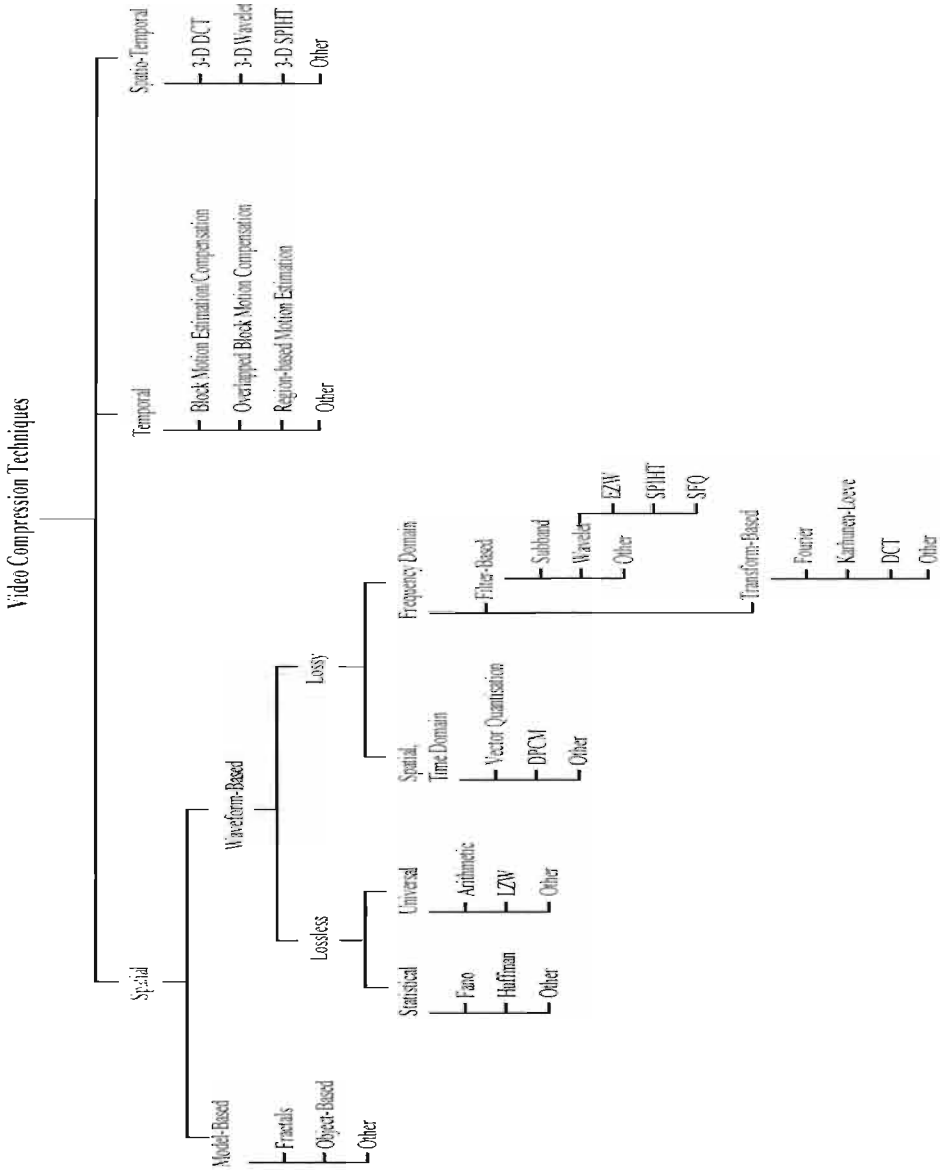


Figure 2.36: A Taxonomy of Some of the Video Compression Techniques Available

Chapter 3 The Proposed Compression Implementation

Chapter 2 gave an insight into available video compression techniques and highlighted some new schemes that have been proposed by the research community. This chapter describes the system that has been implemented for this project and the choices made in the development of the compression scheme are detailed, with justifications for each choice explained. The chapter starts by giving a systems overview to clarify where each implementation block resides in the system as a whole before continuing with the explanation of each facet of the compression scheme. Where deemed relevant some extra theoretic information is provided to explain the benefits of certain choices.

The design of a lossy video compression scheme is governed by a set of trade-offs as illustrated in Figure 3.1. Since the algorithm is lossy, some way of determining acceptable quality is desired. An objective technique that is often used is the PSNR of a signal, the higher the PSNR the closer the decompressed signal is to the original. However, as has been discussed in Chapter 2, this is often not sufficient and a subjective measure is often used to support the PSNR. Typically this is a rating of how the image looks to the eye and whether any degradation of the image is bearable, which can change from person to person and so is very difficult to standardise.

The encoding efficiency (Figure 3.1) is effectively the compression ratios achievable and is related to the other considerations. The higher the encoding efficiency the lower the quality of image, although, more complex schemes generally provide better compression results at the expense of computational power. The compression delay/complexity aspect is very important in a real-time implementation as the longer the compression delay the lower the video frame rate which is problematic when reasonable, non-jerky motion video is required in real-time. Thus a compression scheme tries to minimise the compression delay while maintaining a suitable efficiency and image quality with acceptable complexity. In this video scheme an extra application specific feature is added to try and increase the image quality and compression efficiency with modest increases in delay and complexity.

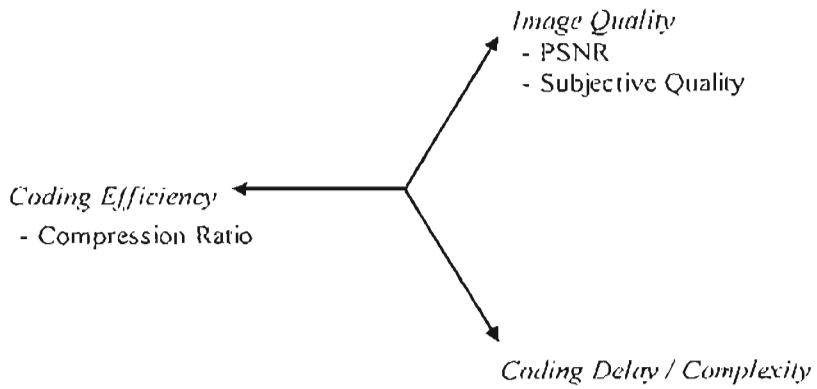


Figure 3.1: Trade-off considerations of a video compression codec [Bhas97]

3.1 System Overview

While there has been a considerable amount of work done in video compression, it has mainly focussed on the general case for compression of video and not on using characteristics of the application for which the video algorithm is destined to achieve better performance specific to that application. In such a case, certain assumptions can be made to allow for higher compression ratios and still have an acceptable

decompressed output. The compression scheme implemented in this dissertation is a purely intra-frame based scheme which uses information specific to certain applications to achieve higher compression results while maintaining acceptable performance for the application. This allows for higher compression ratios to be achieved than those by other available intra-frame schemes such as Motion-JPEG.

As Figure 2.36 shows, the choices for compression are numerous and each has its advantages and disadvantages. In this implementation it was decided to use a wavelet transform approach, as the hardware to be used would be able to cope with the computational demands of the transform. To encode the transform matrix it was decided to use the SPIHT algorithm, which showed, improved compression performance over JPEG and other algorithms of similar type. An arithmetic encoder was added to the system to provide further lossless compression before the compressed data was transmitted.

A basic block diagram of the implemented codec is given in Figure 3.2. The scheme has been implemented to accept video from a video camera, compress this video on a frame-by-frame basis and send the compressed data to the PC where it is decompressed and displayed on the PC monitor through a graphics user interface (GUI). The option of sending the compressed data through the serial link is also available in order to demonstrate operation over an actual communications link. In this case the decoding is performed on a second DSP before being displayed on a TV monitor. The GUI allows the user to change various compression parameters on-the-fly, adjusting the incoming video stream as desired. The application specific block refers to the implementation of a quality box scheme that will be elaborated on in a later section.

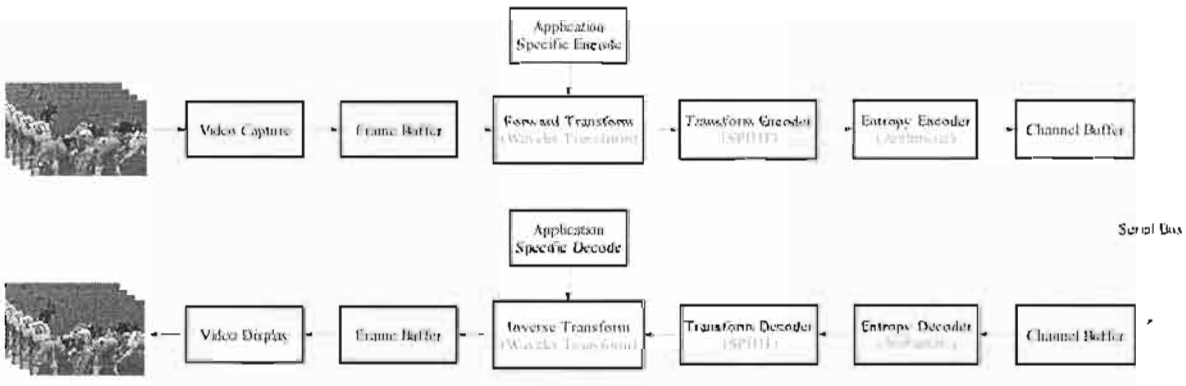


Figure 3.2: Basic Block Diagram of Video Compression Scheme

3.2 *Choices and Justification*

In any design the choices made for achieving the goal must be justified and the remainder of the chapter aims at supporting those decisions made in the design.

3.2.1 Video Capture

The proposed system is able to capture video from a video camera accepting both composite and S-video video formats allowing for both the common composite video cameras and the higher quality S-video input. The support for both these formats is governed by the capabilities of the video capture card being used and the card in this scheme is able to capture in both formats. The typical spatial resolution used for low bit rate video compression is the QCIF spatial resolution [Marp99]. In this codec various spatial resolutions are allowed, with the user being able to dynamically change resolutions between the CIF and

QCIF standard spatial resolutions. An additional resolution catered for is a non-standard ¼ QCIF [8](which from now on will be termed SQCIF).

Since the codec is a real-time, frame-based scheme, limiting the temporal resolution is easily achieved by adjusting the frame capture rate. However, the major factor governing the maximum temporal resolution achievable, is the time taken to compress and decompress a single frame. The main contributing factors to the frame compression are the transform (wavelet) and the encoding of the transform coefficients (SPIHT). Generally the smaller the spatial resolution the less time taken to encode and so the higher the possible temporal resolution. A more stringent form of control has been implemented which allows a user to define a specific frame rate, spatial resolution and data rate and the codec will calculate the best quality video achievable at this rate. For example, a user can define the communications bandwidth as 16kbits and want the video to be QCIF resolution, colour and at 5 frame/sec. The implemented system would then output the best possible quality video meeting the desired constraints. The advantage of such a scheme is the ability to quickly simulate compression performance at various possible rate metrics without having to physically set up a test environment.

3.2.2 Colour or Greyscale?

The ability to handle greyscale or colour has serious implications on the achievable compression ratios and quality. Greyscale video is able convey the required video information and uses less data to represent than does colour, making it very desirable for a low bit rate video compression scheme. The addition of colour, however, adds an extra dimension by providing not only general identification features of the video but also the differentiation of colours that suits the HVS more closely. The drawback is the increase to amount of the data needed to represent the colour information, increasing the compression required to meet the same rate constraints as greyscale.

In this scheme both greyscale and colour video are provided for, with the user being able to dynamically switch between the two. This provides for the user that wants very high compression ratios and is willing to sacrifice colour and, the user that will accept slightly less compression but with the addition of colour. Chapter 2 discusses various successful subsampling techniques used to compress colour. This video compression scheme transforms colour into the YUV domain and uses 4:2:0 subsampling to compress colour information. The 4:2:0 subsampling is a common choice for low bit rate schemes as it minimises the colour information quite substantially while maintaining acceptable colour quality. The choice of the YUV format over other colour formats is to keep in line with the PAL TV standard which uses the YUV colour space [Bhas97].

3.2.3 Choice of Transform - Wavelet

The taxonomy of Figure 2.36 shows two possible paths for a lossy frame-based compression scheme. Either Spatial or Time Domain, such as Vector Quantisation, or Frequency Domain which is effectively a list of useful transforms. From the theory presented in Chapter 2, the transform based schemes have provided much success in video compression and thus it was decided to use one of these transforms

[8] ¼ QCIF = 88×72

The choice of the transform in the implementation is governed by two main concerns; firstly the effectiveness of decorrelation and secondly the speed of implementation. The classic DCT algorithm has been used in many video compression implementations [Miya00, MPEG4] as the already existing and extensive knowledgebase allows for fast and easy implementation. However, the well-known disadvantage is the 'blockiness' introduced as a result of the excessive compression needed for low bit rate video. Thus for this implementation, a solution more suited to high compression was considered, which included using algorithms to correct the 'blockiness' of the DCT or using a different transform. The use of filters, to reduce the blocky affect, is considered a sub-optimal solution to the problem as, although it minimises the blocky effect it does not totally suppress the problem and so it was decided to choose a different transform. The optimal solution for decorrelation purposes would be the KLT but this is immediately dismissed due to its intensive computational requirements and Fractals, while providing good results, are hamstrung by the excessively slow encode times [Uys00]. The wavelet transform, on the other hand, has gained much prominence in image processing since it is shown to provide good image quality at high compression ratios. A particularly impressive result is the good performance at high compression ratios, which makes the DWT particularly suited to the low bit rate application in mind. The filter implementation shows that a fast method of calculating the DWT is possible and thus the wavelet transform is seen as the optimal choice for this system.

However, the choice of using the wavelet transform is not that simple, as wavelet theory provides a basis for wavelet analysis and not the actual wavelet functions themselves. So the selection of which mother wavelet to use for the transform is still required. This choice is limited by the fact the wavelet must be relatively short (meaning as few filter taps as possible) in order for a reasonable implementation time and it must be smooth, allowing for efficient transformation of the smooth images. Both these constraints limit the choice of wavelets to the set known as biorthogonal wavelets [Anto92]. This problem is well-known and as such much research has been done to find the optimal set of wavelets for image compression. Granted, the question may be asked that surely the best wavelet can be found for each image, which is true. However this means that for each image a new wavelet must be derived, not a trivial task and also quite computationally expensive. So research has rather focussed on the best wavelet which performs well over a series of images. Work done on various biorthogonal wavelet bases has revealed that the 9-7[9] biorthogonal wavelet provides the best performance in terms of image compression [Anto92][Vill95]. Antonini et al compared three classic biorthogonal filters in a work published in 1992, which found that for purposes of image compression the 9-7 filter provided the best results and in a separate work, Villasenor et al compared a series of wavelets, the results of which are shown in Table 3.1. The testing process was to transform the desired image and use an adaptive scalar quantizer to compress images to the same compression ratio, then decompress these images and calculate the PSNR. The higher the PSNR the better the wavelet is able to decorrelate the image data. It is clear that of all the wavelet bases tested the 9-7 wavelet provides the best PSNR results. This objective analysis is valid as the point of the testing was to determine which wavelet bases provided the best image decorrelation, which is proven through the PSNR results.

[9] The notation is derived from the number of filter taps used in the wavelet (9) and scaling (7) FIR filters

		Length	Filter Coefficients	PSNR (dB)
1	H_0	9	0.852699, 0.377402, -0.110624, -0.23849, 0.037828	29.67
	G_0	7	0.788286, 0.418092, -0.040689, -0.064539	
2	H_0	13	0.767245, 0.383269, -0.068878, -0.033475, 0.047282, 0.003759, -0.008473	29.57
	G_0	11	0.832848, 0.448109, -0.069163, -0.108737, 0.006292, 0.0141182	
3	H_0	6	0.788486, 0.047699, -0.129078	29.53
	G_0	10	0.615051, 0.133389, -0.07237, 0.006989, 0.018914	
4	H_0	5	1.060660, 0.353553, -0.129078	29.13
	G_0	3	0.707107, 0.353553	
5	H_0	2	0.707107, 0.707107	29.28
	G_0	6	0.707107, 0.088388, -0.088388	
6	H_0	9	0.994369, 0.419845, -0.176777, -0.066291, 0.033145	29.13
	G_0	3	0.707107, 0.353553	
7	H_0	2	0.707107, 0.707107	27.48
	G_0	2	0.707107, 0.707107	

Table 3.1: Results of Villasenor Testing

Although the filter operation is a fast implementation of the DWT, it is not the optimal solution for implementation. Figure 3.3 shows the FIR implementation of the 9-7 wavelet transform. It is evident that to calculate one coefficient through the scaling filter, 9 multiplications and 9 additions are required, and for the wavelet filter, 7 multiplications and 7 divisions. Furthermore, down-sampling is required which involves re-sampling each coefficient of the resulting transform. Some simple arithmetic reveals that for a single iteration of the 2-D wavelet transform, for a 512×512 image, 4194304 multiplications and 4194304 additions are required. This figure excludes the additional down-sampling and border handling requirements which add to it. So even though there are fast FIR implementations in hardware, the number of calculations required for a single iteration is quite substantial.

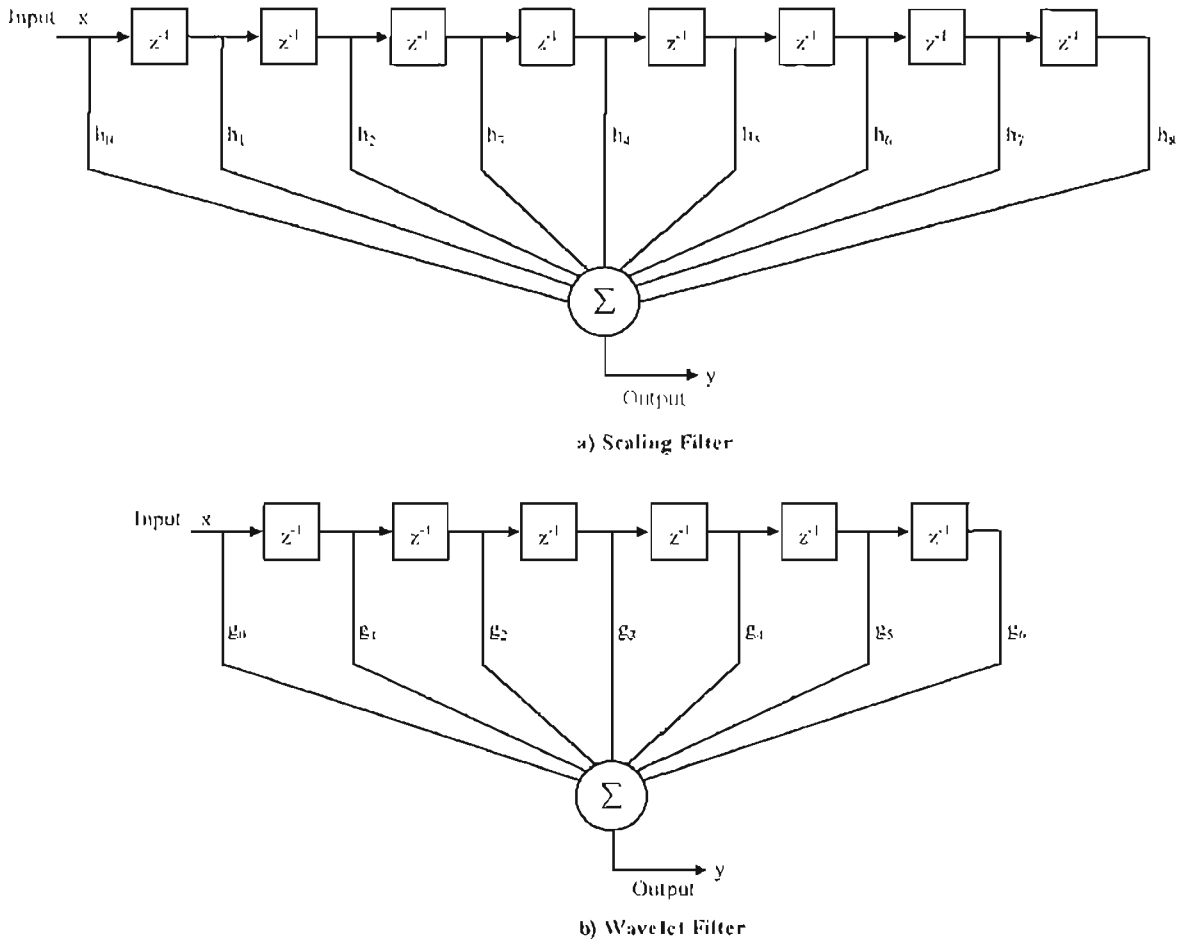


Figure 3.3: FIR Filter Implementation of Wavelet Transform

An alternative method of implementing the wavelet transform that has been newly introduced, is a technique called *lifting* [Swel98]. Lifting combines the process of down-sampling and filtering into a single step which efficiently reduces implementation time for the DWT.

3.2.4 Lifting

Figure 3.4 shows a block diagram of the process of lifting for a 1-D wavelet transform. As the figure shows the input signal is first split into odd (γ) and even (λ) components, a process known as the *lazy wavelet transform*. The lazy wavelet transform does not do much to the signal, other than separate the even and odd parts, the decorrelation is achieved by the *predict* and *update* steps which may involve several stages. The prediction step, also known as *dual lifting*, uses λ data to predict γ data. When the input is highly correlated this step is often very accurate but still can result in a slightly erroneous prediction. But since the prediction error is small the lifting scheme only retains this error information and not the entire γ subset. So the entire γ is replaced by $\gamma_{new} = \gamma - P(\lambda)$ where $P(\lambda)$ is the predicted of γ using λ. This operation, however, means that the transform has lost some important information about the signal (such as the mean value) and so an update, or *primal lifting*, step is needed where the λ set is updated with the γ_{new} subset. This means that the λ_{new} is now replaced by $\lambda + U(\gamma_{new})$ where the U is the update operation. With careful design of the prediction and updating processes the result of the lifting operation is a 1-D wavelet transform.

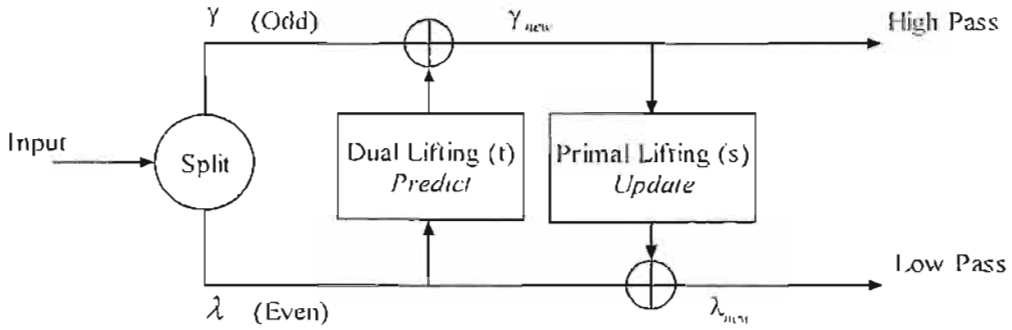


Figure 3.4: Basic Block Diagram of Lifting

The problem comes in trying to obtain the correct prediction and update algorithms required for a wavelet transform. In order to do this, the filter bank wavelet theory and lifting theory need to be unified which is achieved using polyphase representations of the filter banks. The output signal in the z-domain for the filter bank in Figure 3.3 can be represented in the z-domain as in (3.1).

$$\begin{aligned}
&\vdots \\
&y_0 = h_0 x_0 + h_1 x_{-1} z^{-1} + h_2 x_{-2} z^{-2} + \cdots + h_8 x_{-8} z^{-8} && \text{row 1} \\
&y_1 = h_0 x_1 + h_1 x_0 z^{-1} + h_2 x_{-1} z^{-2} + \cdots + h_8 x_{-7} z^{-8} && \text{row 2} \\
&y_2 = h_0 x_2 + h_1 x_1 z^{-1} + h_2 x_0 z^{-2} + \cdots + h_8 x_{-6} z^{-8} && \text{row 3} \\
&\vdots
\end{aligned} \tag{3.1}$$

The process of sub-sampling, which is done in any wavelet transform would remove the second row in (3.1) and all other odd indexed outputs. This results in only the odd indexed filter coefficients affecting odd indexed inputs and even indexed filter coefficients only affecting even inputs. Grouping this together the filter operation could be re-written as in (3.2), where the subscript e denotes even and o odd.

$$y_e(z) = h_e(z) x_e(z) + z^{-1} h_o(z) x_o(z) \tag{3.2}$$

Applying this remodelling of the FIR filter to the wavelet filter bank as seen in Figure 2.16 results in the representation seen in (3.3). The *lazy wavelet transform*, as described earlier, does nothing more than split the input into even and odd components and its polyphase representation is the unit matrix. Once the polyphase matrix has been defined the process of dual and primal lifting needs classification.

$$\begin{pmatrix}\lambda(z)\\ \gamma(z)\end{pmatrix}=\tilde{P}(z)\begin{pmatrix}x_e(z)\\ z^{-1}x_o(z)\end{pmatrix}$$

$$P(z)=\begin{pmatrix}h_e(z) & h_o(z)\\ g_e(z) & g_o(z)\end{pmatrix}=\text{ Polyphase Matrix} \tag{3.3}$$

In primal lifting a new h filter bank is created from the old h and g filters to perform the primal lifting step. (3.4) shows this primal step and (3.5) shows the polyphase representation.

$$h^{new}(z)=h(z)+s(z^2)g(z) \tag{3.4}$$

$$P^{new}(z)=\begin{pmatrix}h_e^{new}(z) & h_o^{new}(z)\\ g_e(z) & g_o(z)\end{pmatrix}=\begin{pmatrix}1 & s(z)\\ 0 & 1\end{pmatrix}P(z) \tag{3.5}$$

The dual lifting process is given as in (3.6) and (3.7) and involves constructing a new g filter from the old h and g filters. This process can be repeated to obtain a concatenation of primal and dual lifting steps to perform the wavelet transform. Mathematically, equations (3.4) and (3.5) describe the process of *lifting* the high-pass with the help of the low-pass sub-band (predict) and lifting the low-pass sub-band with the help of the high-pass sub-band (update).

$$g^{new}(z)=g(z)+t(z^2)h(z) \tag{3.6}$$

$$P^{new}(z)=\begin{pmatrix}h_e(z) & h_o(z)\\ g_e^{new}(z) & g_o^{new}(z)\end{pmatrix}=\begin{pmatrix}1 & 0\\ t(z) & 1\end{pmatrix}P(z) \tag{3.7}$$

With the primal and dual lifting steps defined the problem of factoring the wavelet FIR filter coefficients into lifting steps still remains. Daubechies and Sweldens [Daub98] proved that any polyphase matrix representing a wavelet transform can be factored into a series of unit upper and unit lower triangular 2×2 matrices and a diagonal normalisation matrix as illustrated in (3.8). What remains is the determination of $t_i(z)$ and $s_i(z)$. For primal lifting this means that the Laurent polynomials s(z), $h^{new}{}_e(z)$ and $g^{new}{}_o(z)$ have to be found such that equations in (3.9) are satisfied. A similar constraint is required for dual lifting. Note that the solutions are not unique and so there may be multiple possibilities, the only constraint being that the solution satisfies the equations.

$$P(z)=\overbrace{\begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}}^{normalisation}\prod_{i=1}^{m}\left[\underbrace{\begin{pmatrix} 1 & s_i(z) \\ 0 & 1 \end{pmatrix}}_{primal}\overbrace{\begin{pmatrix} 1 & 0 \\ t_i(z) & 1 \end{pmatrix}}^{dual}\right] \tag{3.8}$$

A and B are normalisation constants

$$\begin{aligned} h_n(z) &= s(z)h_e(z)+h_n^{new}(z) \\ g_n(z) &= s(z)g_e(z)+g_n^{new}(z) \end{aligned} \tag{3.9}$$

In terms of the 9-7 wavelet, Daubechies and Sweldens [Daub98] factor the FIR filters into lifting steps. This leads to the lifting steps as seen in (3.10)[Daub98] which translates graphically as in Figure 3.5.

$$\begin{aligned} s_l^{(0)} &= x_{2l} \\ t_l^{(0)} &= x_{2l+1} \\ t_l^{(1)} &= t_l^{(0)}+\alpha\left(s_l^{(0)}+s_{l+1}^{(0)}\right) \\ s_l^{(1)} &= s_l^{(0)}+\beta\left(t_l^{(1)}+t_{l-1}^{(1)}\right) \\ t_l^{(2)} &= t_l^{(1)}+\gamma\left(s_l^{(1)}+s_{l+1}^{(1)}\right) \\ s_l^{(2)} &= s_l^{(1)}+\delta\left(t_l^{(2)}+t_{l-1}^{(2)}\right) \\ s_l &= \zeta s_l^{(2)} \\ t_l &= \frac{t_l^{(2)}}{\zeta} \\ \alpha &\approx -1.586134342 \\ \beta &\approx -0.05298011854 \\ \gamma &\approx 0.8829110762 \\ \delta &\approx 0.4435068522 \\ \zeta &\approx 1.149604398 \end{aligned} \tag{3.10}$$

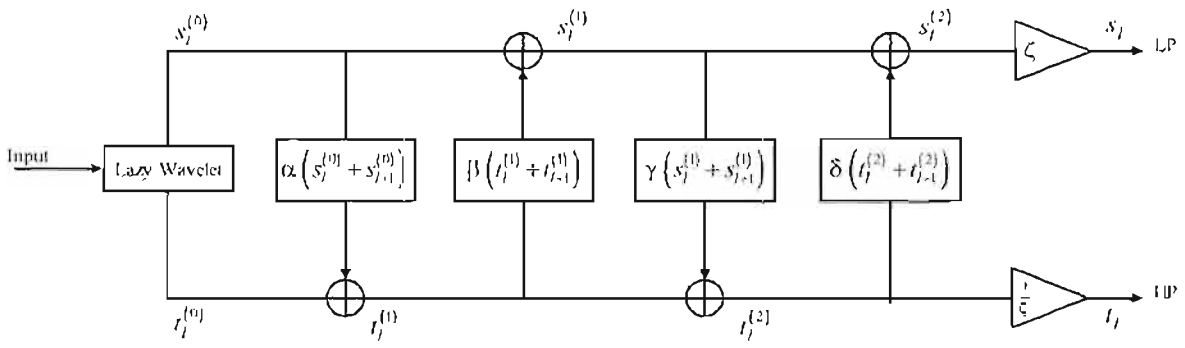


Figure 3.5: 9-7 Lifting Wavelet Transform

The advantage of using lifting over conventional FIR implementation is the increase in speed as the number of calculations required are reduced The inverse of the transform is also trivial to find as it is the forward transform applied in the reverse direction. Thus in this scheme the implementation of the 9-7 wavelet transform is done using the lifting technique.

3.2.5 Choice of Compression Scheme – SPIHT

As is common in most lossy compression schemes, once image data has been transformed some quantisation and entropy encoding is used to compress the transform coefficients. While some success has been demonstrated in the use of scalar and vector quantisation techniques followed by Huffman encoding [Wall91], since the transform being used is the wavelet transform, it makes sense to use the properties of the transform to achieve better compression. Chapter 2 highlighted some of the available and more successful compression techniques that effectively use the properties of the wavelet transform to achieve very impressive compression results. It is worth noting that although the implementation of the transform is not using FIR filters the result of the lifting transform is equivalent to that achieved by using FIR filters.

The choice of the wavelet transform over other transforms means that a suitable compression scheme designed for it must take advantage of the properties of the transform. In this implementation the governing factors were speed of implementation and performance. Table 3.2 shows a comparison of objective performance of the various tree-based schemes that have been discussed. It is clear that the SFQ provides better performance than both SPIHT and EZW, however the high computational cost of the SFQ means that it is unsuited for this application. The SPIHT algorithm provides better performance than the EZW and at similar computational cost [Said96] making it the better choice of the three. Thus for this implementation the SPIHT algorithm was chosen due to its excellent performance at comparatively modest computational cost.

	SFQ		SPIHT		EZW
Rate (bpp)	Lena (PSNR)(dB)	Goldhill (PSNR)(dB)	Lena (PSNR)(dB)	Goldhill (PSNR)(dB)	Lena (PSNR)(dB)
1	40.52	36.70	40.41	36.55	39.55
0.5	37.36	33.37	37.21	33.13	36.28
0.25	34.33	30 71	34.11	30.56	33.17

Table 3.2: Comparison Between SFQ, SPIHT and EZW. Results for SFQ obtained from [Xion97], SPIHT from [Said96] and EZW from [Shap93]

The easiest method for explanation of the SPIHT algorithm is to give it in step format as seen below. Essentially the main idea behind the SPIHT is the quantisation and ordering of wavelet coefficients in order of importance. The initial threshold is calculated as in (2.12).

1. Compute the initial threshold and initialise the lists: LIP – put all the roots of trees into the LIP; LIS – put all the trees in the LIS and assign them as D-Type; LSP – Make the LSP empty.
2. Check the significance of all coefficients in LIP:
   If Significant then output 1 followed by a sign bit and move the coefficient to the LSP
   If not significant output 0.
3. Check the significance of all trees in the LIS according to the type of tree:
   For a D-Type:
      If it is significant, output a 1 and code its children:
         If a child is significant, output 1, then a sign bit and add to LSP
         If a child is insignificant, output 0 and add it to the end of the LIS
         If the children have descendants, move the tree to the end of the LIS as type L, otherwise remove it from LIS.
      If it is insignificant, output 0.
   For a L-Type:
      If it is significant, output 1, add each of the children to the end of the LIS as a D-Type tree and remove the parent tree from the LIS.
      If it is insignificant, output 0.
4. Decrease the threshold and go to Step 2. Repeat until the compression ratio is reached or the threshold is 0.

3.2.6 Application Specific

The application specific block in Figure 3.2 refers to the algorithms employed to allow for greater compression with video quality suitable for certain applications but not for others. Each video application has its own set of unique conditions, specific to it, that can be used to tune a compression algorithm to produce better results. The issue is the choice of tuning algorithm which best suits each application.

In the case of this implementation the idea of providing better quality to certain areas of the video frame has been introduced. This means that in a video scene there exists the possibility of highlighting an area for better quality than the rest of the scene at the expense of degrading the surrounding areas. This

provides the advantage of achieving higher than normal compression ratios while only allocating suitable quality to those areas deemed important to the user.

With the wavelet transform there are a few methods that can be used to introduce the quality box idea. Extra quality in an area means that extra detail is required in that area. The wavelet transform effectively separates detail from the average of the image and so a logical method for increasing quality in a certain area is to increase that areas detail before applying the wavelet transform. In this way those coefficients in the detail regions of the transform would be abnormally higher and thus given higher priority. On encoding, these areas would then be given more encoding priority than others and thus better quality could be achieved. On decoding the affected area could be adjusted back to normal detail with the result of an apparently better quality area. To achieve this, the contrast of the area to be affected can be adjusted before compression thus creating regions of extra detail. This type of adjustment will work but is not desirable as there is no real control on the achievable quality. The contrast must be very carefully adjusted as too much adjustment results in a 'clipping' affect which means image data is lost and too little adjustment and the wavelet results are insufficient. Thus this method has not been used.

Another method is to increase the values of those wavelet coefficients that affect the area of interest. In this manner the subsequent encoding algorithm used will give these coefficients higher priority and thus code them more accurately. On decoding the coefficients could be adjusted to the correct range before applying the inverse transform. In this manner extra quality can be achieved in those areas that are adjusted. However the problem of adjusting the exact amount of quality wanted still is an issue and very difficult to accurately achieve with this method.

An extension of this idea is used in this scheme where coefficients in the affected area are set to 0. In this way the SPIHT algorithm assigns them the least priority of all and so this area contributes minimally to the image. If the area forms a zero tree then it is conceivable that this area could be represented by the single bit expressing the entire zero tree. The area of interest is then coded separately from the background image, according to a separate rate metric, and on decompression superimposed on the total video frame. The advantage here is that the area of interest is now very scalable with a user being able to define a certain quality metric for this area. Chapter 4 describes some implementation issues that concern the realisation of the area of interest.

3.2.7 Choice of Serial Communications

Since the algorithm is designed for low bit rate applications, for effective demonstration purposes it was decided to develop a simple communications protocol for transmitting the compressed video over the serial link. This allowed for testing the performance of the compression scheme in a more realistic environment, where issues of frame synchronisation and communications delay are important. However this was still not a totally realistic environment as channel issues, like bit errors, were not considered.

The serial communications link is configurable to various baud rates from 9600-115200 baud, which allows for testing over a range of low bit rates. The simple protocol is able to set up a link between encoder and decoder and begin transmitting the video packets that are then displayed on the TV monitor. Control of the decoder is achieved via the serial link and the encoder is able to tell the decoder to start and stop encoding. Although the ability exists to simulate the quality of the video over various rate metrics the addition of the serial link allows for a more realistic demonstration of operation and thus is very useful.

3.3 Summary

This chapter has introduced the work performed for this dissertation. It describes, an intra-frame video compression scheme that uses the wavelet transform and the SPIHT algorithm and which has been implemented on a DSP. The scheme is able to capture video from a camera and compress this video for subsequent decompression and display on the PC. A further feature is the optional ability to transmit the compressed video over a serial link and decompress and display the video using a second DSP on a TV monitor. An adjustable area on the video scene can be highlighted for better quality than the rest of the video scene. This can be done during the live video display, allowing for re-definable areas of quality and a higher degree of scalability. A graphical user interface has been designed for the PC whereby a user is able to adjust a multitude of compression parameters that can change the properties of the video during run-time, allowing for easy and seamless scalability.

Chapter 4 Test-bed Implementation

The major portion of this project is the implementation of the video codec using a DSP board and interfacing this with the PC. For this purpose the Philips Trimedia Multimedia Development Board is used. The board is specifically designed for development of multimedia applications and as such is perfectly suited to this application. This chapter will give a brief description of the hardware used for the implementation, before focussing on the specifics of designing the code to be implemented. The main objective is not to produce a highly optimised solution but rather to demonstrate the functionality and the advantages of such a system. However some degree of optimisation is required for effective implementation and demonstration.

4.1 The Test-Bed

PC Functions
- Control Trimedia
- Display Original
- Decode and display compressed Video
- Facilitate dynamic control
- *Act as Communications Device*

Trimedia Functions
- Capture Video
- Compress Video
- Transmit both compressed stream and original video to PC
- *Decompress Video and Display on monitor*

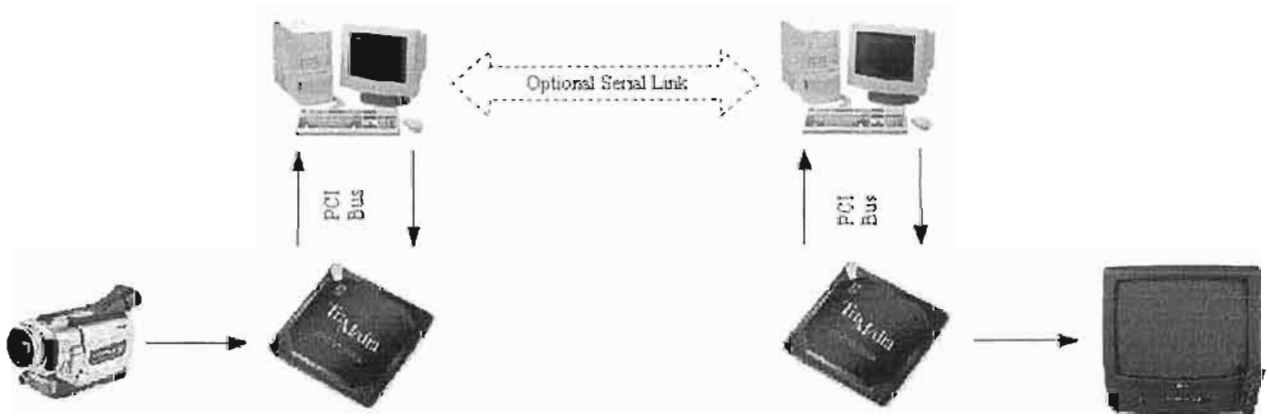


Figure 4.1: Block Diagram of Test-Bed (Italics represents optional extras)

The block diagram of the hardware used for the implemented system is given in Figure 4.1. The PC is the master controller and has the ability to activate and deactivate the Trimedia DSP (from now onwards referred to as the DSP) through the PCI bus. Therefore the PC acts as the interface between the user and the compression algorithm. The PC also acts as the output for the compression scheme by decompressing and displaying the decompressed video along with the original video. This allows for a subjective performance comparison between decompressed output and original input. The PC also acts as the configuration device for the compression algorithm allowing the user to change those variables discussed in Chapter 3 and seamlessly integrate with the DSP to adjust the compression features. Since the PC contains all the compressed data and an easily configurable communications device, in the serial port, it is able to use this device to transmit to a decoding PC, which in turn sends the compressed data to the decoding DSP for decoding and display on a video monitor. The decoding PC, then, is merely an information pump, receiving coded data and pumping it to the decoding DSP. The obvious concern in this system is the apparent waste of resources, in using an entire PC merely as a go-between. However,

everything done by the decoding PC and DSP is also done by the Host PC (decode and display). Thus the only reason for including the decode PC and DSP is for purposes of communication demonstration which is important as it demonstrates the timing reliability of the developed system. Since transmitting the data over the serial port takes time, the developed system is required to illustrate that the extra delays will not adversely affect the quality of the video stream nor produce errors, and this is done through the serial port demonstration. Furthermore it demonstrates that the developed system can function with a communications protocol meaning that the system could be altered to include only the camera, DSP and monitor using any other communications protocol.

4.2 Trimedia Overview

The Philips Trimedia Multimedia Development Board contains the Philips Trimedia TM-1300 at its core which is a Very Long Instruction Word (VLIW) multimedia DSP running at up to 143MHz achieving up to 6.5 BOPS[10] [Trimedia]. The Trimedia has access to 16kB of data cache and 32kB of instruction cache, along with support for up to 64MB of Synchronous, Dynamic RAM (SDRAM). The board used in this implementation contains 16MB, which provides enough memory for storage of image frames and memory requirements of encoding. The Trimedia VLIW architecture makes it particularly suited to complex DSP implementations, as it is uses parallelism to execute multiple instructions in parallel. Reduced instruction set computer (RISC) architectures provide simpler and faster solutions than do complex instruction set computer (CISC) based architectures, but the Trimedia VLIW provides even faster and simpler implementations than that of RISC, although the compiler support needs to be more complex. Trimedia reports [Trimedia] that its experiences show that complex DSP instructions ported to the Trimedia take between 1 and 0.5 times less instruction cycles than on other DSP's. Considering that the Trimedia code is C optimised whereas most DSP code is assembly optimised, this is an impressive result, further highlighting the Trimedia's suitability to complex implementations, as found in video compression.

[10] The Trimedia Documentation quotes this huge figure as a maximum achieved in a special test case which optimally makes use of the VLIW architecture.

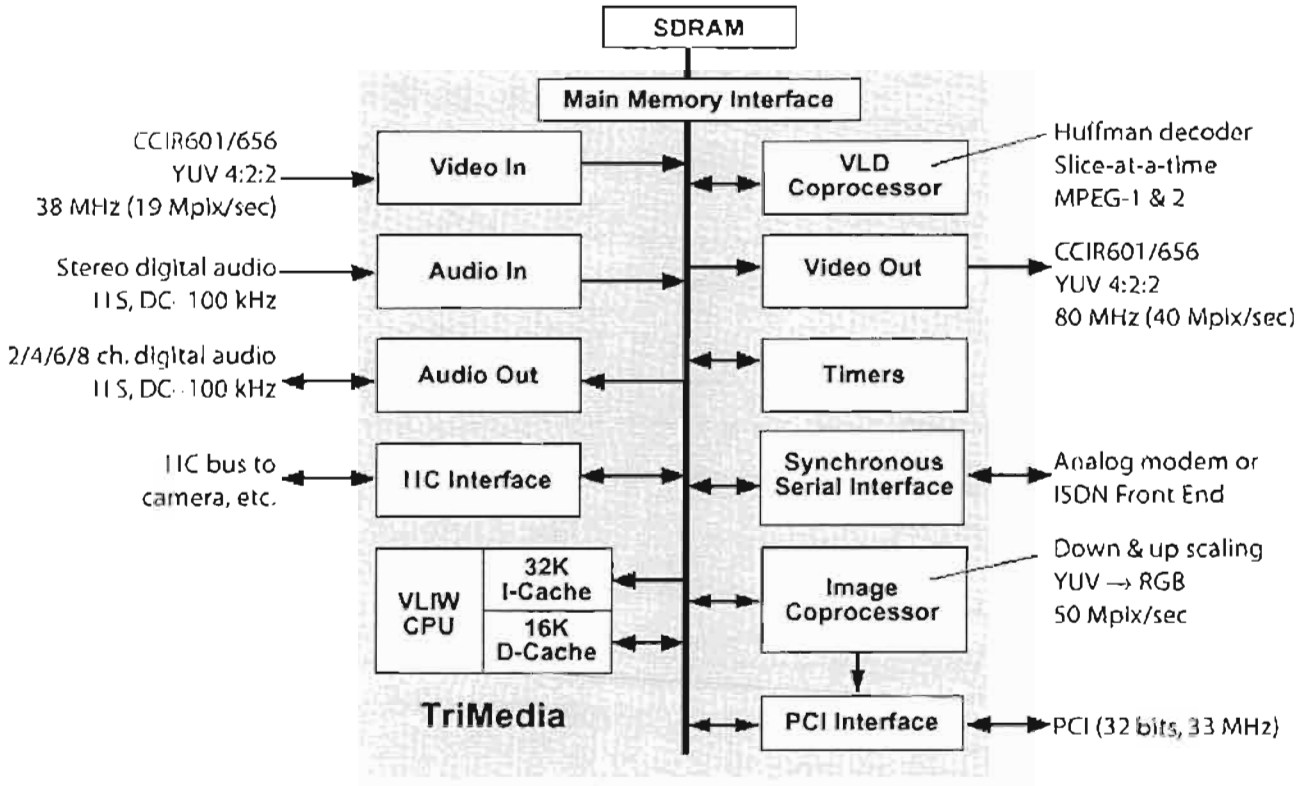


Figure 4.2: Trimedia Block Diagram [Trimedia]

The block diagram of the Trimedia development board, obtained from the *Philips Trimedia Documentation Set* [Trimedia], gives an overview the system. Since the board is primarily a multimedia development system some video and audio I/O is essential. The Trimedia provides for Composite and S-Video I/O with YUV sub-sampled *on-the-fly* to 4:2:2. The I/O meets those of the CCIR601/656 [ITU601] standard video input specifications. There are 2 channels for audio input each allowing for 8- or 16-bit samples using the I^2S and other serial 3-wire protocols with a sample range from DC-100kHz. Audio output allows for 8 channels at 16- or 32-bit sampling rates catering for mono and stereo data formats. The board also implements an I^2C bus for daisy chain communications with other electronic devices. An image coprocessor is included with optimised image based algorithms such as colour conversion and down- and up-sampling. For communication with the PC, a PCI interface is provided for Direct Memory Access (DMA). The peripheral Variable Length Decoding (VLD) block provides optimised solutions for MPEG-1 and MPEG-2 decompression making the Trimedia particularly suitable for these applications. Physically the board looks as in Figure 4.3.

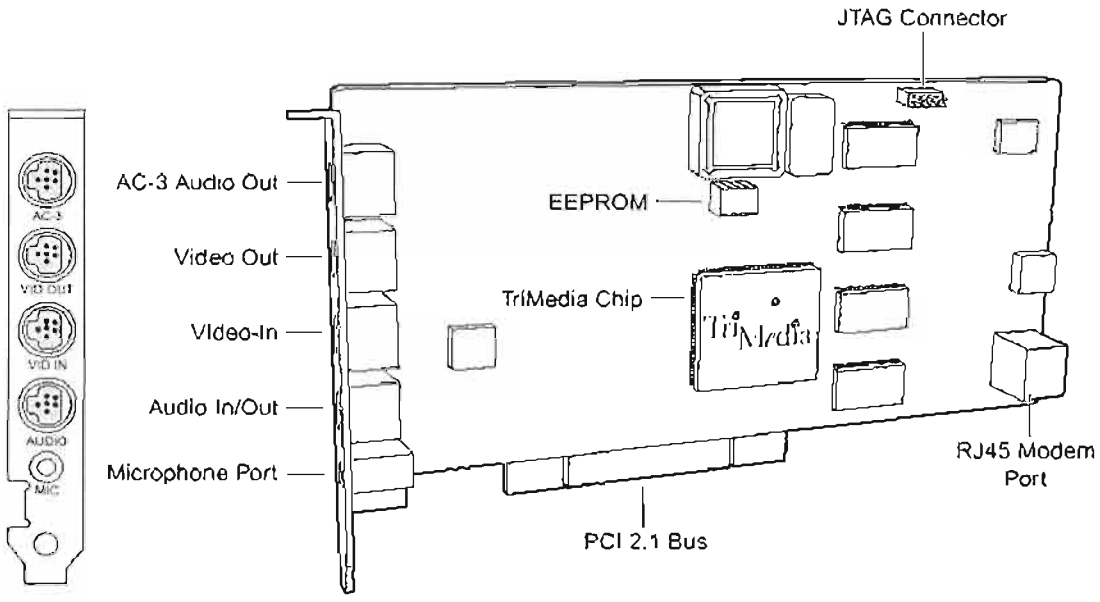


Figure 4.3: Development Board Diagram [Trimedia]

Software support for the Trimedia comes in the form of their software development kit (SDK), which allows for ANSI C and C++ code development with an optimised VLIW compiler. The SDK provides tools for source and machine-level debugging for isolating implementation problems, while providing tools for analysis and enhancement of developed code. The analysis tools provide a break-down of resources used by developed code on a function by function basis allowing for easy isolation of those portions of code which require more optimisation. The enhancement tools analyse compiled code and provide grafting and profile information to the compiler for a more optimised compiled output. A cycle-accurate machine-level simulator facilitates simulation of code for testing timing requirements and speed issues. Finally, the Trimedia uses the pSOS operating system developed by Integrated Systems, Inc. as the Real-time operating system.

4.3 The Initial Test Program

A test program was developed for the PC to test encoding ideas before porting them to the DSP for implementation. The major advantage of testing code on the PC is the more user friendly debugging facilities allowing for faster code development. This program was effectively a still image compression system, which could compress and decompress any greyscale image or colour image. The advantage of the test program was in testing basic code functionality and initial code optimisation before porting to the DSP. A screenshot of the test program is given in Figure 4.4.

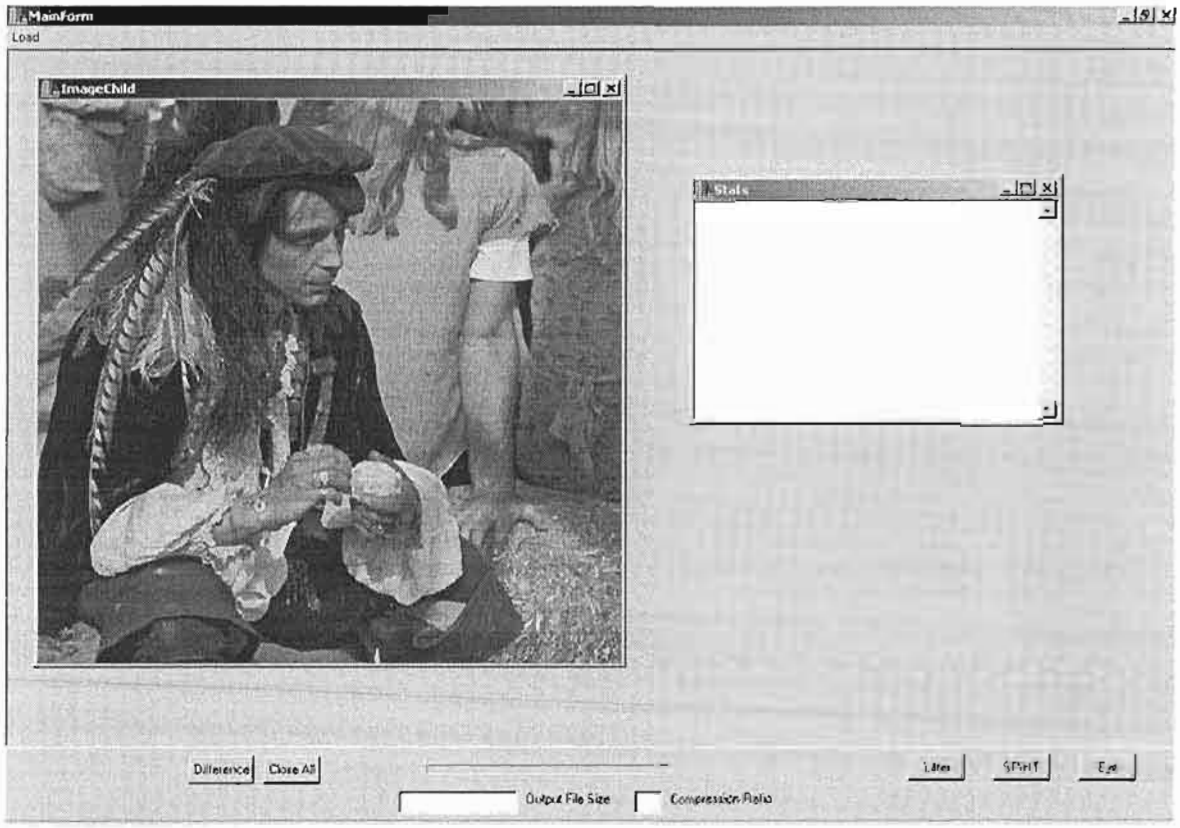


Figure 4.4: Screenshot of PC Test Program

4.4 DSP Software Overview

This section describes the software developed for the Trimedia DSP. Since the DSP is required to compress and sometimes decompress the video stream, the code development for each aspect of compression and decompression is described. Figure 4.5 shows a basic flow diagram for the DSP code. In this flow diagram the process of encoding the frame is represented as a single block for sake of clarity7, however, this block contains the majority of DSP code as it encompasses the Lifting, SPIHT and Arithmetic encoding portions of the code which will each be explained in this section. The set up of the Direct Memory Access (DMA) is directly related to the PC and will be explained in a later section, as will the area of interest software for the same reason. The remainder of this section will explain encoding issues regarding each of these flow diagram blocks.

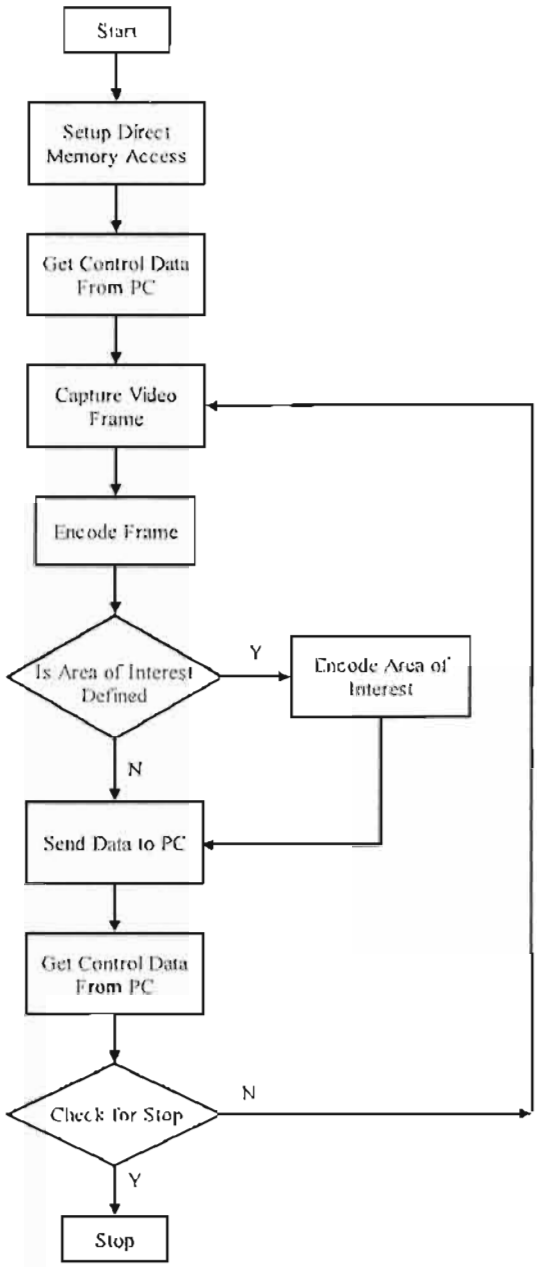


Figure 4.5: Basic Flow Diagram for DSP Code

4.4.1 Lifting code

The implementation of lifting is a direct extension of the diagram shown in **Figure 3.5**. Not shown in this diagram is the handling of the boundary of the image, which is still important. As in the filter bank implementation the lifting algorithm requires some boundary handling for the perfect reconstruction property to be met. In this case boundary extension was similar to that described in [Uyt99] where the signal is extended symmetrically. However as [Uyt99] shows, the symmetric extension must be carefully done to preserve perfect reconstruction. Four cases of symmetric extension are available as shown in Figure 4.6. The numbers in brackets denote the notation (a,b) used to describe each type of extension. If $a = 1$ then the first sample is not repeated whereas if $a = 2$ the first sample is repeated. This is similarly

done with the last sample and *b*. In this implementation for perfect reconstruction to be achieved the boundary extension is as in Figure 4.7.

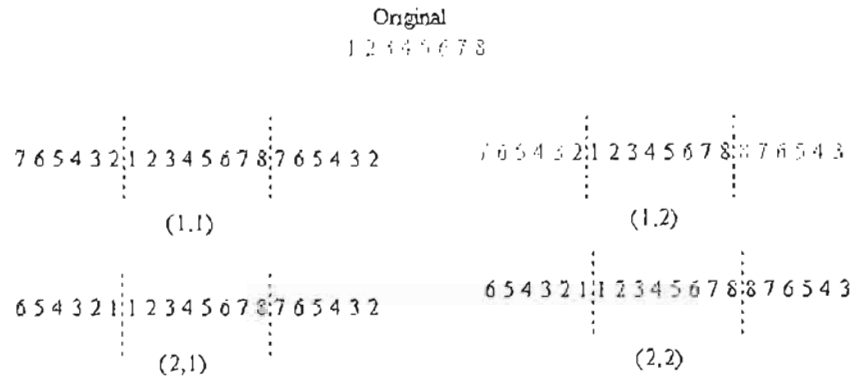


Figure 4.6: The Four Cases of Symmetric Extension

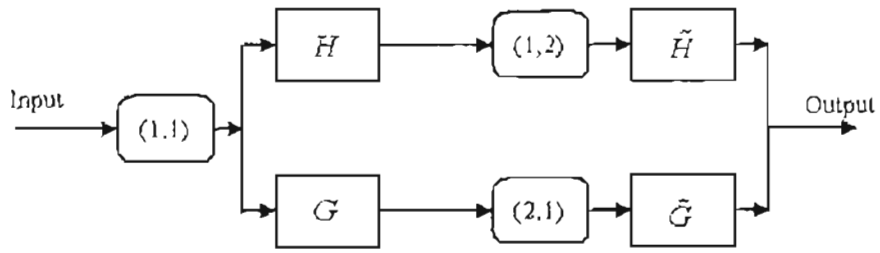


Figure 4.7: Boundary Extension for Wavelet Transform

Figure 4.8 helps to explain the implementation of the code and is effectively a simplified version of the diagram seen in **Figure 3.5**. To help with future explanation the top branch of the diagram contains only even parts of the input while the bottom part contains only odd parts. In terms of images, even parts of the signal are those pixels found at even indexed positions and odd parts of the signal are those pixels found in odd indexed positions. For example, the pixel found at position (2,2) in an image matrix is even. The main loop for a 1-D lifting operation is shown in pseudo code below. This operation is the same for both row and columns but only the row is shown here. To keep the pseudo code relatively simple the code for boundary handling has been omitted, however it is important to mention the effects of the boundary handling, as it can explain some discrepancies evident in the pseudo code. The result of boundary handling is to perform the loop described below with the symmetrically extended boundary as described above. This means that when looking at the code below the values B_2, E_2 and F_2 are effectively preloaded due to the extension being already performed.

```
For ( s = 0 to size of row)
{
      B1 = B2;                          //set the last B term as current
      E1 = E2;                          //set the last E term as current

      A1 = Image[current even];         //get the current even pixel
      A2 = Image[next even];            //get the next even pixel
      D1 = Image[current odd];          //get the current odd pixel
      E2 = α × (A1 + A2) + D1           //calculate the current E term
      B2 =  β × (E1 + E2) + A1          //calculate the current B term

      F1 = γ × (B1 + B2) + E1;
```

```
        Result[odd] = F1 / ζ;          //output the odd placed result
        C1 = δ × (F1 + F2) + B1;
        Result[even] = C1 × ζ;         //output the even placed result
        F2 = F1;                       //F2 is actually the previous F value
                                       //and F1 is the current
}
```

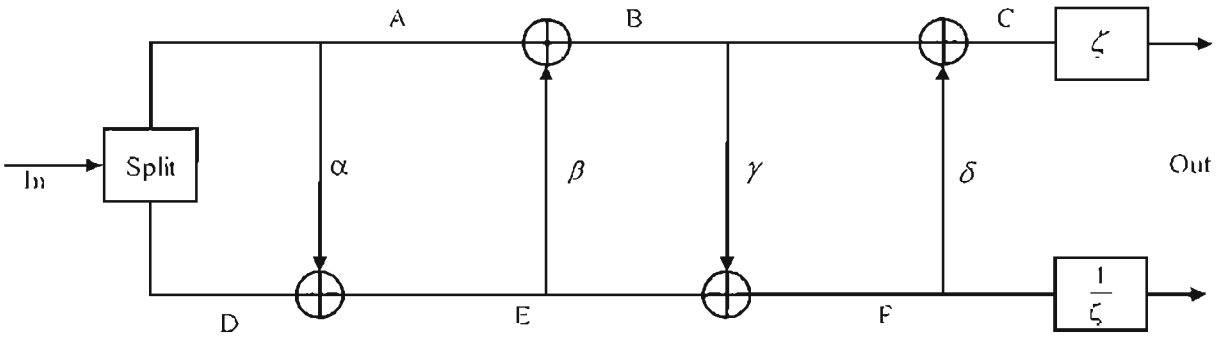


Figure 4.8: Diagram for Explaining Lifting Code

From this basic framework the lifting algorithm is implemented in another control loop Figure 4.9. This effectively iterates the transform until the required number of iterations has been met. This function uses the original image passed to it and allocates memory for a further matrix of the same size for working space. In a single iteration the result of the 1-D transform on the rows is stored in the working matrix and the result of the 2-D transform stored in the original image matrix, overwriting the original. Thus after several iterations the memory allocated for the original image is replaced by the iterated transform. This is a fairly memory efficient implementation and is thus useful for the DSP. However, as the results are stored in one large matrix, careful attention is required in addressing the respective approximate matrices to correctly iterate the transform.

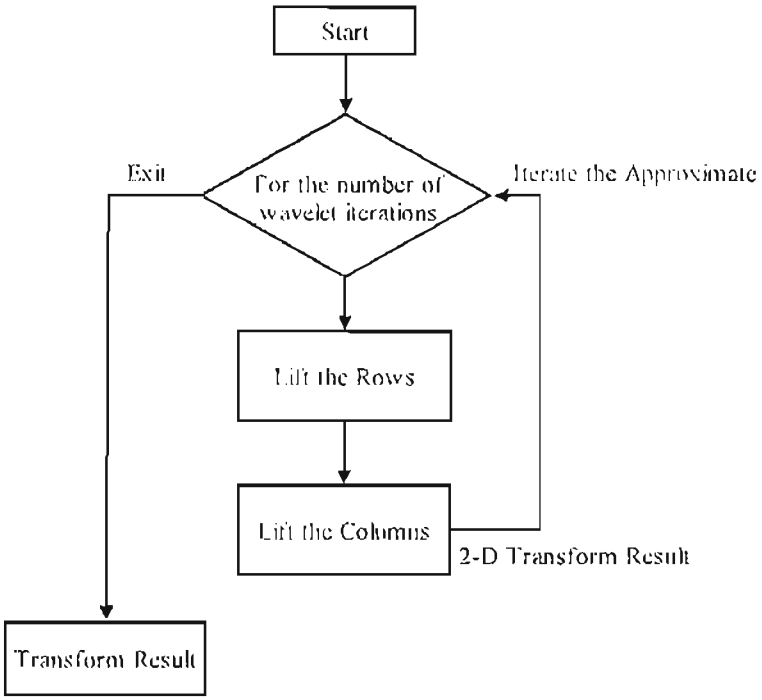


Figure 4.9: Flow Diagram of Control loop for Lifting

4.4.2 SPIHT Code

The SPIHT algorithm was first coded on the PC using a test program, which allowed for testing of encoding ideas and optimisations. SPIHT, as explained in Chapter 3, is effectively a list based program and therefore an obvious encoding implementation was to implement the three lists, LSP, LIP and LIS as linked lists. These lists were implemented as linked lists of structures as shown in Figure 4.10. The SPIHT has a dominant and subordinate pass with the dominant pass being further divided into a two logical paths; checking the LIP and the LIS. The advantage of using the linked list in the SPIHT implementation is that memory is used more efficiently, since as each list grows it allocates new memory and when the list gets smaller it deallocates memory meaning no wasted memory allocations occur.

For an initial test of memory requirements and performance, the SPIHT was coded on the PC test program and applied to the 512×512 image of 'Lena'. At the start the LIP was 256 elements long, the LSP, 0 elements and the LIS 256 elements long. Once the iterations reached a threshold value of 2 the LIP was 73457 elements long, the LSP 168519 and the LIS 4747 elements long. This meant that, in the case of the LSP, a minimum of 168519 memory allocations had taken place, the time taken to reach this point, 5.974s. The main cause for the slow execution time was seen to be the constant memory allocation and deallocation for the lists, which is a time consuming process. In order to test resolutions that will be used for the video compression algorithm, a 352×288 greyscale image of 'Lena' was tested and seen to execute in 975ms for a compression ratio of 10:1 and 2.928s for a compression ratio of 2:1. Although conceptually this solution worked, in terms of optimisation, the SPIHT required a rework.

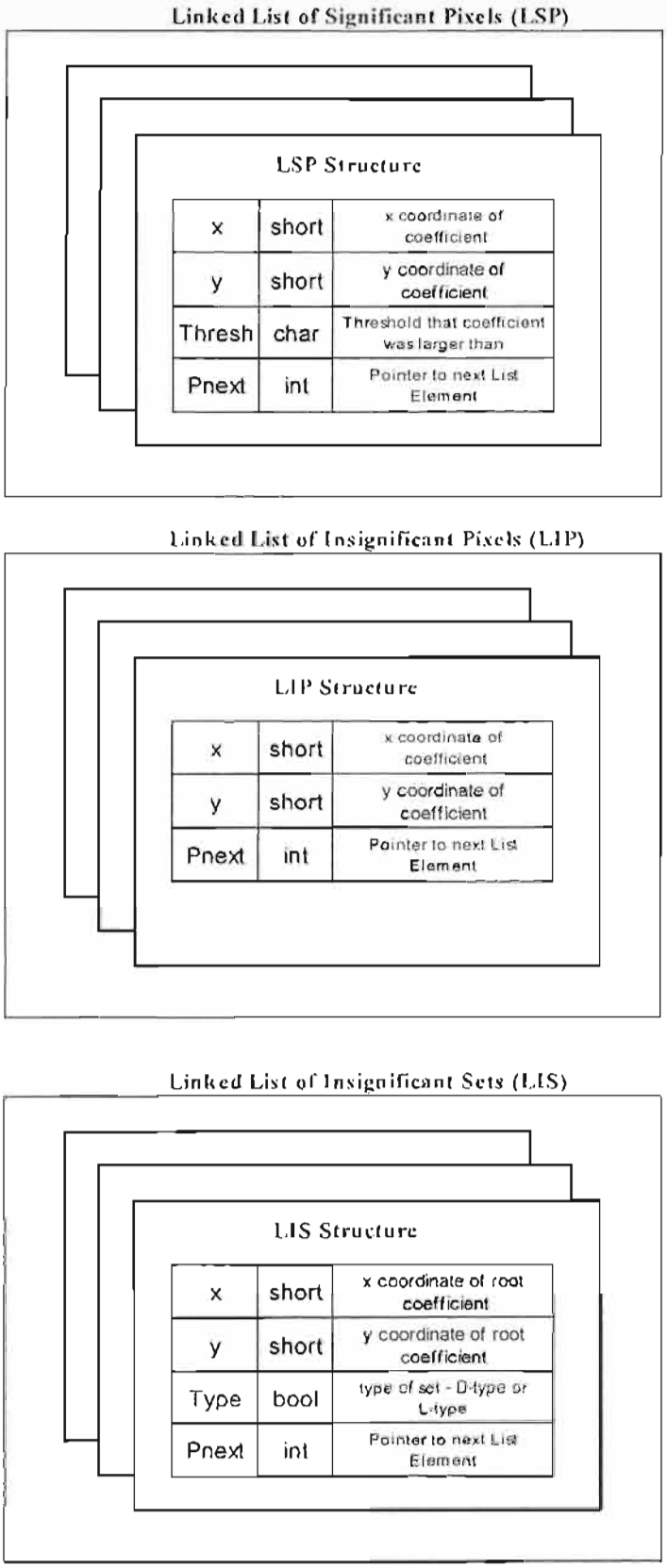


Figure 4.10: The Lists for SPIHT

The immediate concern was the memory usage of the system. For the 352×288 'Lena' image the list sizes at a compression ratio of 2:1 were 31636 for the LIP, 65192 for the LSP and 1127 for the LIS. Some calculation, see below, reveals that the total memory used for the SPIHT lists alone, is approximately 830KB. If the SPIHT were to iterate until the threshold was 0 the LSP would be full, with every coefficient accounted for, and the LIS and LIP empty. This would mean the memory required would be the size of the image times the number of bytes needed for each element, which is approximately 891KB for the 352×288 image. Another important aspect to notice about the lists is that, if a coefficient is present in the LIP then it cannot be in the LSP. This means that the LIP and LSP can be

```
Referring to Figure 4.10.
The LSP Struct contains 9 bytes.
      The LIS Struct contains 9 bytes.
      The LIP Struct contains 8 bytes.

Therefore: Memory      = 65192×9 + 1127×9 + 31636×8
                       ≈ 830KB
```

joined into a single linked-list, saving some processing time. The maximum resolution of a video image is to be 352×288 and the Trimedia has 16MB of SDRAM, so memory is not really too much of a problem however speed of execution is. Also it is apparent that as the compression ratio increases so the time of execution decreases. This is due to less memory allocation and deallocation operations required with smaller lists.

A solution to the linked list problem is in allocating all necessary memory at the beginning of the program, which is possible as the maximum usable memory the SPIHT list is known. The downside to this that the total memory possible for each list is always allocated and is thus not able to be used in other portions of the code. In the case of the three lists the total possible memory for each list is required to be allocated which means that 3×891KB ≈ 2.6 MB is needed to be allocated which is potentially a large portion of the available memory to be used. The upside is the slowdown problems caused by the constant memory allocation and deallocation can be avoided.

In order for this solution to become viable some adjustment to the link lists are required. As there can be no repetition in the LSP and LIP these lists are joined into a single list meaning that only two sets of 891KB allocations are required. In this new representation of the lists they are no longer seen as linked lists but rather as arrays of list structures as Figure 4.11 illustrates. In this new scheme the arrays of structures are allocated at the beginning of the program as two large blocks of memory. In each array element there is a pointer to the next array element and pointer to the previous array element. Although this is not essential it was found that encoding was simplified by allowing each element to point to the previous and next and not just to the next. Now traversing the list was a simple procedure of finding the start of the first element in the array for each type and following the links to the next element of those specific lists.

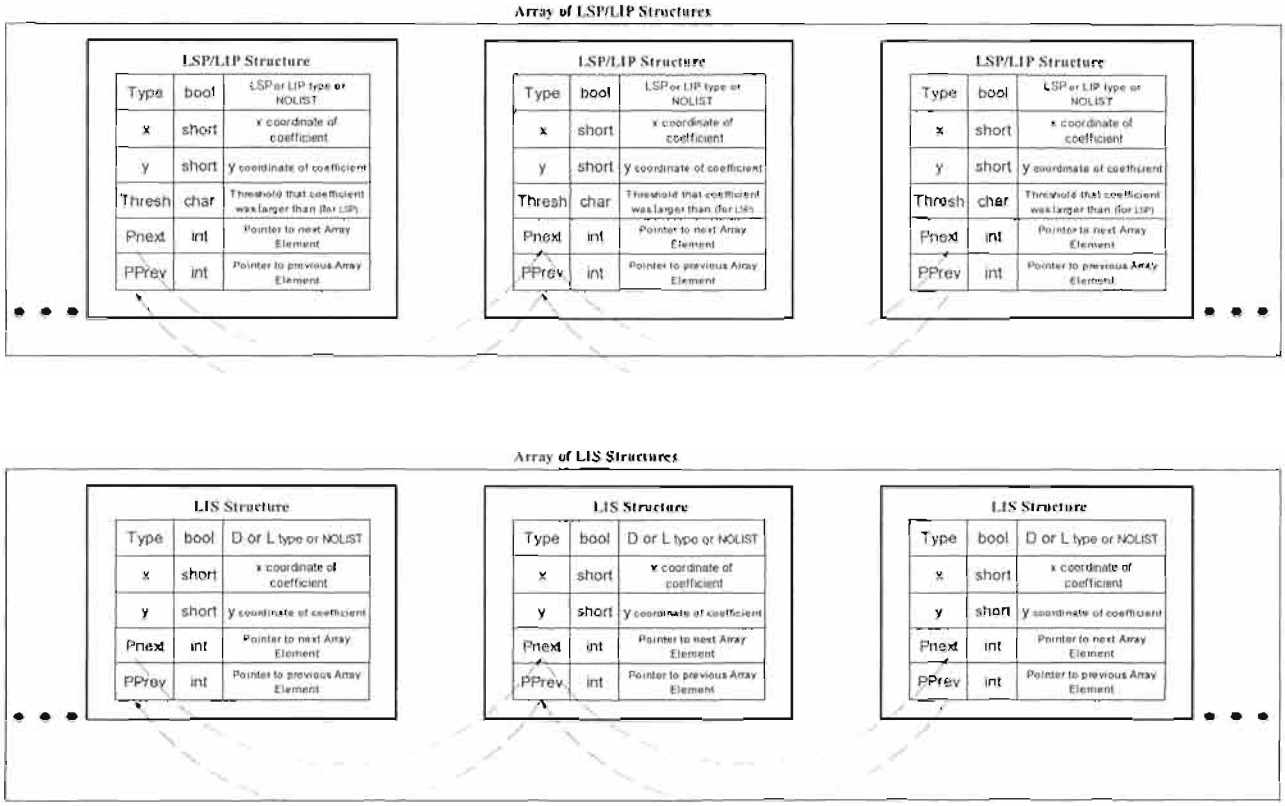


Figure 4.11: Array representation of lists

The removal and addition of elements in the lists still poses a non-trivial problem and requires some explanation. Since it is undesirable to allocate and deallocate memory the removal and addition of the list element does not involve memory handling. To account for such a situation an extra flag of *NOLIST* was added to the list structures. On initialisation (memory allocation) every possible list element is set to *NOLIST*. Thus when a coefficient is to be added to the list its type is now changed from *NOLIST* to either *LIP* or *LSP* for the LSP/LIP array or *D* or *L* for the LIS array. On deleting an element from the list the type is reset to *NOLIST* and the *PNext* and *PPrev* pointers of the elements around it altered to point to the correct elements as illustrated in Figure 4.12.

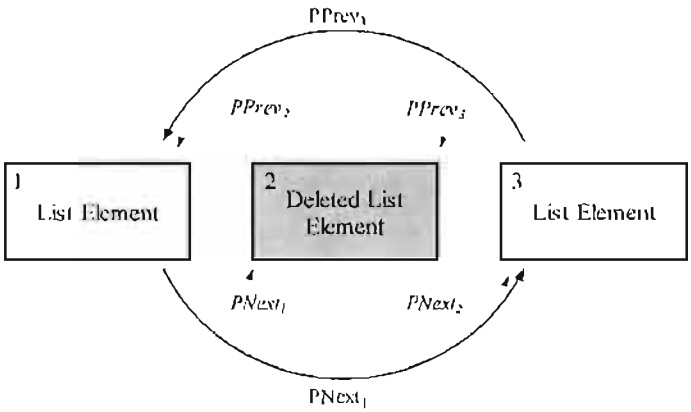


Figure 4.12: Illustration of the Deletion of an Array Element

Notice that although the list is in essence a slightly different implementation of the typical linked list, the need to constantly allocate and deallocate memory for new and old list elements has been eliminated. It is important to note that each list element represents a single pixel position in the 2-D image array. Thus when adding elements to the list their positions in the list are determined by the position the pixel holds in the image array. For example the pixel in position (2,2) of the image array is represented by the array element positioned at [2 + 2x(Image Width)] of the array. So the problem of having to find previously deleted array elements is nullified as each array position represents the list status of a specific pixel in the image. This results in a substantially faster list implementation at the disadvantage of requiring the allocation of the maximum memory needed for each list, which is not memory efficient. Chapter 5 highlights some timing results of compression and decompression times to support this claim.

4.4.3 Capture frame code

The Philips Trimedia Development Board captures video in 4:2:2 YUV format with 8-bit resolution and allows for a full resolution capture (704×576 for PAL) and half resolution CIF capture (352×288 for PAL). The maximum resolution for this video codec is CIF and as such, capture is set up for half resolution. To cater for the other supported resolutions some spatial down-sampling is required to convert the CIF to QCIF or SQCIF. Colour is also further down-sampled to obtain the 4:2:0 colour format used in this codec.

The set up of the video capture is based on recommendations and sample code from the Trimedia documentation [Trimedia]. Capture involves setting up and starting the interrupt based 'Video In' Application Programming Interface (API). The diagram in Figure 4.13 shows the basic calling commands to explain how this is done. Firstly, allocation of memory is required which in this case is the maximum size of an input image. In this system two memory buffers were used to allow for capture and processing. Each buffer had a flag associated with it that indicated whether it was ready for another image or whether it was still being processed. This cyclic buffer management meant that the interrupt would never place new data into a memory buffer before it had been processed. The viOpenAPI function sets up the interrupt function to which the program interrupts when a capture process is complete. The viYUVAPI function sets up the various video parameters such as the size of the image to capture, the x and y starting point to capture from and the memory addresses in SDRAM to dump the captured image data. It also enables the interrupt and thus begins capture. Once this is completed the interrupt merely cycles the memory buffers depending on availability.

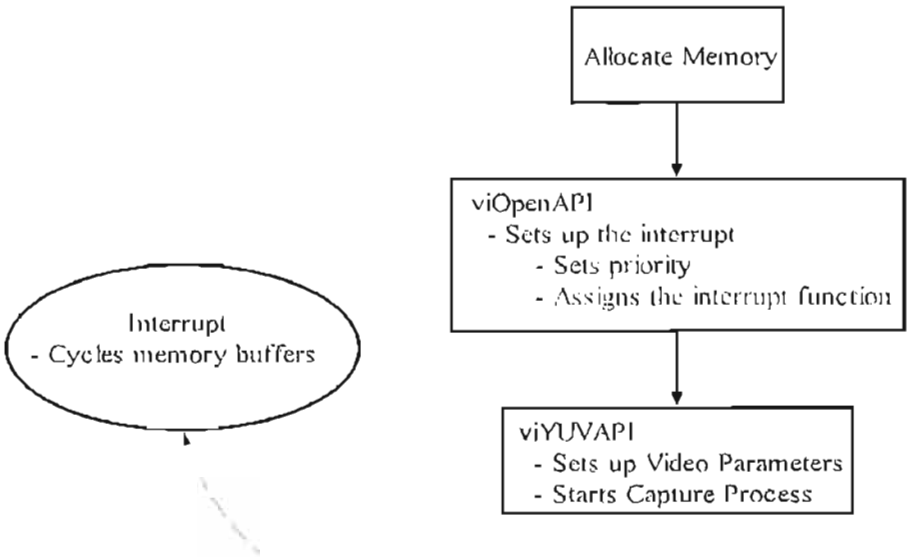


Figure 4.13: Capture of Video

Display of a video frame on a monitor is performed in a very similar process. Similar functions for setting up the video out interrupt and output parameters are called. The video out interrupt, however, cycles through the memory buffers copying frame data to the output video buffers when available.

4.4.4 Arithmetic code

The arithmetic encoder is the final stage of the compression algorithm and the first stage of the decompression algorithm. The encoder takes the binary output data from SPIHT and encodes this data losslessly. The code used for this section is obtained from [Witt87] and was modified only slightly to conform to input and output requirements of the compression scheme and so no further comment is going to be made in this regard.

4.5 PC Software

The main function of the PC is to interface between the user and the Trimedia DSP. The PC, however, also provides a useful tool for displaying and updating compression parameters and for comparison of original and coded data. This means that the PC has the decode software necessary to display the video and has the software for communicating with the DSP. Figure 4.14 shows a very basic flow diagram for the PC code. The purpose of this is to give the reader a better understanding as to where each code portion fits. The rest of this section aims at explaining important aspects of implementation concerning each block in this flow diagram.

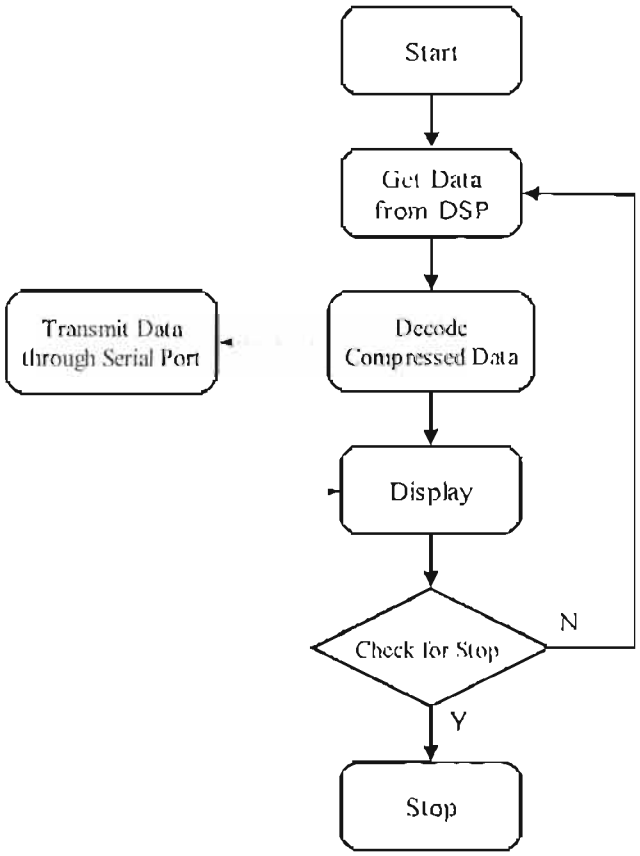


Figure 4.14: Basic Flow Diagram for PC Code

A screenshot of the user-interface is shown in Figure 4.15. The PC code has been written in Microsoft Visual C++.

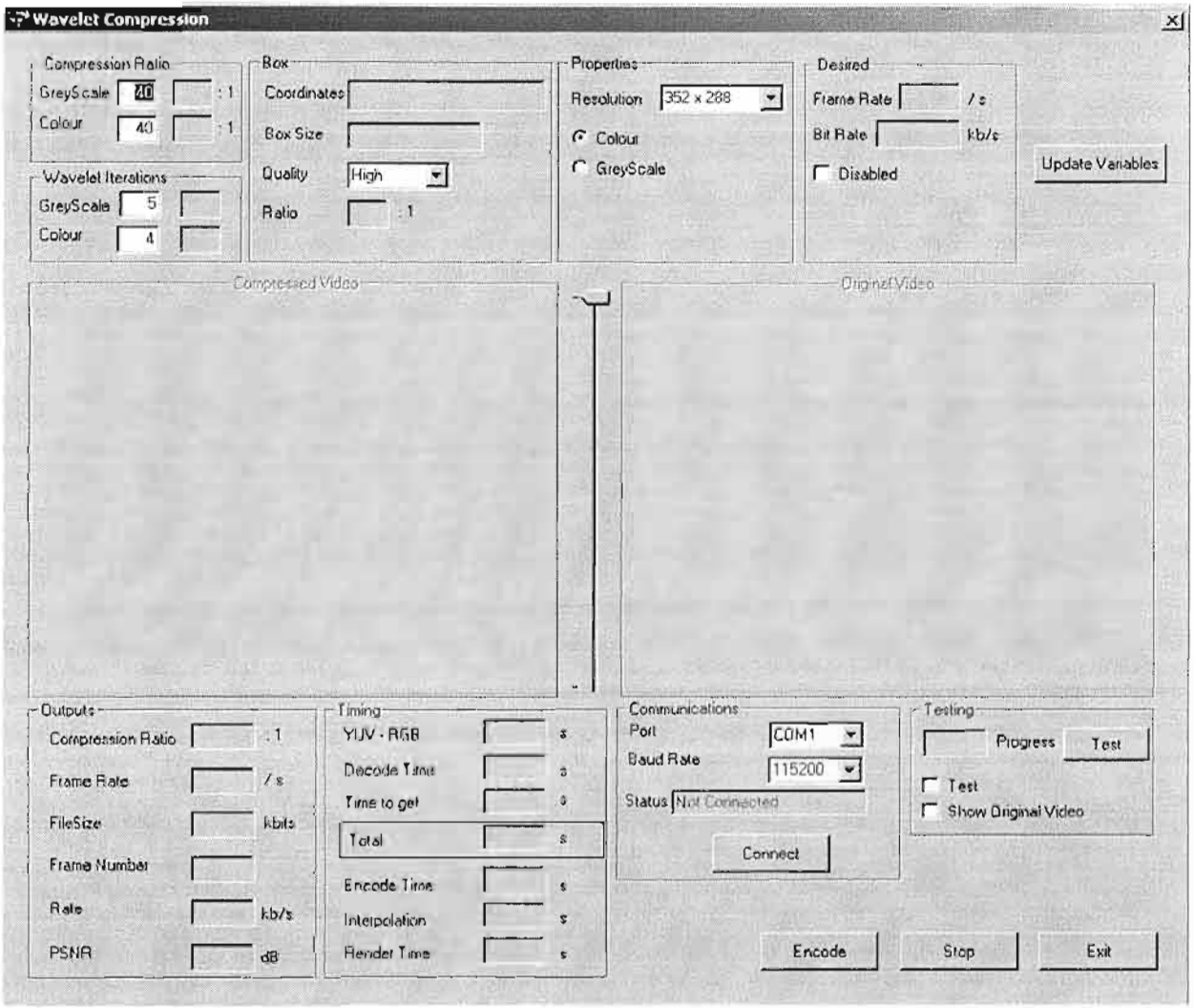


Figure 4.15: ScreenShot of PC Interface Screen

4.5.1 Start and Stop – The Use of the Peripheral Component Interconnect (PCI) Bus

Since the PC is the main controller for the DSP, communication between PC and DSP for purposes of control is vital. This communication is achieved via PCI bus and involves setting up a shared memory location on the PC, allowing data to be passed from DSP to PC through this shared address, which is controlled with the use of semaphores. A semaphore is passed to the unit requiring the use of the memory, once obtained that unit has control of the shared memory. When finished, it passes the semaphore to the next unit and releases control of the shared memory. This code is of vital import as it allows the PC to have total control of the DSP through the PCI bus. In fact, the entire system operates with the PC turning the DSP on and downloading the compiled code to the DSP. The DSP is then started and the system becomes operational. The PC can be used to stop the DSP at any time through the same communications channel. A diagram illustrating the process of setting up the DSP from the PC is given in Figure 4.16. It is evident from this that the DSP set up does two fundamental things; one it starts the DSP running and, two, it sets up the shared memory for future transactions.

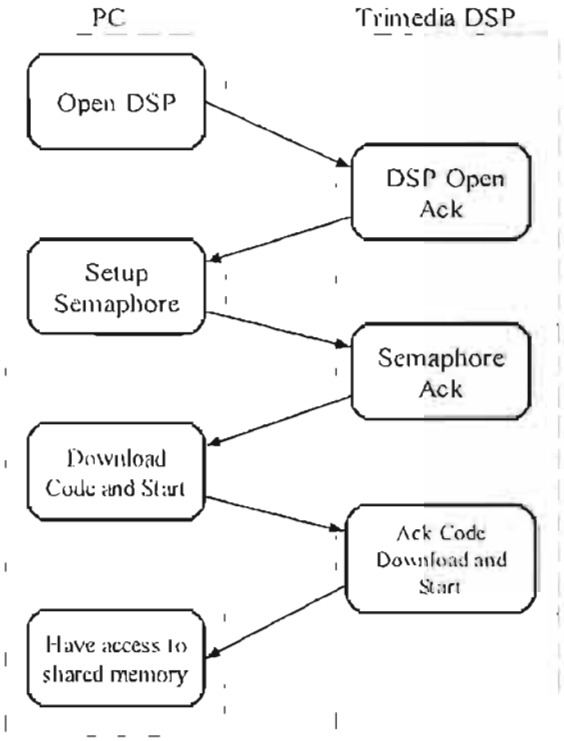


Figure 4.16: Diagram of Setting up DSP from PC

In order for useful communications between DSP and PC to occur some protocol is required. In this case, the protocol developed is basically a master-slave protocol where the PC is the master and controls the access to the shared memory and the DSP is the slave and responds to any requests from the PC. After set up, communication data is divided into two sets; control and image data. The control section contains such control commands as START and STOP and also contains configuration data for adjusting the compression parameters. The image data section contains the image data from the DSP, be this the original frame or the compressed image data. Since the system is re-configurable 'on-the-fly', control data is needed all the time to ensure that each frame is encoded and decoded using the correct

information. Since the PC is the master, communication is always started with the PC sending data to the DSP and then waiting for reception of data. If no data is received within a specific timeout, an error event is generated which results in the PC trying to shut the DSP down. The organisation of the communication data sent by each is given in Figure 4.17.

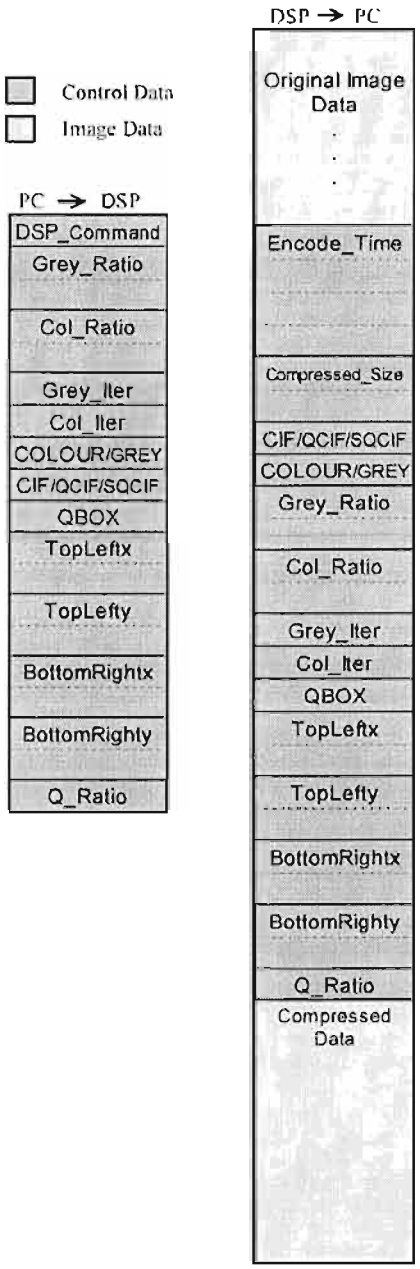


Figure 4.17: The Communications Packets

The first command in this packet is a START, STOP, or CONT command which sets the DSP in start encoding status, stop encoding or continue encoding. In stop encoding status the DSP sits in an idle loop until told to restart or until a SHUTDOWN command (explained later) has been sent by the PC. The rest of the control data sent by the PC includes configuration data. The compression ratio is sent for both greyscale and colour, which allows separate control of colour and greyscale compression. Since most image information lies in the greyscale of the image, it is often useful to compress this data less than that of the colour data as the colour can be compressed more than the greyscale data with less apparent loss in

information. The wavelet iterations to use is a parameter available for adjustment for purposes of testing the decorrelation properties of the transform; the less the iterations, the less the decorrelation and thus less effective compression. A control byte is sent to change between greyscale and colour and a similar byte is sent to adjust the resolution of the video. Lastly the coordinates of the area of interest (QBOX) are sent along with the compression ratio desired for this area.

The response from the DSP is to send the original image data (for use on comparison) along with the control data used to describe the compressed data being sent. This is in the same format as the control data sent by the PC but describes the parameters used to compress the previous frame. As the system is able to dynamically change compression parameters each frame needs to have a set of parameters assigned to it, describing how it was compressed. This avoids any confusion that may result while parameters are continually changing.

This video compression scheme is asymmetric, meaning the decode and encode times do not take the same time to execute. In fact, in this scheme encoding takes longer than decoding and so the limiting factor in the transmission scheme is the encoder. To maximally use resources, once started the encoder is continually encoding image frames. Since this process is slower than decoding, the PC is generally always waiting on the DSP for encoded data and is able to decode and display a frame before the DSP is able to present the next compressed frame. This means that the DSP is never idle during operation and so there is no wasted processing time.

A final word on the PCI bus transfer is the SHUTDOWN command, mentioned earlier. To allow for efficient start and stop of the compression a SHUTDOWN command is added, which is separate from the START and STOP commands. Starting and stopping the encode and decode process does not shut the DSP down and to allow for this, a separate SHUTDOWN command is implemented. The command shuts the DSP down and in order to continue, the DSP must be restarted and the code re-downloaded to the DSP.

4.5.2 The Optional Serial Link

When the serial link is activated, a second PC acts as a data pump between communication link and decoding DSP. This DSP is then able to decode the compressed image data and display it a video monitor as illustrated in Figure 4.1. This additional functionality of allowing the compressed video to be transmitted over the serial link requires a basic communications protocol and the implementation of receiving PC software to receive the data and send it to the decoding DSP.

The serial protocol implemented is very simplistic. This is justified because the focus of the project is not the creation of the robust communications link but rather on the video compression algorithm and the main purpose of the link is for demonstration of functionality. Figure 4.18 shows a basic flow diagram for the serial communications from encoder to decoder. The decoder initially sets up the serial port and then polls this port until it receives a "Ready" command from the encoder. This is an infinite loop until the command is received and provision has been made to force the decoder to break out of this loop for error correcting purposes. On the encoder side, an important aspect is that serial communication is only allowed when the option is activated by the user. In code this is implemented as a flag, which when true, indicates the activation of serial communication. Once the serial port has been set up and encoder made aware of decoder, communication is controlled by sending and receiving a 'go-ahead' signal that tells the decoder to prepare for an incoming video frame. This acts as a form of synchronisation, informing the

decoder of an impending frame to be transmitted. The control data, Figure 4.17, for that frame is sent first followed by the compressed image data. The decoder waits until both control and image data is received before sending the data to the decoding DSP through the PCI bus. The final stage is to check for the 'Stop' command, which is activated by a button on the encoder user-interface. Once activated the encoder sends this command along with the final frame to be decoded. When the 'Stop' command is received the decoder disables the serial port and stops the DSP.

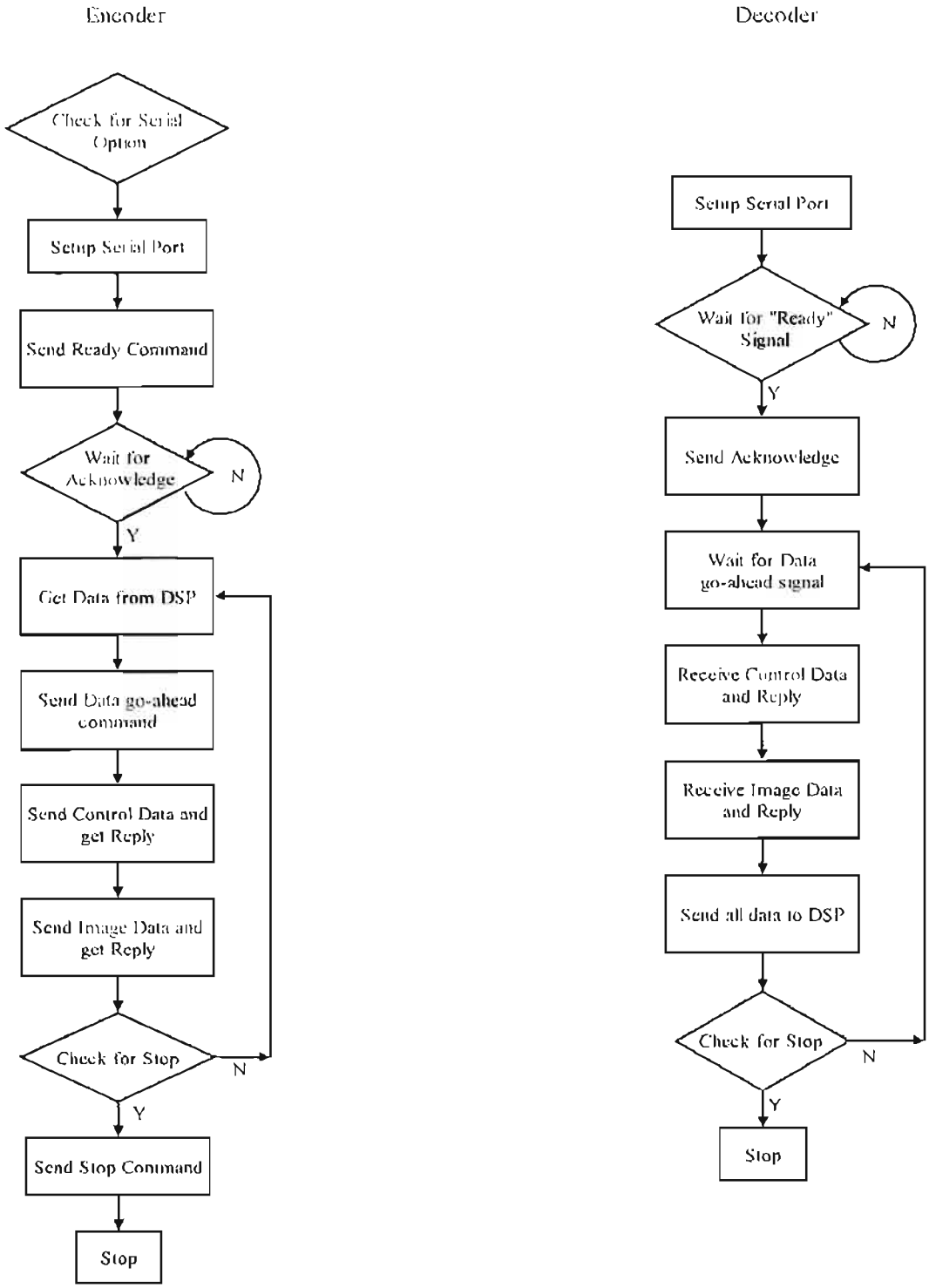


Figure 4.18: Basic Flow Diagram for Serial Link

4.5.3 Displaying Video on the PC Monitor

Displaying an image on the PC monitor requires the colour image to be represented in RGB format with no sub-sampling. Since this video compression scheme results in a 4:2:0 YUV colour video frame, some up-sampling and colour conversion is required. The process of up-sampling requires scaling the U and V matrices to the same size as that of the Y matrix before applying the conversion matrix that results in the desired RGB frame. The obvious method of up-sampling would be to insert zeros in those matrix positions that have no colour component associated with them. While being a fast up-sampling technique the result is a washed out colour image as can be seen in Figure 4.19 and Figure 4.20. In this scheme positions in the up-sampled matrices which don't have values associated with them are filled with copies of neighbouring values. This process has proven to result in unnoticeable colour changes from the original and up-sampled version as Figure 4.21 can testify. A better solution would be to use interpolation which would give a better estimate of unknown values, however the extra processing power introduced is considered unwarranted for the little gain in accuracy achieved.

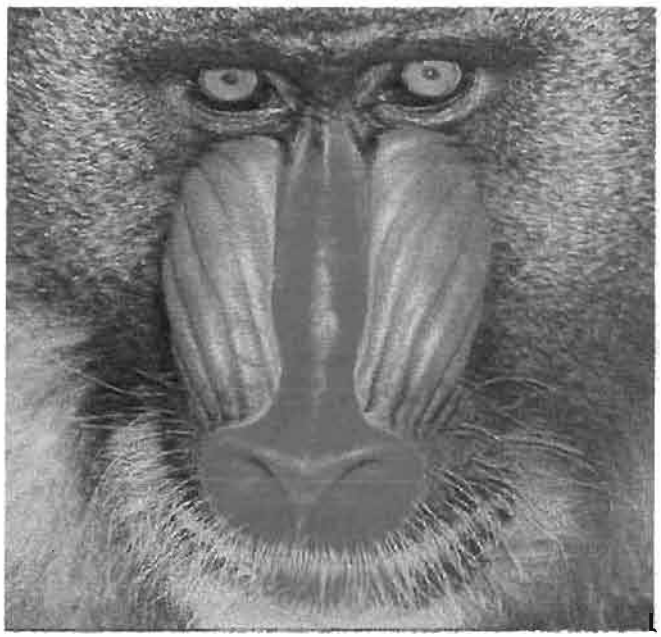


Figure 4.19: Original Colour 'Baboon' Picture

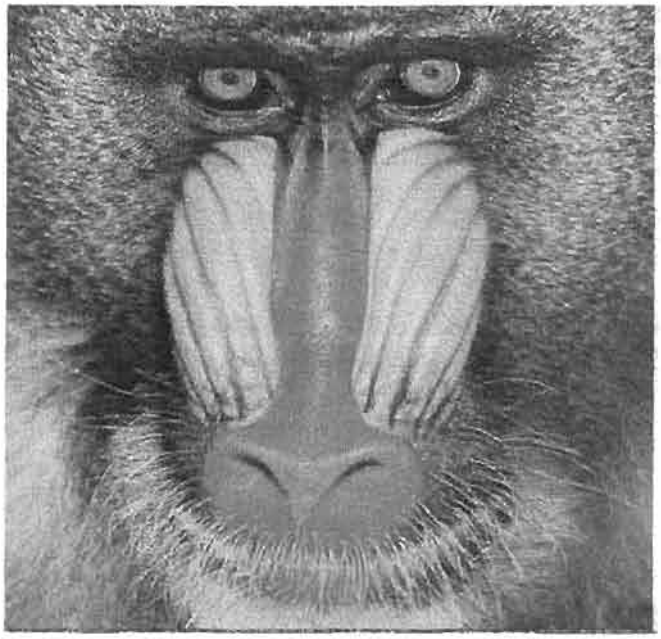


Figure 4.20: The Result of Incorrect Up-sampling

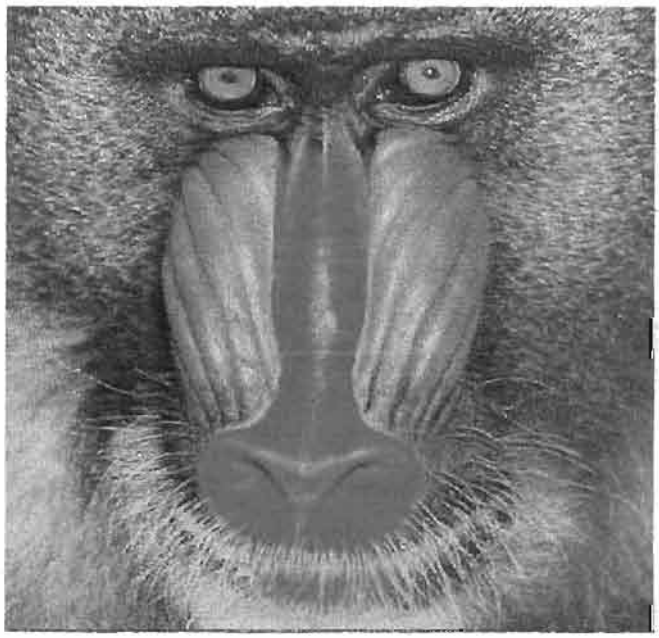


Figure 4.21: Result of Copying Neighbouring Pixels

Once the YUV image has been converted to the RGB space, display of the image is done using the Microsoft Bitmap API. Setting up the display in Microsoft Visual C++ involves setting up a Device Context (DC). This DC is used to create a compatible interface between the program and device drivers specific to the system used. As computers these days are all different, in order to make code that is written on one system compatible with all others it must be compatible with all other system devices. The creation of a compatible DC is therefore important for compatibility. Once this is done, painting of the image on screen involves defining the size and colour depth of the image and then indicating to the system, the coordinates where to place the image.

4.5.4 The Area of Interest

The realisation of a quality box means that the user must be able to define this box somehow. One possibility is the entering of coordinates of the desired box. The problem with this form of input is that it is cumbersome, as the user must now have some reference grid in order to correctly define the box. A better solution is a 'point-and-click' type interface where a user can point to a desired area and be able to interactively drag a box over this area and thus define the area of better quality. This form of user interaction is used in this compression implementation.

Figure 4.22 shows the possible frame sizes of the video being displayed which helps in explaining the restrictions that need to be placed on the drawing of the box. In order to seamlessly integrate a user defined box, the boundaries within which the box are valid must be known. In the case of Figure 4.22 the box coordinates may never exceed the CIF resolution of the frame, however if the frame is QCIF and SQCIF, then it must not exceed these dimensions. Furthermore, provision has been made for automatically resizing the quality box when the resolution changes. For example, if the resolution changes from CIF to QCIF then the quality box is resized to accommodate the new resolution and the video stream updated to provide a seamless change of resolution.

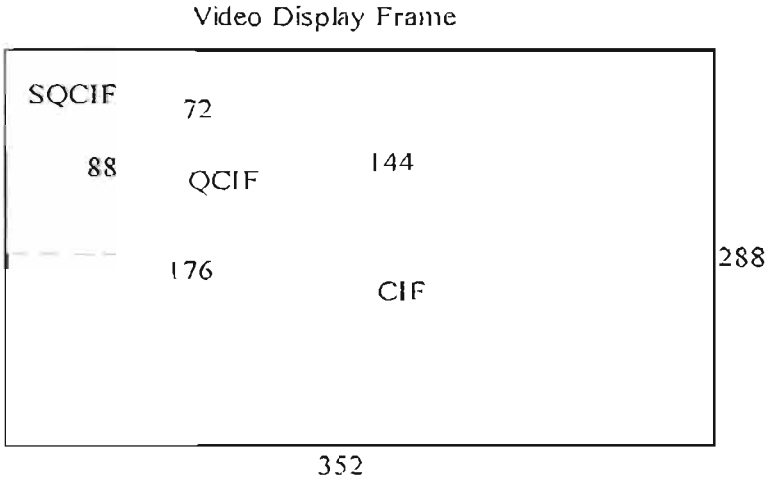


Figure 4.22: Possible Sizes for a Video Display

Another implementation issue worth commenting on, is the redefinition and deletion of an existing box. Currently each time a user defines a new box then the old one is deleted and new parameters calculated for the new box. For deletion of a box a user needs only to click the mouse in the region of the video display frame and the box is deleted and the video restored to meet the best possible quality for the existing rate metric.

Chapter 3 mentions the scalability issue of being able to define a specific quality metric for the quality box, which is achieved by defining a separate quality box compression ratio. However a restriction has been placed on this to make realisation of the box more feasible. Since the scheme dynamically reallocates resources to assign more priority to the area of interest, there exist some conditions that could cause unacceptable results. If the box defined is too large and the quality defined is too high then the possibility exists for there to be inadequate available resources to meet this metric. To account for this a restriction is placed on the maximum box size and maximum box quality where the maximum bandwidth

allowed for the box is half the total available bandwidth and the maximum size of the box is 144×144 for CIF, 72×72 for QCIF and 36×36 for SQCIF.

To allow for some scalability it was decided to have three quality metrics for the box; *High*, *Medium*, *Low*. In *High*, half of the bandwidth available is used to represent the quality box data, for *Medium* a quarter, and for *Low* a sixth. In terms of greyscale, it is trivial to calculate the remaining available bandwidth for use in compressing the rest of the image. However, for colour the problem becomes slightly more complex as a decision is needed on how to distribute the bandwidth over the greyscale and colour portions of the video frame. An obvious solution would be to assign the bandwidth half-half, but this is in fact not an optimal solution when considering that most of the important image information resides in the greyscale. So in this scheme it has been decided to allocate more bandwidth to greyscale than to colour. The next problem is to decide just how much extra to give greyscale. Intensive experimental testing, encompassing a range of video inputs and distribution characteristics, concluded that the optimal tradeoff between image quality and system bandwidth was attained by allocating ¾ of the available bandwidth to the greyscale data and ¼ to the colour. Using this distribution criteria, it was found that there was minimal colour information sacrificed for the extra greyscale information required.

4.6 Summary

This chapter has presented the implementation of the system detailing important coding aspects of each area of the codec. The aim of the chapter has been to explain implementation issues which have been addressed in order to achieve the desired result. The next chapter will examine the results of the implementation and comment on the effectiveness of the issues explained in this chapter.

Chapter 5 Performance Evaluation

Evaluating the performance of the implemented video compression scheme involves firstly, testing the functionality of the features discussed in previous chapters and then comparison with other existing schemes. This chapter begins by evaluating the performance of the system using a pre-recorded video sequence and testing the effects of adjusting various scalability options. PSNR and subjective pictorial results are given for purposes of evaluation. The affect of applying the area of interest on the scene for various bit rates is discussed before providing some comparison with other available schemes. Since this scheme is essentially a frame-based compression system, a comparison between other frame-based schemes is given using standard video sequences. These same video sequences are then used for comparison with some relevant temporal based schemes to provide a more complete evaluation of performance.

5.1 Overall Performance

Initial testing involves showing the performance of the scheme while adjusting the various compression parameters. Useful evaluation measures include the time taken to compress and decompress, the PSNR at various compression ratios, the effect of the area of interest for low bit rate schemes, the functionality of the serial port and its implications and the consequences of adjusting the wavelet iterations.

To test the general performance of the system a simple scene has been recorded containing a background and a single person walking towards the camera. The camera also pans the scene to simulate performance in a relatively high movement situation.

5.1.1 Initial PSNR and Subjective Results

Table 5.1 shows the average PSNR values measured for the video sequence described above for various bit rates and resolutions. The testing was performed at a constant frame rate of 5 frames per second and the PSNR measured for each frame and averaged over the entire sequence. As the luminance component of the video contributes mostly to the perceived quality, only its PSNR is shown for evaluation. The scheme is aimed at low bit rate systems and so only rates up to and including 128kbps were testing for performance. From the table and Figure 5.1, it is clear that as the bit rate increases so the objective quality increases, which is an expected result. In terms of CIF resolution, results for 16kbps were not taken as the resulting video at this rate was intolerably degraded and the PSNR measurement was considered worthless.

Bit Rate (kbs)	PSNR (dB)		Resolution
	Colour	GreyScale	
16	Too Low	Too Low	CIF
	19.43	20.39	QCIF
	21.08	23.07	SQCIF
28	20.30	21.23	CIF
	21.97	23.43	QCIF
	24 49	26.39	SQCIF
32	21.21	22.34	CIF
	22.56	23.65	QCIF
	24.97	27.76	SQCIF
64	23.40	24.71	CIF
	25.33	26.74	QCIF
	31.43	36.80	SQCIF
128	25.62	27.66	CIF
	29.20	31.61	QCIF
	39.72	40.36	SQCIF

Table 5.1: PSNR Results of Video Testing (Frame Rate = 5 fps)

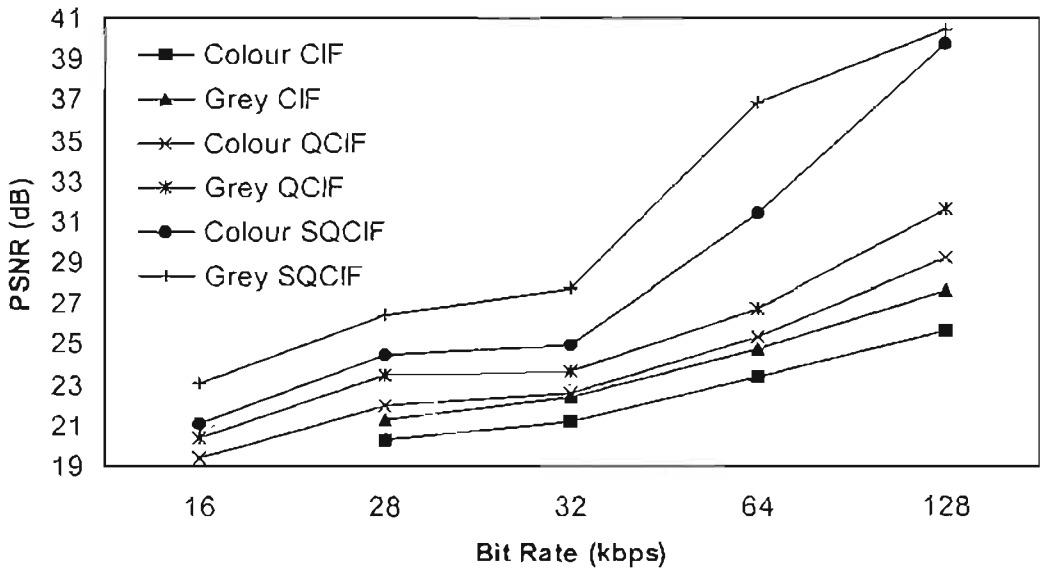


Figure 5.1: Graph of Tabulated Results

Subjective results shown in Figure 5.2, Figure 5.3, Figure 5.4 and Figure 5.5 for both QCIF and CIF video sequences for various low bit rates are given. These pictures give a visual perspective on the affect of adjusting the compression ratio to meet the desired bit rate. It is clearly evident that as the bit rate decreases so the image degradation increases and, in terms of the QCIF images at 16kbps, the degradation is extremely severe. Although the quality of all the images is poor, at 128kbps for both CIF and QCIF resolutions the details of the video are relatively clear and, in fact the quality of the QCIF

video at this bit rate is very reasonable. Even with the addition of the colour information the video sequences remain distinguishable for such a low bit rate.

128kbps
PSNR = 27.66dB

Original

64kbps
PSNR = 24.71dB

32kbps
PSNR = 22.34dB

Figure 5.2: Results for Video Testing at CIF

128kbps
Y PSNR = 25.62dB

Original

64kbps
Y PSNR = 23.40dB

32kbps
Y PSNR = 21.21dB

Figure 5.3: Colour Results for Video Testing at CIF

128kbps
PSNR = 31.61dB

Original

64kbps
PSNR = 26.74dB

16kbps
PSNR = 20.39dB

Figure 5.4: Video Results for QCIF

128kbps
Y PSNR = 29.20dB

Original

64kbps
Y PSNR = 25.33dB

16kbps
Y PSNR = 19.43dB

Figure 5.5: Colour Video Results for QCIF

5.1.2 Timing Issues

A relatively important aspect of the scheme is the timing issues concerned with compression and decompression as it has a direct bearing on the achievable frame rates for real-time operation. Table 5.2 shows encode and decode times for the video taken at various resolutions and for both greyscale and

colour. It is immediately evident that the compression and decompression times generally decrease as the compression ratio increases. The major contributing factor to the decrease in time is the SPIHT algorithm, as, with higher compression ratios, less trees of wavelet coefficients are required to be searched for significance, resulting in a faster execution time.

An interesting point to note is the encode times for QCIF resolution at 32kbps for colour and greyscale. The immediate assumption is that the greyscale video should encode faster than the colour, but, this is not the case. This occurrence is explained when considering that the colour is 4:2:0 sub-sampled and so the chrominance components are SQCIF resolution. The time taken to compress a SQCIF resolution image is a great deal faster than that of a QCIF image and larger compression ratios result in faster compression times. In this case the luminance component of the colour image is compressed more highly than the greyscale image and thus executes faster. Since the chrominance components are SQCIF, encoding them is very fast with the net result of a quicker encode time for the colour image than the greyscale, in this case.

An issue of concern is the large encode and decode times needed for CIF images, be they greyscale or colour. The fastest reported encode time is 204ms which means that the maximum possible frame rate achievable is 4.9 frames a second. This frame rate is less than the minimum considered frame rate requirement of low bit rate video (5 fps). Considering that the software has not been fully optimised it is foreseeable that this problem can be overcome with some further code optimisation. This said, the QCIF and SQCIF encode and decode times are relatively fast allowing for frame rates greater than that of 5 frames per second in all cases.

	CIF (ms)	QCIF (ms)	SQCIF (ms)	Bit Rate (kbps)
Encode	328	132		128
Decode	180	50		
Encode	231	86	69	64
Decode	185	40	20	
Encode	204	91	18	32
Decode	192	30	10	
Encode		76	17	16
Decode		32	10	

GreyScale

	CIF (ms)	QCIF (ms)	SQCIF (ms)	Bit Rate (kbps)
Encode	369	105		128
Decode	290	70		
Encode	310	131	70	64
Decode	205	50	30	
Encode	353	52	20	32
Decode	210	40	10	
Encode		64	18	16
Decode		40	10	

Colour

Table 5.2: Encode and Decode Times for Colour and Greyscale Video

5.1.3 The Effect of Adjusting the Wavelet Iterations

Apart from the low bit rate abilities of the implemented system the high scalability allows for testing of a number of scenarios. One such test is the ability to adjust the wavelet iterations for the wavelet transform. The advantage gained is a possible decrease in encode time with the sacrifice in image quality. To illustrate this point Figure 5.6 shows the affects of adjusting the number of iterations from 2 to 5 while the compression ratio is kept constant. It is obvious from the pictures that as the wavelet iterations increase so the image quality increases with encode time. The increase in the encode times, however, is not a linear process. This is due to the dyadic down-sampling in the wavelet transform, meaning that the next iteration only operates on half the image size of the previous and thus is able of execute faster. So as the number of iterations increases, the extra time needed to execute the remainder of the transform decreases. This is illustrated by the encode times quoted below the pictures, where the addition time needed going from 2-3 iterations is 40ms and from 3-5 only 23ms. From these results it seems that the relatively small gain in encode time is not worth the sacrifice in image quality and so the adjustment of wavelet iterations is not further considered.

5 Iterations
Encode Time = 246ms

Original

3 Iterations
Encode Time = 223ms

2 Iterations
Encode Time = 183ms

Figure 5.6: Results of adjusting the number of wavelet iterations. The compression ratios is maintained at 40:1

5.1.4 Lossless Compression?

Since the scheme is highly scalable it need not be limited to low bit rate applications and as such testing the results for high bit rates is meaningful. To show the performance of the scheme in a lossless[11] scenario, the system is set up and the compression ratio adjusted until visible degradation is observed. The test involved visual comparison between compressed and original video and so is a subjective one. However it still gives a useful indication of the performance of the system in an environment where visually perfect video is desired. Greyscale images were seen to visibly degrade when the compression ratio is increased past 10:1 (for CIF) and manifested in small ringing artifacts around the edges of certain objects in the scene. With colour video the compression ratio achieved was before degradation is approximately 20:1[12] for CIF.

5.1.5 The Serial Port

Testing the serial port feature of the scheme is basically just verifying functionality as all performance evaluation can easily be performed without the serial port through the PC. Such a test requires setting up the serial link and streaming video across for decode and display on a TV monitor. The serial port does add extra delay when the BAUD rate is set very low and the compression ratio is high. Thus testing requires verifying that there is no loss in synchronization between encode and decode DSP's due to the extra delay introduced. This occurrence was tested and the result was a very slow frame rate but the system did not fail, indicating the communications protocol was functioning correctly. Further testing of the video scheme using the serial port was discontinued as the all other features of the scheme could be tested on the PC. However this does not mean the addition of the port is meaningless as it does allow for a more realistic demonstration of operation in a real-world type environment.

5.2 *Effect of the Areas of Interest*

The addition of the user-selectable area of interest feature is this codecs attempt to make the video suitable for low bit rates, while being only appropriate for a selection of possible applications. Thus some analysis of the affect of the area of interest is required.

Tests conducted to determine the usefulness of the area of interest, were to stream video at the various low bit rates tested earlier (16, 32, 64, 128kb/s) and select an area for higher quality. Both colour and greyscale video was examined with the luminance PSNR being calculated for the quality area as well as the entire image, excluding the quality area. The goal was to see if the extra quality provided in the quality box was worth degrading the surrounding image.

Table 5.3 shows some PSNR results for the streaming video at various bit rates. The difference here is that the area of interest has been defined in each case. At 128kbps in QCIF format the resulting video is of very reasonable quality and so there is no real gain in including an area of interest for this bit rate. It is clear that in all cases there is a significant improvement on the PSNR in the quality area at the expense of

[11] In this case, the term lossless is used very loosely and refers to the image before any visible loss of quality when visually compared to the original

[12] This was calculated as the 4:2:0 colour image size / compressed output

the rest of the image. The general trend shown in these results is that the area defined for higher quality is able to achieve an increase in quality which, in some cases, is quite drastic.

An important observation is that the size of the defined box plays a major role in the achievable increase quality. The larger the box size the less extra quality that is able to be achieved. This observation is supported when considering the case for 64kbps CIF on Table 5.3. In this case the PSNR measured for the area of interest is higher in the colour image than in the greyscale image. Intuitively this should not be the case as there should be more bandwidth available for the greyscale. However, the size of the defined box in the colour video was slightly smaller than that of the greyscale video and as such provided a better PSNR. Comparison with this table and Table 5.1 shows that the entire image PSNR has been degraded below what was measured in Table 5.1, in order to allocate the extra bandwidth needed for the quality box, while maintaining the bit rate at a constant value.

Bit Rate (kbps)		CIF		QCIF	
		Area of Interest PSNR (dB)	Image PSNR (dB)	Area of Interest PSNR (dB)	Image PSNR (dB)
16	Colour	Not Measured		30.91	19.21
	Greyscale			29.34	21.00
32	Colour	32.76	22.64	35.63	20.81
	Greyscale	32.94	22.70	33.41	22.17
64	Colour	35.81	21.87	36.80	22.34
	Greyscale	34.32	23.41	36.76	24.49
128	Colour	36.02	23.71	Not Measured	
	Greyscale	36.83	26.32		

Table 5.3[13]: Luminance PSNR Measurements at Various Bit Rates

In the case of greyscale video some of the video frames described in Table 5.3 are presented in Figure 5.7 and Figure 5.8. It is clear that inside the areas of interest the quality of the video appears better while outside the video is degraded. In this case the testing involved defining an area around the head of an individual for increased quality as could be used in a security type application. Thus a useful function for such scheme would be in any application where the surrounding scene is deemed unimportant and only certain areas of the scene are deemed significant. This shows that at very low bit rates an intra-frame video codec can still be useful when using a user-definable quality box idea. The drawback of the system is that, it is now only suitable for a limited number of applications making the codec very specific.

[13] Results were not measured at 16kbps for CIF for the same reason as in Table 5.1

Original Frame

Result at 32kbps
Image PSNR = 22.70dB
Area of Interest PSNR = 32.94dB

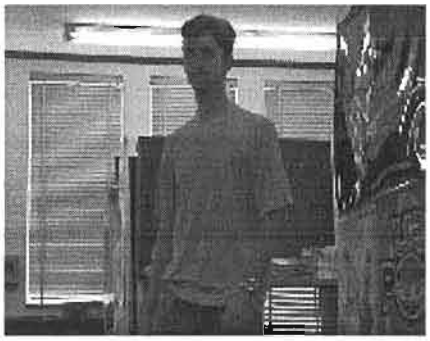
Original Frame

Result at 128kbps
Image PSNR = 26.32dB
Area of Interest PSNR = 36.83dB

Figure 5.7: Area of Interest Results for Greyscale CIF

Original Frame

Result at 16kbps
Image PSNR = 21.00dB
Area of Interest PSNR = 29.34dB



Original Frame

Result at 64kbps
Image PSNR = 22.34dB
Area of Interest PSNR = 36.76dB

Figure 5.8: Area of Interest Results for Greyscale QCIF

5.3 Comparison with Other Intra-frame Schemes

As the codec is an intra-frame compression scheme it is relevant and useful to compare it against other codecs of the same type. MJPEG is a widely used and very popular intra-frame compression scheme as it provides reasonable compression with the additional desirable feature of random access to frames. A new scheme which is to be implemented in the near future is MJPEG2000 which is the video compression extension of JPEG2000. Essentially both these schemes treat the video as a set of still images and compress each image separately to meet the desired rate constraint. Useful results for comparison are the objective (PSNR) and subjective performance of these schemes as they compare to the implemented codec.

In testing the performance, a MJPEG2000 codec could not be acquired and so a simulated system using JPEG2000 was used for testing. It is assumed that this simulated system would provide comparable results as the fundamental concept of MJPEG2000 is to apply JPEG2000 to a series of video frames in compression. Since a specific rate metric can be easily set for JPEG2000, compression ratios required to meet certain desired video rates were calculated and the video frames compressed to these rates using JPEG2000. Although, results generated from this may be slightly incorrect, they are still useful for comparison as they will be close to the actual results achievable.

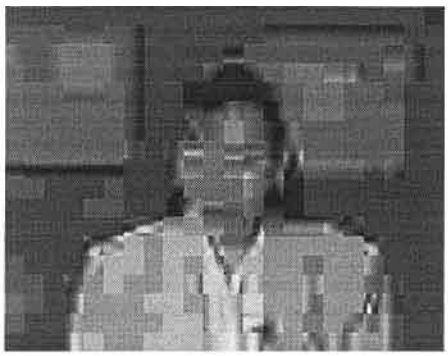
Bit Rate (kbps) at 5 Frames/s	MJPEG			MJPEG2000			Implemented System		
	Foreman PSNR (dB)	Akiyo PSNR (dB)	HallMonitor PSNR (dB)	Foreman PSNR (dB)	Akiyo PSNR (dB)	HallMonitor PSNR (dB)	Foreman PSNR (dB)	Akiyo PSNR (dB)	HallMonitor PSNR (dB)
16				23.56	26.65	22.35	24.39	27.89	23.11
32	22.57	25.44	21.61	26.89	30.64	25.77	27.23	31.20	26.26
64	28.62	32.25	27.97	30.30	36.60	30.61	30.70	35.91	30.59
128	33.01	37.97	33.59	35.87	42.78	37.75	34.89	41.35	36.13

Table 5.4: Measured PSNR for Video Frames of Standard Sequences, 'Foreman' Frame 59, 'Akiyo' Frame 117 and 'Hallmonitor' Frame 132.

Table 5.4 shows measured results for comparison over the three standard video sequences. 'Akiyo' is the classic head-and-shoulders shot typical in videoconferencing scenes, 'Foreman' is a very busy scene with a great deal of camera movement and 'Hallmonitor' is the classic example of a security type monitoring application. These scenes were tested only for greyscale as this comparison would convey the necessary information and it is felt that the extension to colour would not add an extra information for use in comparison. In the case of JPEG, results for 16kbps were unacceptably degraded and as such are not shown. From the table, it is clear that the implemented system clearly outperforms that of MJPEG and in some cases that of the simulated MJPEG2000. As the bit rate increases the MJPEG2000 scheme starts to outperform that of the implemented scheme. Although the results for MJPEG2000 are only a simulated case and may not provide an exact measure, it is felt that the results generated are very close to the actual case and as such the comparison shows that the implemented system is able to compare favourably to MJPEG2000.

Figure 5.9, Figure 5.10 and Figure 5.11 provide some visual results to highlight the subjective performance of the scheme. These were taken at 32kbps for each of the three sequences. It is clear that both the MJPEG2000 and this system outperform MJPEG. Also evident is the presence of heavy 'blocky' artifacts in the MJPEG images, while the MJPEG2000 and the implemented system look very similar in quality.

a) 'Akiyo' Frame 117



b) MJPEG 32kbps

c) MJPEG2000 32kbps

d) Implemented System 32kbps

Figure 5.9: 'Akiyo' frame 117 encoded at 32kbps for 5 frames/s

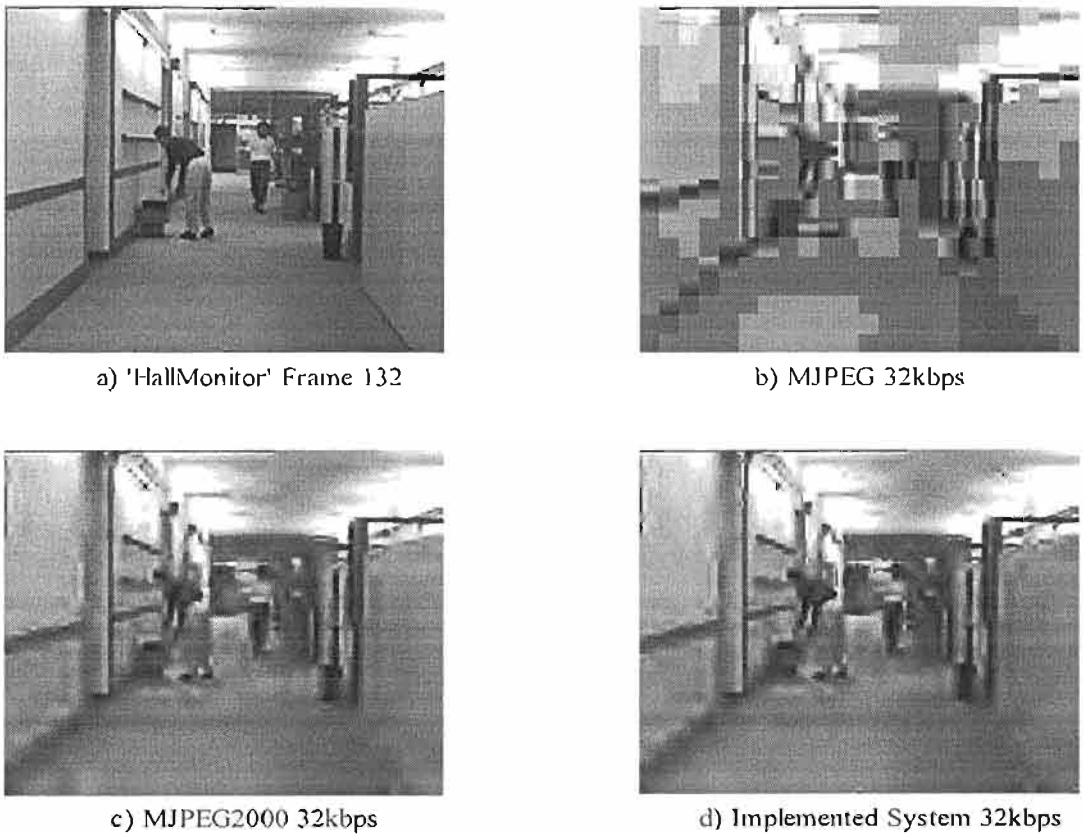
a) 'Foreman' Frame 59

b) MJPEG 32kbps

c) MJPEG2000 32kbps

d) Implemented System 32kbps

Figure 5.10: 'Foreman' frame 59 encoded at 32kbps at 5 frames/s



a) 'HallMonitor' Frame 132

b) MJPEG 32kbps

c) MJPEG2000 32kbps

d) Implemented System 32kbps

Figure 5.11: 'HallMonitor' frame 132 encoded at 32kbps at 5 frames/s

5.4 Comparison with Temporal Schemes

For the sake of completeness it is useful to compare the scheme with some other video compression schemes aimed at low bit rates. These are the PACC [Marp00] and MPEG-4 Verification Model standard schemes which both employ temporal compression. The PACC scheme was chosen as it represents the very latest in video compression technology and provided excellent results. MPEG-4 VM was chosen to conform to the literature as this scheme is used in most research works for comparison and is thus very familiar. The system was compared using the same standard video sequences of 'Akiyo', 'Foreman' and 'HallMonitor' with subjective and objective results taken. The PACC results are quoted only for the bit rates and frames shown in Table 5.5 and so, for comparison, the implemented system was tested with the same parameters.

Sequence	Bit Rate (kbps)	Frame Rate f/s	MPEG-4 PSNR (dB)	PACC PSNR (dB)	Implemented System PSNR (dB)
Akiyo	24.15	7.5	37.38	38.36	25.44
HallMonitor	25.90	10	33.75	34.51	21.12
Foreman	50.30	7.5	32.03	32.91	26.91
	123.70	15	34.29	35.16	27.70

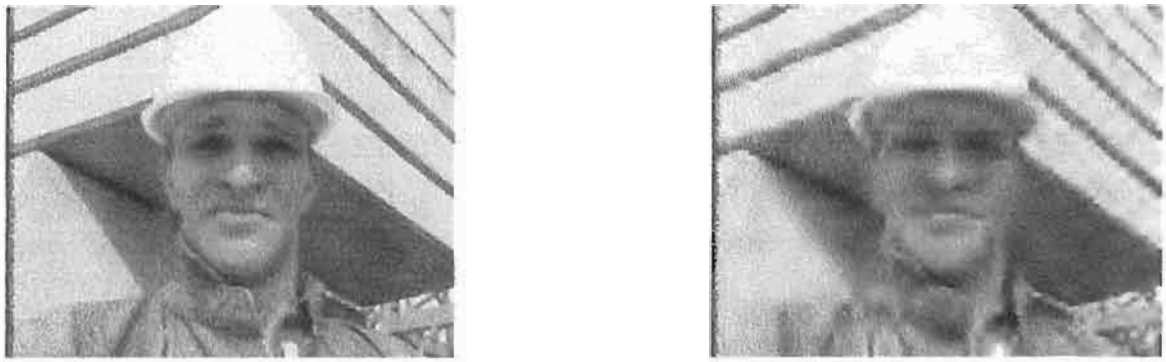
Table 5.5: Luminance PSNR Results for Entire Video Sequences

Table 5.5 shows the PSNR results of PACC and MPEG-4 (obtained from Marpe et al [Marp00]) compared to the implemented system. It is clear that both the PACC and MPEG-4 schemes easily outperform the intra-frame based scheme, especially in the 'Akiyo' and 'Hallmonitor' sequences. This is as expected as, the addition of temporal compression increase performance a great deal, especially in sequences of low motion, as with 'Akiyo' and 'Hallmonitor'. In the 'Akiyo' and 'Hallmonitor' scenes there is very little movement in the scene, with the majority of the background remaining static. Thus the temporal based schemes are able to achieve enormous compression results by using previous and future frames, whereas the intra-based scheme struggles.

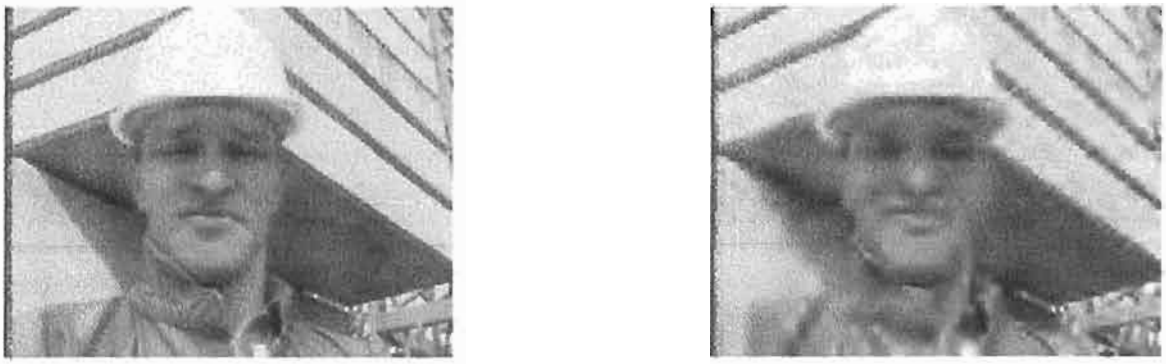
However in the busy 'Foreman' sequence there is a great deal of temporal movement with the camera panning the scene and the Foreman's head moving. In this scene the intra-frame based scheme is able to perform more comparably, although still losing out to the temporal schemes. Figure 5.12 and Figure 5.13 shows the visual results of the intra-frame system and this supports the objective results. In the case of the 'Akiyo' and 'Hallmonitor' sequences, the degradation is intolerable whereas the 'Foreman' sequence performs reasonably acceptably.

With the addition of the area of interest, the compression scheme is able to make up some ground on the temporal schemes, especially with the 'Foreman' sequence. Figure 5.12 and Figure 5.13 shows the result of applying the area of interest on the face in 'Foreman' and 'Akiyo' and on the person in 'Hallmonitor'. In the 'Foreman' sequence the luminance PSNR inside the box reaches 36.61 dB on the frame shown and visually the face is very recognizable. In the case of 'Akiyo', the face becomes recognizable but still degraded as the bit rate is still extremely low and not enough bandwidth is available to generate any significant extra quality. The same can be said for 'Hallmonitor', with the person inside the area of interest being visible and actions performed clear, but outside the area the scene is merely a blur.

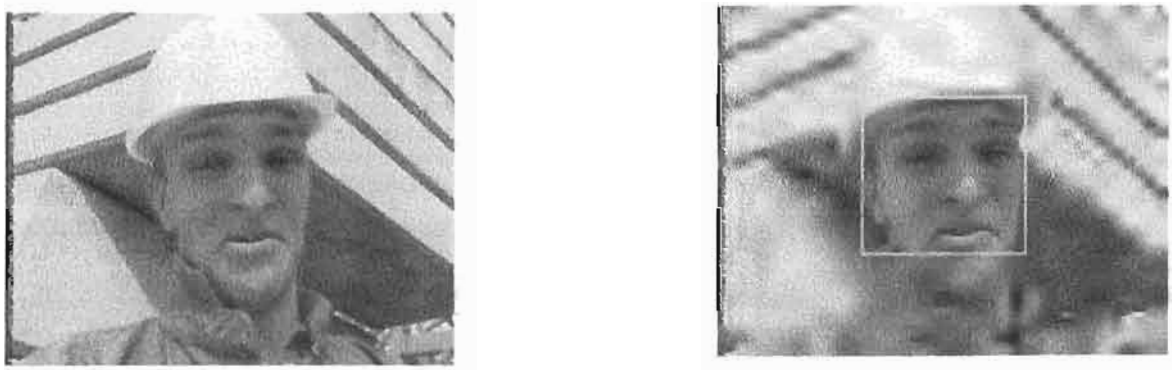
So the intra-frame scheme is able to comparably perform with the temporal schemes only in scenes of high movement and with higher bit rates. When the bit rates become too low the use of temporal compression adds an extra performance dynamic which the spatial scheme cannot meet. However with the introduction of the area of interest, the very degraded intra-frame video compression is able to show useful video results as demonstrated in Figure 5.12 and Figure 5.13. This highlights the usefulness of applying some application specific algorithms to try and improve performance in bandwidth starved environments.



'Foreman' Frame 53 at 50.30 kbps and 7.5 fps
Y PSNR = 26.75 dB

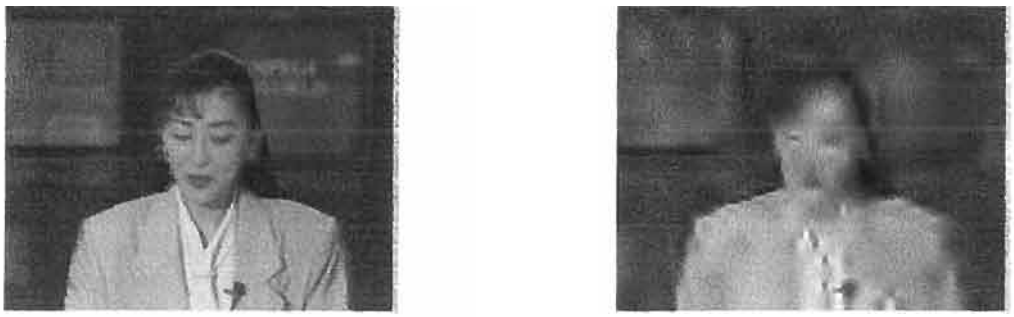


'Foreman' Frame 23 at 123.7 kbps and 15 fps
Y PSNR = 27.04 dB

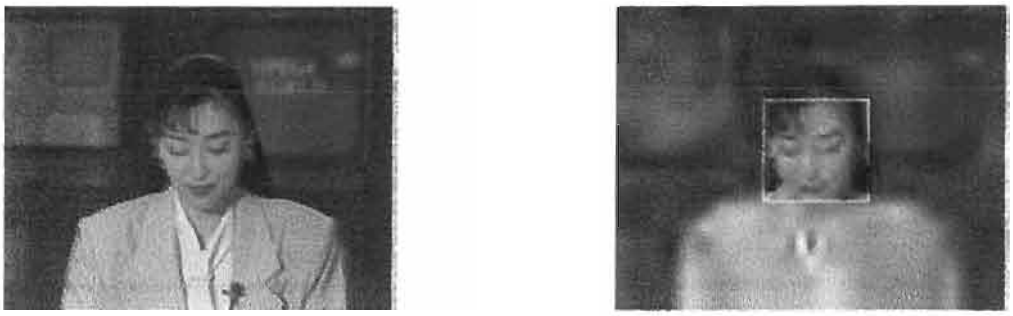


'Foreman' Frame 170 at 123.7 kbps and 15 fps
Box Y PSNR = 36.61 dB

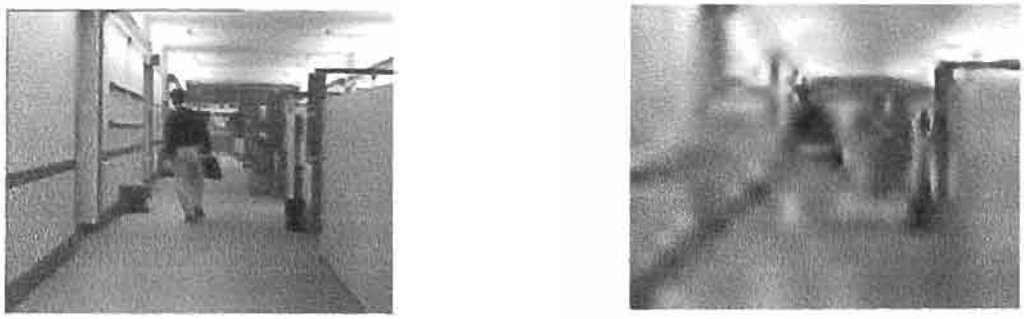
Figure 5.12: 'Foreman' Sequence at various rates



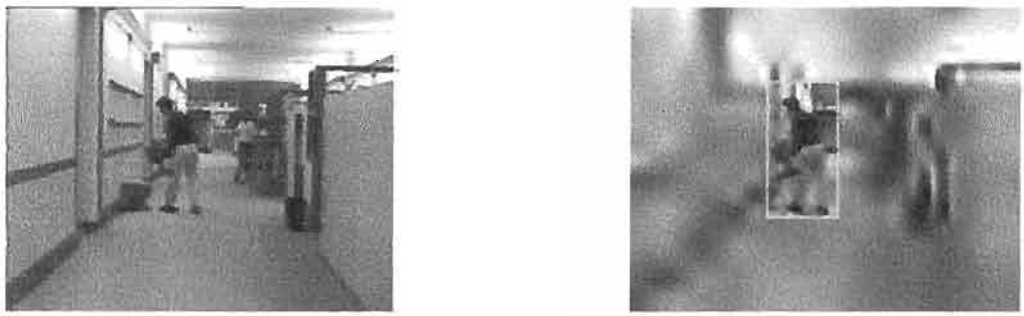
'Akiyo' Frame 80 at 24.15 kbps and 10fps
Y PSNR = 25.37 dB



'Akiyo' Frame 266 at 24.15 kbps and 10fps
Box Y PSNR = 26.42 dB



'HallMonitor' Frame 71 at 25.9 kbps and 10fps
Y PSNR = 21.23 dB



'HallMonitor' Frame 94 at 25.9 kbps and 10fps
Box Y PSNR = 22.34 dB

Figure 5.13: 'Akiyo' and 'Hallmonitor' at various rates

5.5 Summary

This chapter presented an evaluation of the performance of the implemented system. Some initial tests conducted illustrated the scalability of the scheme and showed results when adjusting the various parameters of the scheme. Compression times showed that for SQCIF and QCIF resolutions, reasonable frame rates are achieved while at CIF some further code optimisation was identified. The user-interface allows the user to adjust a great deal of parameters and thus test a multitude of different compression combinations. This diversity meant that evaluation needed to be limited to showing the performance of relevant areas. Thus the evaluation limited itself to only low bit rate concerns as, this is the focus of the dissertation. It was shown that in very low bit rate situations the introduction of the area of interest allowed for a useable video sequence where certain important areas of the video sequence were given extra quality at the expense of the rest of the scene. The important result here, is that the scheme is able to produce distinguishable video while limiting the number of applications that would find it useful. Comparison with some other intra-frame based schemes showed favourable results with the scheme easily outperforming MJPEG. In terms of temporal based schemes, the codec was not able to compare as favourably, which is expected. However in sequences with high motion the affects of temporal compression are somewhat dulled and the codec is able to make up some ground. With the introduction of the area of interest, it was demonstrated that very usable video could be generated at these low bit rates while limiting the applications where this video could be useful.

Chapter 6 Recommendations for Future Work

The scope for future work using this scheme is relatively vast. Basically what has been developed is a testing platform for wavelet based video compression schemes. The code has been written in a modular fashion allowing code blocks to be added and removed at will. The user interface software allows seamless communication with the DSP device for testing compiled code and adjusting parameters. So the focus here is to present some recommendations to make the current system better.

As mentioned throughout the dissertation the need for code optimisation is paramount. Although in some cases the code was optimised somewhat, further optimisation is required to make the scheme more useable at higher spatial resolutions and extend its use to higher bit rate applications. Since the SPIHT and Wavelet Transform implementations are the areas that require most processing time the optimisation of these code portions are required. In the SPIHT algorithm the searching of trees for significance occupies a major potion of processing time and so optimising this search process would go a long way in improving execution times.

The introduction of temporal compression schemes, such as motion compensation/estimation, would improve on compression results while making the area of interest implementation slightly more complex. However, introducing motion compensation/estimation presents some extra problems. Firstly, the processing time for the video compression will increase, as motion compensation/estimation is a computationally expensive task. Therefore, before such a scheme can be introduced some code optimisation to speed up the existing system's execution times is required. Secondly, when considering block-based motion compensation the resulting block artefacts introduced will decrease the performance of the wavelet transform, as the transform will assign higher priority to the high frequency components introduced at the edge of the block artefacts. Hence some form of overlapped block motion compensation is required to suppress the block artefacts, which adds to the processing time.

On the area of interest side, the selection of the area of interest could be complemented with some form of tracking. Thus, once the area of interest has been defined on an object, this object could be automatically tracked and the area of interest moved along with it. A further extension is to define arbitrarily shaped areas of interest and not just a box shape. The usefulness here is to define a face as the area of interest, which is shaped arbitrarily and then track the face over the sequence of video frames while applying the quality constraints on the scene.

Finally, some work has been performed on 3-D based schemes [Kim00]. A future system could implement a 3-D wavelet and SPIHT and then apply the area of interest concept on this scheme. Although, a promising idea, it is felt that this implementation may suffer somewhat due to the extra processing requirements of 3-D schemes and as such may not meet a real-time realisation, until significantly faster hardware becomes available.

Chapter 7 Conclusions

This project began life as an ill-defined requirement to illustrate how application specific characteristics could be used to improve compression ratios in video compression techniques. A particular goal was to target the transport of usable video streams over a low bandwidth (in the order of 16kb/s) data link. The application and the techniques were left unspecified as a subject for research. The visualised application that was settled on, was one where it could generally be said that the reproduced quality of the entire scene was of lower importance than that of some specific, user definable area. Scenes such as monitoring personnel in a largely static background and tracking vehicles in a fixed landscape were considered to be typical such applications. Thus, the focus of the project was to determine techniques for providing differential quality video transmission and high compression ratios. The final implementation used spatial compression only, in the form the Discrete Wavelet Transform with a dual area approach being used to provide the differential quality *'user-box'* where the user can dynamically alter the region of interest and the reproduction quality of both the background and the *'user-box'*.

The motivation for such a project was found in the ever increasing number of important applications for digital video and how the world is affected by the increasing use of the media. The meteoric rise of multimedia services, of which digital video plays a major role, has been pointed out and the problem of large resource requirements highlighted. The need for the compression of video to meet the demands of the digital age has been explained and some insight into the fundamentals of compression has been explained.

With the fundamentals defined a look into the principles of video generation, with a bias towards digital video, was deemed necessary to better understand the media. This short discussion focussed on familiarising the reader with digital video and the terminology and concepts used to compress the video. Such a discussion on video generation then lead to an explanation of important concepts used specifically for video compression. This discussion included the important spatial transforms, such as DCT and Wavelet, and compared advantages and disadvantages of some spatial compression techniques used in video. Although no temporal compression was implemented, an analysis of video compression was deemed incomplete without some explanation of temporal techniques and, as such, some basic temporal compression theory was introduced. Once the fundamentals of video compression had been explained the next logical step was to introduce some existing schemes and algorithms. This included an explanation of existing standards and a discussion of the latest advances in the field.

The theoretical background provided a decent grounding from which the design of the implemented system could commence. Firstly, it was deemed prudent to give a brief overview of current video compression technology as presented in Chapter two. From this taxonomy, the explanation of various design choices could more readily be supported. The implemented system used a wavelet transform as it was considered the optimal transform for a spatial based scheme. The scheme was aimed at low bit rate applications and so some degradation of the video was anticipated. Thus a compression system which could provide a more pleasing form of degradation was desirable. It was shown that the wavelet transform resulted in a more pleasing form of degradation than the 'blocky' effects of the DCT and thus it was considered the method of choice.

Once chosen, the implementation of the transform became important as, a fast implementation time was required. It was shown that the *lifting* scheme resulted in a very fast execution time and thus was used for

the wavelet transform. In compressing the wavelet transform coefficients, a compression scheme which used the properties of the wavelet transform was desired. The SPIHT scheme provided good compression with reasonable complexity and execution time and was therefore chosen for implementation of the compression system.

The goal of the compression scheme was to use an application specific idea to create a usable video compression scheme for low bit rates. The scheme was a purely spatial-based one, where temporal compression was not used. This limited the achievable compression ratios somewhat, but the idea of restricting the scheme to certain classes of application meant that some extra information could be used to further compress the data and still result in useful video. The compression scheme implemented a combination of techniques described in chapter two and added a quality box feature. This feature allowed for a user-definable area in the video scene of extra quality at the expense of the rest of the scene. A user-interface was described which allowed for adjustment of various compression parameters and display of original and decompressed video, while also showing important compression results.

With the system defined, implementation could begin. The hardware used for the implementation was of vital import and so some description of this hardware was given with a discussion on those aspects that were deemed most relevant to the system. The design of the software code was presented to illustrate how certain features and functionalities were achieved. Preliminary optimisations were presented which allowed for a near real-time implementation to be achieved. The communication and control of the DSP through the PC user-interface was discussed, showing how the system integrated software interface issues with hardware implementations.

Having implemented the design, an evaluation of the performance was given to test its successfulness. Results were collected from preliminary testing where the system was set up with a camera and video streamed. Various parameters of the system were adjusted and PSNR and subjective results gathered. These initial evaluations highlighted the system performance over low bit rates (16 – 128 kbps) and over various spatial resolutions. The frame rates were kept at a constant five frames per second to meet with accepted low bit rate frame rates. These initial tests showed that the system performed reasonably well for SQCIF and QCIF resolutions over most bit rates with quality degrading greatly at rates below 32kbps. In terms of CIF, the extra image resolution meant that the image was far more degraded than at other resolutions. The addition of the quality box showed that at low bit rates, while the rest of the image was of poor quality, the region inside the area of interest retained acceptable image quality. This result demonstrated the value of an area of interest concept as it showed that for applications that require only certain areas to be of good quality, such a scheme is useful.

Other features of the scheme were then adjusted for evaluation. The effect of adjusting the wavelet iterations was shown to adversely affect the image quality without providing any meaningful gain in processing time. This scheme was aimed at low bit rate concerns, but the scalability of the designed system meant that just about any compression ratio could be set for any bit rate requirement. With this in mind, it was deemed useful to test the system for the compression ratio when visible image degradation was noticed. Although a subjective test, it was still useful for a measure of performance when bit rate is not an issue. This testing showed that the compression ratio could be adjusted to 10:1 for greyscale before the image appeared to degrade. The operation of the serial port was tested and verified to function correctly. Additional serial port testing was deemed meaningless as the goal of adding the serial port was

to test functionality of the system over a real communications system and any other evaluation could easily be achieved without the serial port.

Finally, the system was compared to current spatial and temporal schemes such as MJPEG and MPEG-4. This comparison showed that the system was able to perform comparably well against other spatial-based schemes but not able to compare with temporal-based schemes. However, it was shown that with the addition of the area of interest feature, the implemented scheme was able to generate a useful video stream at similar bit rates of the temporal schemes. Thus, while the system was not able to perform as well as temporal based schemes, the addition of the quality box was shown to be valuable for providing useful information in certain areas of the video scene, adding value to the video for certain applications.

While the initial goal of the dissertation has been reached the design of the system has yielded a flexible software implementation where many features could be added and removed for testing of various video requirements, making the system a useful starting point for future work. Some recommendations for future additions and implementation ideas have been presented focussing on optimisation and additions to be made to improve the current system. Such schemes as 3-D video processing are recommended useful for future consideration and could be implemented using the designed schemes flexible base.

To conclude, this dissertation has presented a theoretical background into video compression before presenting an application specific, wavelet based video compression scheme to be implemented. The implementation issues are discussed before some meaningful performance analysis shows the usefulness of the area of interest idea for low bit rates and the flexibility of the implemented system. This flexibility has then prompted some recommendations for extensions on the current work.

Appendix A Colour Spaces

A.1 The Meaning of Colour Sub-Sampling Notation [Silb00]

The ratios 4:2:2 and 4:2:0 are common terms in expressing the sub-sampling of colour formats in todays digital video environment. But what do the numbers mean and where to they come from?

The notation comes from the digital sampling rates of a single-channel, composite signal as in NTSC and PAL, where sampling is synchronised at either 2, 3 or 4 times the sub-carrier frequency. The shorthand for these rates is 2fsc, 3fsc and 4fsc respectively. In three-channel component systems, the sampling shorthand becomes a ratio where the first number refers to the luminance signal and the other two the red and blue chrominance signals. So the 4:2:2 notation refers to a luminance component sampled at 4fsc (14.3 MHz for NTSC) and the two chrominance signals at 2fsc (half the luminance).

However, with digital video, sampling schemes based on multiples of NTSC or PAL sub-carrier frequencies have been abandoned in favour of a single sampling standard for both 525- and 625-line component systems, with the shorthand of 4:2:2 remaining as a legacy. Currently the '4' usually refers to the internationally accepted sampling rate of 13.5MHz and the other numbers the corresponding ratios. So a 4:1:1 system describes a signal where luminance is sampled at 13.5MHz and the chrominance at 3.375MHz. However the 4:2:0 notation seems to imply that there is no blue chrominance component which is not the case. In discussions for the MPEG standard it was decided to introduce the 4:2:0 scheme to provide a more averaged colour component by describing 2 colour components (one read, one blue) for each four luminance samples. Unlike 4:1:1, the chrominance samples don't come from the same line as the luminance samples, but are averaged from two adjacent lines in the field.

A.2 Colour Sub-Sampling Transforms [Bhas97]

A.2.1 YC_BC_R

Given the RGB signal the conversion the YC_BC_R is as follows:

$$\begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{A.1}$$

The inverse is:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.0 & 1.4021 \\ 1 & -0.3441 & -0.7142 \\ 1 & 1.7718 & 0.0 \end{bmatrix} \begin{bmatrix} Y \\ C_B \\ C_R \end{bmatrix} \tag{A.2}$$

A.2.2 YUV

YUV is the common transform used in PAL TV systems and is converted from RGB as follows:

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.144 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{A.3}$$

The inverse is computed as:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.0 & 1.140 \\ 1 & -0.395 & -0.581 \\ 1 & 2.032 & 0.0 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix} \tag{A.4}$$

A.2.3 CIELAB

The Commission Internationale de l'Eclairage (CIE) proposed its colour system called CIEL$^*a^*b^*$ or CIELAB. This system is contains two processes: 1) a linear transformation from RGB to CIE XYZ and 2) a non-linear transformation from CIE XYZ to CIELAB. In this system the RGB values must be defined between [0,100] as per ITU-R Recommendation BT.709, then the transform from RGB to CIE XYZ is given as below:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.35758 & 0.180423 \\ 0.212671 & 0.71516 & 0.0721169 \\ 0.019334 & 0.11919 & 0.950227 \end{bmatrix} \begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} \tag{A.5}$$

Now the conversion from CIE XYZ to CIELAB requires the XYZ data to be expressed in terms of a reference. In this case let X_n, Y_n and Z_n be XYZ values for reference white. Then:

$$
\begin{aligned}
L^* &= 116 f\left(\frac{Y}{Y_n}\right) - 16 \\
a^* &= 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right] \\
b^* &= 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right] \\
f(r) &= \begin{cases} r^{\frac{1}{3}} & if\ r > 0.008856 \\ 7.7867r + \dfrac{16}{116} & otherwise \end{cases}
\end{aligned}
\tag{A.6}
$$

The inverse of the above is given below:

If $L^* < 8.0$, then:

$$
\frac{Y}{Y_n} = \frac{L^*}{903.3} \tag{A.7}
$$

Else

$$
\frac{Y}{Y_n} = \frac{1}{100}\left[\frac{L^* + 16}{25}\right]^3 \tag{A.8}
$$

Once Y is obtained,

$$
\begin{aligned}
\frac{X}{X_n} &= \left[\frac{a^*}{500} + \left(\frac{Y}{Y_n}\right)^{\frac{1}{3}}\right]^3 \\
\frac{Z}{Z_n} &= \left[\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - \frac{b^*}{200}\right]^3
\end{aligned}
\tag{A.9}
$$

Then the inverse linear transform is as:

$$\begin{bmatrix} R_{709} \\ G_{709} \\ B_{709} \end{bmatrix} = \begin{bmatrix} 3.240479 & -1.53715 & -0.498535 \\ -0.969256 & 1.875991 & 0.041556 \\ 0.055648 & -0.20404 & 1.057311 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (A.10)$$

Appendix B Search Algorithms for Motion Compensation/Estimation

This appendix gives an overview of common block searching techniques used in temporal compression of video. The algorithms are explained and the advantages of each given. To limit processing the search for a matching block is usually limited to a search area which is user-defined. Figure B.1 shows the basic search principle employed in motion compensation/estimation. Once the block is chosen a search begins for the best match using the MAE criteria. Generally a search area is employed to limit the processing requirements of the search. This area is defined by considering the maximum amount of movement expected in the video sequence. In general, for TV this is usually with p = 15 and for high motion sequences p = 63. The rest of this appendix briefly describes some of the popular search methods used to find the best matching block.

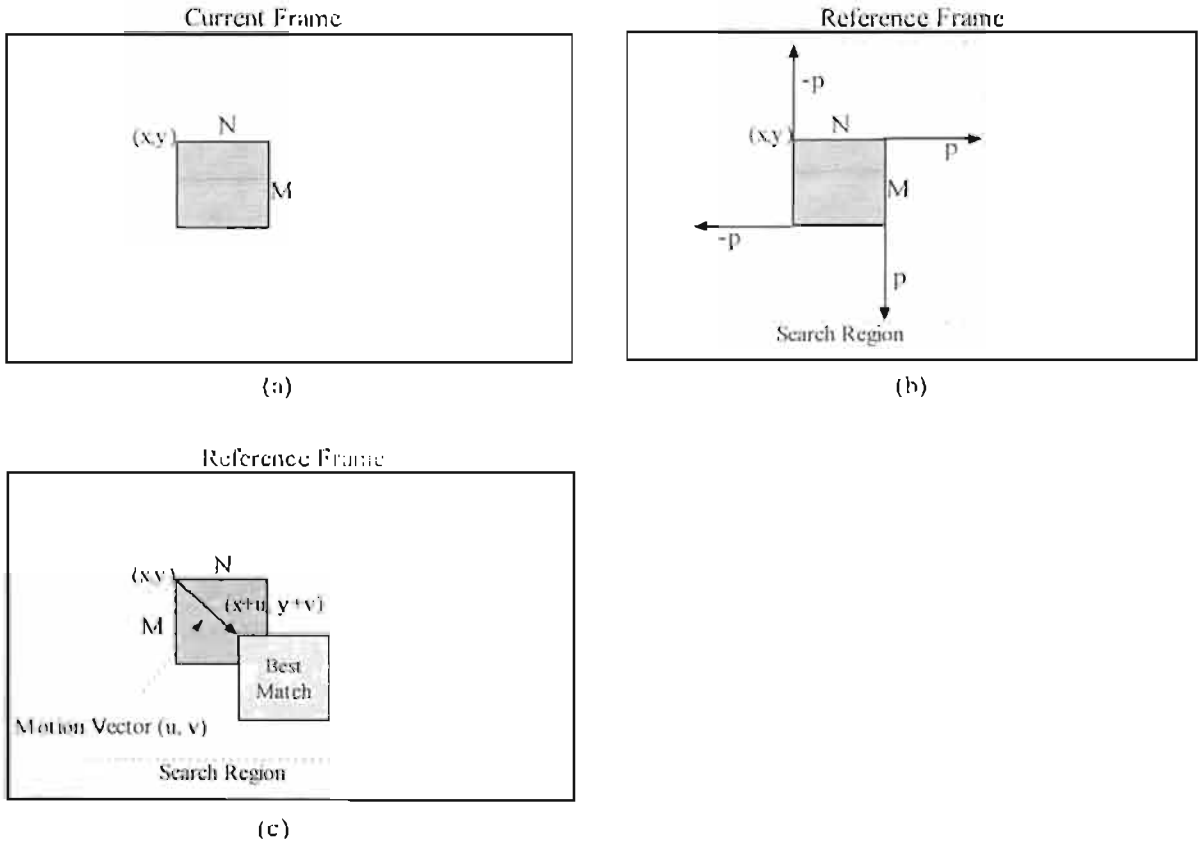


Figure B.1: Motion Estimation [Bhas97]

B.1 Full-Search

As the name suggests the full search algorithm searches the entire window area pixel-by-pixel for a matching block. This brute-force approach requires a great deal of computational time while offering a guaranteed best match block in return. Due to the excessive resources required this method of searching is not often employed.

B.2 2-D Logarithmic Search

In 1981 Jain et al.[Jain81] presented a search algorithm, now known as the 2-D logarithmic search, which provides good results at less computation expense. The 2-D logarithmic search is an iterative system very similar to a binary search. The first step is to define the search region [-p, p] (notation as used in Figure B.1). The region is then divided into an inner and outer area, the inner area being [-p/2, p/2] (illustrated in Figure B.2). Only the major 8 points of the inner region are searched; (0,0), (0,d_1), (0,-d_1), (-d_1,0), (d_1,0), (d_1,d_1), (d_1,-d_1), (-d_1,d_1) and (-d_1,-d_1). Where d_1 is given as in (B.1).

$$d_1 = 2^{k-1}$$
$$k = \lceil \log_2 p \rceil \qquad \text{(B.1)}$$

When the best match is found the algorithm redefines the search area centred around the coordinates of the match. It then continues the searching this region at its' 8 major points defined by $d_2 = d_1/2$. The search iterates until the best match is found or the search region becomes 0 (in that case the last found block was the best match).

This iterative scheme executes faster than the Full-Search scheme but does not guarantee finding the optimal match and can introduce distortion.

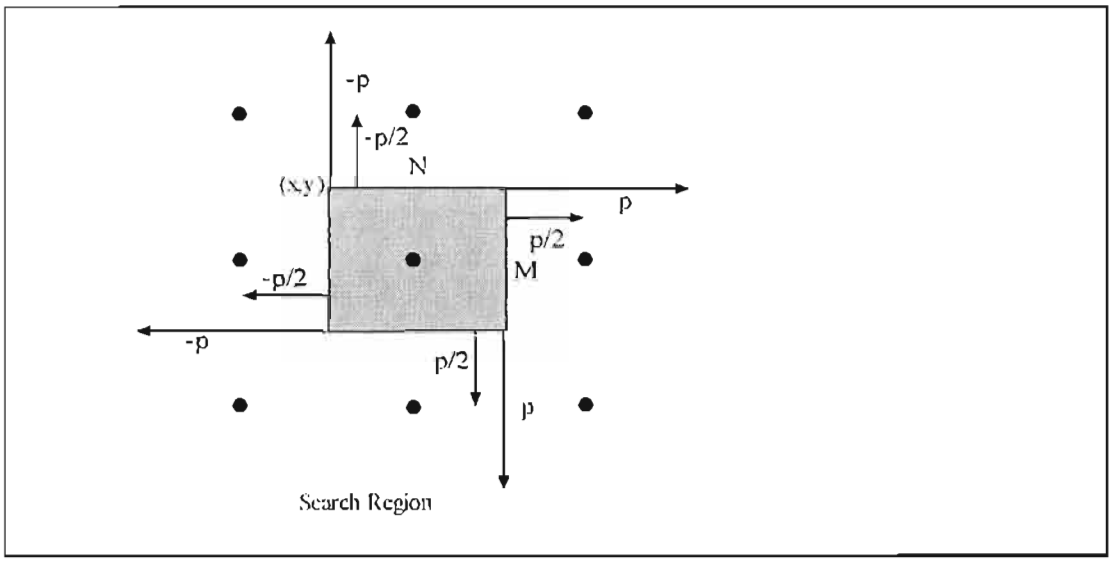


Figure B.2: Example of Logarithmic Search

B.3 Hierarchical Search

Hierarchical block matching seeks to take advantage of larger macro-blocks for searching. As the video frame is divided into macro-blocks, the more macro-blocks the more searching that is required, however too few macro-blocks and the effect of motion compensation becomes nullified. But the smaller the macro-block the better the motion compensation and the higher the computational expense.

In hierarchical block matching, the frame is first divided into large macro-blocks and then filtered to remove fine detail information. This fuzzy version of the block is used to arrive at the first approximation of motion. This approximation can then be successively refined by searching smaller and smaller regions within the approximations.

This technique is useful as it first establishes a general trend of motion and then is able to refine to better estimates. This iterative process has proven quite successful in providing better motion estimates than other sub-optimal schemes like the 2-D logarithmic search .

B.4 The Three Step Search

The Three Step Search (TSS), presented by Koga et al [Koga81], is popular due to its simplicity, robustness and reasonable estimation performance. This algorithm searches for the best motion vectors in a coarse to fine search pattern.. As its' name suggests, it uses three steps to achieve its goal. Step one is to select an initial step size and pick the eight blocks at a distance of the step size from the centre for comparison. Step two, chooses the best matched block from step one and then halves the step size. Step three is to repeat steps one and two. This algorithm is very similar to the 2-D logarithmic search as described earlier, however, the price for simplicity and fast execution is that the TSS uses a uniformly allocated checking pattern in its' first step and thus is ill-suited for small motion estimation.

B.5 Sub-Pixel Schemes

The search algorithms presented are limited to the integer pixel grid in order to make their estimate. This does not model true motion effectively as motion often does not follow a grid-like pattern. To increase the accuracy of the search, schemes which extend the search to sub-pixel domains can be used. The advantage is a better motion estimate with the disadvantage of increased execution time. The most common of these schemes is half-pel, Figure B.3, accurate motion estimation. In this scheme a motion vector with half-pel accuracy is measured using interpolation between neighbouring pixels. From here the matching criterion is applied to determine the best match. The results from using this scheme show a better estimate of the motion and the H.263 standard employs this method in advanced mode for a better motion estimate. However the trade-off is an increase in execution time.

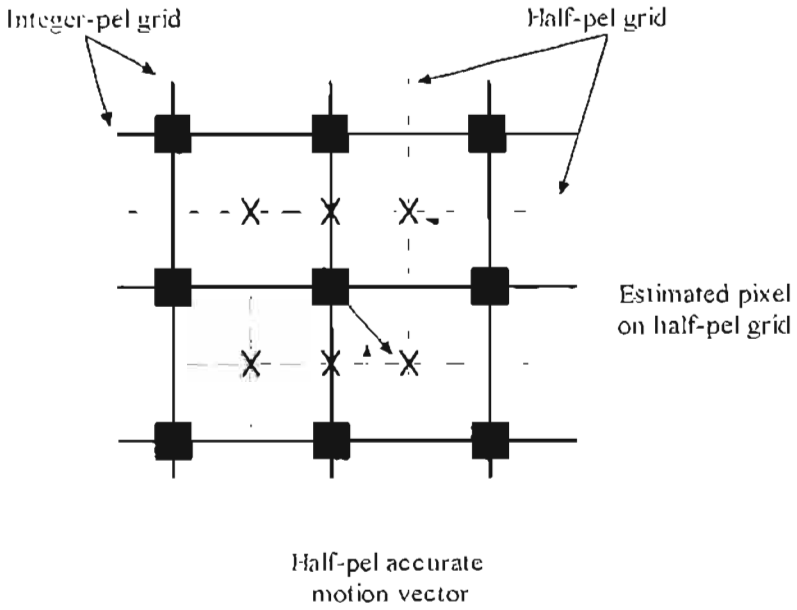


Figure B.3: Half-pel Motion estimation [Bhas97]

References

[Anto92] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, *"Image Coding Using Wavelet Transform"*, IEEE Trans. On Image Proc., Vol. 1, No. 2, pp.205-220, April 1992.

[Bhas97] V. Bhaskaran, K. Konstantinides. *Image and Video Compression Standards – Algorithms and Architectures*, 2nd Edition, Kluwer Publishers, 1997

[Bris96] C.M. Brislawn, *"Classification of Nonexpansive Symmetric Extensions Transforms for Multirate Filter Banks"*, Applied and Computational Harmonic Analysis, Vol. 3, pp. 337-357, 1996

[Brya95] J. Bryan, *Compression ScoreCard*, http://www.byte.com/art/9505/sec10/art5.htm, May 1995, Byte

[Chan95] Y.T. Chan, *"Wavelet Basics"*, Kluwer Academic, 1995

[Chen89] C-T Chen, *"Adaptive Transform Coding Via Quadtree-Based Variable Blocksize DCT"*, International Conference on ASSP, pp. 1854-1857, 1989

[Cho91] N.I. Cho, S.U. Lee, *"Fast Algorithms and Implementation of 2 – D Discrete Cosine Transform"*, IEEE Transactions on Circuits and Systems, vol. 38(3), pp. 297-305, Mar. 1991.

[Chui92] C.K. Chui, *"An Introduction to Wavelets"*, Academic Press, 1992

[Ckim00] C. Kim, J-N. Hwang, *"An Integrated Scheme for Object-based Video Abstraction"*, Proceedings of ACM, pp. 303-311, 2000

[Cohe92] A. Cohen, I. Daubechies, J.C. Feauveau, *"Biorthogonal Bases of Compactly Supported Wavelets"*, Comm. On Pure and Applied Math., Vol. 45, pp. 485-560, 1992

[Dang95] V-N. Dang, A-R. Mansouri, J. Konrad, *"Motion Estimation for Region-Based Video Coding"*, Proc. Of the IEEE. I.C.I.P., pp. 189-192, 1995

[Daub88] I. Daubechies, *"Orthonormal Bases of Compactly Supported Wavelets"*, Comm. On Pure and Applied Math., Vol. 41, pp.909-996, 1988

[Daub98] I. Daubechies, W. Sweldens, *"Factoring Wavelet Transforms into Lifting Steps"*, J. Fourier Anal. Appl., Vol. 4, No. 3, pp.247-269, 1998

[Elli82] D.F. Elliot, K.R. Rao, *"Fast Transforms: Algorithms, Analysis, Applications"*, Academic Press, 1982

[Esta77] D. Estaban and C. Galand, *"Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes"*, Proc. ICASSP, pp. 191-195, May 1977.

[Fall73] N. Faller, *"An Adaptive System for Data Compression"*, Record of the 7th Asilomar Conf. on Circuits, Systems and Computers, pp. 593-597, November 1973

[Farv97] N. Farvardin, H. Jafarkhani, J. Johnson, R. Bhattacharya, *"Real-Time Wavelet-Based Video Compression System Using Multiple TMS320C40s"*, Texas Instruments Application Note, 1997

[Fowl95] J.E. Fowler, K.C. Adkins, S.B. Bibyk, S.C. Ahalt, *"Real-Time Video Compression Using Differential Vector Quantization"*, IEEE Transactions on Circuits and Systems for Video Technology. Vol. 5, pp. 14-24, 1995

[Gall78] R. G. Gallager, *"Variations on a Theme by Huffman"*. IEEE Trans. Inform. Theory Vol. 24, pp. 668-674, November 1978

[Gall91] D. Le Gall, *"MPEG: A Video Compression Standard for Multimedia Applications"*, Communications of ACM, Vol. 34, No. 4, pp 46-58, April 1991

[H.263] Telenor White Paper on H.263, http://www.fou.telenor.no/brukere/DVC/h263_wht/h263wht.html

[Hech87] E. Hecht, *"Optics"*, Second Edition, Addison-Wesley, 1987

[Hoan01] D.T. Hoang, J.S. Vitter, *"Efficient Algorithms for MPEG Video Compression"*, John Wiley & Sons, New York, NY, to appear 2001.

[Huff52] D. Huffman, *"A Method for the Construction of Minimum-Redundancy Codes"*, Proceedings of the I.R.E. pp. 1098-1101, September 1952

[Image] Image Power Inc. http://www.imagepower.com

[ITU601] International Telecommunications Union http://www.itu.int

[Jacq93] A.E. Jacquin, *"Fractal Image Coding: A Review"*, Proceedings of the IEEE, vol. 81, No. 10, October 1993

[Jain81] J.R. Jain, A.K. Jain *"Displacement measurement and its application in interframe image coding"*, IEEE Transactions on Communications, Volume COM-29, Number 12, p 1799 - 1808, December 1981

[JPEG] JPEG compression software, http://www.webattack.com/Freeware/gmm/fwgcomp.shtml

[JPEG2000] JPEG 2000 WebSite, http://www.jpeg.org/JPEG2000.htm

[Kama82] F.A. Kamangar, K.R. Rao, *"Fast Algorithms for the 2 – D Discrete Cosine Transform"*, IEEE Transactions on Computers, vol. C – 31(9), pp.899 – 906, Sept. 1982.

[Knut85] D. E. Knuth, *"Dynamic Huffman Coding"*, J. Algorithms, Vol. 6, pp.163-180, June 1985

[Koen99] R. Koenen, *"MPEG-4: Multimedia for our Time"*, IEEE Spectrum, pp.26-33, February 1999

[Koga81] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, T. Ishiguro, *"Motion-compensated interframe coding for video conferencing"*, Proceedings NTC'81 (IEEE).

[Main] MainConcept GmbH, http://www.mainconcept.com/products.shtml

[Mall89] S. Mallat, *"A Theory in Multiresolution Signal Decomposition: The Wavelet Representation"*, IEEE Trans. Pattern Anal. Machine Intell., Vol 11, pp. 674-693, July 1989

[Marp99] D. Marpe, H.L. Cycon, *"Very Low Bit Rate Video Coding Using Wavelet-Based Techniques"*, IEEE Transactions on Circuits and Systems for Video Technology, Vol.9, No.1, pp. 85-94, 1999.

[Mart97] S.A. Martucci, I. Sodagar, T. Chiang, Y-Q. Zhang, *"A Zerotree Wavelet Video Coder"*, IEEE Transactions on CSVT, Vol. 7, No. 1, February 1997

[Miya00] H. Miyazawa, *"H.263 Encoder: TMS320C6000 Implementation"*, Texas Instruments Application Note, December 2000

[MPEG4] Overview of MPEG-4 Standard, http://drogo.cselt.stet.it/mpeg/#The_MPEG_standards

[Nich96] J.A. Nicholls, D.M. Monro, *"Scalable Video by Software"*, Proceedings of ICASSP, 1996

[Ogun00] P. Ogunbona, *"JPEG2000 - The New Wave in Image Compression"*, International Magazine for Closed Circuit Television, pp. 36- 38, Jan. 2000

[Okub95] S. Okubo, K. McCann, A. Lippmann, *"MPEG-2 Requirements, Profiles and Performance Verification – Framework for developing a Generic Video Coding Standard"*, Signal Processing Image Communication, pp. 201-9, July 1995

[Papa92] M. Ali, C. Papadopolous, T.G. Clarkson, "*The Use of Fractal Theory in a Video Compression System*", IEEE Data Compression Conference (DCC'92), 24-27, March 1992

[Poyn95] C. A. Poynton. "*Frequently asked Questions about Colour*", 1995, ftp://ftp.inforamp.net/pub/users/poynton/doc/colour

[Riou92] O. Rioul, P. Duhamel, "*Fast Algorithms for Discrete and Continuous Wavelet Transforms*", IEEE Trans. Inform. Theory, Vol. 38, No. 2, pp.569-586, 1992

[Robi97] J.A. Robinson, "*Efficient General-Purpose Image Compression with Binary Tree Predictive Coding*", IEEE Transactions on Image Processing, vol 6, no. 4, April 1997

[Said96] A. Said, W. Pearlman. "*A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees*", IEEE Transactions on CSVT, Vol. 6, June 1996.

[Shap93] J.M. Shapiro, "*Embedded Coding Using Zerotrees of Wavelet Coefficeints*", IEEE Transactions on Signal Processing, Vol. 41, pp. 3445-3462, Dec. 1993

[Silb00] M. Silbergleid, "*A Guide to Digital Television*", 3rd Edition, Chapel Hill NC, 2000.

[Skro00] A.N. Skodras, C.A. Christopoulos, T. Ebrahimi, "*JPEG2000: The Upcoming Still Image Compression Standard*", Proceedings of the 11th Portuguese Conference on Pattern Recognition, pp. 359-366, May 2000

[Sola97] S.J. Solari, "*Digital Video and Audio Compression*", McGraw-Hill, 1997

[Swel98] W. Sweldens, "*The Lifting Scheme : A Construction of Second Generation Wavelets*", SIAM Journal on Mathematical Analysis, Vol. 29, No. 2, pp. 511-546, 1998.

[Syme01] P. Symes. *Video Compression Demystified*, McGraw-Hill, 2001

[Taub86] Taub, H., Schilling, D. L., *Principles of Communication Systems*, 2nd Ed., McGraw Hill, 1986

[Trimedia] Philips Trimedia Documentation Set, Philips Semiconductors, 1999

[Turl93] T. Turletti, "*H.261 Software Codec for Video Conferencing over the Internet*", Technical Report 1834, INRIA, January 1993.

[UEC] UEC Technologies (Pty) Ltd. http://www.uec.co.za

[Uys00] R.F. Uys, *"Parallel Implementation of Fractal Image Compression"*, Masters' Dissertation, University of Natal, 2000

[Uyt99] G. Uytterhoeven, *"Wavelets: Software and Applications"*, PhD. Thesis, KU Leuven, April 1999

[Vet92] M. Vetterli, C.Herley, *"Wavelets and Filter Banks: Theory and Design"*, IEEE Trans. On Signal Proc., Vol. 40, No. 9, pp. 2207-2232, Sep. 1992

[Vil95] J.D. Villasenor, B. Belzer, J. Liao, *"Wavelet Filter Evaluation for Image Compression"*, IEEE Transactions on Image Processing, Vol.2, pp. 1053-1060, August 1995.

[Vit87] J. S. Vitter, *"Design and Analysis of Dynamic Huffman Codes"*. Journal of Assoc. for Computer Machinery, Vol. 34, pp. 825-845, October 1987

[Wal91] G.K. Wallace, *"The JPEG Still Picture Compression Standard"*, Comm. Of the ACM, Vol. 34, No. 4, pp. 30-44, 1991

[Wat91] H. Watanabe, S. Singhal, *"Windowed Motion Compensation"*, Proc. SPIE Vol. 1605, pp. 582-589, Nov. 1991.

[Wee96] A.R. Weeks, *Fundamentals of Electronic Image Processing*, SPIE/IEEE Press, 1996

[Wel84] T.A. Welch, *"A Technique for High-Performance Data Compression"*, IEEE Computer, pp.8-19, June 1984

[Wen99] J. Wen, J. Villasenor, *"Robust Video Coding Algorithms and Systems"*, Proceedings of the IEEE, Vol. 87, No. 10, pp. 1724-1733, October 1999

[Wit87] I.H. Witten, R.M. Neal, J.G. Cleary, *"Arithmetic Coding For Data Compression"*, Communications of ACM, Vol. 30, No. 6, June 1987

[Wu97] X. Wu, *"Lossless Compression of Continuous-Tone Images via Context Selection, Quantization, and Modelling"*, IEEE Transactions on Image Processing, vol. 6, no. 5, May 1997

[Xio97] Z. Xiong, K. Ramchnadran, M.T. Orchard, *"Space-frequency Quantisation for Wavelet Image Coding"*, IEEE Transactions on Image Processing, Vol. 6, No. 5, pp. 677-693, 1997

[Ziv77] Ziv, J. and Lempel, A., *"A Universal Algorithm for Sequential Data Compression"*, IEEE Transactions on Information Theory, May 1977

[Ziv78] Ziv, J., and Lempel, A., *"Compression of individual sequences via variable-rate coding"*, IEEE Transactions on Information Theory, Vol. 24, pp. 30-536, 1978.