Fast Implementations of Block Motion Estimation Algorithms in Video Encoders

By

Naga Rohini Koduri

Supervisors:

Mqhele E. Dlodlo

Gerhard de Jager

Co-supervisor:

Keith L. Ferguson

This thesis is submitted in partial fulfillment of the academic requirements

for the degree of

Master of Science in Electrical Engineering

in the Faculty of Engineering and The Built Environment

University of Cape Town

2011

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

Name: Prof. Mqhele Dlodlo, UCT Associate Professor

Signed: _____

Date: _____

Name: Prof. Gerhard de Jager, UCT Emeritus Professor

Signed: _____

Date: _____

Declaration

I declare that this thesis is my own work. Where collaboration with other people has taken place, or material generated by other researchers is included, the parties and/or materials are indicated in the acknowledgements or are explicitly stated with references as appropriate.
This work is being submitted for the Master of Science in Electrical Engineering at the University of Cape Town. It has not been submitted to any other university for any other degree or examination.

---------- ----------

Name: Naga Rohini Koduri Date

Dedication

This work is dedicated to my dad C.Lakshmi Narasaiah

with deep gratitude for his unconditional love and sacrifice. You inspired a great number of students and me throughout your life.

I hope I made you a proud father. I did and will love you nanna with all my heart.

Abstract

The recent technological developments in telecommunications and Information Technology (IT) are due to convergence of different technologies. Multimedia data transfer has been of great interest and is a potential reason for the emergence of video and image coding in the mass market. A video can be represented as a series of snapshots/images. The strategic approach of dividing a snapshot into smaller sub-blocks and predicting the whole image from other similar images is the key to block motion estimation. Many fast Block Matching Algorithms (BMAs) have been proposed through time to quicken the coding process.

This research is aimed at designing and implementing novel fast algorithms for speeding up the encoding process. The objective of this study was to come up with algorithms which can estimate the data significantly faster than the existing algorithms, whilst ensuring acceptable video quality. The research proposed four new fast algorithms out of which two of them were single resolution algorithms and the other two were multi-resolution algorithms. Special focus was laid on the multi-resolution technique and its influence on the estimation process.

Simulation results reveal that the proposed non-hierarchical fast algorithms were up to 78% faster than the standard Full Search Algorithm (FSA) and were 33.5% faster than the existent algorithms, with no significant degradation in the video quality. The proposed multi-resolution combination algorithms were up to 75.8% faster than the FSA and were 17% faster than the prewritten hierarchical algorithm. The improvements in speed were achieved with either no significant change or impressively with a slight increase in video quality when compared to the project version of multi-resolution algorithm. The proposed fast algorithms proved more efficient for active video sequences (like sports sequences). Thus, mentioned qualities of the new algorithms make them potentially fit for applying in the real-time video compression environments.

One of the proposed algorithms, Small Diamond Hierarchal (SDH) search algorithm was successfully implemented in real-time scenario in the project undertaken by Council for Scientific and Industrial Research (CSIR), South Africa and proved to speed up the end to end compression process by 40% (compared to FSA). The success in faster compression of 'real' data not only answers the bandwidth limitation issues but also will take information access through the internet to higher levels where advertising and multimedia applications would be easier and profitable at the same time.

Acknowledgements

I would like to thank my supervisors, Professor Gerhard de Jager (UCT), Professor Mqhele. E. Dlodlo (UCT) and Dr. K. L. Ferguson (CSIR) for their guidance, encouragement and enthusiasm along the course of my Masters.

To my husband, Niranjan Koduri for his moral support and valuable advice throughout my course period without which I could have not accomplished anything I did. I would also like to thank my son Anish Koduri for, his smile kept me going. Gratitude extends to my mother, PhaniRekha for her constant support through the write up period. I also thank my in-laws and my sister for their help and advice.

I also thank my friends Guy-Alain Lusilao-Zodi and Charles Lubobya for the intellectual support. Thank you Ralf Globisch for the technical help you provided- much appreciated.

The Council for Scientific and Industrial Research (CSIR) has supported this work through the DST-Innovation Fund on project no.T70026: Bandwidth Adaptive real-Time Video Broadcasting over Internet.

Table of Contents

List of Figures

List of Tables

Glossary

Artifacts: Noticeable distortion caused in the original information due to applying compression techniques.

B-frame: Frame which is predicted bi-directionally based on the information in past and future frames.

Bits per pixel: Number of bits to represent the information in each pixel, represented by 'bpp' and it is harmonious to bit rate.

Bit rate: Rate of bits transmitted per second.

Chroma: Colour component in the media data (audio, video or image).

Frame: Still shot of a moving video at a particular instant. A set of frames form a complete video sequence.

Frame rate: Number of frames displayed per second to play a video or movie. It is denoted in frames per second or 'fps'.

Global minima: The minimum energy difference block in the entire frame.

Gnuplot: It is a command-line driven graphing utility for Linux.

Sub- pixel regions: Interpolated regions between pixel regions. Sub-pixel (half-pixel and quarter pixel) context is generally used applied in complex luma and chroma sampling.

I-frame: Frame which is intra frame compressed, a standalone video frame.

Local minima: The minimum energy difference block in a given confined area.

Lossless Compression: Compressing the media information without any information loss.

Lossy Compression: Compressing the media information with loss of information yet visually imperceptible.

Luma: Luminance component in the media data.

Macroblocks: An array of pixels.

Motion Compensation: Process of compensating the motion differences(for a block) between the current and reference frames.

Motion Estimation: Estimating the best match for a block in current frame in the reference frame.

Motion Range: Maximum range over which a motion vector can be placed.

Motion Vector: A two-dimensional vector pointing in the same direction of the motion (in this context, motion of the macroblock).

P-frame: Frame which is compressed using previous frames.

Pixel: It is a picture element, a single point in an image.

Prediction: Process of constructing a frame using other frames.

Residual: Difference in the current frame and reference frame.

Resolution: Number of pixels used to represent the data, resolution is directly related to quality.

Search Window: A confined area in a frame in which the search is executed.

YCbCr: Sampling pattern depicting luma (Y) and chroma components (blue and red).

List of Acronyms

Acronym	Expansion
2D	Two-Dimensional
3D	Three-Dimensional
3SS	Three-Step Search
AVC	Advanced Video Coding
BMA	Block Matching Algorithm
bpp	bits per pixel
CPU	Central Processing Unit
CSA	Cross Search Algorithm
DCT	Discrete Cosine Transform
DPCM	Differential Pulse Code Modulation
DSA	Diamond Search Algorithm
DVD	Digital Versatile Disc (or) Digital Video Disc
E3SS	Efficient Three-Step Search Algorithm
fps	frames per second
FSA	Full Search Algorithm
HS	Hierarchical Search Algorithm
HT	Integer Transform
H.26L	H.26x Long-term series
HDTv	High Definition Television
IDCT	Inverse Discrete Cosine Transforms
IDE	Integrated Development Environment

IT	Information Technology
ITU-T	International Telecommunication Union-Telecommunication Standardization Sector
JPEG	Joint Photographic Experts Group
JVT	Joint Video Team
LDS	Large Diamond Search Algorithm
LDSP	Large Diamond Search Pattern
LS	Logarithmic Search Algorithm
MAD	Mean Absolute Difference
MAE	Mean Absolute Error
MPEG	Moving Picture Expert Group
MSE	Mean Squared Error
NSS	Nearest Neighbour Search
OOP	Object Oriented Programming
OS	Operating Systems
PSNR	Peak Signal to Noise Ratio
PSTN	Public Switched Telephone Network
Qf	Quality factor
RAM	Random Access Memory
R-D	Rate-Distortion
RGB	Red-Green-Blue
RLE	Run-Length Encoding
SAD	Sum of Absolute Differences

SAE	Sum of Absolute Errors
SDH	Small Diamond Hierarchical Search Algorithm
SDS	Small Diamond Search Algorithm
SDSP	Small Diamond Search Pattern
TTHS	Two-Tier fast Hierarchical Search Algorithm
VC++	Visual C++
VLC	Variable Length Coding

Chapter 1

1 Introduction

The emerging technologies like video conferencing, digital television and internet streaming, video applications form a major share in present day broad casting and entertainment industries. Video compression has become an essential component in multimedia scenario. Handling video data efficiently through the storage and transmission process is challenging due to its bulky nature. A video compression system aims to reduce the amount of data required to store or transmit video whilst maintaining an acceptable video quality. With extensive and good compression techniques, it is possible to send a greater amount of multimedia data without using significant bandwidth. Entities like cheap but powerful processors, fast network access, the ubiquitous internet, large-scale research and standardisations have all contributed to the need for development of image and video coding and compression. Figure1.1 below illustrates a simple application, which involves capture, compression and transmission of audio/video data [1].

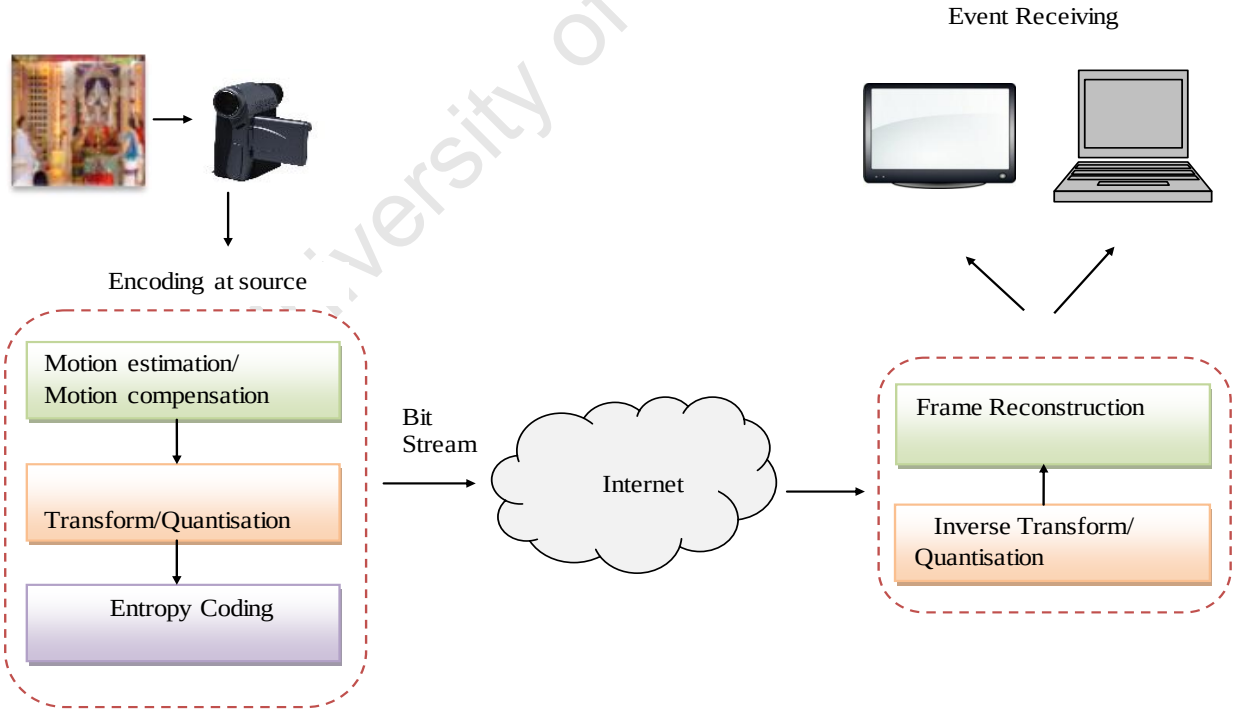


Figure 1.1: Transmission of audio/video data over the Internet

Modern data compression techniques offer the possibility to store or transmit vast amount of data that is necessary to represent digital images and video in an efficient and robust way. New audio-visual applications in the field of communication, multimedia and broadcasting became possible based on digital video coding technology. The importance of these techniques will become even more prominent in a world where there are productivity gains through digital communications.

Compressing video information is performed at several stages to effectively minimise the amount of data transmitted. Initially, video is transmitted as a progressive set of pictures, which constitute a video. The main idea in compression is to avoid transmitting the redundant data and send only the new or necessary information. This encoding process naturally calls for a technique that uses the existing data to represent a new set of data. This representation is achieved by the combination of estimation and compensation techniques. Coding of video sequences normally involves creating a model of the current data using already transmitted data. The motion estimation algorithms also known as Block matching Algorithms (BMAs) are central in the motion estimation. Each algorithm achieves a different compression rate depending on its total number of computations. This research studied various existing BMAs and eventually proposed new algorithms which, when simulated, proved faster than the similar algorithms that were previously incorporated in the project with no significant degradation in the video quality [1].

1.1 Problem Statement

The multimedia data has to be highly compressed for transmission and storage purposes. Bulky videos demand more buffering time thus interrupting smooth running of a video sequence In addition to the bulky nature of data, the bandwidth limitations make it a considerable issue.

With the multimedia data playing a principle part in various web applications, effective data compression techniques are crucial. In video compression most of the time is consumed due to the computations involved in predicting the current frame. Reducing the size of the video data at the prediction level might significantly improve the speeds of the whole process, thus, achieving faster estimation. Efficient compression will not only answer the prevailing bandwidth constraints but also achieves higher data rates and smoother transmission of the video data over

the internet. Fast compression techniques at the prediction level in the encoding process can prove beneficial for a wide range of video applications like real-time streaming, data transmission and mobile video applications [1].

1.2 Objectives of Research

To achieve greater speeds in video transmission applications, the video data needs to be compressed effectively. This research aims at speeding up the estimation process so that the total encoding-decoding time is reduced. As a result of applying fast implementations of estimation algorithms, the total time for compressing the video is also minimised. However, the quality constraints still need to be met for satisfactory resolution of the transmitted video.

The objectives of this research are to:

- Procure a new or modified version(s) of both non-hierarchical and hierarchical algorithms which reduces the block comparisons significantly, thus being computationally faster than the existing algorithms contained in this project.
- Measure the performance of the generated algorithms by measuring the time taken for the compression-decompression process and the Peak Signal to Noise Ratio (PSNR) for the quality assessment.
- Analyse the impact of applying hierarchy technique in motion estimation process.
- Propose algorithms with faster computation capability yet render acceptable video quality.

1.3 Scope of Research

The project covers research on different motion estimation algorithms in video encoding process. It focuses on reducing the total compression time by concentrating on the computational time attached to a BMA. The scope of the research includes:

- Developing new or modified algorithms better suited for the project.
- Assaying the computational results for different resolution or quality of the video.
- Considering parameters like PSNR, quality factor, bits per pixel (bpp) for every algorithm proposed.

However, some limitations which need to be acknowledged are:

- Video content compressed using a standard cannot be decompressed with a different standard which clearly indicates that the encoder and decoding should use the same algorithms. Nevertheless it is possible to implement many different algorithms in the same software or hardware, which would then enable multiple formats to be compressed.
- The implementation and testing of the proposed algorithms was done in H.263.When desired to implemented in newer standard, H.264, the algorithm might present itself naïve of the new features. This might call for slight modifications [3].

This project attempts to study only a few popular fast algorithms which were proven to render satisfactory output. The older and slower algorithms are likely to be excluded.

1.4 Thesis Outline

This rest of this thesis is organised as follows:

Chapter 2, sets a background for the research area and discuses different concepts that are related to the digital video and digital video compression. A Block diagram is illustrated and thoroughly explained. This chapter also contains information on the various video compression standards which prevail so far.

Chapter 3, focuses on estimation and prediction levels in video compression. Terms and concepts which relate to motion estimation are defined which would serve as a foundation to upcoming chapters. Some of the techniques that are adopted in this research are also stated.

Chapter, 4 is on motion estimation algorithms. This chapter explains algorithms that were already incorporated in the project and were used as metrics for the newly added algorithms. This chapter also details other algorithms whose modified versions have been proposed in the research.

Chapter 5 elucidates the four proposed algorithms in step by step manner with their pattern illustrations. It justifies the choosing of different search patterns and explains their effect on the search procedure.

Chapter 6 deals with the simulation environment wherein the software and hardware specifications of the research are stated. This chapter also deals with the parameters of interest and the method of analysing.

Chapter 7 presents the results and their analysis. Further, this chapter justifies various behaviours of the proposed algorithms. In addition, this chapter performs a comparative study on the proposed algorithms to the existing algorithms. Following the results, Chapter 8 concludes the thesis and comments on the implemented algorithms and their applications.

Chapter 2

2 Background

Considering the new technologies, South Africa has a greater need to find better ways to compress the data due to its bandwidth constraints. Faster and adept compression techniques will possibly fill the gap between user's demands and the limited capabilities of transmission networks and storage devices. The dawn of the multimedia era calls for new techniques to handle bulky multimedia data. Encoding and decoding segment of the digital video is an integral part of the business, education and entertainment. Digital video coding technology has developed into a mature field and a diversity of products have been developed, targeted for a wide range of emerging applications, such as video on demand, digital Tv/HDTv broadcasting, and multimedia image/video database services. In using a digital video, file size is an important concern because digital video files tend to take up a lot of storage space, and they have to be compressed [2], [4].

This part of the thesis provides some background information on a digitisation of video data in the multi-media era. This chapter intends to provide information on topics such as sampling and representation pertaining to the digital video thus strengthening the base concepts relating to this study.

2.1 Digital Video

Digital video is generally represented in a binary format, is a sequence of digital data rather than a continuous signal (as used in analog information). The concept of a digital video is essentially, a sampled two-dimensional (2D) version of a three-dimensional (3D) scene. Initially a real-time video is continuous both spatially (space wise) and temporally (time wise). Obtaining a digital version of a the video is done in two stages-firstly capturing; which involves denoting of a video in terms of a continuous electrical signal, and secondly, digitising which is sampling the electrical signal and converting each sample to set of numbers [4],[5].

2.1.1 Sampling a digital video

A motion video sequence is represented as a series of images captured in equidistant time intervals. Each image is a ‘frame’ and can be interpreted as group of ‘pixels’ which are the smallest addressable elements in an image. A video sequence is continuous in both time and space. Thus, to sample a video sequence, samples must be taken both temporally and spatially.

2.1.1.1 Temporal Sampling of a digital video

Temporal samples of a video can be obtained by taking snapshots of that video in regular intervals of time; a balance is made between the number of samples and quality of the reconstructed video at the destination. Normally frame rate for television applications is 20-30 frames per second (fps). At higher frame rates more information is captured which aids in better reconstruction of the frame at the receiving end [1]. Figure 2.1 shows the temporal sampling procedure.

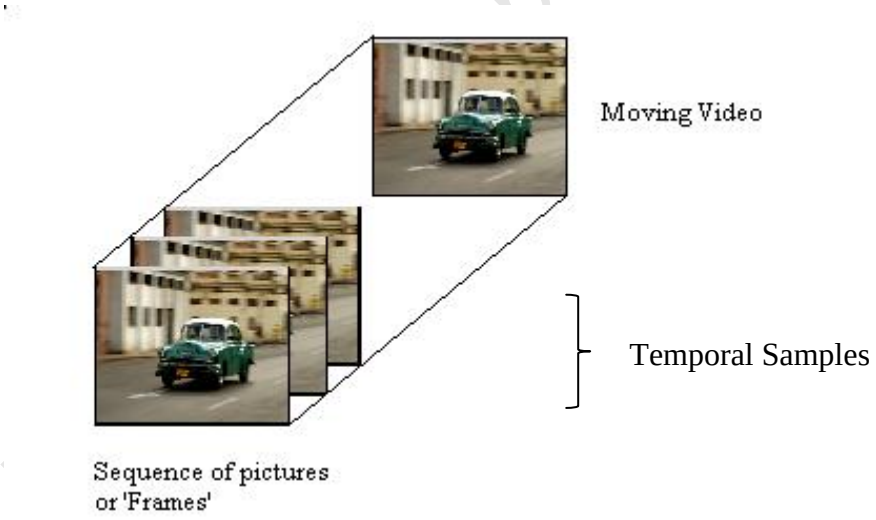


Figure 2.1: Temporal sampling of a video

Sampling which involves a whole frame is ‘progressive sampling’. However, using ‘interlaced sampling’ the smoothness of the video can be improved. In interlaced sampling, a frame is divided into two ‘fields’, each field has odd or even numbered lines, thus each having half the information of the frame. Thus instead of capturing 25frames/second progressively, 50

interlaced frames can be captured per second. Therefore, by doubling capture frequency save half of the bandwidth could be saved [4], [7]. This interlaced sampling is of special interest for lower bit-rate video communications. The interlaced sampling is illustrated in Figure 2.2.

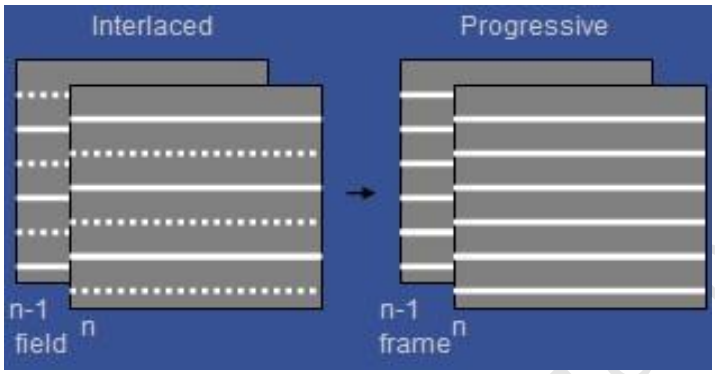


Figure 2.2: Interlaced sampling vs. Progressive sampling [7]

2.1.1.2 Spatial sampling of a digital video

Spatial sampling is generally done by taking samples of the pixels from the frame at some defined points. A rectangular grid is superimposed on the frame to be sampled. The intersection points of the grid determine the sampling points [1]. When reconstructing the image each sample denotes one picture element, a pixel, as seen in Figure 2.3.

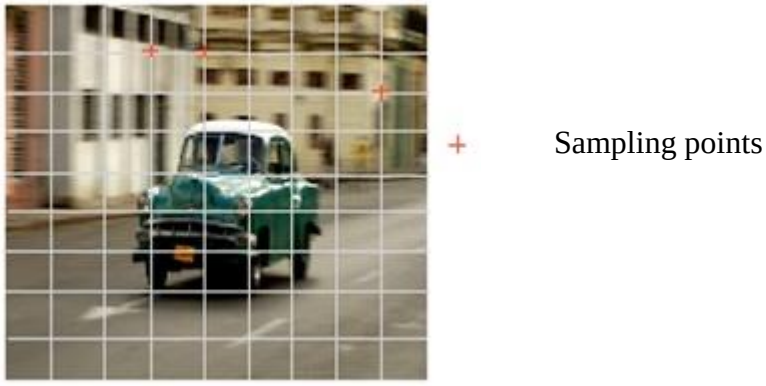


Figure 2.3: Spatial sampling of a frame using a rectangular grid

2.1.1.3 Colour sampling of a digital video

The information in any picture or a frame can be represented by its luminance components and colour components. The luminance detail is critical and must not be compressed; however, the detail in the colour components can be reduced by filtering or sampling schemes that exploit the poor colour acuity of human vision. A picture may be interpreted in three components at chroma level - red component, blue component and green component that is popularly known as the Red Blue Green (RGB) format. The complete information on colour can be compressed if represented in YCbCr format (alternatively known as YUV) where Y is the luma component and Cb, Cr being the colour components of blue and red respectively. The Y:Cb:Cr format is generally designated as 4:n:n. Initially, the first digit denoted horizontal sampling rate for luma, the second and third digits denoted horizontal sampling rates for Cb and Cr respectively. More recently, after the vertical colour sampling was developed the first digit still represents horizontal sampling of luma, the second digit represents the horizontal subsampling of both Cb and Cr with respect to luma and the third digit represents vertical subsampling of Cb and Cr [8].

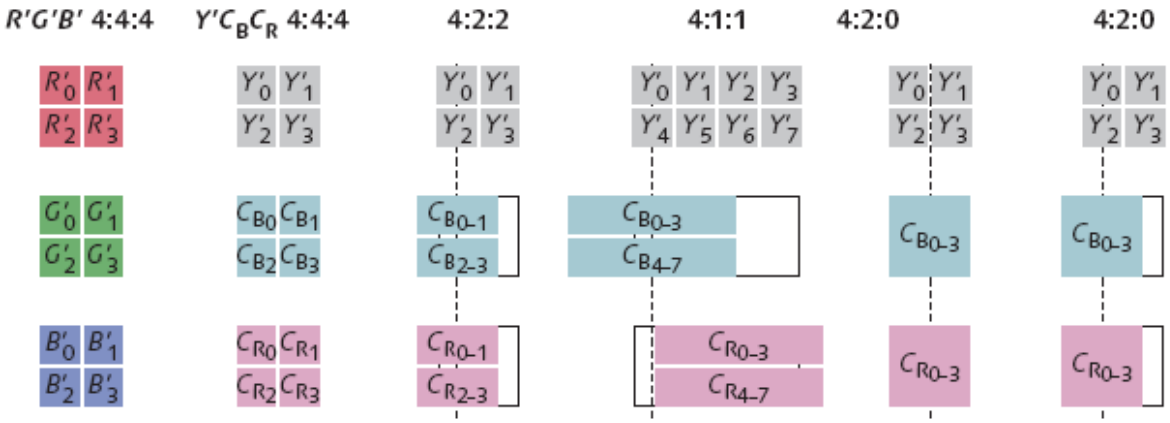


Figure 2.4: Chroma subsampling formats [8]

In the Figure 2.4 the prime "'" denotes that data has not been altered or compressed thus in the R'G'B' representation there is no colour compression employed and the entire colour information is transmitted. The R'G'B' can be transformed to the 4:4:4 Y'CbCr scheme, wherein

there is no loss in any luminance information but the colour components have been transformed. In the 4:2:2 scheme, Cb and Cr components are each subsampled by a factor of 2 horizontally. In the scheme 4:1:1, the Cb and Cr components are subsampled by the factor of 4 horizontally with respect to every fourth sample of luma. The 4:2:0 sampling technique the Cb and Cr are subsampled by 2 both vertically and horizontally for every fourth luma sample [8]. MPEG-2 uses 4:2:0 chroma sampling as a part of the standard.

2.1.2 Frame Prediction

In video compression, the frames are classified into three groups, 'I frames', 'P frames', and 'B frames'. An I frame (Intra-coded picture) is conventionally full-coded without any prediction. A P frame (Predicted picture) is predicted from its previous frame, it holds only the changes in the image from the previous frame. When the background of a frame is exactly same as that of the previous frame, the encoder does not need to store the unchanging background pixels in the P frame, so saving the number of transmitted bits. P frames are also known as delta frames. A B frame (Bi-predictive picture) uses differences between the current frame and both the preceding and following frames to specify its content [4], [9]. Only the information which is unique to the current frame and which is not present in past and future frame is encoded therefore saves even more bits. The I.B and P frames are illustrated in the Figure 2.5.

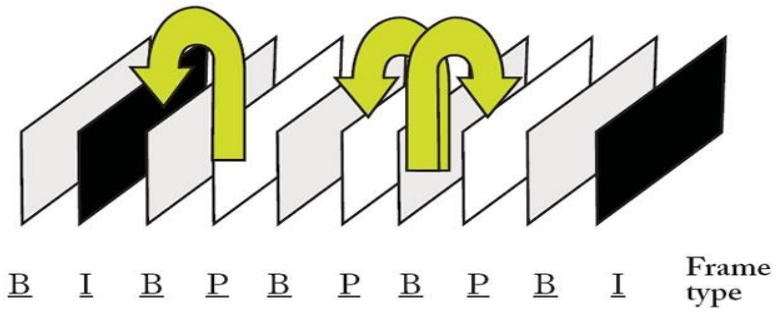


Figure 2.5: Illustration of inter-frame prediction [9]

2.2 Video Codec

A video codec is a combination of ‘co’ding and ‘dec’oding systems. The encoder attempts to reduce temporal and spatial redundancies using different compression methods. The key concept in removing temporal redundancies is by constructing a prediction of a current frame. Due to high similarities in frames, a frame can be constructed using information in other (previous or future) frames. The main idea is to predict as much as possible and then transmit only the ‘residual’ which is the information unique to the current frame. This residual is the input to the spatial model. The spatial model then checks for similar pixels inside the residual frame and attempts to compress it even further. The temporally and spatially compressed residual frame is then coded and transmitted. The decoder must reverse the processes of the encoder to obtain the original frame as shown in the Figure 2.6.

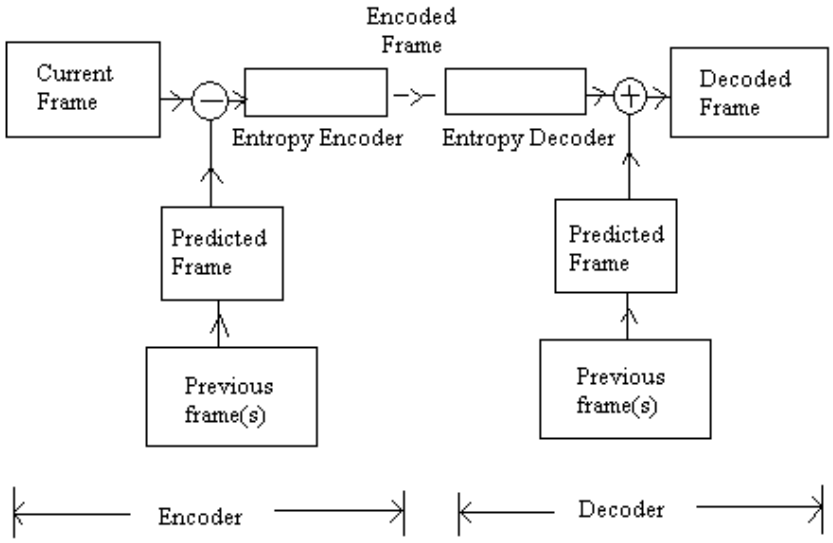


Figure 2.6: Video codec [4]

The strategy in compressing a video is to send much less information to reconstruct each frame. Sending the difference between the frames instead of the whole frame itself would reduce the data to be transmitted to an extent but sending the error between the original and its prediction has been the key to efficient compression. Constructing a frame uses data prediction and BMA performs the construction of the frames. The desirability of an algorithm depends

mainly on its compression speed. However, the quality of the data achieved is also to be considered, thus a trade-off between the speed and quality is inevitable.

2.3 Block Diagram of Video Compression System

As discussed earlier, the process of video compression involves reducing the data size at different levels. The block diagram presents the complete video compression process and illustrates the significance of different compression techniques at different levels of compression process.

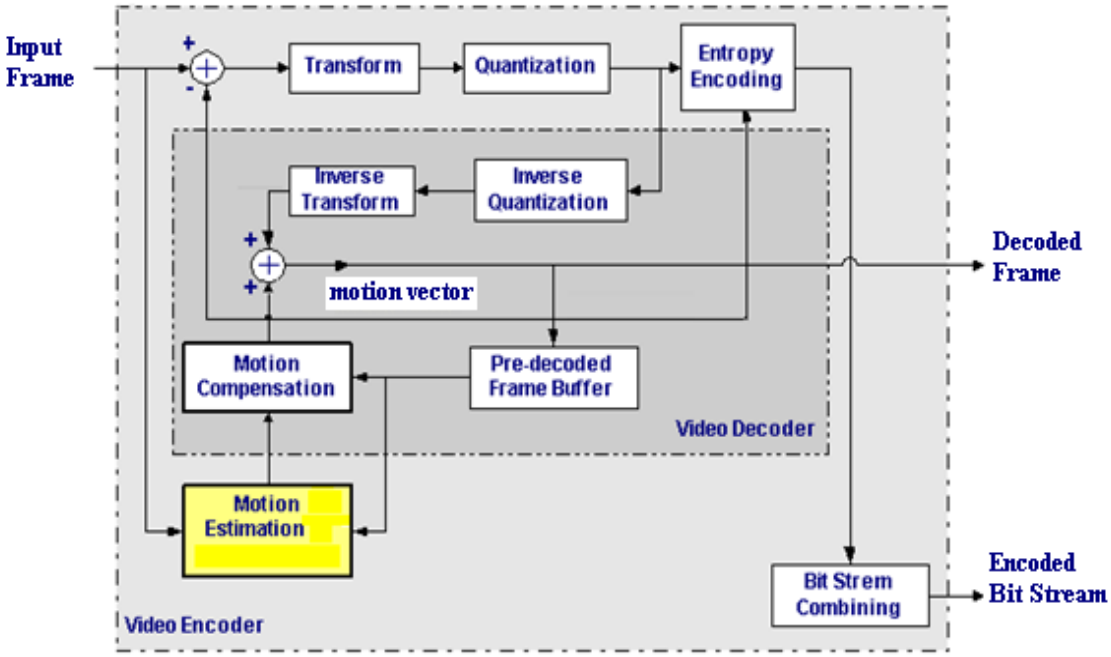


Figure 2.7: Block diagram of video compression [10]

Referring to Figure 2.7, the video encoder block is present at the source and the video decoder part is present at both source and destination. The macroblocks of the frame which is to be coded are motion estimated and compensated using the previous frame. The previous frame which is in the buffer must be stored in original format (un-coded) for comparison which, demands a decoder block inside the encoder. The resulting predicted frame is subtracted from original frame for a subsequent residue. The residue is then frequency transformed using

Discrete Cosine Transform (DCT) thus coefficients are quantised. The entropy encoder encodes the coefficients with the motion vector to a bit stream which is transmitted. At the destination, the decoder performs the inverse of every process and reconstructs the frame with the help of the residual and the motion vector(s) [1], [10].

2.4 Video Compression Standards

The overwhelming demand for video applications aroused the need for standards in video compression techniques. The history of standardisation started as early as in 1980s when a codec was designed based on differential pulse code modulation (DPCM).This codec had a very low temporal quality as DPCM worked on pixel basis which led to the design of block-based codecs. In parallel to the International Telecommunication Union (ITU-T), the Joint Photographic Experts Group (JPEG) designed a codec, which used Discrete Cosine Transform (DCT) as the main unit of compression. Meanwhile, ITU-T group made recommendations on using inter-frame DPCM for compression. The standard definition was launched in 1989, it was H.261.In the early 90s the Motion Picture Experts Group (MPEG) released their first generation standard called MPEG-1 standard. Early versions of MPEG-1 were called MPEG-1+ and these versions led to MPEG-2. These were used in digital satellite television, Digital Versatile Disc (DVD) as well as other applications. ITU-T adopted MPEG-2 and made the H.262 standard, more precisely the MPEG-2/H.262 standard. MPEG continued to develop MPEG-3 for High Definition Television (HDTV), meanwhile Public Switched Telephone Networks (PSTN) was demanding very low bit rates .In 1997, the MPEG group joined ITU-T and Joint Video Team (JVT) was formed. This team intensively worked on lower bit rates and H.26L which was named H.264 by ITU-T and MPEG-4 version 10 by MPEG [5]. This evolution process through the time is summarised in the Figure 2.8.

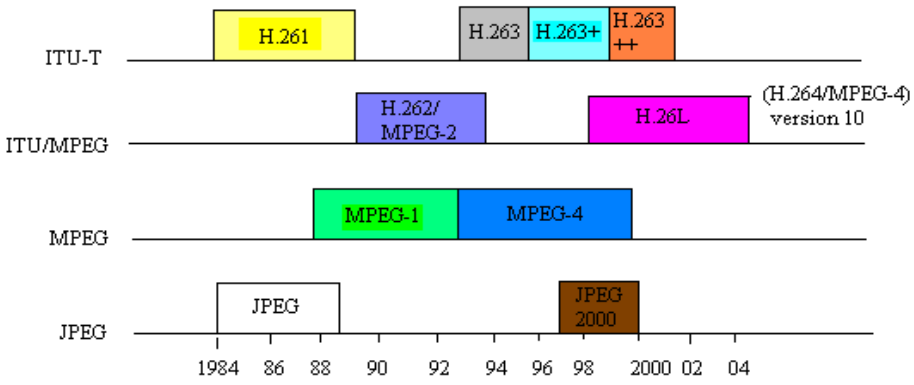


Figure 2.8: Evolution of video coding standards [5]

Chapter 3

3 Overview of Motion Estimation and Other Compression Techniques in Video Compression

Motion estimation is a significant process in the entire video compression process. An improvement in this sector would impact the overall throughput to a great extent. This chapter gives an overview of different compression techniques with special focus on motion estimation algorithms. This chapter explains motion estimation with an introduction on the video compression process. Different compression techniques applied at various levels of compression process are discussed and the over-all block diagram of the video compression scenario is presented. More importantly, it introduces and explains motion estimation, which is the central topic in this research.

Video compression (or video coding) is a technique to reduce the amount of data in a video file to a manageable size. It remains an essential technology for applications which make extensive use of video data such as digital television, DVD, Video, mobile devices, videoconferencing and internet video streaming. A Video compression system comprises of an encoder at the source which compresses the data and a decoder at the destination which decompresses the data. A video codec will remain a vital part of the emerging multimedia industry through the future allowing designers to make the most of the available resources [1].

3.1 Compression Techniques

A video compression technique makes an attempt to transmit the current frame with minimum number of bits. The process of compressing a video involves a flow of prediction, estimation, compensation and then encoding the compressed frame.

There are two basic categories of compression; lossless and lossy. Lossless compression will allow the exact original data to be reconstructed from the compressed data. In lossless compression, there is limited reduction of data. Lossy compression on the contrary means that

the data is minimised largely through the compression process. Here, the reconstructed data differs from the original. These differences are called 'artefacts' [1].

3.1.1 Block Motion Estimation

This remains the focused area throughout the research. The basic premise in block motion estimation is coding the error between the frames. Considering two consecutive frames there is a possibility that a significant part of the frames would be same as the time interval between frames is just $1/30^{th}$ of a second. However, the recognition of the complete object in a frame becomes difficult so, the image is split into smaller blocks called 'macroblocks' with a size of 8x8 or 16x16 grouped pixels and are separately handled. The entire motion estimation is illustrated in Figures 3.1, 3.2, 3.3 and 3.4.

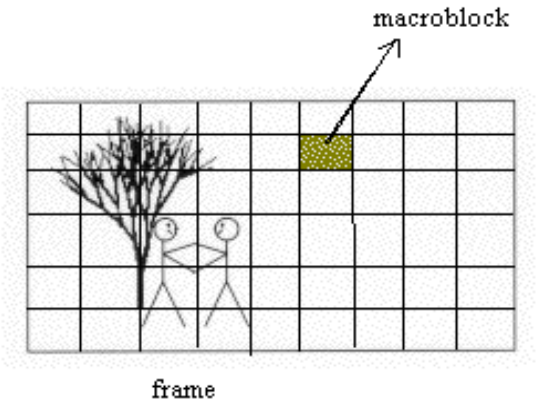


Figure 3.1: Frame and macroblock [11]

The prediction process depends upon the type of frame dealt with. The I frames are images that permit direct access to the individual sections in itself so it can be self-predicted. The P frames are predicted from the I frame and the B frames are interpolated from the previous or following P or I frame(s). Figure 3.2 shows the current frame (frame 2) with its previous frame (frame 1). For the example frames presented in Figure 3.2, the motion estimation process is elucidated as follows.

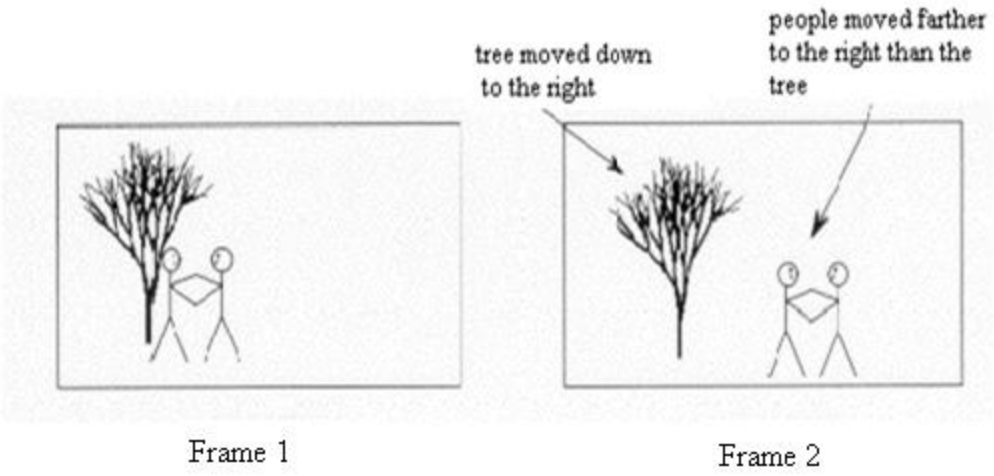


Figure 3.2: Example frames [11]

The Figure 3.2 above refers to a set of consecutive frames which have a visible difference. The current frame is supposed to be a P frame, is coded using the information in its previous frame, frame 1. Assume that the macroblock to be coded is the one shown in Figure 3.3, which is from frame 2 [11].

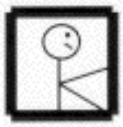


Figure 3.3: Macroblock to be encoded [11]

To find a match for this macroblock all the neighbouring macroblocks in frame 1 are searched and the best match is chosen as seen in Figure 3.4. The macroblock with least energy difference is selected as a best match.

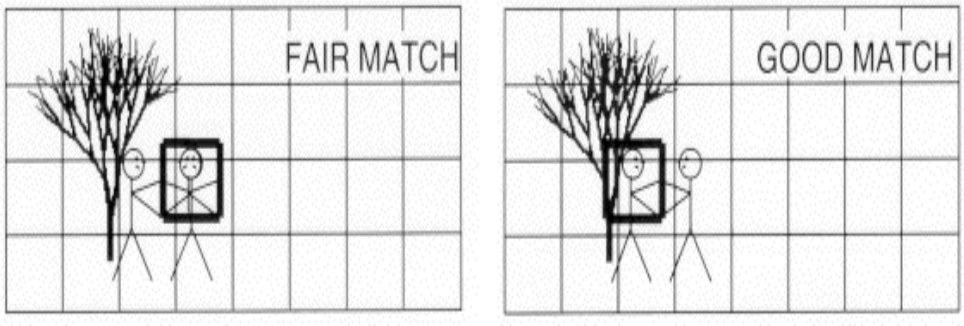


Figure 3.4: Possible matches for the macroblock [11]

This block matching process is repeated for every macroblock and a prediction of the desired picture is estimated. This is motion estimation .

3.1.1.1 Energy Measures in Motion Estimation

The motion estimation technique uses energy measure as a matching criterion. In the estimation process, energy difference between a current block and its neighbouring blocks in the reference block is calculated. The block with least energy difference in the reference frame is the best match for that block in the current frame. There are three different energy measurements: Mean Squared Error (MSE), Mean Absolute Difference (MAD), which is also called Mean Absolute Error (MAE) [5], [12], and Sum of Absolute Errors (SAE), also called Sum of Absolute Differences (SAD) [4]. Any of these three energy measures can be employed to estimate the energy difference between the blocks.

Considering a N x N sample block where C_{ij} is a sample of the current block and R_{ij} is a sample of the reference block, the energy measures can be calculated as follows:

$$MSE = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}(C_{ij} - R_{ij})^2 \qquad (3.1)$$

$$MAD = \frac{1}{N^2}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|C_{ij} - R_{ij}| \qquad (3.2)$$

$$SAE = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|C_{ij} - R_{ij}| \qquad (3.3)$$

This project adopted MAD as the matching criterion as MAD is easier to calculate than MSE yet gives a reasonably good approximation [4].

3.1.2 Motion Compensation

Motion compensation allows regions in the prediction frame to be generated from shifted regions of the previous decoded frame. This is done by the motion vector, a two-dimensional vector used for inter prediction that provides an offset from the coordinates in the decoded picture to the coordinates in a reference picture. Each block is assigned motion vector(s), which informs about the direction of relative shift. Motion vector(s) are transmitted to the decoder with the encoded residual so that the frame can be reconstructed at the destination [11], see Figure 3.5.

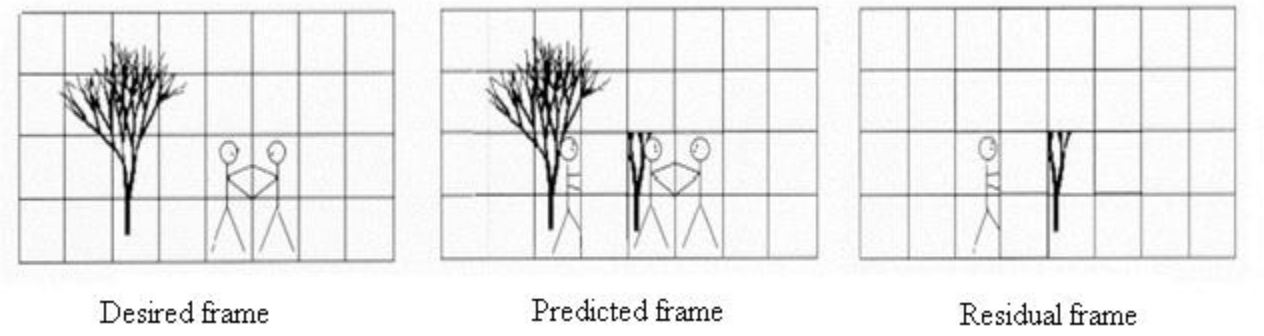


Figure 3.5: Prediction and residual [11]

3.2 Discrete Cosine Transforms (DCT) and Integer Transforms (HT)

DCT on a frame is used to suppress spatial redundancies, as the human eye is not sensitive to high frequencies. Human eye is unlikely to notice the difference when the pixels are reconstructed from similar ones in a big uniform patch. DCT is based on Fourier transformations that present a signal as a combination of sine signals of different frequencies and amplitudes. The Fourier transformation yields frequency and amplitude distribution values from the location of pixel values in an image. This means that large, regular areas in the image are then represented

more in the lower frequency parts, whereas finer details are in the higher range. DCT transforms the display 8x8 macroblock into an 8x8-coefficient matrix [11]. For dealing with the 4x4 blocks, the H.264 version of the project uses Integer Transforms, which will be referred as HT in the thesis as in paper [13].The Figure 3.6 shows the conversion of DCT coefficients into HT coefficients.

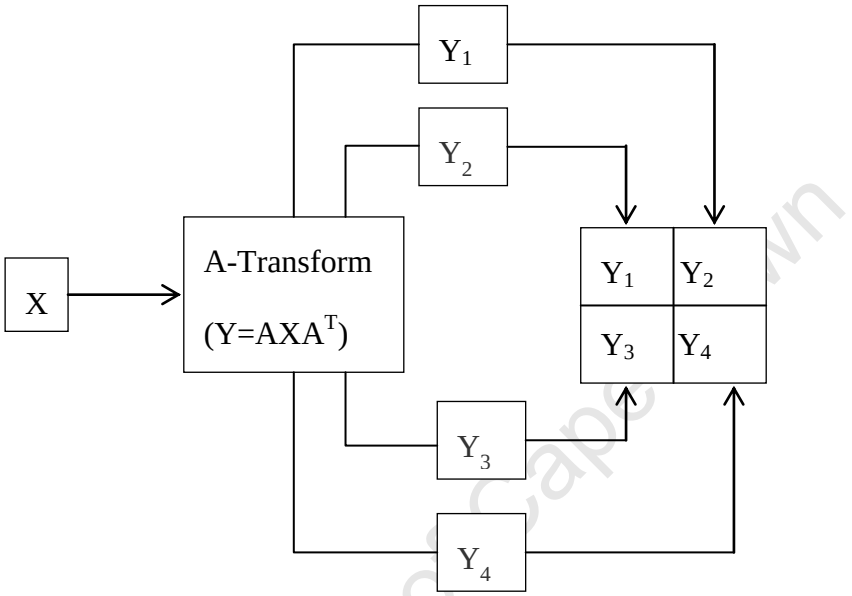


Figure 3.6: DCT to HT conversion [13]

In the above figure, X is the input DCT coefficient matrix. For an NXN input block, A is the 8X8 transform matrix that is used to produce Y, which is the coefficient matrix for HT. Thus, for an NxN sample block the Y is given by:

$$Y = AXA^T \quad (4)$$

and the inverse is given by:

$$X = A^TYA \quad (5)$$

3.3 Entropy Encoding

Entropy coding is a lossless compression technique. The DCT coefficients of the residual frame and the motion vectors are entropy coded before transmission. Most lossless compression programs first generate a statistical model for the input data, and then use this model to map input data to bit sequences in such a way that data that is more probable or redundant data is denoted by shorter bit sequence and longer bit sequences refer infrequent data thus compressing the bits transmitted [5]. Some widely known entropy coding techniques are discusses below.

3.3.1 Shannon-Fano Coding Technique

This technique was proposed by Shannon (1948) and is attributed to Fano (1949). In this technique, the message and its probability are listed in a decreasing probability order. This list is then divided in two groups such that the two groups have approximately same total probability. The first digit code for the first group starts with '0' and the second group with '1'. Continue sub dividing the groups in the similar manner i.e. having approximately same total probability counts and code digits are appended accordingly [4], [14]. For better understanding, an example of Shannon-Fano coding technique is presented for the data shown in Table 3.1 and illustrated in the Figure 3.7.

Table 3.1 Example input data [15]

Symbol	A	B	C	D	E
Count	15	7	6	6	5

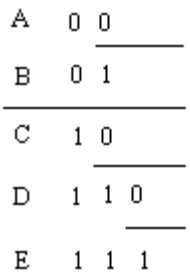


Figure 3.7: Grouping in Shannon-Fano coding [15]

The grouping technique explained above is shown in Figure 3.7 above. From the Table 3.2 it is seen that more probable symbol would need less number of bits thus saving significant number of bits while representing redundant information. The overall number of bits for this particular input can be calculated as shown in the Table 3.2.

Table 3.2 Total bits to representing the data using Shannon-Fano technique [16]

Symbol	Count	Code	Subtotal of bits
A	15	00	30
B	7	01	14
C	6	10	12
D	6	110	18
E	5	111	15
		Total bits:	89

3.3.2 Huffman Coding Technique

Huffman proposed this entropy coding technique in 1954 in the paper [16]. Unlike Shannon-Fano coding where the approach is from highest probable symbol, Huffman coding starts from the lowest probable symbol and then climbs up to the highest possible symbol.

Initially all the symbols are a part of the list. Two symbols with lowest probabilities (the child symbols) are picked and are grouped into a parent symbol and the sum of their counts is assigned to the parent. The parent symbol is added to the list and the two child symbols are eliminated to form a new list. This process is repeated until there is only one symbol left in the list [16]. For the same input data as in the Shannon-Fano example, Huffman coding procedure can be explained as shown in Figure 3.8 where 'P' refers to the parent symbol.

To generate codes for the symbols the upper symbol is assigned '0' and the lower symbol is assigned '1' in any group. The code for the child is generated by tracing the child through its subsequent parents starting from the right most parent. For example, the code for 'C' would be 101(P4 →P3, P3→P2, P2→C) [17].

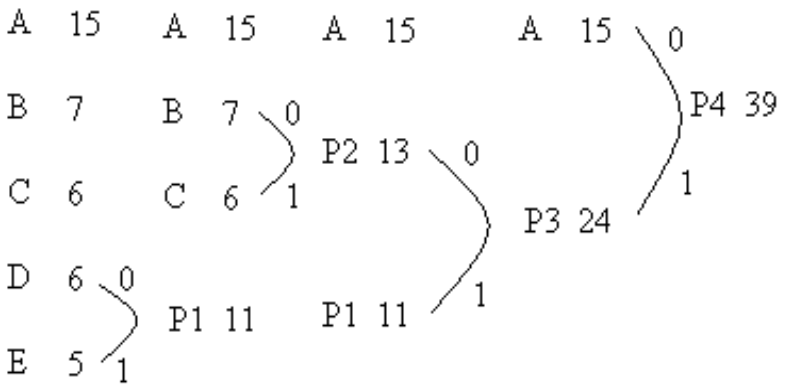


Figure 3.8: Huffman coding technique [17]

Thus, the end codes of all the symbols and the total bits for transmitting the data is listed in the Table 3.3.

Table 3.3 Total bits to represent the data using Huffman technique [17]

Symbol	Count	Code	Subtotal of the bits
A	15	0	15
B	7	100	21
C	6	101	18
D	6	110	18
E	5	111	15
		Total bits: 87	

3.3.3 Run-Length Encoding (RLE)

Run-Length Encoding is an algorithm which is used as a data compression technique in image and video compression. RLE reduces the length of a repeated string of data, here referred as 'run'. For example consider a typical grey scale data as shown in Figure 3.9, where there is a run of white and black pixels. The Run-Length algorithm codes it by using two bytes the first byte is the length of the run and the second byte being the character of the run.

Figure 3.9: Example scan line for RLE algorithm

If the white pixels are represented as 'w' and black pixels by 'b', the code for this set of data would be '1w1b1w5b4w3b1w'.This type of encoding is highly efficient when there is great

runs of data in the image or a video sequences. Generally, RLE scans and codes the frame in sequential manner-it might be either sequentially horizontal (along x-axis) or sequentially vertical (along y-axis) or even in tiles of 8x8 pixels [5], [18].

3.3.4 Variable Length Coding (VLC)

VLC is the final lossless stage of the MPEG video compression unit. VLC is applied to further compress the quantized image. VLC consists of three steps: zig-zag scanning, Run Length Encoding (RLE) followed by Huffman coding. At the decoder, VLC is the first step in the decoding process [1], [19].

Figure 3.10: VLC at the encoder [19]

3.4 Chapter Summary

This chapter has provided thorough summary of various compression techniques in video coding process with a special focus on motion estimation in particular .It serves as a foundation for the upcoming topics and discussions related to motion estimation algorithms and estimation procedure.

Chapter 4

4 Related Motion Estimation Algorithms

This portion of the thesis is mainly on the motion estimation algorithms and their search logic. This chapter is divided into two main segments. The first segment explains the algorithms from the literature review, whose modified versions were developed as new proposed algorithms. The second segment is on algorithms that were pre-written in the project which were used as gauges to evaluate the newly proposed algorithms. Motion estimation has a significant effect on overall codec performance. A good algorithm tends to minimise the residual energy, by efficiently exploiting redundancies between the frames. Some factors need to be considered while choosing an algorithm for motion estimation process are the:

- Effectiveness of algorithm to compress residual frame
- Rate-distortion performance
- Computations required in the algorithm
- Compatibility for specific hardware or software

As mentioned earlier, a macroblock is a set of pixels and these pixels can be interpreted as luminance samples. Thus, a 4x4-sample block will have 16 luminance samples. For a given set of current and reference frames, a motion estimation algorithm aims to find an identical block in the reference frame for a block in the current frame [4].

Theoretically, to find the best match for a possible block all the blocks in a reference frames must be compared. This would result in a true minimum point thus minimising the energy in the residual frame. The Full Search Algorithm (FSA) carries out comparisons with all the possible blocks in a reference frame. Thus for every block in the current frame, all the blocks of the reference frame are searched. The FSA is bound to find the best possible match for each block of the current frame, thus creating the closest replica of the current frame. On the other hand, searching the whole frame for each current block is impractical, for the reason that mostly

the best match for the current block is found in its immediate neighbourhood. Hence, in practical implementations, the full search is limited to a 'search window'. Search window is a region in the reference frame typically centred on the same position of the block as in the current frame [4]. Although the search window concept reduces the search count there is a possibility that the minimum block resides outside the search window, which implies that the best match found is a 'local minima' and not the true best match which is termed as 'global minima'.

Although FSA is forbiddingly complex to implement in real-time scenarios, the project includes it as a metric for measuring the performance of other fast algorithms. The FSA in the research proceeds in a raster order as shown in the Figure 4.1. The FSA in the research is limited to a search window sized 15x15 blocks, each block constituting 16x16 pixels. The search starts at the corner of the search window and then proceeds in a raster order ending up searching all the points in the search window as shown in Figure 4.1.

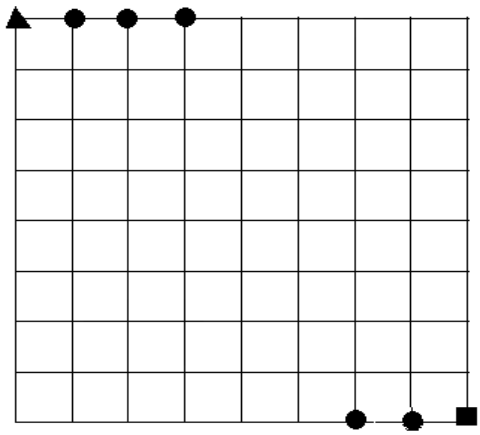


Figure 4.1: Full search algorithm [1]

A fast search algorithm attempts to reduce the number of computations involved in the algorithm. The key idea is to sample search the positions, thus avoid searching all possible locations. There have been many fast algorithms proposed over time to achieve quicker estimations of the frames; Three-Step Search (3SS) [13], Cross Search Algorithm (CSA) [20], Logarithmic Search (LS) [21], Nearest Neighbour Search (NNS) [22] and Diamond Search Algorithm (DSA) [23] are few to name. A fast algorithm may or may not locate the true best

match, as the ‘real’ minimum block might never have been searched. The result is that the best match found by the fast algorithm has more energy difference than the one found by FSA. Thus, using a fast algorithm calls for a trade-off between the speed of the compression process and the quality of the video transmitted [1].

Multi-resolution algorithms are a special case in fast algorithms. In normal algorithms, the search takes place at a single resolution level, implying that the current frame and the reference frame will be at the same resolution. The multi-resolution algorithm employs the idea of estimating a single frame at different resolutions [4]. A multi-resolution scheme is advantageous in environments where different systems use different resolutions of a video. A version of the multi-resolution algorithm is implemented in the project, which will be discussed in this chapter.

4.1 Related Fast Algorithms from the Literature Review

Following the literature review, few algorithms and search patterns were chosen to form the basis of new motion estimation algorithms. This portion of the thesis is dedicated to the algorithms that relate to the proposed algorithms and to the algorithms that inspired the development of new motion estimation algorithms.

4.1.1 Three-Step Search Algorithm (3SS)

Koga et al. (1981) proposed the Three-Step Search algorithm, as an attempt to deal with the computational complexity of FSA. 3SS is a fast search and considerably simpler than FSA which performs the search in iterations. In an iteration, eight positions around the centre are searched to get the block with minimum energy difference. The step size is halved through every iteration performed where, the ‘step size’ is defined as the distance between the centre and the search points. The step size for 3SS would be generally be 4. The 3SS algorithm can summarised as follows [1]:

1) Identify the centre location (0, 0).
2) Search 8 locations +/- 4 around (0, 0).
3) Pick the location with minimum energy difference and make it the new centre.

4) Halve the step size (now step size is 2).

5) Search 8 locations, 2 locations far around the centre.

6) Find the best matched location and with this new origin search 8 locations around it with step size 1.

7) The resulting best match is the result of the 3SS algorithm.

Figure 4.2 illustrates the iterations and the procedure adapted by 3SS algorithm.

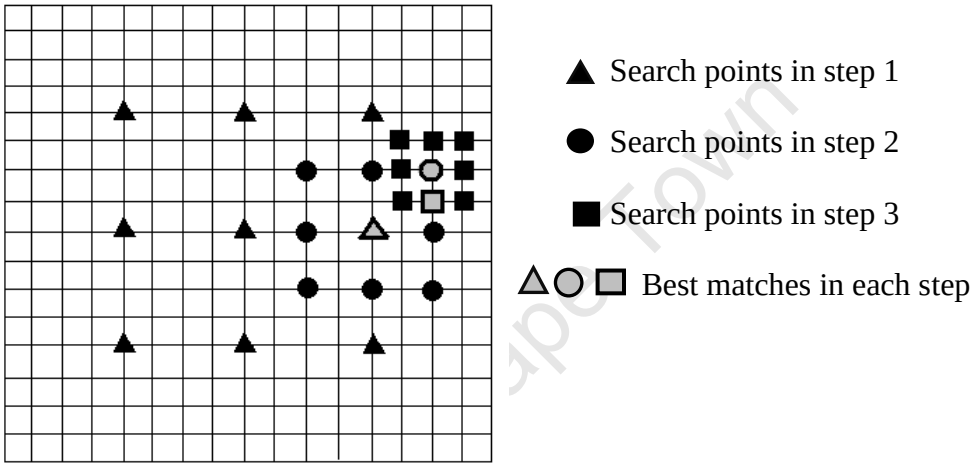


Figure 4.2: Three-step search [1]

4.1.2 Cross Search Algorithm (CSA)

CSA was a fast algorithm proposed by Ghanbari (1990). This algorithm adopted logarithmic step approach to generate iterations. Four points are searched in each iteration and these search points can be imagined as end-points of a cross [20]. According to [20], the CSA can be described as follows:

1) The location is centred at (0, 0). Step size is set as half of the maximum motion displacement.

2) Search the four locations which form the ends of a cross with (0, 0) as the centre of the cross.

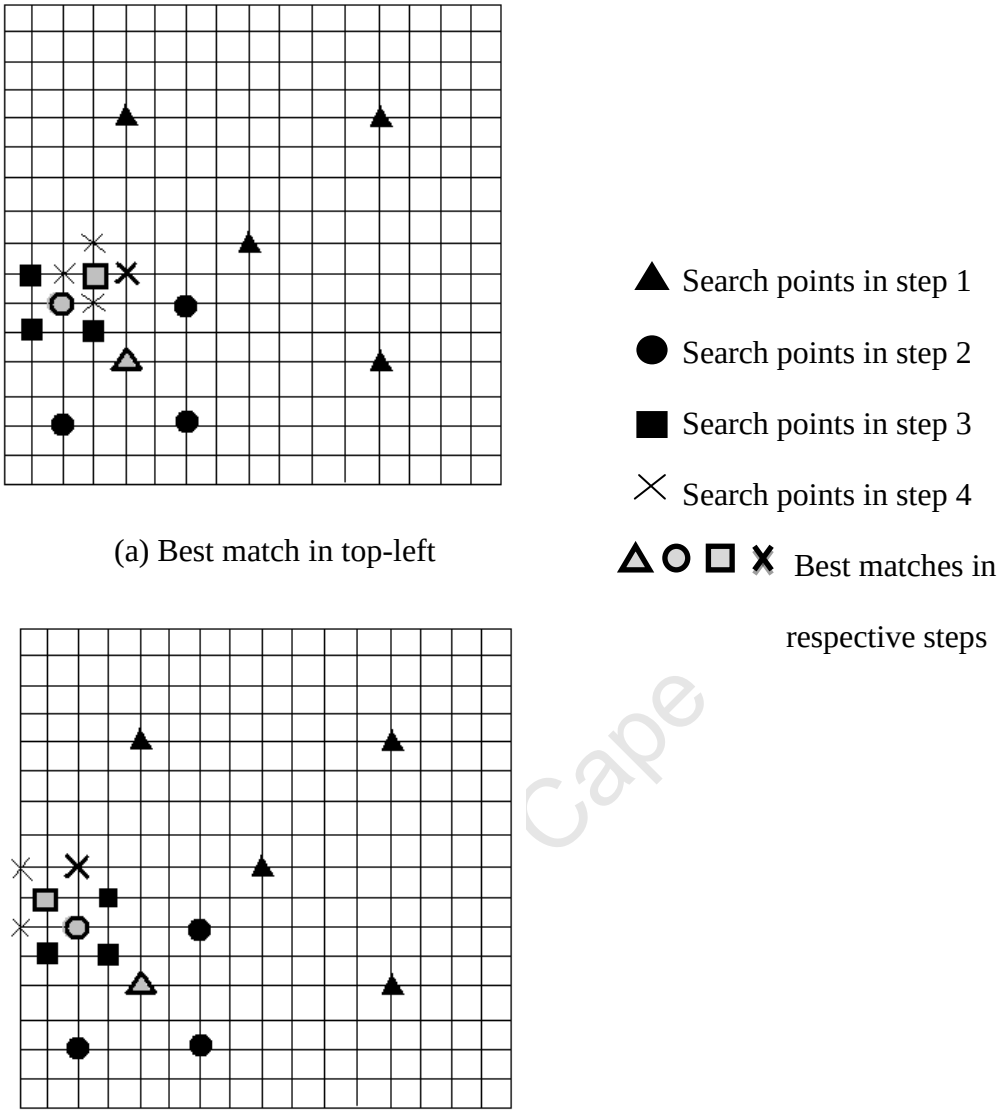
3) For step size less than zero, halve the step size.

4) With the best match from the first iteration as centre, search the ends of cross again.

5) Repeat until step size is one. This is the penultimate step.

6) This is the last step and the step size =1. If the location of the match from the previous step is top left or bottom right, search the locations that are ends of a 'X'. If the best match is at bottom left or top right search the locations that are ends of a '+' (to avoid repeating search on same pixels).

7) Best match found is the final result, either they are the end points of a 'X' or a '+'.



(a) Best match in top-left

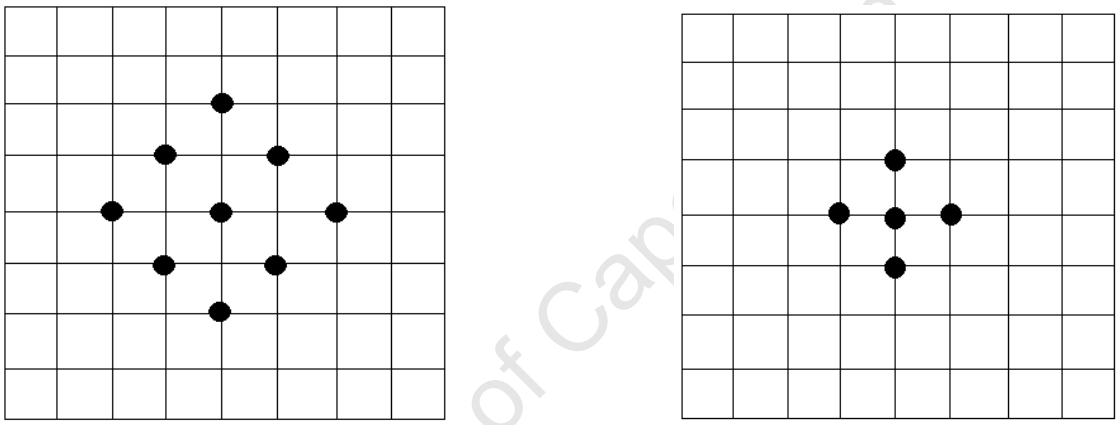
(b) Best match in top-right

Figure 4.3: Cross search algorithm [6]

The search pattern of CSA is shown in the Figure 4.3 with two different possibilities in its penultimate level.

4.1.3 Diamond Search Algorithm (DSA)

The mentioning of this algorithm is important because of the two search patterns employed by DSA. These patterns were incorporated in the novel proposed algorithms. The patterns are as shown in the Figure 4.4 in which the first one is Large Diamond Search pattern (LDSP) that has nine search points in total (eight points forming a diamond shape and the centre) and the second one is the Small Diamond Search Pattern (SDSP) which searches five points (four points making a smaller diamond shape and the centre) [23].



(a) Large Diamond Search Pattern (LDSP) (b) Small Diamond Search Pattern (SDSP)

Figure 4.4: Search patterns in DSA [23]

The DS algorithms can be explained as follows [23]:

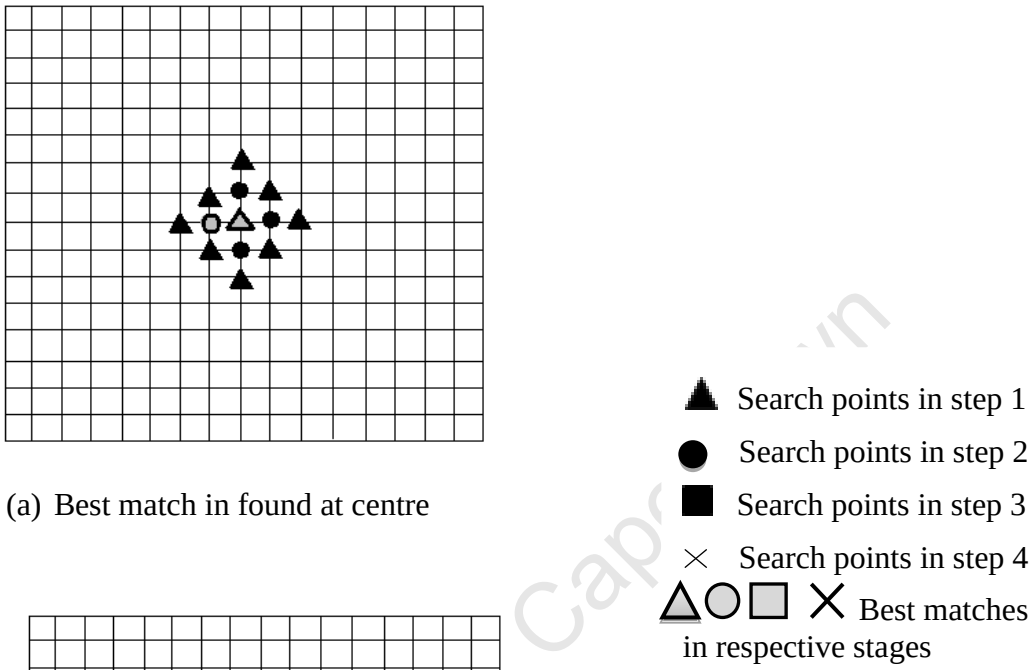
1) Initially the search starts with LDSP, nine points including the centre of the search are tested for the best match. This calls for considering two cases-

   Case (a): If the best match is the centre, SDSP is applied .The best match found in this step is the final match.

   Case (b): If the best match is not the centre and is found in one of the eight bordering points, LDSP is applied with the winning point as the new centre.

This procedure isrepeated until the best match forms the centre, then SDSP is eventually applied to get the final result.

Both the cases explained above are illustrated in Figure 4.5.



(a) Best match in found at centre

(b) Best match not in the centre

Figure 4.5: Diamond search algorithm [23]

4.1.4 Multi-resolution or Hierarchical Algorithm

In all the previous algorithms discussed, the estimation takes place at a single level of resolution. Contrarily, hierarchical algorithms perform search through different levels of resolutions. This division of the chapter starts with the procedure by which the resolution levels were generated and proceeds to detailing the search logic adopted by the hierarchical algorithm. This algorithm formed the principle idea for all the faster estimation schemes developed in the research.

4.1.4.1 Generation of Hierarchy Levels

A hierarchical algorithm needs a single frame to be represented at different resolutions. The frame at its full resolution is considered as the base level and higher levels of hierarchy are generated out of it. This naturally results in a pyramid structure, which is often termed as 'hierarchy' where the base of the pyramid refers to the frame at its full resolution and the resolution/quality of the frame decreases as we move towards higher levels in the hierarchy.

Levels can be yielded either by sub-sampling or by averaging the pixels in the original frame [18]. In the sub-sampling method, a frame with NxN blocks can be sampled by a factor of two in both dimensions that would result in a frame with N/2xN/2 blocks. This N/2xN/2 can further be sub-sampled until the required resolution is obtained. This creates a 'sub-sampling' hierarchy. Here, the information (block energy) stored is directly related to the base frame, thus an exact recovery of the original frame is possible. On the other hand, the sampled points might not represent the entire information in the sampled area, which results in poor spatial correlation among the levels. Thus, it becomes difficult to run the estimation process through the levels [18].

The correlation problem in the sub-sampling method would be eliminated by performing averages instead of sampling. Constructing a 'mean hierarchy' involves forming a hierarchy by averaging over blocks of pixels to form bigger blocks. For example, two 4x4 blocks are averaged to form one 8x8 block. The mean values are generally better interpretations of an area than taking just one sample point out of the whole region. This project uses an averaging technique to generate the required levels [18]. The generation mean hierarchy is shown in Figure 4.6.

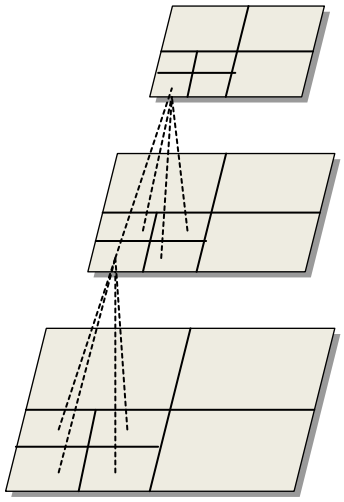


Figure 4.6: Mean hierarchical pyramid [18]

Normally a multi-resolution algorithm that uses sub-sampling proceeds as follows [4]:

1) Level 0 has the reference and current frames in their full resolutions. Sub-sample level 0 in both horizontal and vertical directions to produce level 1. Further sub-sample level 1 to produce level 2.
2) Search all points in the highest level (here level 2) for the minimum point. All the points are searched at this level. The motion vector assigned at this level is the initial coarse vector.
3) In the lower level (here level 1), search eight points around the coarse vector to find a better match.
4) Repeat this until the lowest level which is the level at full resolution is reached. This is the final result of the algorithm.

A multi-resolution algorithm proceeds as shown in Figure 4.7.

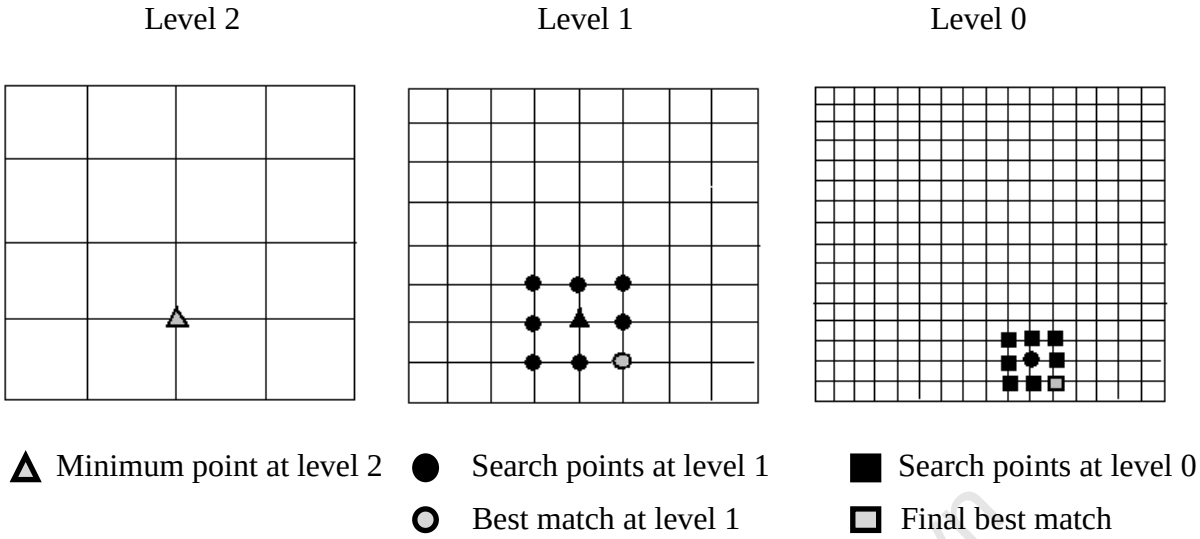


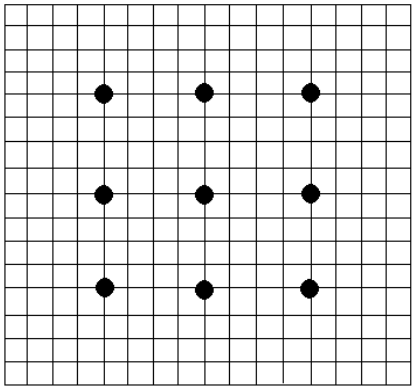
Figure 4.7: Multi-resolution search using subsampling technique [4]

4.2 Existent Fast Algorithms in the Project

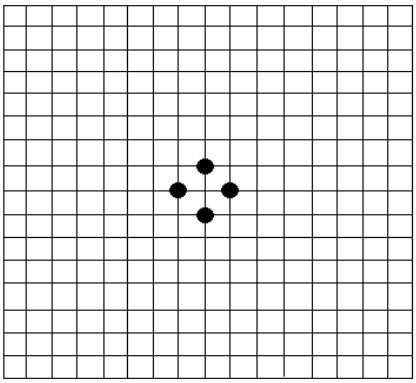
The project already contained three motion estimation algorithms in total out of which one was the FSA and the other two were fast search algorithms. This portion of the chapter explains these two existing algorithms. The objective of the research was to propose new algorithms that are more efficient than these pre-written algorithms.

4.2.1 Efficient Three-step search Algorithm (E3SS)

Xuan Jing and Lap-Pui proposed this algorithm in an effort to minimise computation in Three-Step search (3SS) [1]. This search involves a combination of two search patterns, first is a 9-point grid and the second, a 4-point small diamond as illustrated below in Figure 4.8.



(a) Grid pattern



(b) Small Diamond pattern

Figure 4.8: Search patterns used in E3SS [25]

The step-by-step approach by E3SS is as follows [25]:

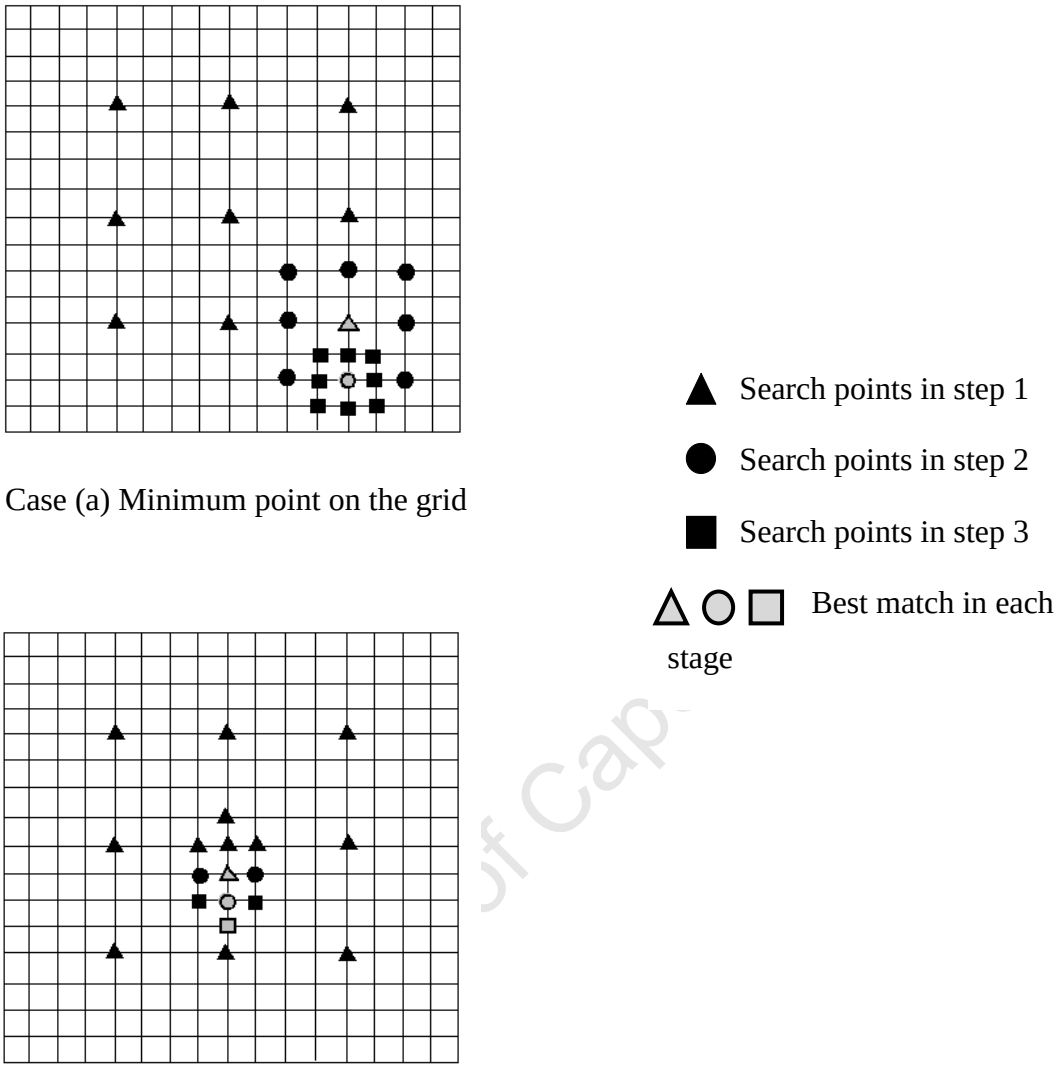
1) In the first step, nine points of the grid and four points of the small diamond are searched. The width of the grid in this project is four blocks from the centre and the small diamond, is placed one block away from the centre.

2) The minimum point found in step 1 now becomes the new centre. If the minimum point is at the centre of the grid, the search ends; if not, there are two cases to be considered, (see Figure 4.9 for illustration):

   Case (a): When the minimum point is on the bordering points of the grid, it follows the 3SS concept where the width of the grid is halved and eight points of the grid around the match are searched. This process repeats until the best match is found in the centre.

   Case (b): When the minimum point sits on the small diamond, the centre of the diamond is moved to the minimum point and the points on the newly positioned diamond are searched. This process repeats until the best match is found at the centre of the diamond.

Every time this algorithm performs a search, it only searches the points that were not searched previously.



Case (a) Minimum point on the grid

Case (b) Minimum point on the small diamond

Figure 4.9: Efficient three-step search algorithm [25]

4.2.2 Project version of Hierarchical Search Algorithm (HS)

A version of the Hierarchical algorithm was included in the project for the fast motion estimation process. This version will be referred to as 'HS' in this thesis. The levels in this algorithm were generated using the averaging technique, thus creating a mean pyramid, as mentioned in the previous section. Once the levels are set, the search for the match starts in the

highest level, level 2 and reaches to the full image resolution, level 0. For this algorithm, the macro blocks are sized 4x4 pixels at the original resolution. The algorithm performs full search at all the three levels of the hierarchy.

The systematic explanation for the project version of HS is as follows:

1) Start with level 0 at the full resolutions. Get level 1 from level 0 and level 2 from level 1 thus; the three- level mean pyramid is constructed.

2) At the highest level (level 2), calculate the energy difference between the current block and the zero vector difference block in the reference frame. Note the energy difference in the blocks; this point forms the measure and also serves as the centre for the search at level 2. Perform a full search in the area of 13x13 blocks. The block with the lowest energy difference forms the centre for the next level, level 1. The vector associated with this search forms the coarse motion vector.

3) With the new centre, search the next lower level, level 1. Again a full search is carried out, but this time confined to a search window of a 4x4 block area. The block with the lowest energy difference is the best match and this gets set as a centre for level 0.

4) With the best match from level 1 as the centre, all the points which fall in the 4x4 search window around the centre are searched. This will lead to the final result of the algorithm.

The project dealt with a 352x288 image dimensions and extracted two more resolutions for implementing multi-resolution search. The table below shows the specifications at each level of the hierarchy.

Table 4.1: Specifications at different levels of Hierarchy

Specifications	Levels of Hierarchy		
	Level 0	Level 1	Level 2
Image Width	352	176	88
Image Height	288	144	72
Macroblock Width	16	8	4
Macroblock height	16	8	4
Motion Range (full pixels units)	32	16	8

Table 4.1 implies, a block of 16x16 pixels at level 2 would be two blocks with 8x8 pixels each at level 1 and four 4x4pixel blocks at level 0.

For the purpose of clarity and better understanding, the illustration below has limited the search window at level 2 to a 4x4 area instead of a 13x13. The grey area in each level represents the search areas for the levels. The Figure 4.10 below clearly illustrates the transition through different levels.

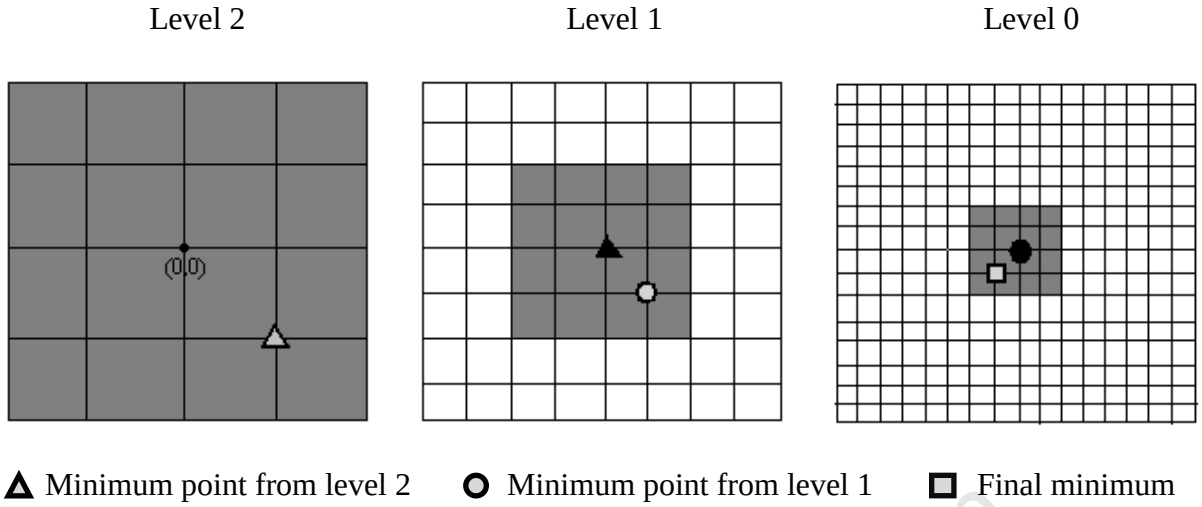


Figure 4.10: Project version of hierarchical algorithm

4.3 Chapter Summary

This chapter has explained motion estimation algorithms that led to the development of proposed algorithms. It briefly defined the FSA and fast search algorithms including the hierarchal setting by explaining the method of developing levels to perform a hierarchical search. Further, the chapter thoroughly explained all the algorithms that were pre-written in the project. The existing algorithms formed the standards against which the proposed algorithms got rated. This chapter forms a foundation for the next chapter, which is on the designed algorithms.

Chapter 5

5 Proposed algorithms

This chapter presents the algorithms that were designed, tested, and proposed for future applications in real-time scenario. The proposed algorithms are categorised into two segments: non-hierarchical algorithms and hierarchical algorithms. These algorithms are a result of a combination of concepts from different algorithms that were explained in the previous chapter.

5.1 Non-Hierarchical algorithms that were implemented

Different search patterns were considered while experimenting with the motion estimation algorithms out of which two search patterns were selected for further consideration: Large Diamond Search Pattern (LDSP) and Small Diamond Search Pattern (SDSP).

5.1.1 Large Diamond Search Algorithm (LDS)

LDS is a combination of Three-Step Search Algorithm (3SS) and the LDSP search pattern in the Diamond Search Algorithm (DSA). The diamond pattern used in the LDS searches nine points in total, eight border points and the centre (although the centre is calculated separately). The size of the diamond can be termed as the 'width' of the diamond that is analogous to the step size in 3SS. The width is the distance of the diamond points from the centre. LDS performs the search in iterations as in 3SS unlike in DSA where the width of the diamond is constant as there are no iterations performed. A maximum of eight points are searched in every iteration and the width of the diamond changes for every iteration (precisely halves) and converges onto the best match of the previous iteration.

Generally, the best match for a block in the current frame would be found at or near the same positioned block in the reference frame. This block in the reference frame with zero position difference is termed as the 'zero motion vector' point.

The sequence of the LDS is as follows:

1) The search starts with the zero motion vector difference point as the centre because it is the most likely to be the matching candidate. The energy difference between the blocks is calculated and this forms a reference for further searches.
2) The first iteration starts with searching the eight points around the origin to find a block with the minimum energy difference. This minimum position will form the centre for the next iteration.
3) Halve the width of the diamond, search eight diamond points around the centre and look for a better match. The iterations continue until the diamond size reaches the minimum width, w=2 (considering w=1 would demand half pixel computations).
4) The end result of the algorithm is compared to the zero vector difference block and the block with the minimum energy is the final result of the estimation.

The Figure 5.1 details the discussed iterations in LDS algorithm.

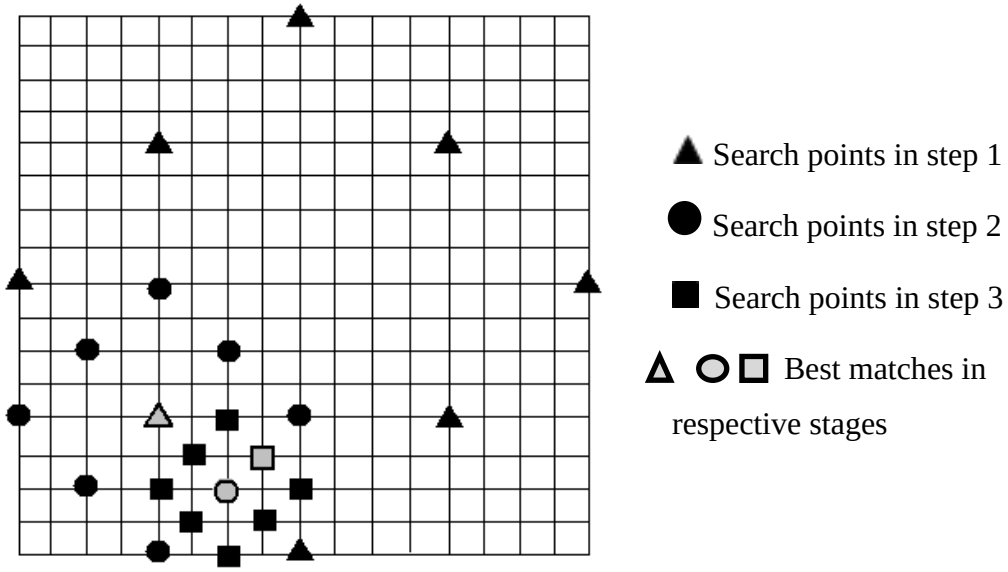


Figure 5.1: Large diamond search algorithm

5.1.2 Small Diamond Search Algorithm (SDS)

SDS is a result of combining 3SS and SDSP. It is very similar to the LDS except that only four points form the diamond pattern in contrast to LDS where eight points constitute the diamond shape. The SDS algorithm was implemented in order to analyse the effect of computations on the speed of the estimation process and the quality of the resultant video.

The sequence of the SDS algorithm can be explained as follows, see Figure 5.2 for an illustration:

1) As in LDS, SDS also starts with measuring the block difference between the current block and the zero motion vector block.
2) With the centre being the zero vector point, search four points which are at the ends of a diamond. The match found in this iteration will serve as a centre for the next iteration. Halve the width of the diamond.
3) With the new centre, search four more points. Continue until the diamond reaches the minimum width, w=1. The final result is compared to the centre block (the zero motion vector block) and the block with minimum residual energy is considered to be the final best match.

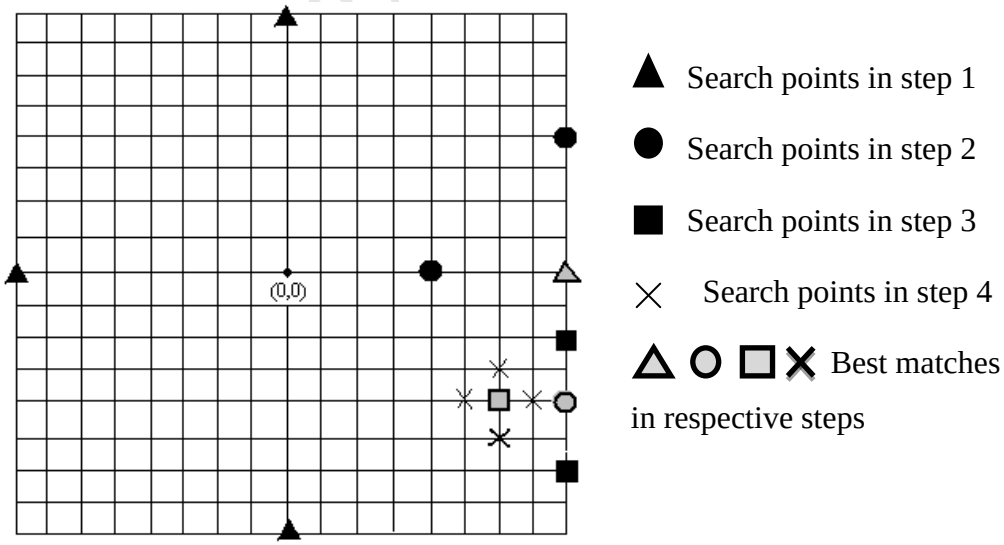


Figure 5.2: Small diamond search algorithm

5.2 Hierarchical algorithms that were implemented

Multi-resolution algorithms concentrate around the most probable area of finding the closest match for a block, which most of the times aids faster estimation. Two multi-resolution algorithms were developed and implemented for speed improvements. The algorithms were mainly hybrids of two or more algorithms discussed in Chapter 4.

The project version of hierarchical search (HS) employs full search at all the three levels where as the implemented multi-resolution algorithms employ a fast search at level 2 over the entire motion range, a 16x16 area rather than a 13x13 search window as in HS. Similarly a wider area is searched when a fast search is applied at level 1 (a 32x32 area) rather than a confined 4x4 area as in HS. These modifications were made to increase the possibility of finding the true best match and avoid the situation where the match found is a local minima.

5.2.1 Small Diamond Hierarchical Search Algorithm (SDH)

This algorithm was implemented as a primary effort to come up with a faster motion estimation algorithm. The main idea was to shrink the number of computations at the level 2 in HS algorithm to speed up the estimation. SDH was a result of combining SDS with the multi-resolution concept. In SDH, the highest level (level 2) is modified for the reason that one may still derive a similar coarse motion vector even with a fast search is applied (instead of a full search). This algorithm resulted in decreasing the search point count to a certain extent, thus resulting in faster estimation as expected.

Similar to HS, the implemented algorithms also have three levels of hierarchy for estimation. The calculation of the energy difference for the zero vector difference block is done prior to applying the algorithm. The result of the algorithm is matched to the zero vector difference block and then final minimum point is chosen.

The systematic explanation for SDH is as follows:

1) Start with the full resolutions of the current and the reference frames; this is level 0.Generate level 1 and then level 2 using averaging technique.

2) Employ SDS algorithm at level 2 with the width of the diamond equal to the motion range at level 2 (see Table 1). Halve the width for each iteration and converge the

diamond points around the best matches. Repeat the process until the minimum possible width is reached (w=1). The best match found in the last iteration will be the centre for the search process at level 1.

3) Compare the result of the algorithm with the zero motion vector block and derive the final best match for the current block.

The Figure 5.3 below gives a pictorial version of SDH. The grey area denotes the search area in each level. At level 2 the search is carried out in motion range (16x16 area) rather than search window (14x14 area), but for clearer illustration of the algorithm, the shaded area is limited to 2x2.

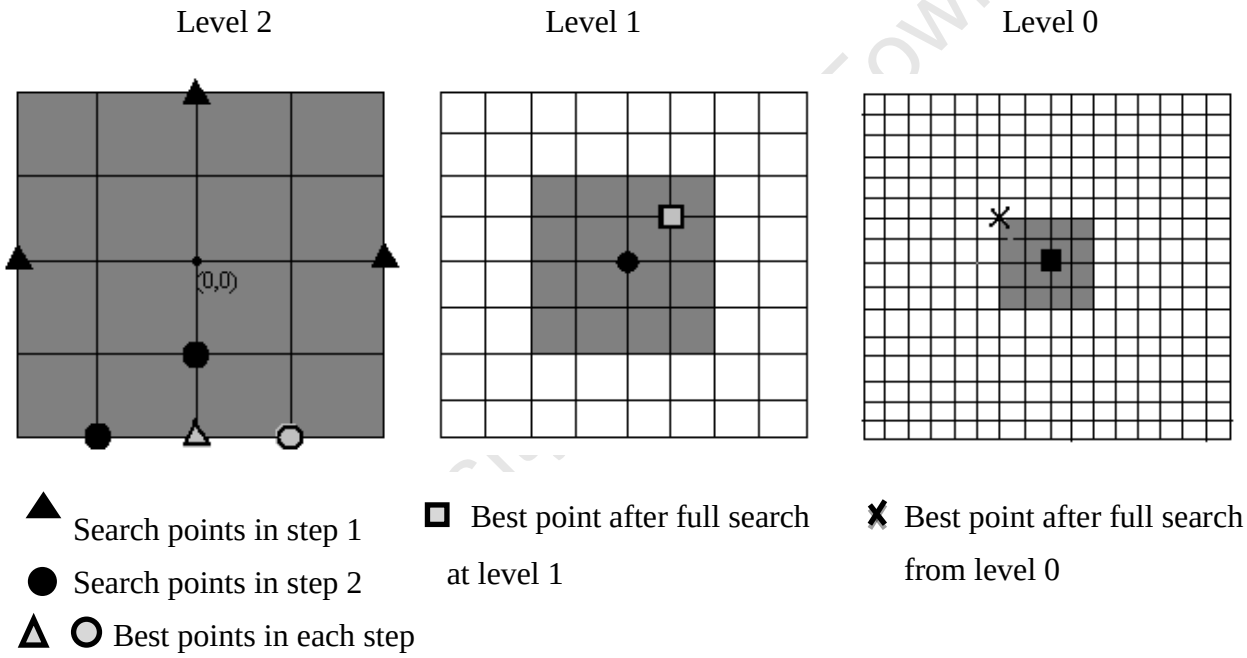


Figure 5.3: Small diamond hierarchical search algorithm

5.2.2 Two-Tier Fast Hierarchical Search Algorithm (TTHS)

This algorithm was experimented to study the effect and quality consequences of applying more than one fast search to the hierarchical design. It modifies the multi-resolution algorithm at two levels unlike the previous algorithms where the modification was done only at

one level, level 2. TTHS uses two different fast search patterns at two different levels: SDS at level 2 and grid search (which was mentioned in ESS) at level 1. The grid search in this algorithm can be appropriately referred to as a 'square' search pattern for the reason that the centre is always calculated separately and not included the search pattern. Similar to the previous algorithms there is a three-level hierarchy with the original resolutions at level 0.

Employing a fast search at level 1 called for balance on the quality of the output video as searching fewer points would mean decreasing the number of search points. To keep the search count higher than the SDS, a square search was considered. They were two reasons for preferring a square search pattern to any of the diamond patterns at level 1:

- To search a greater number of points (than SDS) in an iteration to find a better match, thus getting a better prediction for a given block.
- To accommodate a greater number of iterations to search more locations (LDS is not possible for w=1) for extracting a better match for a current block.

The TTHS is detailed as follows:

1) Apply SDS at level 2. Search four points in each step and halve the diamond width, therefore converge the diamond pattern onto the best match in every step. The best match in the last step forms the centre for the search at level 1.
2) At level 1 search eight points of a virtual square around the centre. The width of the square for the first iteration is equal to the motion range of the motion vector at level 1 (w=16). Halve the width of the square and search eight more points around the new centre. Repeat the process until the square reaches its minimum size (w=1). This is the last step and the best match from this iteration will be the centre for the search at level 0.
3) At level 0 perform a full search in the standard search window area. This is the final result of the algorithm.
4) Compare the result from the algorithm to the zero motion vector block, the block with lower residual energy amongst the two is the final match.

See Figure 5.4 for an illustration of TTHS algorithm. It can be noted that a larger area is searched at level 2 compared to previous algorithms which is equal to the area searched at level 1. Nevertheless, at level 0 a limited search window of 4x4 size is searched as in HS.

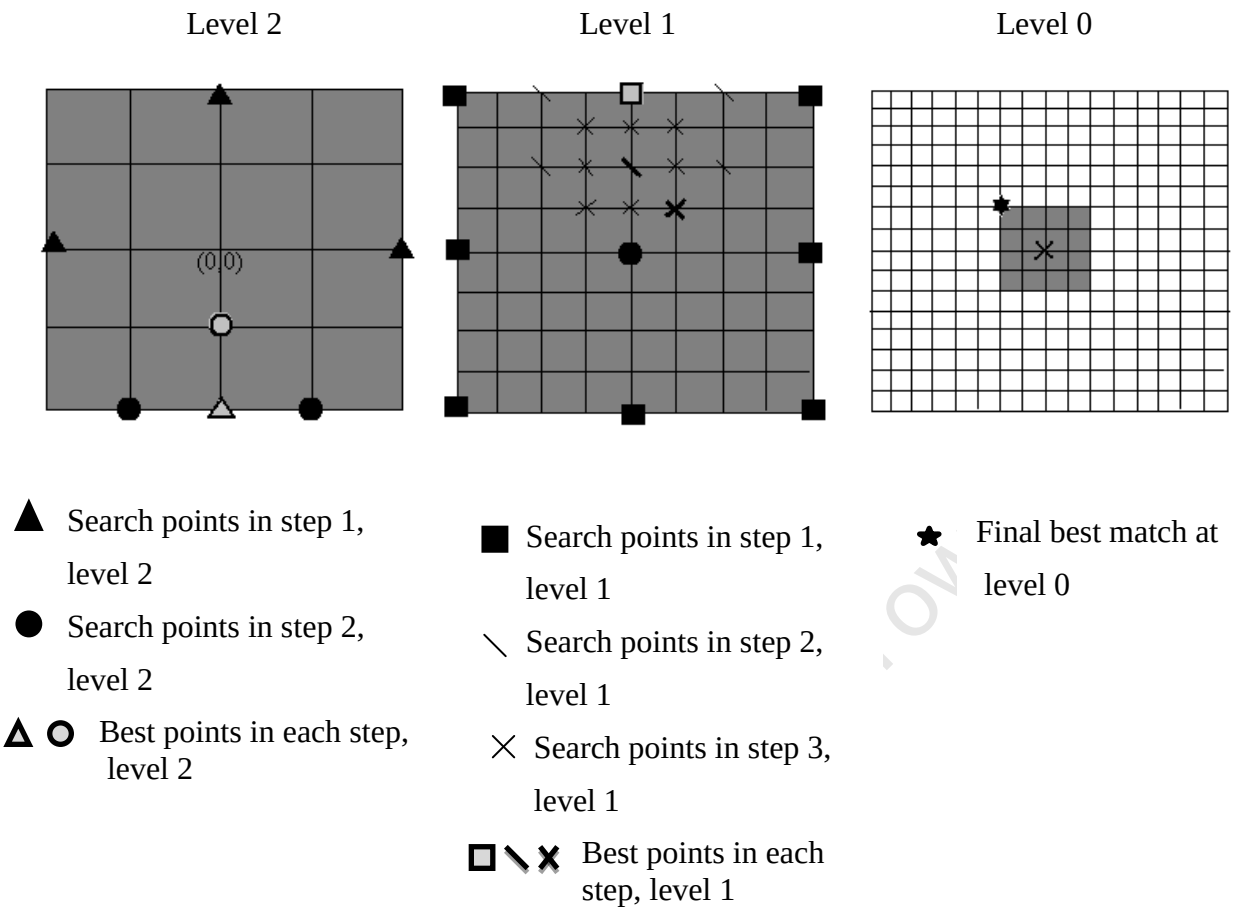


Figure 5.4: Two-tier fast hierarchical search algorithm

Considering the proposed algorithms, there are two factors that need to be justified. Firstly, it is evident that the majority of the proposed algorithms use diamond search as their main search pattern. The reason for selecting the diamond pattern over square pattern is based on the assumption that the video sequences are usually non-monotonic (random). The diamond pattern has points that are not equidistant from the centre, which makes the pattern more suited for random movements. Secondly, the limiting of the window to 14x14 or 15x15 to avoid the trapping of the best match in the local minima rather than achieving the global minima [23].

5.3 Chapter Summary

This chapter explained all the algorithms that were implemented in this research. It showed the relationship between the algorithms from the previous chapter and set forward the new algorithms, which were modifications of these ideas. Hierarchical and non- hierarchical algorithms were distinctively detailed following a step-by-step explanation with relevant figures. This chapter also mentions the reasons for choosing the selected implementations.

Chapter 6

6 Overview of Simulation Environment

This chapter presents the nature of the test bed in which the proposed algorithms (those discussed in Chapter 5) were simulated. In addition, it explains the project code architecture focusing on different code components and details the methodology adopted to attain the results. The chapter further focuses on the parameters that were used to evaluate the performance of the algorithms.

6.1 Test bed Specifications

The system used for all the implementations for the project was an LG Flatron desktop machine with Pentium (R) dual core CPU E5200@2.5GHZ, 1.98GHZ of RAM with Windows XP professional Version 2002 operating system (OS).

Microsoft Visual Studio 2008 was chosen as the IDE (Integrated Development Environment) for compiling and development in the project space. Microsoft Visual Studio is an IDE from Microsoft, which is used to develop console and graphical user interface applications along with Windows Forms application [26]. The project was written in C++ programming language exploiting the object oriented programming (OOP) concepts and thus the implementation also followed the C++ script [27]. Visual Studio compiler was used to run and analyse the code on the windows platform. The results were plotted in Linux platform using gnuplots.

6.2 Software Components in the project

The project package was a collection of all the tools and elements that constitute the entire process of coding and decoding of a video sequence. The MSVS IDE supported the project package and allowed additions on the implementations. The project mainly used the H.263 codec for simulation purposes. The complete package of files and classes (received from CSIR for research purposes) can be divided into four groups:

a) Build set:

This set contains all the application files pertaining to the 'debug' and 'release' modes in the compiler. Codec Analyser that is the application to evaluate different algorithms is also in this build set. There are a few linker files to link the complier to the application and the project. This build set also contains extension files which allow the project to be updated from H.263 to H.264.

b) Library file set:

This is a set of the library files that were provided to carry out implementations. This set consists of all the library files, general utilities, codec utilities and image utilities to perform encoding and decoding procedures on images.

c) Project file set:

The notable file in this pack is the launcher; it was used to launch the Microsoft visual studio through the command prompt window. It also contains other VC++ projects which are more image-related than video-related.

d) Source file set:

This forms the major sector in the project that includes all the fragments of the video coding and encoding process. The source code is further classified into three groups:

- Codec Analyser: An application wherein all the classes and implementations regarding the settings and functioning of the codec analyser application are written.
- Codecs: This contains standard implementations based on H.263 and H.264 codecs. This also contains the utility files of the codec which essentially include all the files which handle the process pertaining to coding-decoding such as sampling, estimation, compensation, quantisation, entropy coding etc.
- RtvcLibs: The name 'RtvcLibs' is purely project specific and it contains classes and implementations which define image related functions (block patterns, luma patterns, memory handling for the data etc.).

6.3 Source code Architecture

The project code is designed in a manner such that it is both effective yet flexible for changes and additions. This part of the thesis explains the architecture of the code and various notable classes included in the source code.

6.3.1 The Main Class

The backbone of the source code is the main class, which is a H.263 codec based implementation. This class serves as a software codec and can access all the codec related classes .This class configures various image objects such as image memory, macroblock objects etc. and functional objects such as motion compensator, filters and colour converter.

The functions of this main class include:

- Allocating image memory for luma data followed by Cb and Cr.
- Configuring overlays (explained under next immediate heading).
- Configuring macrobolck data objects like macroblock height and macroblock width.
- Configuring colour convertors.
- Instantiating DCT filters for transform coding and create the coefficient quantisers.
- Configuring motion estimators and motion compensators.
- Configuring image plane encoders and decoders.
- Instantiating the RLE.
- Configuring VLC encoders and decoders.
- Configuring bit streams.

6.3.2 The Overlay Classes

The motion estimation process involves performing searches and often relocating the origin of the search to different points around the reference area. In hierarchical searches, this relocation process gets complex due to the changing sizes of the macroblocks and the search

windows in each of the hierarchy levels. Overlay classes are included in an effort to make the motion estimation process simpler.

An overlay class has fixed parameters for example, macroblock height, macroblock width and motion range etc. and 'latches' onto the reference and input blocks. The search proceeds as per the dimensions in the overlay class. Thus, a motion estimation class performs search in two steps firstly, it 'overlays' the search grid upon the input and the reference areas and secondly, it performs the search on the overlay to find the minimum point. To accommodate multi-resolution algorithms, the code contains three different overlays for each level – level 0, level 1, and level 2 overlays. It is worth noting that the input and reference blocks have the same dimensions and macroblock sizes throughout but the positioning of the overlay brings the input and the reference blocks to the different levels of resolution by using averaging technique. However, when the algorithm is not a multi-resolution type, the standard overlay (a level 0 overlay) is in use. The illustration below (Figure 6.1) explains the setting.

More importantly, it is to note that only one overlay is overlaid at any point of time generating a single hierarchy level at that instance. The overlays make the estimation simpler as the size of the blocks is now taken care by the overlays and the estimator need not keep track of the levels. These overlay classes can also be appended with an additional functionality called 'extended boundary'. This would increase the boundary by one block size (in the respective level) such that the macroblocks at the boundary are always included in the search even when a part of them has exceeded the normal boundaries.

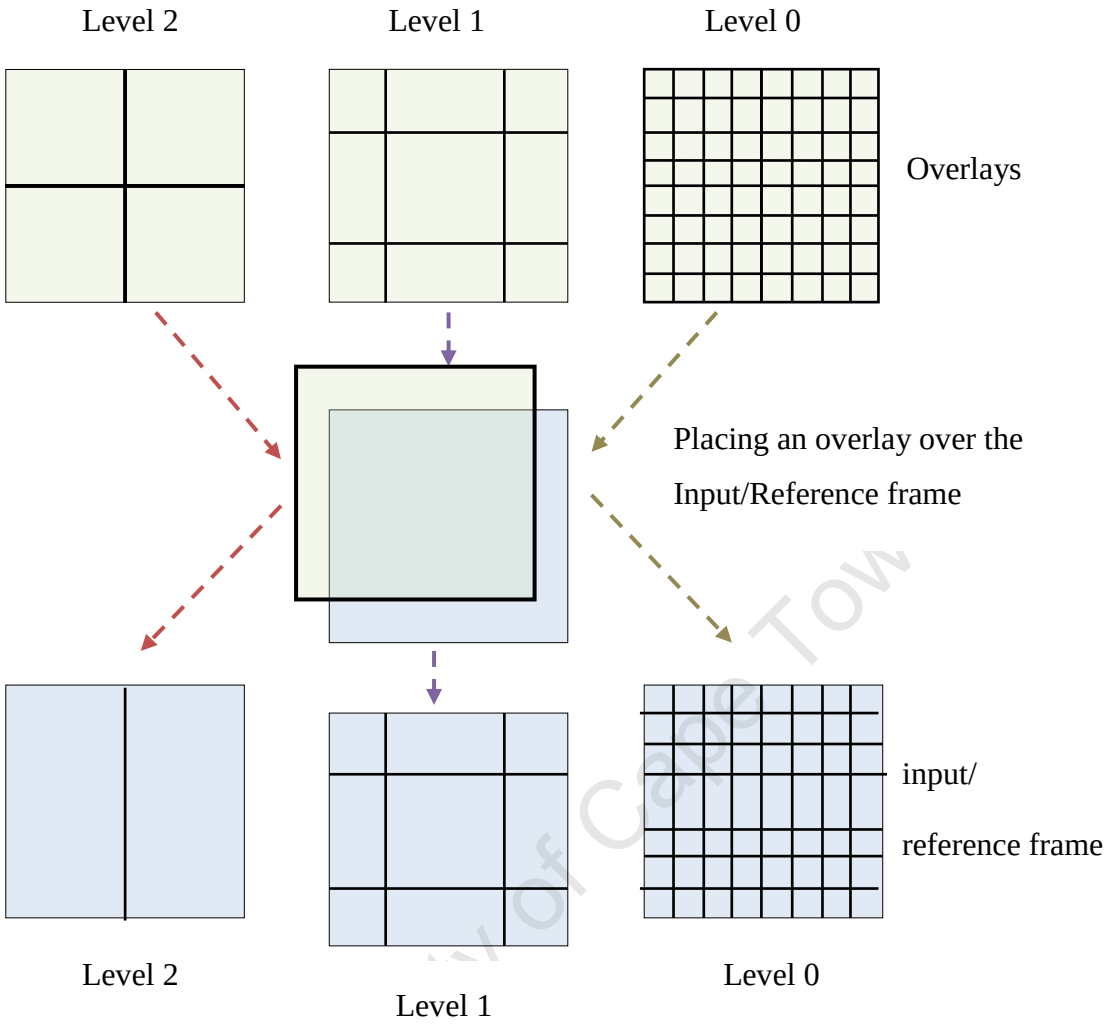


Figure 6.1: Overlaying onto the input/reference frames to generate levels of hierarchy

6.4 Measurement Parameters

This section defines the factors that were considered in analysing the efficiency of the different proposed algorithms. To study the performance of the algorithms two main attributes were under consideration - the speed of the encoding/decoding process and the quality of the video at the receiving side. Generally, the following metrics are considered for analysing motion estimation algorithms: speed (for a constant set of frames), Peak Signal to Noise Ratio (PSNR) and Quality factor (Qf) [28], [5].

6.4.1 Speed of the algorithm

The speed of the algorithm is analysed by studying the time lapse during the whole process of encoding and decoding a video. The units used for the time measurement are milliseconds (ms). It should be noted that the simulation equipment is capable of giving the total time for the entire process rather than just the time taken for the estimation process. However, the total encoding and decoding time is considered as motion estimation and compensation processes contribute the majority of the time in the entire encoding and decoding process for a video sequence. Figure 6.2 below shows an example of a typical speed-reading during the experiments.

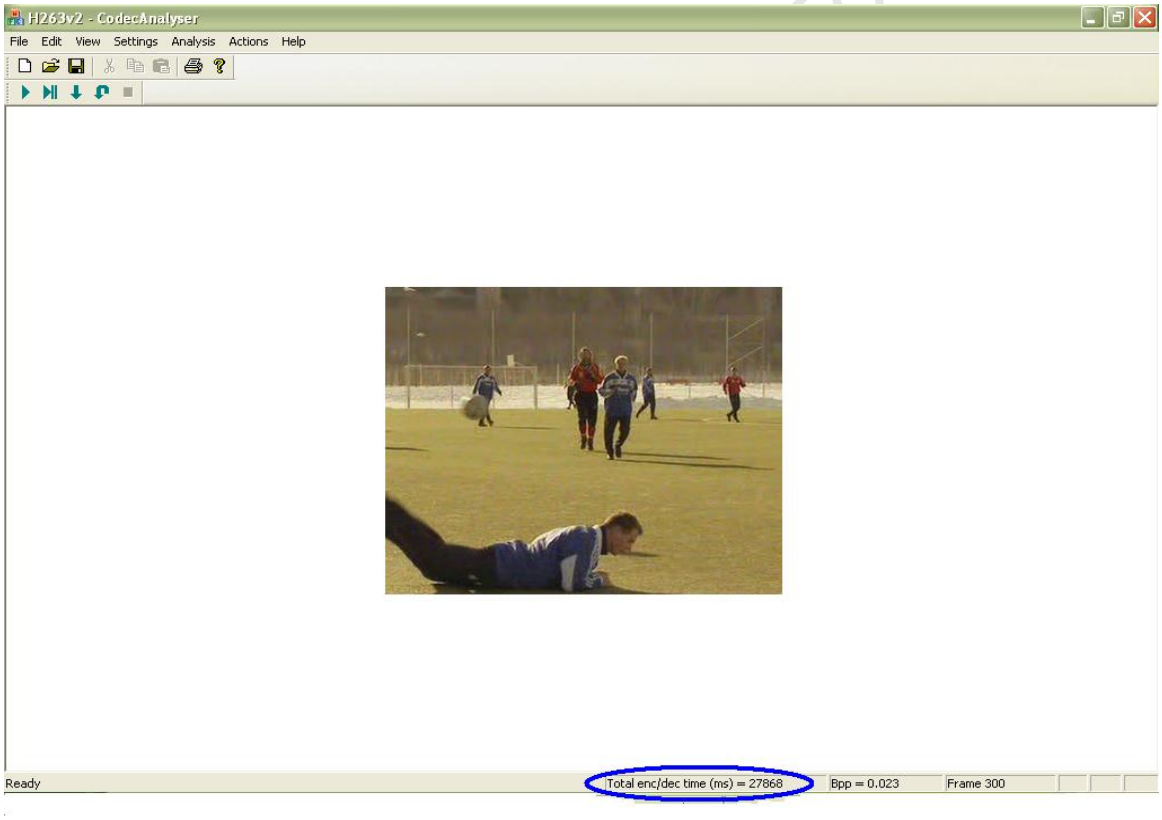


Figure 6.2: Total encoding/decoding time measurement in codec analyser

6.4.2 Peak Signal-to-Noise Ratio (PSNR)

The PSNR is a general quality metric in the analysis of a video [29]. The PSNR in video compression is the ratio of the highest possible quality for a number of bits to the error between the original and the reconstructed version. In other words, it is the ratio of the peak signal possible to the noise occurred due to the compression process. Mathematically, PSNR is given by the formula, see equation 6.1 and is measured in decibels (dB) [1].

$$\mathrm{PSNR}_{\mathrm{dB}} = 10 \log_{10} \frac{(2^{\mathrm{n}}-1)^2}{\mathrm{MSE}} \qquad (6.1)$$

where, $(2^{\mathrm{n}} - 1)^2$ is the square of the highest signal possible given n is the number of bits per image sample and MSE is the mean error between the original and the impaired image.

The Table 6.1 below gives the relation between theoretical calculations and subjective opinions over the received data. Different ranges of PSNR yield different Mean Opinion Scores (MOS) [30], [31] meaning a higher PSNR would yield more viewer satisfaction.

Table 6.1: Mapping of MOS with PSNR range [30]

MOS rating of the video	PSNR (dB) Range
Excellent	>37
Good	31-37
Fair	25-31
Poor	20-25
Bad	<20

Although PSNR is widely used for its simplicity [29], it cannot be perceived as a best metric for validating the 'real feel' due to the non-linear behaviour of the human eye for being more sensitive to lower frequencies than higher frequencies [31], [32]. However, [32] states that PSNR can be considered a good metric when the content and the codec are fixed over the entire process, for example when the same video is being assessed over the same codec but cannot be used to compare quality across different video contents or different codecs.

6.4.3 Quality factor (Qf)

In low bit rate scenarios, the number of bits transmitted to depict a single frame (or even a pixel) plays an important role. Theoretically, the fewer number of bits sent, the faster is the transmission however, fewer bits also means degraded information. For example, if only one bit is used to represent a sample, it would be either 1 or 0 resulting in depicting either black or white. On the other hand if two bits are used, one can represent have four values: 00, 01, 10, and 11 that can be any of the four shades of grey [33].

The Quality factor can be defined as the number of bits assigned to denote each pixel in a given video sequence. Qf for a given bit rate (bits/second) for a specific video resolution can be calculated by:

$$Qf = \frac{b}{v*h*f} \qquad (6.2)$$

Where Qf is the quality factor,b is the bit rate, vand h are the vertical and horizontal resolutions respectively and f is the frame rate of the video. Practically, the term 'v ∗ h ∗ f' points to the video quality and it is evident from equation 6.2 that $Qf \propto \frac{1}{\text{video quality}}$ thus, a lower value of Qf indicates higher bits/pixel condition which in turn renders higher video quality. The Figure 6.3 shows the quality of a video frame at Qf= 1 and Qf= 31.

Video quality at Qf = 1 Video quality at Qf = 31

Figure 6.3: Frame qualities at different quality factors

Although Qf is widely used for estimating video quality, it is not an ideal metric for comparing between video sequences as different video sequences may require more or fewer bits to have the same resolution. For example, a busy video sequence may require a greater number of bits for a specific quality than a subtle one. But once again like PSNR it gives an appropriate analysis for this setup where a single video sequence encoded using different algorithms but a given number of bits/pixel or Qf.

6.5 Method of Simulation

The first step in the research was to develop different codes for different algorithms with the core search pattern embedded in a function. Each algorithm thus was a separate implementation of the main class and for the main class to access the algorithms, instances of these classes were created. An example of the instantiation is shown in the code below, see Figure 6.4.

```
_pMotionEstimator = new MotionEstimatorSmallDiamond(  (const void *)_pLum,

                                (const void *)_pRLum,

                                _lumWidth,

                                _lumHeight,

                                motionVectorRange);

  _pMotionEstimator = new MotionEstimatorLargeDiamondMultiRes(   (const
void *)_pLum,

                                (const void *)_pRLum,

                                _lumWidth,

                                _lumHeight,motionVectorRange);........
```

Figure 6.4: Script for instantiating new implementations

Following the instantiation, the motion estimation class then would perform the estimation using the embedded algorithm as a part of the complete encoding process in the main class Therefore, project solution is compiled and built leading to the appearance of 'Codec Analyser' window as shown in the screenshot, see Figure 6.4.

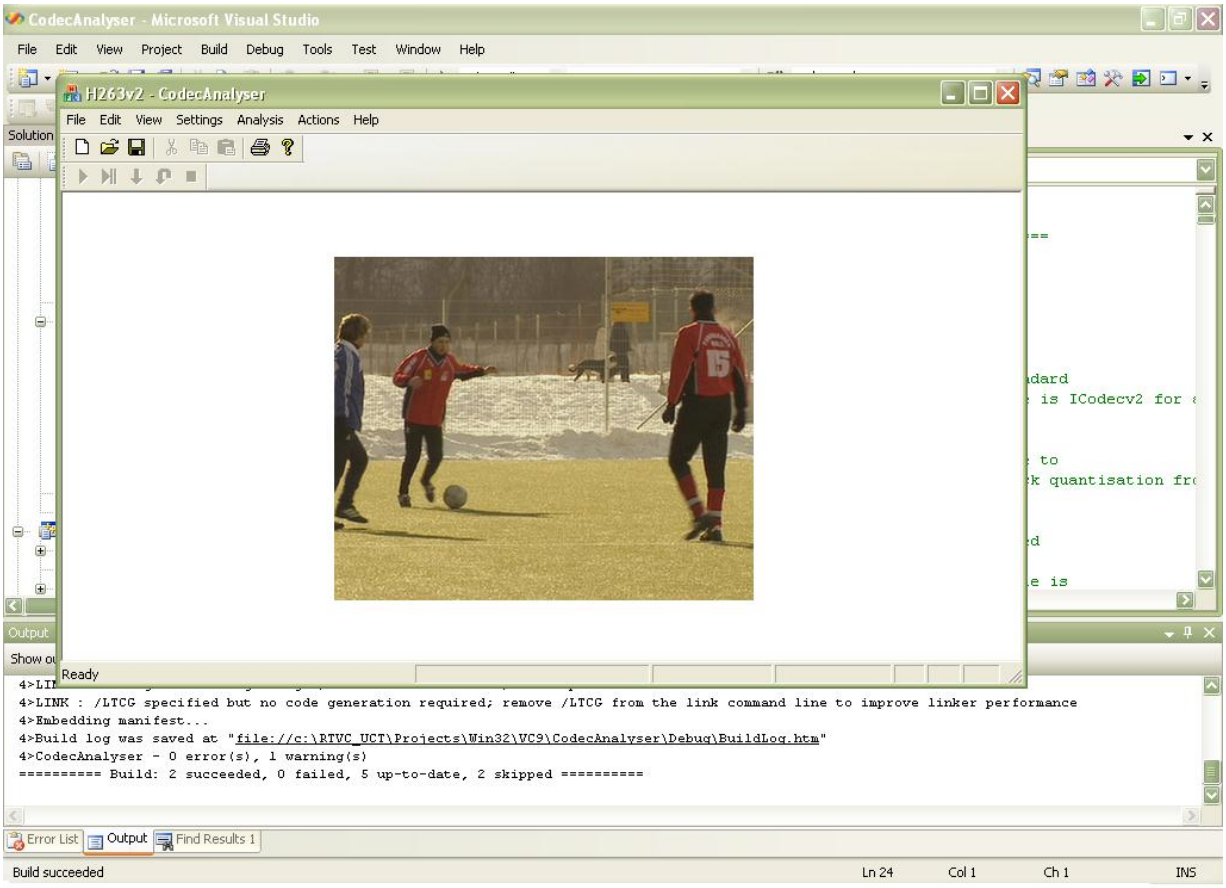


Figure 6.5: Successful build scenario leading to the codec analyser window

The window shows a successful compile and build scenario that leads to the codec analyser window, which is ready for simulations. More details on code analyser follow under the next heading.

6.5.1 Codec Analyser

The 'Codec Analyser' is a software application provided in the project for simulation and analysis of parameters regarding the encoding and decoding process. It exists as a run-time class and various implementations inside this class further define the properties and settings of this application. The codec analyser facilitates the use of both H.263 and H.264 standards for simulating purposes and allows the user to manage the input parameters. The typical parameters set during the simulation are shown in the Figure 6.6.

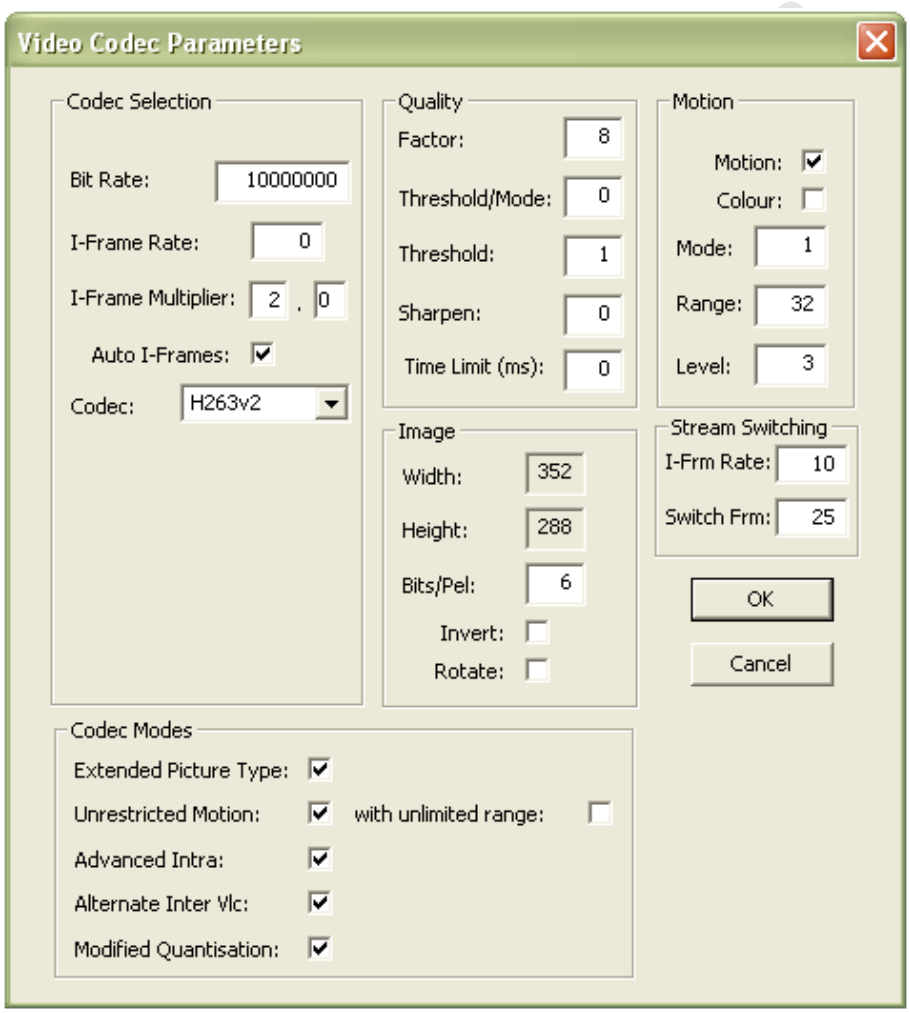


Figure 6.6: Video codec parameters

During the simulation, the above shown parameters were constants throughout the simulations except the bit rate and Qf. The adjustments in bitrate (mostly increasing the bitrate) were either to aid the coding of a bulky video sequence or to study the output at superior Qf (s).

Analysing a file starts by selecting that particular video file into the codec analyser window and continue by using the 'analyse' tool in the codec analyser. The codec analyser application is linked to the main class that steers the video compression process through various sub-processes according to the selected instances in the main code. Thus, the codec analyser application then simulates the entire encoding-decoding process on that video sequence with the set codec parameters. In the end. the codec analyser presents the sum of the encoding and decoding time for that particular video sequence at those parameters, see Figure 6.6. Finally, the codec analyser stores the list of PSNRs and bits per pixel (bpp) values for each frame in the video sequence in a data file.

6.6 Method of Generating the Results

The experiments were conducted with luminous components of three standard video sequences: "Foreman" (352X288, 30 frames/second (fps)), "Soccer" (352X288, 30 fps) and "Mobile" (352X288, 30 fps) having 300 frames each [34]. The reason for selecting the above sequences is that they vary in their characteristics as; the "Foreman" sequence is relatively less active than the "Soccer". The "Mobile" sequence is different from the first two with slow moving objects but involves complex motions. This created an opportunity to study the efficiency of different motion estimation algorithm over wider range of input videos.

Each of the videos was tested in permutation with all the proposed algorithms (LDS, SDS, SDH and TTHS) and summate of the encoding through decoding time was noted for all the combinations. The PSNR and bpp were individually averaged over 300 frames for each video sequence over a range of Qfs from 1to32. Following the calculations, Rate - Distortion (RD) graphs [1], [35] having PSNR as the y-axis and bpp as the x-axis were generated for each algorithm to compare their performances of each algorithm. The analysis was done from two different perspectives firstly by comparing all the proposed algorithms for the videos and secondly by comparing only the hierarchical algorithms.

6.7 Chapter Summary

This chapter provided all the hardware and software details in the test bed environment. Each of the code components were explained including codec analyser that was a simulation tool/application used. Metrics of interest were discussed and appropriate reasons were given for choosing them in this research. The chapter also presented the systematic approach for analysing the algorithms and the theory was supported by related screenshots for better understanding. The chapter concluded by giving information on the input video sequences used and on the formatting of the readings in a presentable manner.

Chapter 7

7 Results and Analysis

This chapter presents various simulation results over the parameters for different algorithms that were discussed in the previous chapter (speed, PSNR, Qf and bpp). The quality and performance tests are carried out on all the three video sequences ("Foreman", "Soccer" and "Mobile") and a comparative study was made between the existent algorithms (including the FSA) and the proposed motion estimation algorithms. Furthermore, the results are presented in a second perspective where the non-hierarchical algorithms are compared with the hierarchical algorithms to study the influence of the hierarchy technique on the performance of an algorithm. In addition to the presentation of results, this chapter also assesses each of the retrieved results with the explanation of possible reasons for the specific behaviours of the algorithms.

7.1 Total Encoding and Decoding time: Comparison and Analysis

As previously mentioned, (in Chapter 6) this research takes into account the time taken for the entire encoding - decoding process to analyse the speed of the algorithm. The speed of an algorithm is principally dependent on the number of computations that are done for each block in the current frame. The larger the number of search points, the more computations are needed leading to a longer estimation time. In an algorithm, the exact numbers of computations performed are uncertain because a block may require more or less number of searches in the due course of finding the match. Thus, the numbers of computations vary between the maximum and minimum search points for that specific algorithm. Table 7.1 shows the maximum number of computations that can take place in an algorithm for finding the best match for a macroblock, and was tailored for both the existent and proposed algorithms.

Table: 7.1: Maximum possible computations for each macroblock in different BMAs

BMA	Maximum possible computations for one macroblock
Full Search Algorithm (FSA)	256
Efficient Three-Step Search Algorithm (E3SS)	33
Large Diamond Search Algorithm (LDS)	25
Small Diamond search Algorithm (SDS)	17
Project version of Hierarchical Search Algorithm (HS)	228
Small Diamond Hierarchical Search Algorithm (SDH)	49
Two-Tier Fast Hierarchal Search Algorithm (TTHS)	72

From the Table 7.1 it is evident that FSA performs the largest number of computations than all other fast algorithms and HS performs the most number of computations amongst the hierarchial algorithms. The total time for encoding and decoding is expected to be proportionate to these values and the relevant simulation results are presented in the forthcoming sections.

To assay the data from various perspectives, the results are segregated into non-hierarchical and hierarchical segments.

7.1.1 Non-Hierarchal Algorithms

The overall encoding and decoding time was noted for all the test video sequences. The plots in this section show summate of the encoding through decoding time (for all the 300 frames) with different non-hierarchical algorithms. The graph is plotted for various timing values (in milliseconds (ms)) over a range of quality factors (Qf). Figure 7.1 is plotted for the "Foreman" sequence.

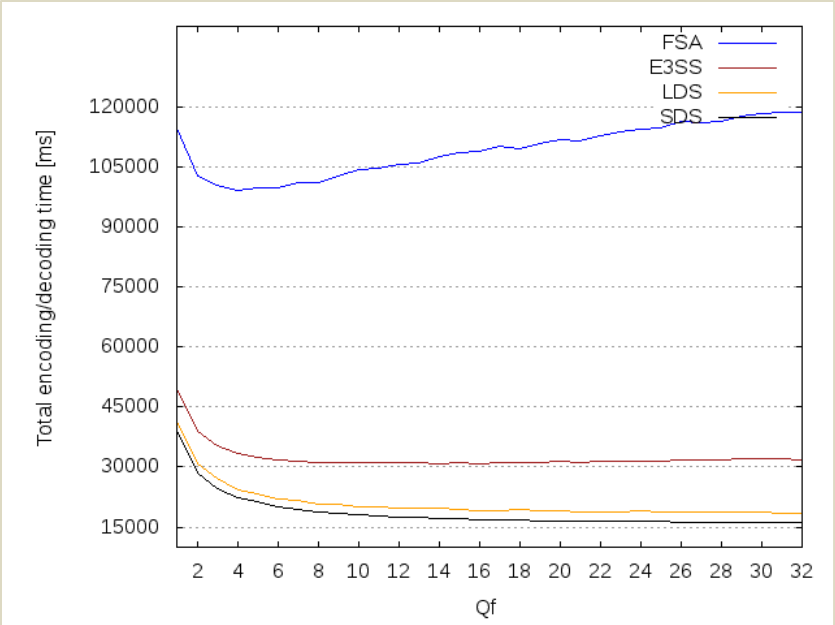


Figure 7.1: Total encoding and decoding time for "Foreman" using non-hierarchal algorithms

Considering the non-hierarchical batch in the Table 7.1, it is noted that FSA has the most number of computations followed by E3SS, LDS and SDS where SDS has the least number of computations. The Figure 7.1 testifies that the number of computations is analogous to the speed of the algorithm. The video encoding and decoding process is slowest when FSA is employed for motion estimation process and is fastest when SDS is employed. However, the time taken by FSA increases with increase in Qf and the possible reason for this unusual behaviour is that FSA is performing extra computations at the boundaries. To emphasise the relation between the number of computations and the speed of the algorithm the simulation is carried out for the remaining test videos. Figure 7.2 presents the results for “Soccer” video sequence and Figure 7.3 for “Mobile” sequence.

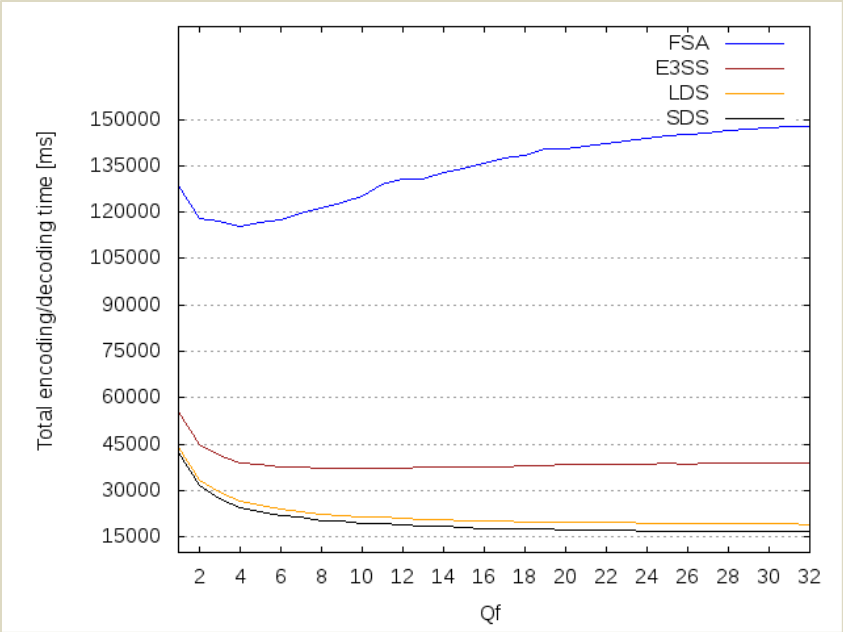


Figure 7.2: Total encoding and decoding time for "Soccer" using non-hierarchal algorithms

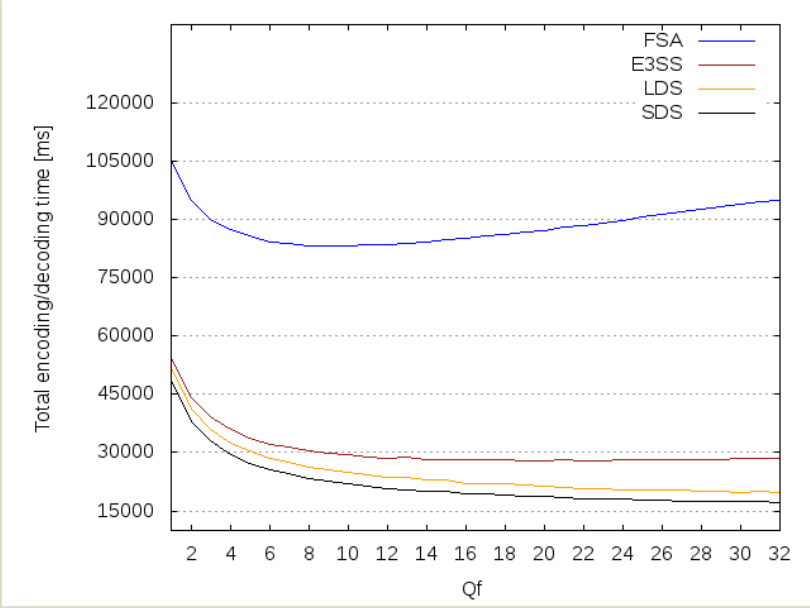


Figure 7.3: Total encoding and decoding time for "Mobile" using non-hierarchal algorithms

It can be seen that Figure 7.2 and Figure 7.3 are similar to Figure 7.1, which establish the affiliation between the search count and the speed of the algorithm. Thus, decreasing the search count results in faster motion estimation and in turn decreases the total encoding and decoding time. The proposed algorithm SDS proves to be the fastest among all the non-hierarchical algorithms in this project for any type of video input.

7.1.2 Hierarchical Algorithms

As discussed earlier (in Chapter 5 and 6), the hierarchical algorithms in this research perform computations at three different resolutions for each macroblock and the search points at each level account for the total number of computations in that particular hierarchical search algorithm. In Table 7.1, it is shown that HS has the maximum number of computations among all the hierarchical algorithms and the proposed algorithms (SDH and TTHS) reduced the computations at specific levels by incorporating fast searches. Speed simulations were conducted to study their impact on the total time taken for the entire coding-decoding process. Figures 7.4, 7.5 and 7.5 show the results for “Forman”, “Soccer” and “Mobile” sequences respectively.

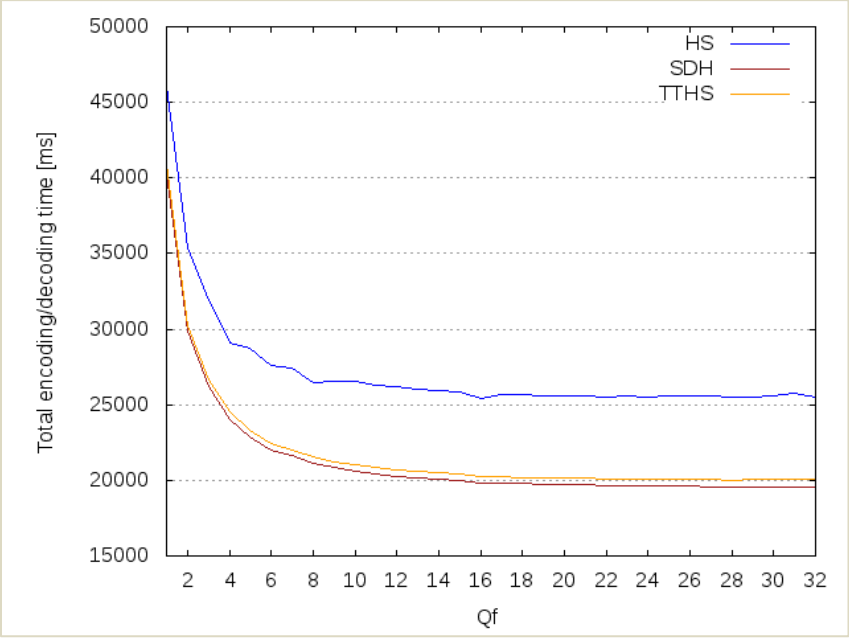


Figure 7.4: Total encoding and decoding time for "Foreman" using hierarchal algorithms

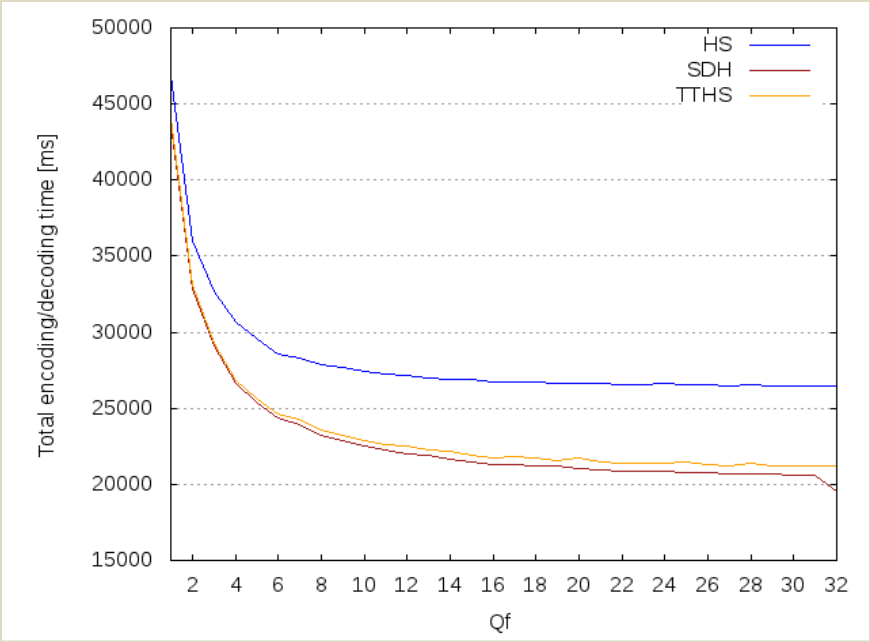


Figure 7.5: Total encoding and decoding time for "Soccer" using hierarchal algorithms

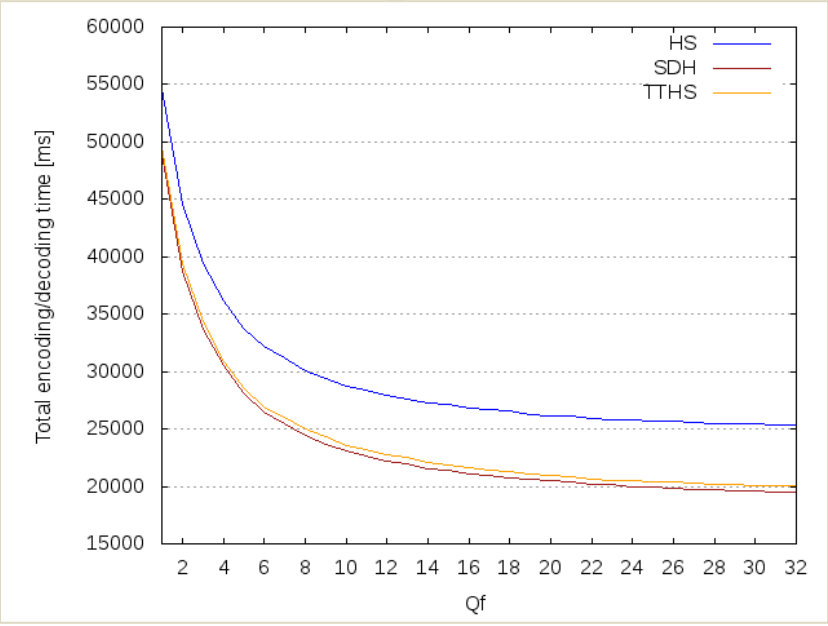


Figure 7.6: Total encoding and decoding time for "Mobile" using hierarchal algorithms

From the graphs presented in the Figures 7.4, 7.5 and 7.6, it is evident that HS, which performs a greater number of computations, takes more time to estimate than the two proposed hierarchical algorithms. SDH and TTHS are faster due to their reduced number of search points. Although TTHS performs fast search at two levels, this algorithm is slower than SDH because it searches greater number of blocks than SDH (see Table 7.1). The effect of decreased search points is consistent over all the three test video sequences.

7.1.3 Non-Hierarchical vs. Hierarchical

To further study the capability of the hierarchical algorithms, there is a need to compare the non-hierarchical algorithms with hierarchical algorithms. This section compares the fastest algorithm of non-hierarchical algorithm, the SDS with the fastest of hierarchal search algorithms, the SDH. The Figures 7.7 present the simulation results for "Foreman" video sequence.

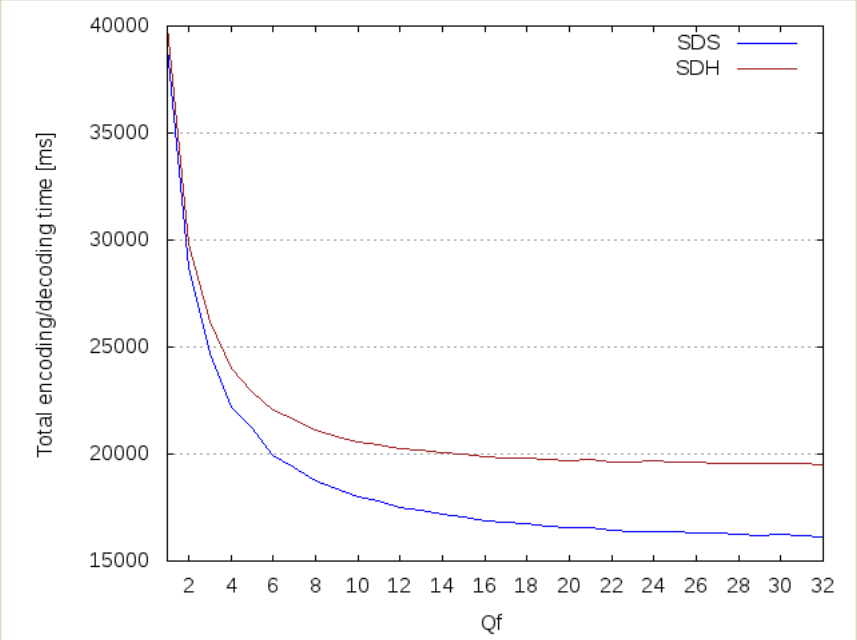


Figure 7.7: Total time comparison: SDS vs. SDH

It can be seen from the above figure that the hierarchical algorithm is slower than the non-hierarchal algorithm. The reason for this behaviour by SDH is due to the additional computations

done at different resolutions. Thus SDS is apparently the fastest of all the pre-written and the proposed algorithms in this research.

7.2 Qualitative Performance: Comparison and Analysis

In addition to the speed of the algorithm, the quality of the video at the destination also determines the efficiency of the algorithm. The qualitative performance of an algorithm is estimated by two parameters: the first one is the Peak Signal to Noise Ratio (PSNR) which estimates the noise (or distortion) due to applying compression techniques and second one is the bits per pixel (bpp) which estimates the rate of compression by the number of bits required to represent each pixel. The bpp is proportional to the total number of bits transmitted to denote required information in a frame. This section presents Rate – Distortion (RD) curves for different algorithms. The RD curve plots PSNR against bits per pixel (bpp) and is generally a standard depiction to analyse the qualitative performance of an algorithm.

7.2.1 Non-hierarchal Algorithms

The RD plots for the non-hierarchical algorithms are presented for "Foreman", "Soccer" and "Mobile" test sequences. The speed of an algorithm is usually disproportionate to the performance of the algorithm for the reason that the algorithms that are more accurate tend to be slower than the less accurate ones. Figures 7.8, 7.9 and 7.10 illustrate the simulation results for "Foreman", "Soccer" and "mobile" sequences respectively.

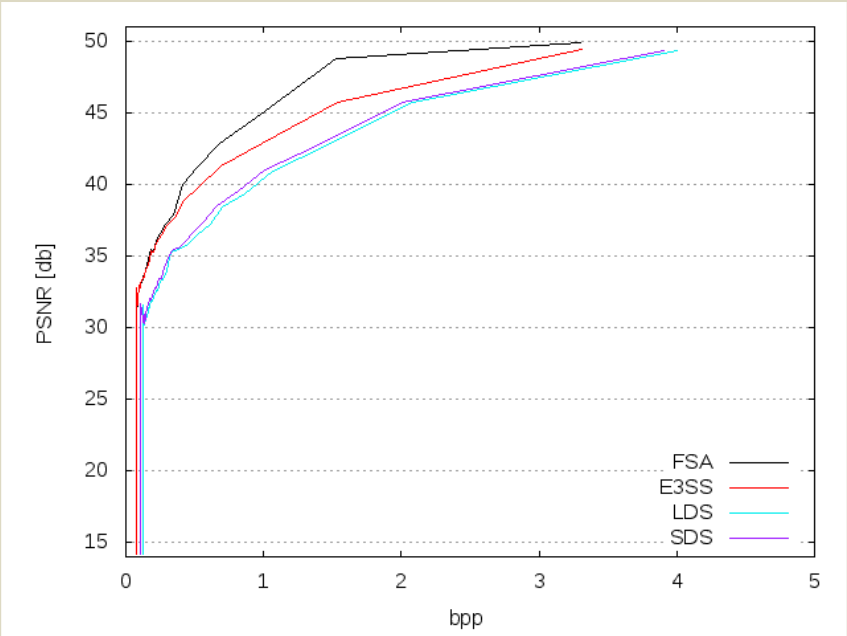


Figure 7.8: RD curves of non-hierarchical algorithms for "Foreman"

The RD chart in Figure 7.8 shows that FSA has the best performance for an obvious reason that is it searches every possible block in the reference frame for every block in the current frame. Although E3SS is a fast algorithm, it performs quasi to FSA for the "Foreman" series. Further, it is also evident from the graph that SDS outperforms LDS despite the fact that SDS searches lesser number of points. The reason for this behaviour of LDS is its search pattern. LDS searches eight points for every iteration but fails to search the adjacent points of the zero motion vector point as the iterations end at w=2, thus the points which are distanced at w=1 are left unsearched. This forms a bottleneck for LDS as the possibility of finding the best match is generally around the zero difference point. On the other hand, SDS searches a lesser number of points but the neighbouring points of the centre (zero motion vector point) are searched as SDS accommodates search at w=1. Figure 7.9, presents the performances of the non-hierarchical algorithms with "Soccer" and Figure 7.10 with "Mobile "Sequence.

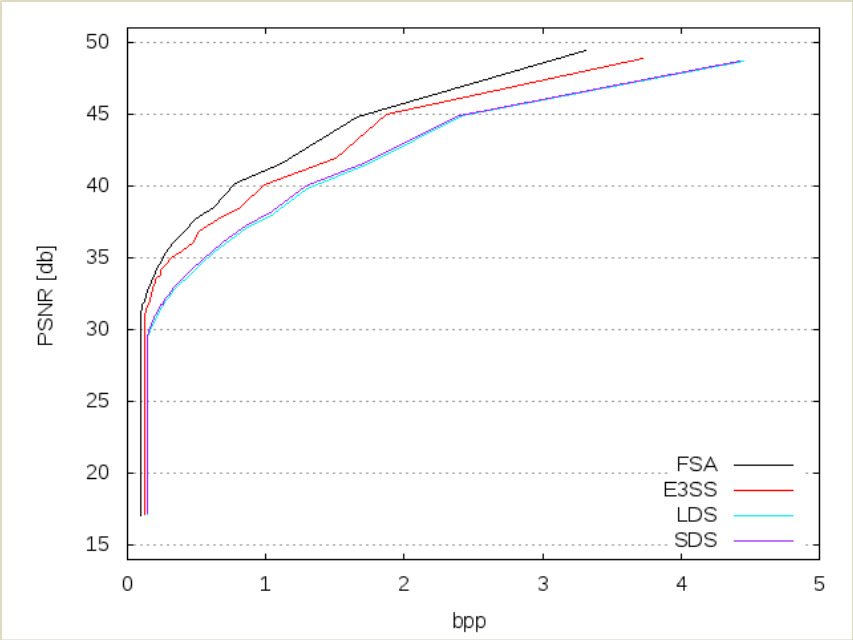


Figure 7.9: RD curves of non-hierarchical algorithms for "Soccer"

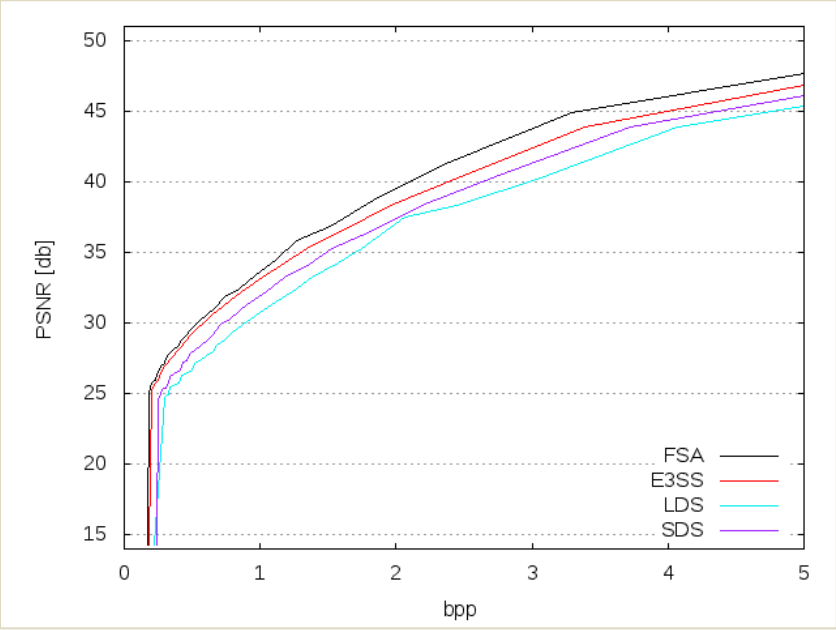


Figure 7.10: RD curves of non-hierarchical algorithms for "Mobile"

For all the three video sequences, FSA has explicitly better performance see, Figure 7.8, 7.9 and 7.10 however if only fast algorithms are considered then E3SS has better performance than SDS or LDS. Further, it is notable that the bpp for FSA and E3SS is significantly lower than SDS and LDS with LDS having marginally higher bpp rates than SDS due to lower PSNR counts.

7.2.2 Hierarchical Algorithms

This segment is on the conduct of various hierarchal motion estimation algorithms. Theoretically, a faster algorithm has low performance and higher bpp rates. Figure 7.11 illustrates the RD chart for "Foreman" sequence. It can be seen from the RD chart that HS has more signal rates than other two algorithms that employ fast searches. TTHS incorporates a two tier fast search that has significant noise component when compared to SDH and HS.

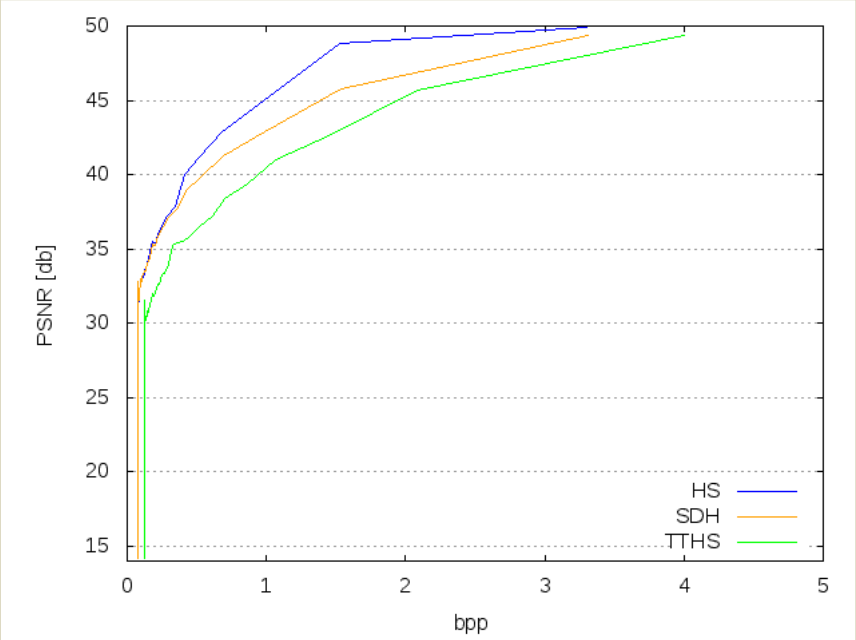


Figure 7.11: RD curves of hierarchical algorithms for "Foreman"

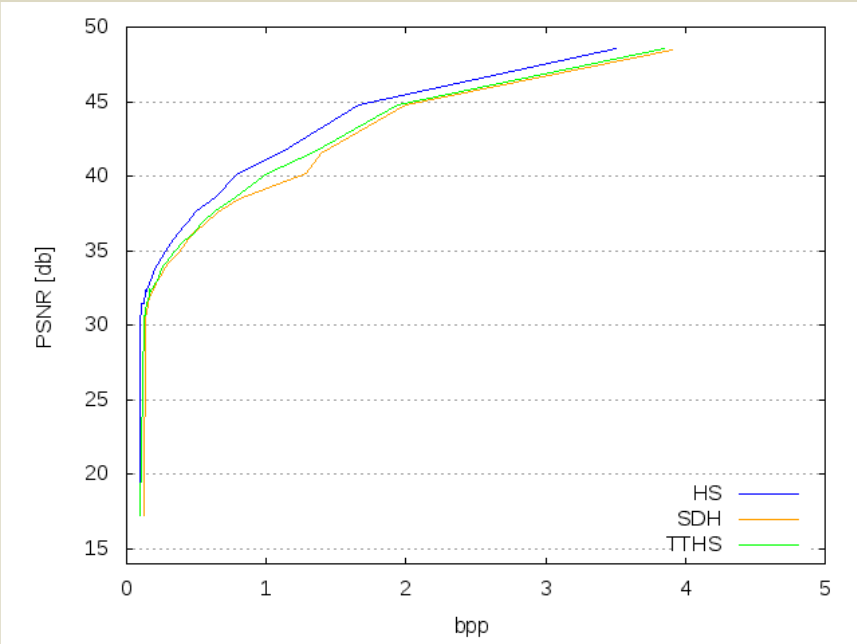


Figure 7.12: RD curves of hierarchical algorithms for "Soccer"

Figure 7.12 replicates the throughput of algorithms when a faster video sequence, in this context being "Soccer" is fed as input. The graph presents E3SS as a better algorithm than LDH and SDH. However, it is worth noting that LDH has higher PSNR rate at a given Qf due to its wider search areas in two levels yet HS makes a better RD curve because of its bpp rates.

Although SDH also employs a wider search window size at preliminary level (level 2), it confines itself at the next levels. Thus, SDH has two drawbacks – firstly reducing the number of search points that reduce the PSNR and secondly confining the search in the lower levels which hinder searching extended areas. On the other hand, even though HS does not employ any fast search it retains its efficiency due to the full search at preliminary level that yields a better bpp rate. This explains HS giving better PSNR than SDH.

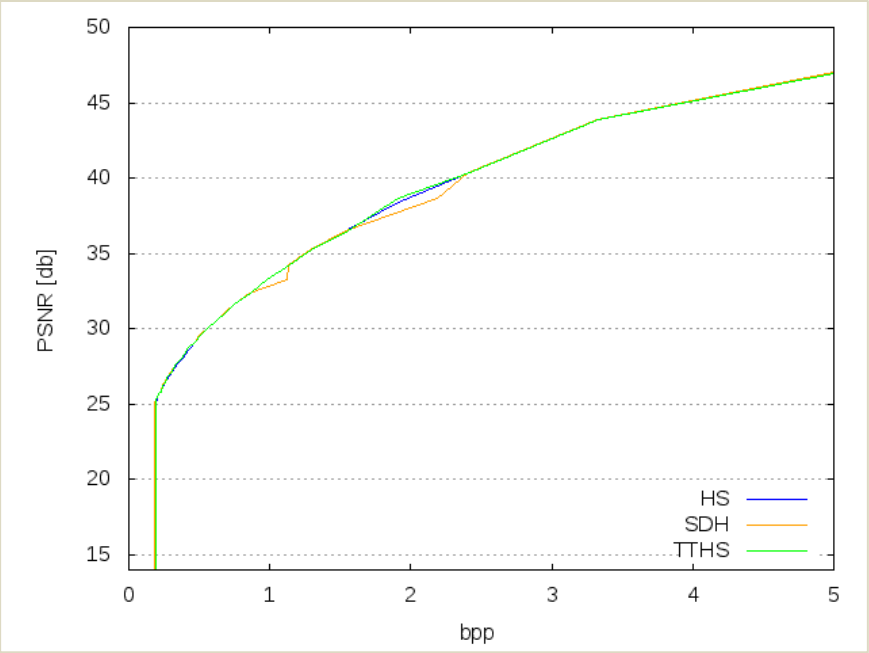


Figure 7.13: RD curves of hierarchical algorithms for "Mobile"

The Figure 7.13 shows the efficiency of the hierarchal algorithms when exposed to "Mobile" sequence. It is seen that all the hierarchal algorithms behave analogous to each other with SDH dropping the performance (either low PSNR or high bpp) at some specific points. This might be due to the poor matching of blocks for the reason that "Mobile "sequence is fairly complex. Furthermore, TTHS has marginally better throughput than HS for this sequence.

7.2.3 Non-Hierarchical vs. Hierarchical

In the previous section, HS performs well for "Foreman" and "Soccer" whereas TTHS proves to be better for "Mobile". On the other hand, E3SS has better performance among non-hierarchical fast algorithms. A comparative analysis was made between the best algorithms in both segments. Figure 7.14 illustrates the outcome when E3SS was compared to HS for a "Foreman" sequence.

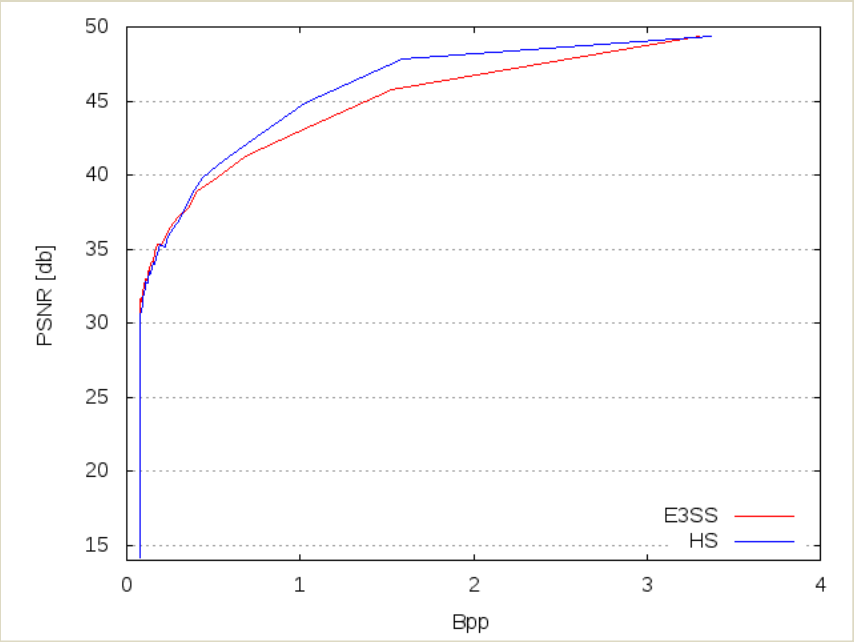


Figure 7.14: RD curve comparison: E3SS vs. HS for "Foreman"

It is evident from Figure 7.14 that HS performs better than E3SS for “Foreman” series. The argumentation for this is that HS uses a full search at different levels thus has increased chances of finding a better match thus giving results that are more similar to the results of the FSA. Similar is the performance for “Soccer” sequence. Figure 7.15 shows the comparison between E3SS and HS where HS performs almost as impressive as E3SS.

The plot in Figure 7.16 is between E3SS and TTHS for “Mobile” sequence. It is clear from the RD graph that both hierarchical and non-hierarchical algorithms perform similar for videos with complexity.

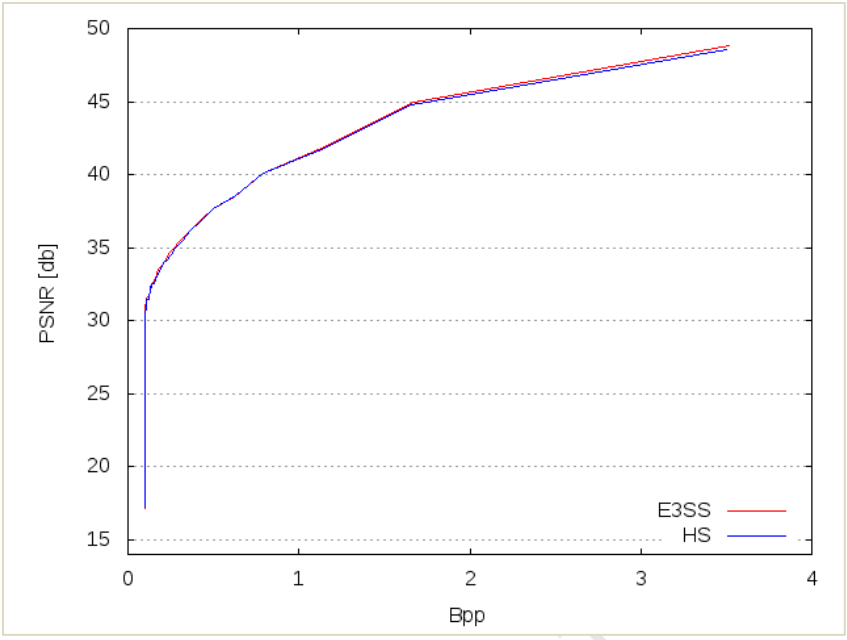


Figure 7.15: RD curve comparison: E3SS vs. for "Soccer"

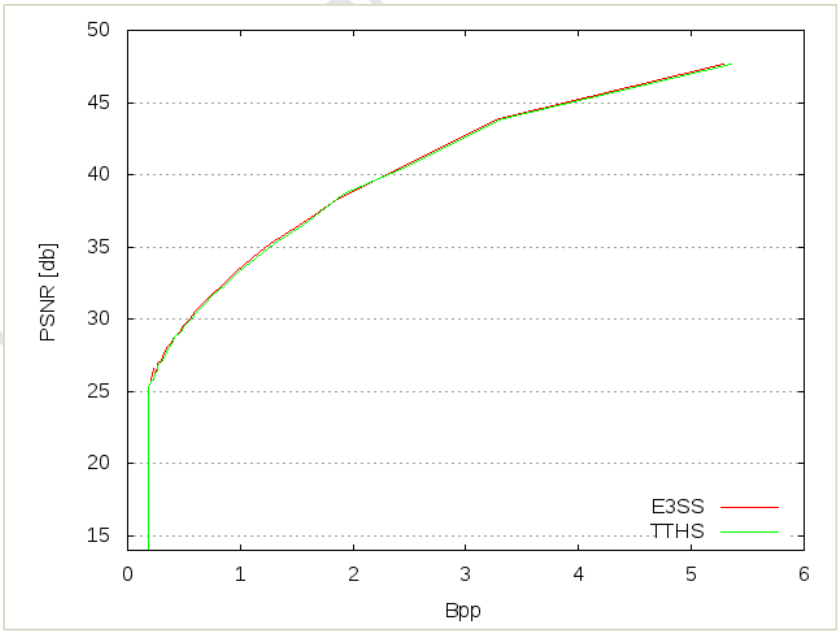


Figure 7.16: RD curve comparison: E3SS vs. TTHS for “Mobile"

7.3 Discussion on Overall Performance of Proposed Algorithms

The previous section had presented and justified the results for different algorithms in the segments of speed and PSNR separately. This portion of the thesis performs a comparative analysis on the manner in which a proposed algorithms accord both speed and PSNR for different input sequences. The data presented in this section is the relative data in percentiles. The notation '+' in the presented tables (only in this part of the thesis) represents increase in value and '-'represents decrease in value.

Usually when an algorithm is fast, it implies to have searched fewer points and is expected to render lower PSNR rates. On the other hand, when an algorithm is slow, it generally signifies that the algorithm searches more points and is expected to generate better quality video at the destination. However, there might be exceptions for this general behaviour of algorithms where the algorithm searches the most probable locations and ignores many other improbable search points thus managing to find best matches yet in limited time.

7.3.1 Overall performance of SDS

When the speed of the proposed algorithms is considered, SDS is proves to be the fastest of all non-hierarchal algorithms and hierarchical algorithms (from Figures 7.1, 7.2, 7.3 and 7.7) but the signal delivery is not the highest when compared to FSA or E3SS. Table 7.2 gives a better understanding of the overall performance of SDS where difference in the measured values at Qf = 4 (for a fairly good quality output) is presented.

Table: 7.2 Overall performance of SDS with respect to FSA at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+77%	-4.2%	-48.5%
Soccer	+78%	-0.4%	-65%
Mobile	+66%	-1.5%	-16%

It is evident from Table 7.2 that SDS has decreased values with respect to FSA in PSNR and bpp and increased values for speed. However, the magnitude of speed variation is much greater than the variation in PSNR or bpp. This table also confirms the positive ability of fast algorithms against an active video like "Soccer" where in the SDS proves to be significantly faster with no significant degradation in video quality (PSNR).

Table: 7.3 Overall performance of SDS with respect to E3SS at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+33.5%	-0.6%	-46%
Soccer	+37%	-0.38%	-30%
Mobile	+18%	-0.02%	-11.7%

Table 7.3 has compared SDS with a pre-existent fast algorithm, E3SS. Although SDS has degraded quality, it has improves the speeds significantly for "Soccer" sequence. Interestingly SDS gives a lower bpp rate for "Mobile" sequence at Qf= 4 yet speeding up the estimation.

7.3.2 Overall performance of LDS

Despite the fact that LDS searches more number of points, the performance of LDS does not exceed SDS for the reason that it fails to perform search in the immediate neighbourhood of the zero motion vector point. However, it can be seen from Figures 7.1, 7.2 that it performs similar to SDS for "Foreman" and "Soccer" in the speed aspect. Table 7.4 shows LDS compared to the standard FSA.

Table: 7.4 Overall performance of LDS with respect to FSA at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+75.5%	-4.4%	-57%
Soccer	+77%	-0.8%	-66%
Mobile	+62.9%	-1.5%	-30.3%

7.3.3 Overall performance of SDH

This section focuses on the SDH algorithm with respect to the standard FSA and to HS, the existent fast algorithm. FSA (as expected) outperforms SDH in both PSNR and the bpp rate categories but to compare the algorithms overall performance, the improvement in speed must also be considered. Table 7.5 gives the idea of its all-round throughput.

Table: 7.5 Overall performance of SDH with respect to FSA at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+75.8%	-1.4%	-44%
Soccer	+76%	-0.02%	-39.7%
Mobile	+64%	-0.7%	-5.3%

From Table 7.2 and Table 7.5 we can realise that SDH performs better than SDS with respect to FSA in quality aspects yet giving significant speeds that makes SDH more desirable than SDS.

The efficiency of SDH is further analysed by comparing it with HS, to acknowledge its performance amongst proposed hierarchical algorithms see Table 7.6.

Table: 7.6 Overall performance of SDH with respect to HS at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+17%	-0.23%	-36.1%
Soccer	+13%	+0.01%	-37%
Mobile	+12.6%	+0.8%	-3.6%

The data in Table 7.6 proves that SDH is the faster algorithm for all the test video sequences with negligible signal loss compared to HS. Also to be noted that SDH renders better PSNR output than HS for "Mobile" sequence at Qf = 4 but with a penalty of an increased bpp.

7.3.4 Overall performance of TTHS

In this segment of efficiency comparisons the last proposed algorithm, TTHS is analysed with respect to FSA and all with respect to the HS to study the improvement by TTHS over HS. HS has advantage with the PSNR at level 2 for the fact that it performs a full search, however searches only 4x4 points around the centre at level 1 and level 0. On the other hand TTHS has an advantage to fast search at two levels (level 2 and 1) and searching wider areas at those levels. Tables 7.7 and 7.8 show the results for TTHS against FSA and HS.

Table: 7.7 Overall performance of TTHS with respect to FSA at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+75.2%	-1.4%	-27%
Soccer	+76.8%	-0.02%	-29%
Mobile	+64.5%	-0.7%	-1.5%

In Table 7.7, TTHS reduces that total time less than half of the time taken by FSA. Nevertheless, FSA gives better quality throughput than TTHS.

Table: 7.8 Overall performance of TTHS with respect to HS at Qf=4

Parameter/ Video sequence	Speed	PSNR	bpp
Foreman	+15.8%	-0.18%	-20.8%
Soccer	+14.2%	+0.04%	-27.8%
Mobile	+15%	+0.8%	0%

The Table 7.8 shows that not only TTHS is faster than the pre-existent algorithm but also renders better PSNR rates that HS for “Soccer” and “Mobile” sequences at Qf=4. The improvements are not significant but the reason for their existence is due to the wider search areas employed by TTHS. However, due to the dropping of the bpp rates the RD curve depicts HS to be a better algorithm.

7.4 Chapter Summary

This chapter presented various simulation results for different algorithms over various input sequences. It analysed both non-hierarchical and hierarchical algorithms in both quality and time perspectives followed by studying the overall performances of the proposed algorithms with respect to the existing ones.

Chapter 8

8 Conclusions

The primary objective of this work was to design and propose new or modified versions of motion estimation algorithms, which would estimate faster than the existing algorithms. However, this project also concentrated on the multi-level hierarchical motion estimation algorithms and the impact of hierarchy technique. This demanded for a comparative analysis between the hierarchical algorithms and non- hierarchical algorithms for different input videos.

Four new fast motion estimation algorithms were designed – Large Diamond Search algorithm (LDS), Small Diamond Search algorithm (SDS), Small Diamond Hierarchical Search algorithm (SDH) and Two-Tier Hierarchical Search algorithm (TTHS). These algorithms theoretically aimed to reduce the computations when compared to the existing algorithms. All the algorithms were simulated using Microsoft Visual Studio 2008 and codec analyser application and each algorithm was experimented with three different video sequences, various parameters were analysed. Following the experiments and simulations, the conclusions drawn are as follows:

8.1 Faster estimation by proposed algorithms

It can be concluded that the proposed non-hierarchical algorithms, SDS and LDS estimate faster than the existing single level algorithms (FSA, E3SS) and the proposed hierarchical algorithms, SDH and TTHS prove faster when compared to the existing HS.

SDS proves to be 64-78% faster than the FSA and 18-37% faster than E3SS with a maximum degradation of 4.2% and 0.6% when compared to FSA and E3SS respectively. Thus, SDS renders significant speed with marginal quality loss. LDS on the other hand also increases the estimation speed by a range of 62-77% compared to FSA but at the cost of 4.5% maximum degradation. However, both the algorithms demand greater bit number of bits to represent a pixel compared to FSA the ranges of increased bit rates are 16-65% for SDS and 30.0-66% for LDS. Nevertheless, the total number of bits transmitted by employing FSA includes information on every block where as a fast algorithms sample the search points thus sending information on less

number of blocks. Thus, the total bit rate is analogous to the number of search points making fast searches more acceptable in real-time environments.

Considering the hierarchical algorithms, SDH and TTHS were evidently faster than the standard FSA and HS. SDS reduced the computation time by 64-76% compared to FSA with a reduction in the PSNR by a maximum of 1.4%. Analysed with respect to HS, SDS offered 64-75.8% higher speeds with a quality degradation of 12.6-17%. Interestingly, TTHS showed slight improvements in the PSNR compared to HS due to its greater search ranges.

8.2 Increased PSNR with hierarchy technique

By employing the hierarchy technique, the estimation algorithm delivers higher PSNR than a non –hierarchical algorithm. To further support this argument Table 8.1 can be considered to where PSNR by E3SS is compared to that of TTHS at Q=4. E3SS uses grid search and small diamond search in a single level where as TTHS uses the same search patterns but in two different levels. Although Figure 7.16 shows that both E3SS and TTHS perform similarly, the breakdown of individual PSNR values contributes to a better understanding.

Table 8.1: PSNR comparison between E3SS and TTHS

	E3SS	**TTHS**
Foreman	41.3	42.6
Soccer	40.1	40.16
Mobile	38.37	38.68

Although TTHS succeeds to deliver better PSNR rate compared to E3SS it is not always the case with HS as HS has full search incorporated which is guaranteed to give better quality output. However, the proposed hierarchical algorithms perform very close to HS in addition to improving estimation speeds.

Acknowledging the performance of SDH, this first hierarchical algorithm proposed is now part of the real-time project by CSIR where it has been proved successful with live streaming. On the other hand TTHS is pending to be tested in the real-time environment which proved better than SDH in the simulation environment.

8.3 Better response to fast videos

The overall performance and throughput of the proposed algorithms is higher than the pre-written algorithms for any type of video input. However, the proposed algorithms are notably effective while dealing with faster video sequences.

It can be noted that all the fast algorithms perform better with "Soccer" sequence, which is active and involves more movement. The hierarchical algorithms are further more impressive with fast sequences like "Soccer" than compared to slower ones like "Foreman". A single level algorithm manages to limit the maximum degradation around 0.8% whereas for a hierarchal algorithm it is 0.04%. Hence, the proposed algorithms are significantly efficient for active video sequences and thus could be applied in a broadcasting system to encode to stream faster videos like a live sports event.

8.4 Future Work

This work incorporated new algorithms into the existing project to maximise the efficiency in the motion estimation sub-system. Proposed algorithms decreased the overall time for streaming by tackling the speeds at the estimation level. However, sometimes (for slower videos) the existing algorithms performed better at the cost of additional time and these algorithms can still be implemented in certain applications where speed constraints are liberal.

Acknowledging the above situation, a sub-system that would consider both the characteristics of the input and the application scenario might form a legitimate solution. In other words, the encoder automatically selects the best-suited algorithm to estimate the video sequence. This technique would not only aid efficient usage of the motion estimation algorithms, but also to prioritise between the speed and the quality for a video application in a particular network.

References

[1] I. E. G. Richardson, "H.264 and MPEG-4 Video Compression", *John Wiley & Sons Ltd*, England, UK, 2003.

[2] Axis Communications, "H.264 Video Compression Standard: New possibilities within video surveillance", white paper, 2008, pp. 1-10. Available at: http://www.axis.com/files/whitepaper/wp_h264_31669_en_0803_lo.pdf. Accessed on 14th November 2010.

[3] Gary J. Sullivan, "Video Compression-From Concepts to the H.264/AVC Standard", *Proceedings of the IEEE,* Vol. 93, No. 1, January 2005, pp.18-20.

[4] I. E. G. Richardson, "Video Codec Design: Developing Image and Video Compression", *John Wiley & Sons Ltd*, England, UK, 2002.

[5] Y. Q. Shi, H. Sun, "Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards, *CRC Press*, Second Edition, 2008, pp.1-93.

[6] M. Ghanbari, "Standard Codecs: Image Compression to Advanced Video Coding", *Institution of Engineering & Technology*, London, U.K, 2010, pp. 1-53, 151-169.

[7] F. June, "Video Compression: Spatial and Temporal Sampling". Available at: http://www.webkinesia.com/games/vcompress2.php. Accessed on 23rd November 2010.

[8] C. Poynton, "Chroma Subsampling Notation", 2008, pp.1-3. Available at: http://www.poynton.com/PDFs/Chroma_subsampling_notation.pdf. Accessed on 20th March 2011.

[9] J. Ozer, "Encoding options for H.264 video", 2009. Available at: http://www.adobe.com/devnet/flashmediaserver/articles/h264_encoding.html. Accessed on 25th December 2010.

[10] Y. Wang, "Low Complexity H.264 Decoder: Motion Estimation and Mode Decision", *University of Columbia.* Available at: http://www.ee.columbia.edu/~ywang/Research/camed.html. Accessed on 1st November 2010.

[11] D. Marshall, "Video and Audio Compression: Motion Estimation", 2001. Available at:

http://www.cs.cf.ac.uk/Dave/Multimedia/node259.html#SECTION042913000000000000 00. Accessed on 10th December 2010.

[12] Y. Fok, O. C. Au, R. D. Murch, "Novel Fast Block Motion Estimation in Future Subspace", pp. 1-4. Available at: http://www.ece.ust.hk/~eefok/paper/ICIP95.pdf. Accessed on 19th January 2011.

[13] J. Xin, A. Vetro, H. Sun, "Converting DCT Coefficients to H.264/AVC", TR-2004-058, June 2004, *Mitsubishi Electric Research Laboratories*. Available at: http://www.merl.com/reports/docs/TR2004-058.pdf. Accessed on 24th January 2011.

[14] C. E. Shannon, "A Mathematical Theory of Communication", *The Bell System Technical Journal*, Vol. 24, July, October 1948, pp. 13-19.

[15] D. Marshall, "Video and Audio Compression: The Shannon-Fano Algorihm". Available at: http://www.cs.cf.ac.uk/Dave/Multimedia/node209.html#SECTION042420000000000000 00. Accessed on 27th January 2011.

[16] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", *Proceeding of the IRE*, September 1952, pp. 1098-1101.

[17] D. Marshall, "Video and Audio Compression: Huffman Coding". Available at: http://www.cs.cf.ac.uk/Dave/Multimedia/node210.html. Accessed on 27th January 2011.

[18] M. Rabbani, P. W. Jones, "Digital Image Compression Techniques", *SPIE Press*, Vol. TT7, pp. 190-209.

[19] L. Pillai, "Variable Length Encoding", Xlinx, XAPP621 (v1.1) January 2005, pp. 1-5. Available at: http://www.xilinx.com/support/documentation/application_notes/xapp621.pdf. Accessed on 15th March 2011.

[20] M. Ghanbari, "The Cross-Search Algorithm for Motion estimation", *IEEE Transactions on Communications*, Vol. 38, No. 7, July 1990, pp.950-953.

[21] J. R. Jain, A. K. Jain, "Displacement Measurement and Its Application in Interframe Image Coding", *IEEE Transactions on Communications*, Vol. 29, No. 12, December 1981, pp. 1799-1808.

[22] M.Gallant, G. Cote, F. Kossentini, "An Efficient Computation-Constrained Block-based motion Estimation Algorihtm for Low Bit Rate Video Coding", *IEEE Transactions on Image Processing*, Vol. 8, No. 12, December 1999, pp.1816-1821.

[23] S. Zhu, K. Ma, "A New Diamond Search Algorithm for Fast Block-matching Motion Estimation", *IEEE Transactions on Image Processing*, Vol. 9,No. 2, February 2000, pp. 287-290.

[24] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion Compensated Interframe Coding for Video Conferencing," *Proceedings NTC 81*,December 1981, pp. C9.6.1-9.6.5.

[25] X. Jing, L. Chau, "An Efficient Three-Step Search Algorithm for Block Motion Estimation", *IEEE Transactions on Multimedia*, Vol. 6, No. 3, June 2004, pp.435-438.

[26] Electric Cloud, "MS Visual Studio Integration", Technical Notes, Version 1.0, October 2008, pp.1-16. Available at: https://electriccloud.zendesk.com/attachments/token/p0r5vr41xqbp4e1/?name=ec_visualStudioTechNote1_0.pdf. Accessed on 25th April 2011.

[27] B. Stroustrup, "The C++ Programming language", Addison-Wesley, Second Edition, 2000.

[28] G. M. Callico, S. Lopez, O. Sosa, J. F. Lopez, "Analysis of Fast Block matching Motion Estimation Algorithms for Video Super-Resolution Systems", *IEEE Transactions on Consumer Electronics*, Vol. 54, No. 3, August 2008, pp.1430-1438.

[29] J. Korhonen, U. Reiter, J. You, "On the Relationship between Perceptual Impact of Source and Channel Distortion in Video Sequences", *Proceedings of IEEE 17th International Conference on Image Processing*, 2010, pp. 2933-2936.

[30] Z. Orlov, "Network-driven adaptive Video Streaming in Wireless environments", *Proceedings of IEEE 19th International Symposium 2008*, pp.1-6.

[31] M. S. Alencar, V. C. da Rocha, "Communication Systems", Springer Science+Bussiness Media, New York. U. S. A , 2005.

[32] Q. Huynh-Thu, M.Ghanbari, "Scope of validity of PSNR in image/video quality assessment", *Electronics Letters*, Vol. 44, No. 13, June 2008, pp. 800-801.

[33] Apple, "Video Sample Rate and Bit Depth", Documentation. http://documentation.apple.com/en/finalcutpro/usermanual/index.html#chapter=C%26section=11%26tasks=true. Accessed on 26th April 2011.

[34] Video Library and Tools, Raw Sequences from the H.264/SVC Group. Available at: http://nsl.cs.sfu.ca/wiki/index.php/Video_Library_and_Tools Accessed on 22nd November 2011.

[35] J. Yang, H. Wang, C. Tseng, "High Quality and Fast H.264/AVC Video Encoder: Enhanced Intra 4x4 Mode Decision for H.264/AVC Coders", *IEEE Transaction on Circuit and Systems for Video Technology*, Vol. 15, No. 8, August 2006, pp.378-401.