H.264 MOTION ESTIMATION AND MOTION COMPENSATION

Darshankumar Shah
B.E., Veer Narmad South Gujarat University, India, 2007

PROJECT

Submitted in partial satisfaction of
the requirements for the degree of

MASTER OF SCIENCE

in

ELECTRICAL AND ELECTRONIC ENGINEERING

at

CALIFORNIA STATE UNIVERSITY, SACRAMENTO

FALL
2011

H.264 MOTION ESTIMATION AND MOTION COMPENSATION

A Project

by

Darshankumar Shah

Approved by:

____________________________________, Committee Chair
Jing Pang, Ph.D.

____________________________________, Second Reader
Preetham Kumar, Ph.D.

____
Date

Student: Darshankumar Shah

I certify that this student has met the requirements for format contained in the University format manual, and that this project is suitable for shelving in the Library and credit is to be awarded for the Project.

___________________________, Graduate Coordinator _________________
Preetham Kumar, Ph.D. Date

Department of Electrical and Electronic Engineering

Abstract

of

H.264 MOTION ESTIMATION AND MOTION COMPENSATION

by

Darshankumar Shah

As technology advances, multimedia applications increase exponentially in day-to-day life. Multimedia applications such as video telephony, video conferencing, TV, streaming video/audio online, and many other applications are in demand in video industry. These applications usually require high bandwidth, large storage, and high latency time to send on network. To conserve resources, it is required to compress the video data before sending them to the network by sender side. It is also required to decompress the video data at receiver end before broadcasting. Many different video codec standards such as H.261, MPEG-1, MPEG-2, H.263, and H.264 are implemented. H.264 is latest international video codec standard. This protocol was developed jointly by International Telecommunication Union – Telecommunication Standardization Sector (ITU-T) and International Organization for Standardization (ISO).

The objective of this project is to explore different blocks of H.264 in MATLAB environment. This project first briefly describes about encoding and decoding process, and then it discusses more details about different modules of encoder and decoder, and the related algorithms. Finally, it compares the result of different algorithms based on compression ratio, peak signal to noise ratio, time required for encoding process, and storage. A video file is given as input to encoder, the video file is converted to a number of frames using video codec, and fixed size macro block is defined in each frame for encoding process. Motion search algorithm finds motion vector after macro block definition, then compensated image is generated based on reference frame and motion vector by video codec. Redundancy is removed from current frame by subtracting compensated image. Compression of residual information is performed using transformation, quantization, and entropy coder. Compressed data are given as inputs to decoder, and decoder process the image to reconstruct image. To enhance image quality and reduce blocking artifact, the image frame is passed through filter. The project is completed successfully by reconstructing video with reasonable quality.

_______________________, Committee Chair
Jing Pang, Ph.D.

____
Date

ACKNOWLEDGMENTS

I would like to express my sincere thanks to my project advisor Dr. Jing Pang for helping and guiding me through this hard project work. I am thankful to her for providing me a good opportunity to work on this project, which enhanced my knowledge in the field of video compression and decompression. She provided all resources, and guidance, which helped me in finding out right ways to complete this project. Her expertise and experiences in the field of video codec were very helpful for me to finish this project successfully. Without her knowledge and guidance, I could not able to finish this project on time.

I would like to thank Dr.Preetham Kumar as my graduate coordinator and second reader. He reviewed my report and provided me valuable feedback to improve my project report. I would like to thankful to all EEE faculty, and department staff members, who helped me directly or indirectly for help me to complete my Master program study.

I would like to give special thanks to all of my family members for their supports to me to finish this project successfully. Special thanks to my parents, and grandparents who provide me all strength and inspiration during critical phase. Special thanks to my brothers and sisters, who cheered me up at my hard time. Special thanks to my fiancée, for believe in me in all situation and encouraged me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

Demand of video applications such as video telephony, video conferencing, online stream video, mobile streaming, TV, 24 hours surveillance system and many others are increasing exponentially because of the evolution of video compression technology. Bandwidth and storage space are crucial parameters in video communication. Their requirements are also increased exponentially with increasing demand of video applications. New video compression standard needed to satisfy growing demand of video applications and be more effective than previously developed standards. It should use less bandwidth, and storage space as compared to previous video codec standards.

Two major groups are actively participating in enhancement of video codec, first is Video Coding Experts Group (VCEG) from ITU-T, and the other is Moving Picture Experts Group (MPEG) from International Organization for Standardization (ISO)/International Electro-technical Commission (IEC). A previously developed video codec standards, and their released year are mentioned here. The first video codec standard is H.120 developed in 1984 by ITU-T. It is lossy. The newer standard defined in 1990 by ITU-T, is called H.261. It is popular for video conferencing and telephony. ISO/IEC also released MPEG-1 part 2 in 1993, which is implemented in video CD. In 1995 ITU-T and ISO/IEC jointly released their new video codec known as H.262, popular in Video CD/DVD/Blu-ray, Video broadcasting. In 1995, H.263 and 1999 MPEG-4 part 2 are developed by ITU-T and ISO/IEC respectively. Finally, H.264 video codec was announced by ITU-T and ISO/IEC, which is popular for all applications mentioned above [8].

H.264 is widely accepted video codec standard. It is also known as MPEG-4 part-10 Advance Video Codec (AVC). It took approximately four years. Trade off in terms of coding efficiency, complexity on hardware, and cost; are balanced in new proposed standard [1]. Some of its features are high definition resolution, interlaced/progressive scan mode, variable frame rate, high bit rate, supporting I, B, & P-frames, 9 different prediction modes, variable size block matching motion estimation, and motion compensation, quarter pixel accuracy, adaptive length coder, and image enhancement filter. H.264 is compatible with almost all kinds of recent video coding tool as opposed to previous standards. As a result, H.264 is most popular nowadays.

1.1 Purpose of the Study

Video signal passes through a long chain of components from source to destination. This chain is broken into two main parts compression and decompression. Also known as encoding and decoding. Encoder and decoder have different small blocks to perform compression and decompression successfully. The purpose of this project is to simulate sub blocks of encoder and decoder using H.264 standard in MATLAB environment, and compare results based on following parameters PSNR, compression ratio, number of steps, speed and cost

These long chains of encoder and decoder modules have been simulated and implemented using previously defined standard. H.264 has opened new horizon of video codec, it uses different algorithm to implement frame prediction, and motion compensation. Frame prediction is done using inter frame prediction and intra frame prediction, which explain in detail in chapter 2. The developed algorithm has compatibility of previous video codec tool, and achieve high compression ratio, high bit rate, with reasonable cost complexity. The significance of H.264 video codec is low latency, less storage, minimal bandwidth, loss less compression and reasonable complexity [1].

1.2 Organization of Report

Chapter 2 depicts about fundamental of digital video representation, format, and color space. It explains top-level block diagram of encoder and decoder modules. Basic operation of each module has been described in this chapter.

Chapter 3 describes h.264 prediction. It includes intra prediction, and motion search algorithms. It also discusses different algorithms of motion search, such as full search, three-step search, and four-step search. Finally, it compares motion search algorithms based on speed, and number of computational steps required in each algorithm.

Chapter 4 describes motion compensation algorithms. It compares fixed size block matching, and variable size block matching motion compensation algorithms. It calculates motion vector size and peak signal to noise ratio for comparison.

Chapter 5 describes image compression block. It will mainly discuss discrete cosine transform, quantization, entropy coder, and de-blocking filter in depth.

Chapter 6 is conclusion of this project, and discusses about advance algorithms for future that reduce computational steps, size on disk, and improves speed for motion search.

Chapter 2

BASIC OF H.264 VIDEO ENCODER AND DECODER

2.1 Basic of Color Image Processing

Image processing block, has an image as an input, and encoded frame at output. An image can be represented in terms of RGB color model, HSI color model, or Ycbcr color model. RGB color model is used widely. RGB represents image in three-color components – red, green, and blue. This color information is used for transmission and storage. Figure 1 shows that the original image and following are the R, G, and B – component of original image.

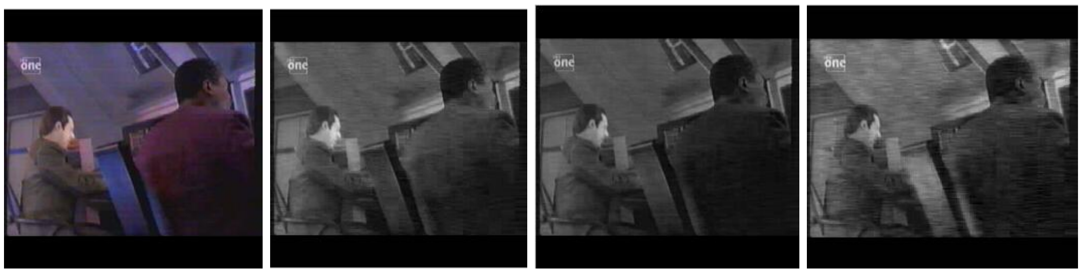


Figure 1 A color image followed by extracted R, G, and B-component

The human visual system (HVS) also works as RGB model. Cones work as a sensor in human visual system, which senses red, green and blue color component and merges them to create a 2-D image in HVS. The human visual system is less sensitive to color component than luminance/brightness of image. Figure 2 shows the Ycbcr image and followings are Y, Cb, and Cr - component of image. Now onwards we will discuss everything in luminance (Y) and chrominance (Cb, Cr) in whole project. Another fact of human eye is that if we pass at-least 15 frame per second in front of human eye, human eye will not differentiate frame boundary and it seem as movie. These limitations of human eye are advantageous to our video compression system.

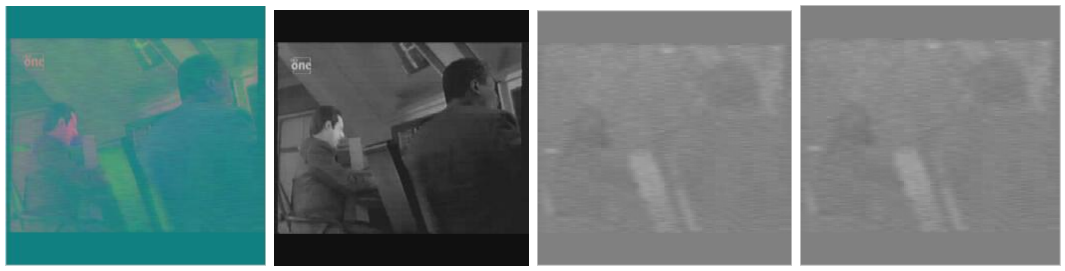


Figure 2 The Ycbcr image and followings are Y, Cb, and Cr - components of image

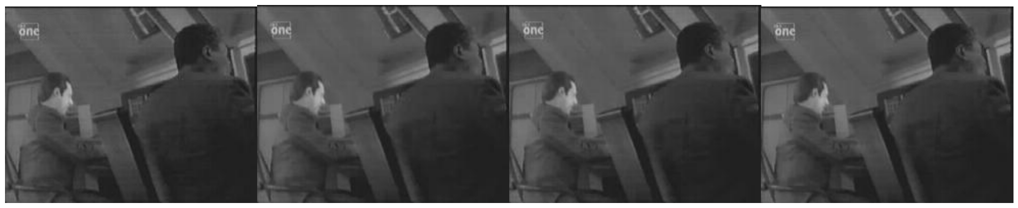


Figure 3 The four consecutive frames with redundant information

The fundamental concept of video encoder is to remove the redundant data from the current image. The redundant data can be removed by predicting redundancy from previous frames and future frames. Figure 3 shows the four consecutive frames with redundant information. There are very large amount of similarities are observed in those four consecutive frames shown in Figure 3 [2]. These similarities are defined as the redundant data between the frames. The process of removing redundancy from consecutive frames is called Compression. Figure 4 shows four consecutive frames with necessary information for reproduction. There are four types of redundancy.

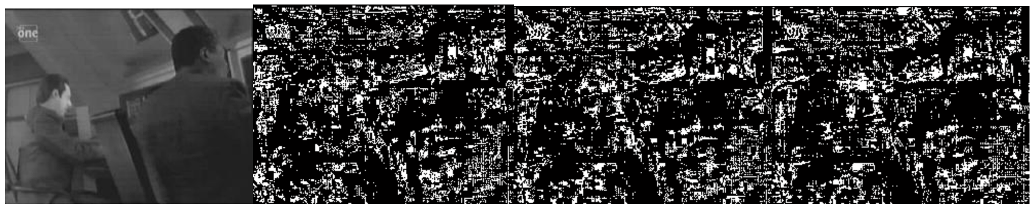


Figure 4 The four consecutive frames with necessary information for reproduction

2.2 Type of Redundancy

As mentioned in the previous section, there are four different types of redundancy [2]. It is divided based on similarity between neighboring frames, neighboring pixel, and the fact of human visual system (HVS).

2.2.1 Temporal Redundancy

It is referred as redundancy between the consecutive frames in video. Let us consider first frame as reference frame (I), and the following all frames are considered as the current frame (P) in figure 3. The frame (I) is subtracted from the frame (P), so the resultant frames is shown in figure 4. Due to subtraction of frames, the current frames contain very less amount of energy for transmission or storage. The subtracted information is called temporal redundancy.

2.2.2 Spatial Redundancy

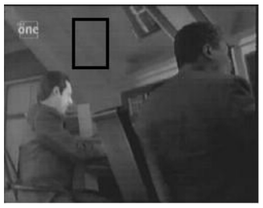


Figure 5 Spatial redundancy indicated by rectangle box

Spatial redundancy refers to the correlation between the neighboring pixels within a frame. Figure 5 shows spatial redundancy in rectangular box within the frame. The huge amount of similar information is marked in rectangle box in figure 5. From rectangle box, partial information is stored and copied to all other locations when needed. It is a simple approach to remove spatial redundancy within a frame. As a result, a frame contains very small amount of energy information, which is needed to transmit or send over the network by sender.

2.2.3 Color Spectral Redundancy

A human eye identifies small changes in brightness of a picture, but a human eye is insensitive for color variations within the same image. We can save bandwidth or storage space if we only send selective samples of chrominance components. As from figure 2, the first image is represented in YCbCr, the second image refers to luminance component of image, and the last two images refer to chrominance part of image. The pixel values are same within the chrominance picture. It is observed in figure 2, and it is true for consecutive frames of video [2].

2.2.4 Phyco-visual Redundancy

It refers to a high-level limitation of HVS. When HVS visualizes an image, the partial information of image is important. The rest of the information is less important, so we can represent less important data with less number of bits. In the figure 1, humans to be considered as important information in the image, so the important information is represented with 256 bit in the image. The rest of the information is represented with only 64 bits instead of 256 bits. HVS visualizes 64 bits represented image as an original image, and it is not able to identify even small differences of resolution.

2.3 Video Coding Layer

An H.264 video encoder is mainly comprised of motion estimation, motion compensation, intra frame prediction, discrete cosine transformation, quantization and entropy encoding [3]. Figure 6 shown block diagram of H.264 Encoder.

The brief overview of H.264 block is as follows. Encoder has intra prediction mode, which removes spatial redundancy from the frame. The feedback path of the decoder module is an access point, which is used to decode intra predicted frame correctly. It works on different intra mode to remove spatial redundant data from the reference frame

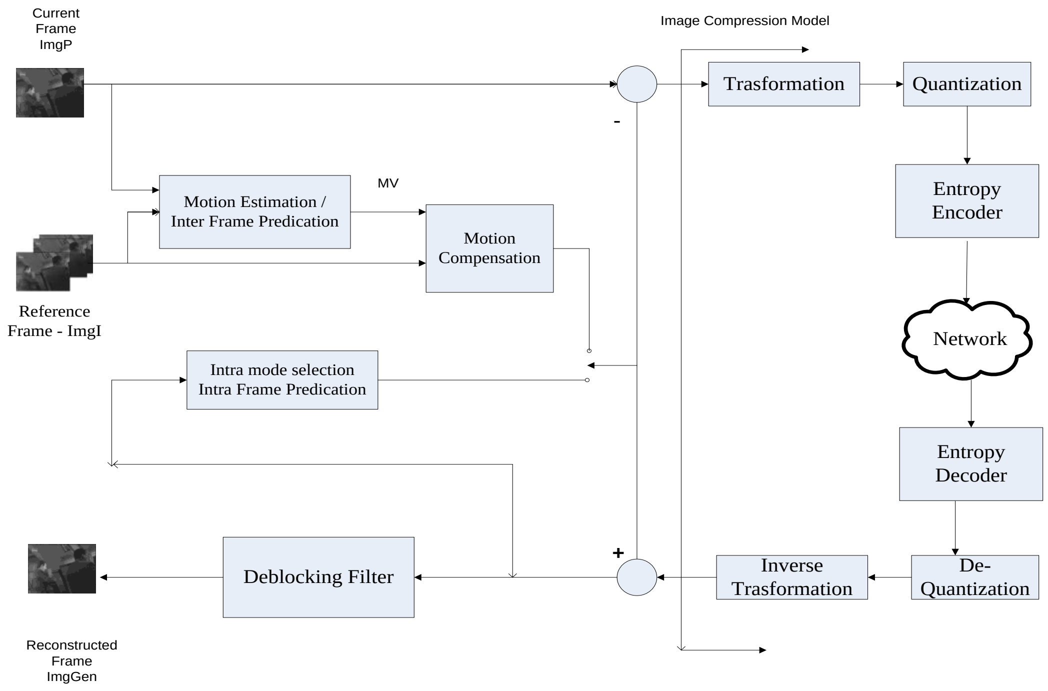


Figure 6 Block diagram of H.264 Encoder [7]

Motion estimation block is also known as inter frame prediction. It is a more effective algorithm for compression, because it removes temporal redundancy from the image. The current frame is predicted using previous frames and future frames. The current frame is predicted on macro-block level. Each of the macro-blocks in the current frame are compared with a predefined search window of the past frames or future frames. A MAD, MSE, or SAD algorithm will decide whether it is a match or not between the macro blocks of two frames. The coordinate of the matched macro block is saved in the motion vector (MV). The procedure continues with the next macro-block in the current frame and It ends at boundary of the current frame

Motion compensation generates the compensated motion picture using the reference frame and the motion vector. We subtract the reference frame from the current frame to remove temporal redundancy without compromising the motion. This temporal redundancy is removed by motion compensation. Consider a blank frame that is the same size of current frame, and fill it with the macro-blocks of the reference frame and the motion vector information of the current frame. The resultant frame is known as the compensated current frame. It will generate a residual frame (imgRes) by removing temporal redundancy from the current frame.

This residual image moves further to the image compression module. The image compression module is a combination of both lossy and lossless compression. H.264 has lossless image compression. The image compression block is the same as in JPEG. It is the process of separating frequency components, and keeping low frequency data by quantization. The high frequency data is discarded which causes distortion in a reconstructed image. The output of quantized DCT data is MxN, which will convert in a serial stream to 1-by-(M * N) for entropy coding.

2.3.1 Intra Frame Coding

Intra prediction is applied to remove spatial redundancy within the frame. Intra mode prediction is selected if there is not enough temporal redundancy in the two frames.

So the current frame is coded by itself. It was present in previously adopted video codec standards, but it was less effective than the H.264 video codec. The H.264 has nine different modes of intra frame prediction. The block diagram for intra frame coding is shown in figure 7.

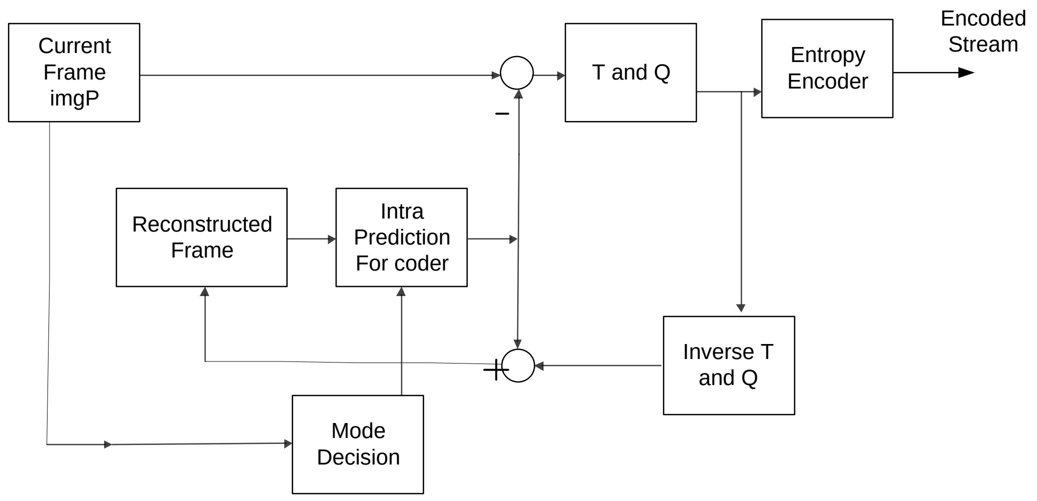


Figure 7 Block diagram for Intra frame prediction

Intra coded macro block is predicted from the previously coded macro block. The feedback is provided to the intra prediction block. It encodes a residual signal between the current block and the previously coded block. We will discuss more about intra frame prediction later in the following chapter.

2.3.2 Inter Frame Prediction

Inter frame prediction is used to achieve a higher compression ratio in video codec standards. It takes advantage of temporal redundancy between multiple frames.

The current frame is predicted from multiple reference frames, which are available in picture buffer memory.

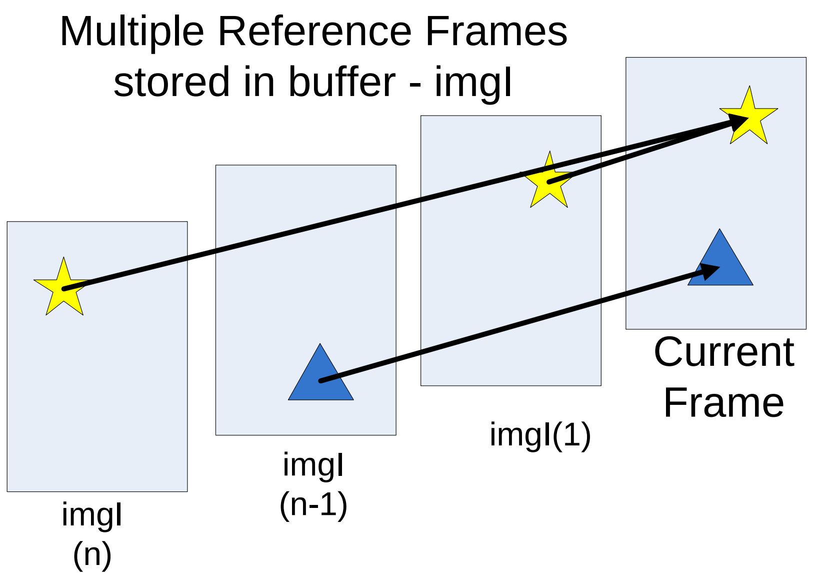


Figure 8 The prediction of current frame with respect to multiple reference frames

Figure 8 illustrates that the current frame is predicted from multiple reference frames (imgI). In the figure 8 shown, the yellow star has different coordinates in all reference frames, and the position of the yellow star in the current frame will be predicted with respect to all reference frames. The triangle does not change its position in the current frame with respect to the reference frame img(n-1). When a Current frame is predicted with respect to the past and future frames, it is called a B-frame or bidirectional frame shown in figure 9. When a current frame is predicted with respect to a reference frame only, it is known as a P-frame shown in figure 9. The figure 9 shows I-frame, P-frame, and B-frame.

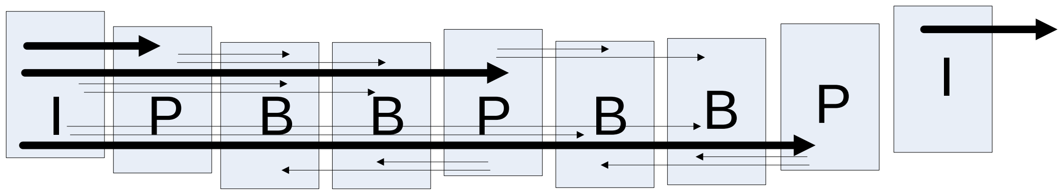


Figure 9 I-frame (Reference frame), P-frame (Predicted frame), and B-frame (Bi-directional frame)

The frames are divided into a number of macro-blocks (MB) before processing. The inter frame prediction module searches the position of the current macro block of the current frame in the reference frame. It uses a matching algorithm to find the maximum match. The motion search is trying to locate the maximum match of the current MB value in the reference frame window, and stores the x-y positions of the current MB. The x-y coordinates are also known as the motion vector (MV). This motion vector is passed to the next block for further processing. Many different motion search algorithms are implemented to generate a motion vector. These algorithms will be explained in the following chapter.

2.3.3 Motion Compensation

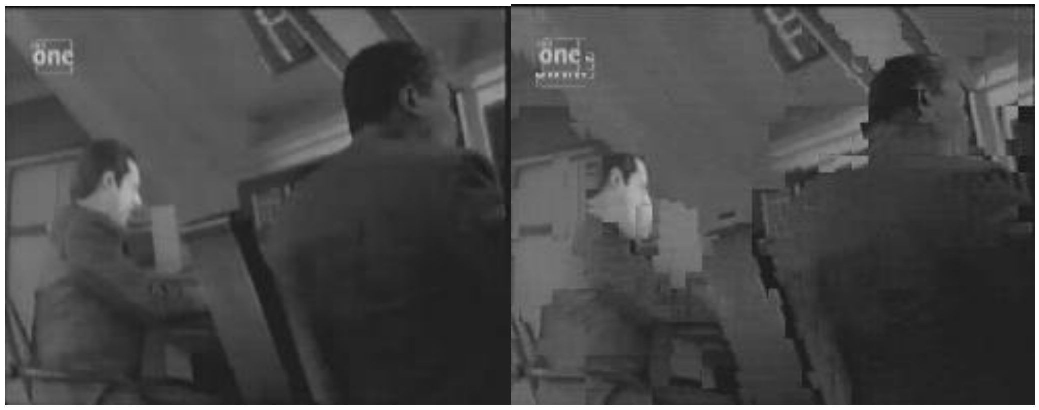


Figure 10 Current frame and motion compensated frame

Motion compensation represents a motion in terms of renovation of a reference frame into a current frame. It means that the motion compensation block takes input as a motion vector and reference frame, and gives the output as a compensated frame [4]. The reference frame might be a previously coded frame or a future frame. Figure 10 shows the current frame and compensated frame. A motion compensated image is generated using many different algorithms. The compensated image matches very closely with the current frame. It can be represented in terms of the equation shown below.

$$MV(i, j) = (p, q) \quad (2.1)$$

$$MBCF(i, j) = MBRF(i+p, j+q) \quad (2.2)$$

The difference between the current frame and the compensated frame is known as the residual frame. The compensated frame is not an exact match with the current frame.

The residual image still contains a small amount of information, which is needed at the decoder side to reconstruct a current frame. Figure 11 shows a residual image and its 3D mesh plot of energy distribution. The residual image is applied to an image compression model for additional data compression.

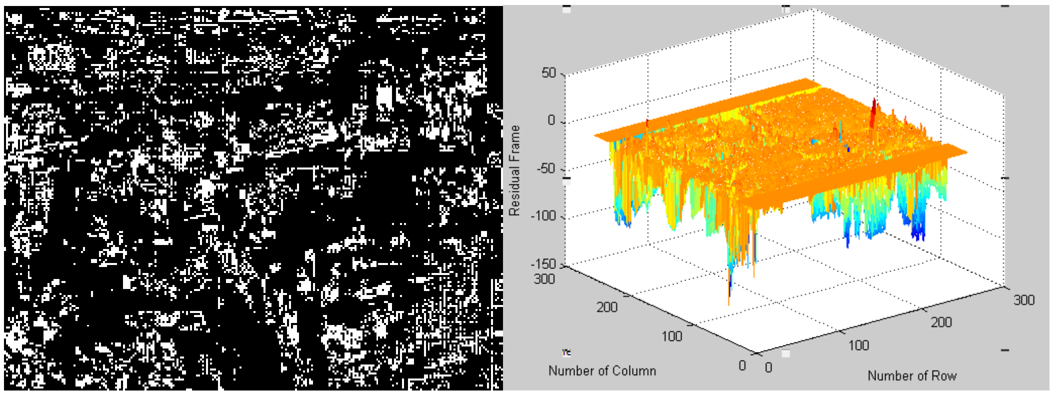


Figure 11 Residual frame and 3D mesh plot of residual image

2.3.4 Image Compression Model

The image compression model is comprised of DCT, Quantization, and Entropy coder. The DCT and quantization are applied to remove high frequency component of an image. The part of information of an image is lost after applying DCT and quantization on the image. The Entropy coder is a lossless block of image compression model. The data is compressed by replacing long series of the same data with the data and the frequency of occurrence. The image compression model is discussed in chapter 5.

2.4 Video Decoding Layer

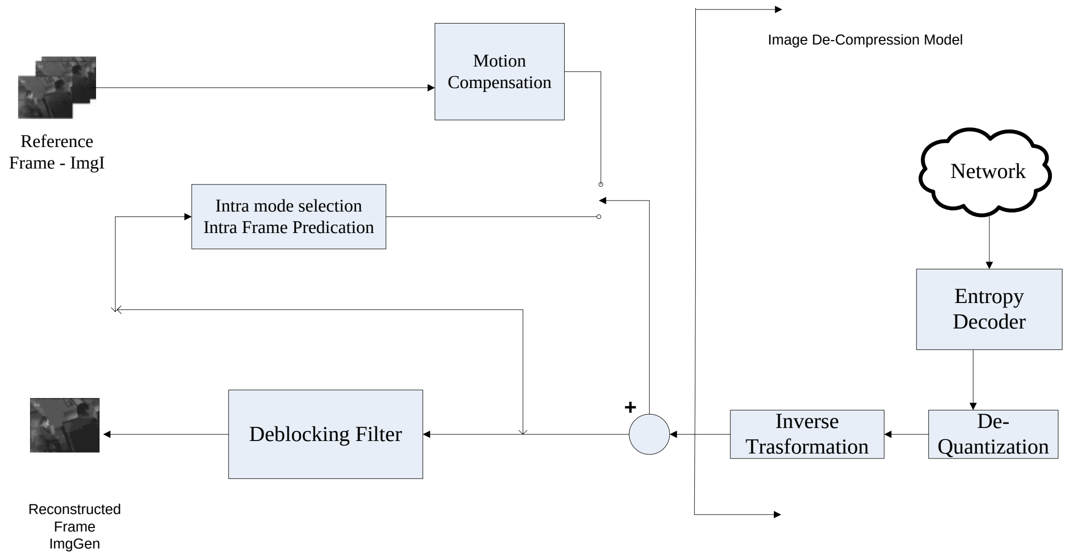


Figure 12 Top level block diagram H.264 Decoder

A decoder is mainly comprised of motion compensation, intra frame decoding, inverse DCT, de-quantization, entropy decoder, and de-blocking filter. Figure 12 shows top-level block diagram of decoder. A video decoder is a reverse process of video encoder.

Once receiver gets a data at the receiver end by users, it is inputted to the entropy decoder. The entropy decoder works same as the entropy coder, but in the reverse direction. The data is copied based on number of occurrence into a serial stream. The long chain of data is converted into MxN matrix The MxN matrix is applied to de-quantization and the inverse-DCT to generate the residual image.

First, the H.264 decoder generates a compensated frame with help of the motion vectors and the reference frame. If the received frame is a P-frame or B-frame then the current frame is reconstructed by adding the compensated frame into the residual frame. If the received frame is the reference frame, an I-frame is reconstructed using intra mode predictions. The reconstructed image is filtered to improve visual quality, and to remove blocking artifact. The detail of image decompression and de-blocking filter are discussed in chapter 5.

Chapter 3

H.264 PREDICTION AND MOTION ESTIMATION

Chapter 2 gives a brief overview of the video coding layer. Prediction is important layer in video codec, and playing important role in compression and decompression. Prediction is used to remove spatial redundancy from individual image, and temporal redundancy from moving images. Intra frame prediction , and inter frame prediction are two types of prediction is used in video codec standards. This chapter will discuss theory and different algorithms for intra coding and inter coding.

3.1 Intra Frame Prediction

Intra frame prediction is one form of compression that looks at information of an individual frame, and tries to reduce the amount of information with minimum loss and high quality. By doing intra prediction, spatial redundancy is removed from individual frame.

H.264 is macro block oriented and motion compensation based video codec standard. In H.264, macro block can be a variable [1]. Macro block size could be 16x16 and divided into size of 16x8, 8x16. Macro block size could be 8x8 and divided into size of 8x4, 4x8 or merged into size of 8x16, or 16x8. 4x4 macro block could not be divided into sub block. A bigger macro block covers large area of frame and more information. A higher macro block size is used to code continuous area of picture. A smaller macro-block size is used to capture minor changes in frame with respect to another frame. In other word, higher macro block size reduces the quality of reconstructed image, but also decrease computation cost and complexity of algorithm. Smaller macro-block size improves a quality of reconstructed image, but also increases computational cost and complexity of algorithm. Impact of macro-block size on compression is explained in figure 13. The figure 13 shows variable size of macro-blocks. A 16x16 macro-block size is encoded using single vector, but if 16x16 macro-block is divided into 16x8 or 8x16, 16x8 or 8x16 block, it is encoded using two vectors. As we process 16x16 macro-block size with 4x4 macro-block size, number of encoded vector is sixteen. Thus, a selection of macro-block size is tradeoff between the precision in motion and the computational cost. An 8x8 macro-block size is considered for simulation in this project.

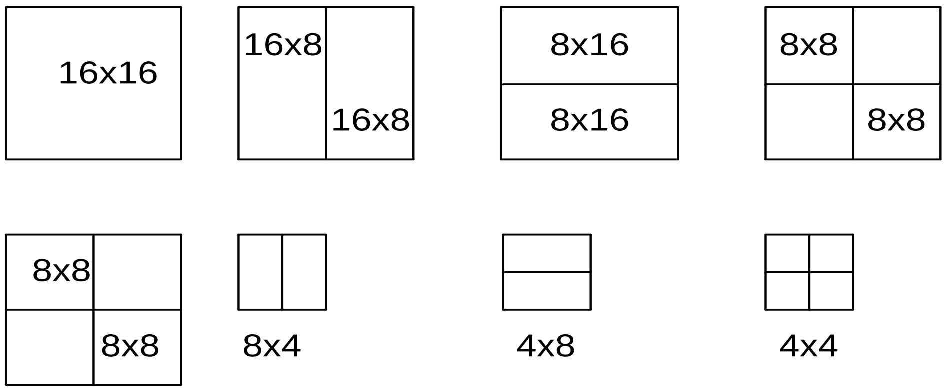


Figure 13 H.264 macro block size 16x16 – 4x4 [3]

To improve bit rate and latency, H.264 standard introduces 9-intra prediction modes as shown in the figure below. Figure 14 shows the direction of prediction of intra mode.

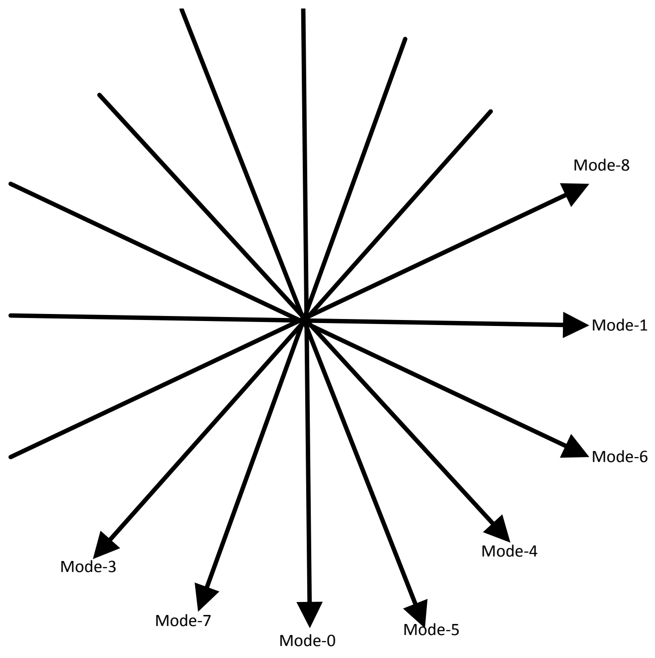


Figure 14 H.264 intra prediction direction, 0-vertical, 1-horizontal, 2-DC, 3-diagonal down left, 4-diagonal down right, 5-vertial right, 6-horizontal down, 7-vertical left, 8-horizontal up [3]

The implementation of intra mode-0, mode-1 and mode-2 for 8x8 macro-block size will be discussed. If macro-block or sub-block is encoded in intra mode, prediction block is formed based on previously encoded and reconstructed block [16]. As shown in block diagram, intra frame prediction has two inputs called the reconstructed frame and the original frame. The reconstructed intra image is unfiltered and fedback as input to the intra prediction block. Figure 15 shows block diagram of intra prediction.

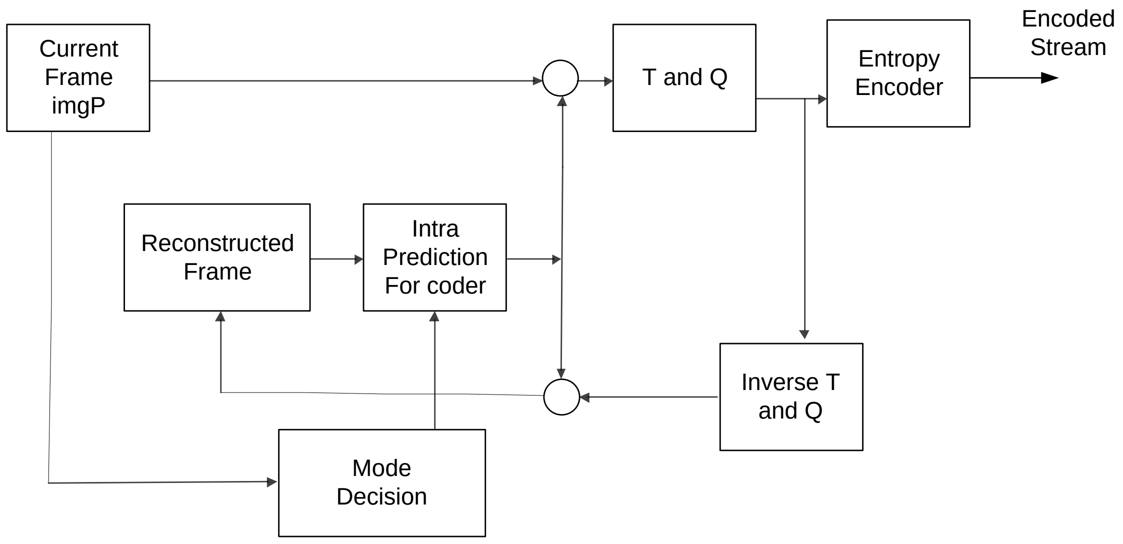


Figure 15 Block diagram of Intra frame prediction

Intra frame is divided into 8x8 macro-block size. Each macro-block is inputted to the mode selection block. The mode decision block selects one mode out of nine intra modes. A selection of intra mode is based on sum of absolute error (SAE), or mean of absolute difference (MAD) algorithms. SAD, or MAD is calculated at mode selection point, and the minimum result mode would be considered for only the current macro-block. The implementation of SAD or MAD could be possible using following equations respectively,

$$SAD = \sum_{i=1}^{N} \sum_{j=1}^{M} |Cij - Rij| \ldots\ldots\ldots\ldots\ldots\ldots..(3.1)$$

And

$$MAD = \frac{1}{NxM} \sum_{i=1}^{N} \sum_{j=1}^{M} |Cij - Rij| \ldots\ldots\ldots\ldots..(3.2)$$

The predicted macro-block is generated with respect to the reconstructed macro-block and the intra mode. The predicted macro-block is subtracted from the original macro-block, and resulted as the residual macro block. The residual macro block is further processed by discrete cosine transform and quantization. The output of quantization is applied to inverse quantization, and inverse DCT, and then generates the residual macro-block. The reconstructed residual macro-block is used as reference for next macro-block in queue. All nine modes are self-explained, but we will discuss only mode-0 (vertical), mode-1 (horizontal), and mode-2 (DC). Figure 16 shows the direction of prediction in intra coding.

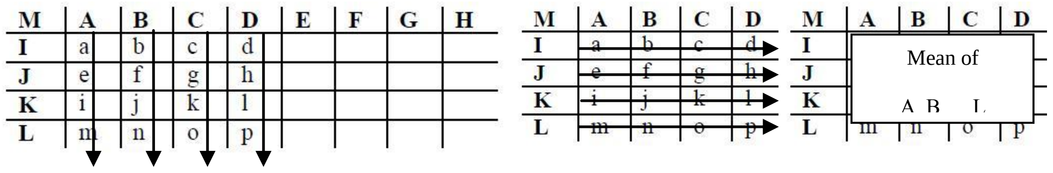


Figure 16 The direction of prediction of intra mode mode-0,1,2 (left to right) [3][7]

The mode-0 has vertical direction of prediction shown in left most macro-block size. The small letters (a,b,c,d…p) is pixel values of current processing macro-block. Those current pixel values are predicted from reference pixels reprinted by capital letter (A, B… M). Arrows show the directions of prediction; in vertical, mode direction of prediction is top to bottom; in horizontal mode, direction of prediction is left to right; DC mode can predicted by mean value of A,B…M. One can understand other intra mode, and direction of prediction of intra mode.

3.2 Inter Frame Prediction

Inter frame prediction is also referred as motion search or motion estimation. Inter frame prediction is used to reduce temporal redundancy among the frames. Frame can be predicted by single reference frame or multiple reference frames in inter frame prediction. The direction of prediction can be one directional or bidirectional, which means a frame can be predicted considering previously coded frame or future frame or both. H.264 video codec standard improves performance because H.264 standard add features such as accuracy at pixel level, multiple referencing, and variable macro-block size as compared to previous standards.

The tradeoff and impact of macro-block size were discussed earlier in this chapter for intra frame prediction. It is also applicable to inter frame predictions. .Considering figure 13, each macro-block size is encoded with single motion vector. Four motion vectors are needed for encoding 16x16 macro-block size with 8x8 sub macro-block size. If we further divide a 8x8 macro block into 4x4 sub macro-block size, sixteen motion vectors are needed to encode 16x16 macro-block size with 4x4 sub macro-block size. Thus, a macro-block is divided into multiple sub macro-blocks to capture smaller motion in image frame and needs higher number of motion vectors to locate multiple sub macro-blocks.

Inter frame prediction is the process of determining motions from one image to another. Motion vector defines displacement of a macro-block from one image with reference to another in terms of 2D coordinates of macro-block. Motion estimation creates dummy current frame by modifying reference frame such that dummy current frame closely matches original current frame. The dummy current frame is called motion compensated frame, which will be discussed later in chapter 4.

The objective of motion search is to estimate motion vector of latest frame captured at time t_2 with reference to another frame captured at time t_1. Figure 17 shows broad classification of motion search algorithms.

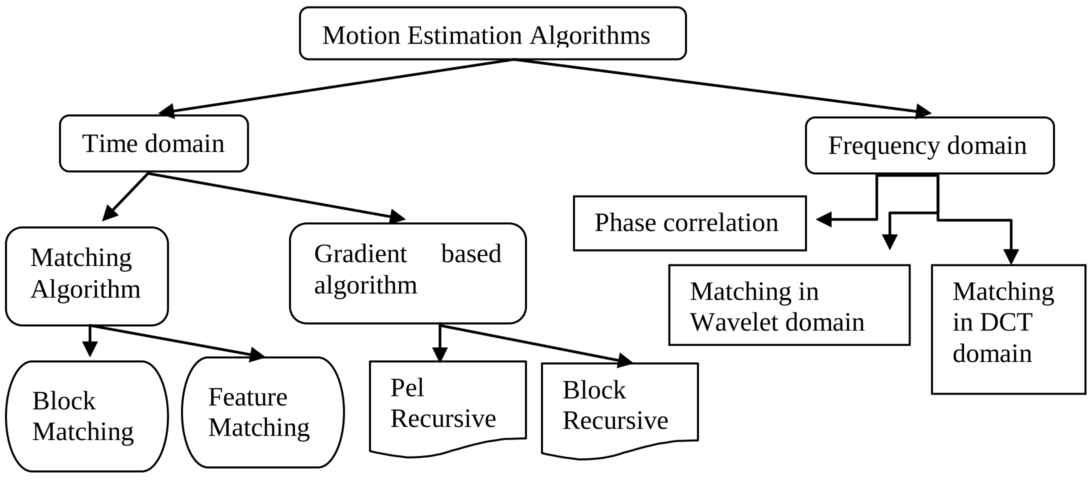


Figure 17 The Broad classification of motion estimation algorithms [9]

The scope of this project is limited to the block matching algorithms. Video codec standard tries to reduce computational complexity, improves compression ratio without compromising quality of image. The computational complexity is calculated based on search algorithm, search area, and cost function. Cost function is based on distance criterion between two macro-blocks. MAD, MSE, and SAD are three types of cost function we have seen in intra prediction section.

Motion estimation searches for motion vector of macro-block in current frame with respect to reference frame, and searching motion vector is heavy computation task in any video codec standards [10]. The broad classification of motion estimation algorithms are given in figure 17. In this project, three algorithms are studied in detail. Exhaustive search algorithm is one simple search algorithm. It provides best picture quality in terms of PSNR with very high computational cost as compared to other algorithms [10]. Three step search, and four step search algorithms have less computational cost compare to full search algorithm with compromising image quality. The macro-block of the current frame is searched into the reference frame, and it is shown in figure 18. This process is repeated until closest match is found in reference frame. Figure 18 shows macro-block of current frame (MB in light blue shade) is searched in reference frame (I in yellow shade), within predefined search area (P in gray shade).

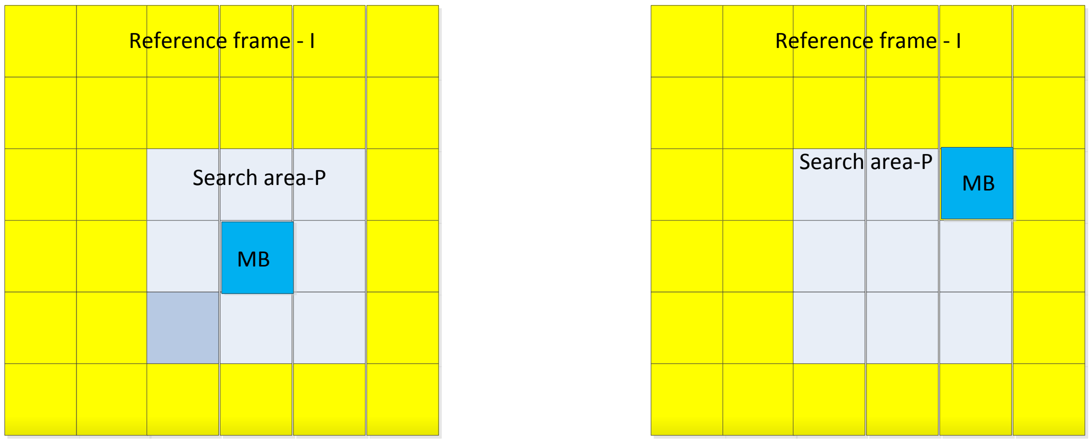


Figure 18 Reference frame (I), search area (P), current macro-block (MB).

A video has captured multiple image frames in a second. Those consecutive frames do not have much movement with respect to one another. The regular interval frame is chosen as a reference, and the following frames of the reference frame are known as a current frame. Reference frames have high movement, and unacceptably high values of cost function. In figure 18, the reference frame and the current frame are shown with the size of MxM, and are divided into equal number of non-overlapped mxm macro-blocks. Each-macro-block of current frame is compared with limited number of macro-blocks in the reference frame as shown in the above figure. That limited number of search is varies by changing restricted search area or search size. It is too costly to examine macro block of current frame with all macro-blocks of reference frame. MAD, SAD, and MSE are algorithms to get best match. The macro-block with minimum cost function would be the best candidate for motion-vector calculation. How much the macro-block of the current frame is moved with respect to the reference frame is defined as the motion-vector. This motion-vector will use by motion compensation block. The rest of chapter is dedicated to three motion estimation algorithms.

3.2.1 Full Search Algorithm

The full search algorithm is also known as exhaustive search algorithm. The name explains that algorithm calculates cost function at every possible location within search area. This would be considered as an advantage because the search is resulted into best matching candidate with lowest cost function and highest peak signal to noise ratio

(PSNR). Matching macro-block of current frame with all possible macro-block of search area of reference frame would require the high computational complexity, and latency. PSNR is directly proportional to the number of matching performed within search area. Computational equation for full search algorithm is shown below.

$$\boldsymbol{computational\ cost = m^2\ x\ (2p+1)^2} \qquad \ldots\ldots\ldots\ldots\ldots\ldots(3.3)$$

Where, m=length of macro-block, p=search area

Above equation is used for the macro-blocks with equal height and length. The total number of comparisons would be counted from the equation 3.3 for single macro-block of current frame. The computational cost could be reduced by either limiting the number of checking points in the search area or reducing the computation of absolute matching by subsampling and other techniques [10]. Fast search algorithm improves computational efficiency by reducing the number of checking points in the search area.

3.2.2 Three Step Search Algorithm

The three step search is categorized as fast block matching algorithm. The figure 19 mentioned three step search algorithm. It became very popular estimation algorithm because of its simplicity and robust implementation.

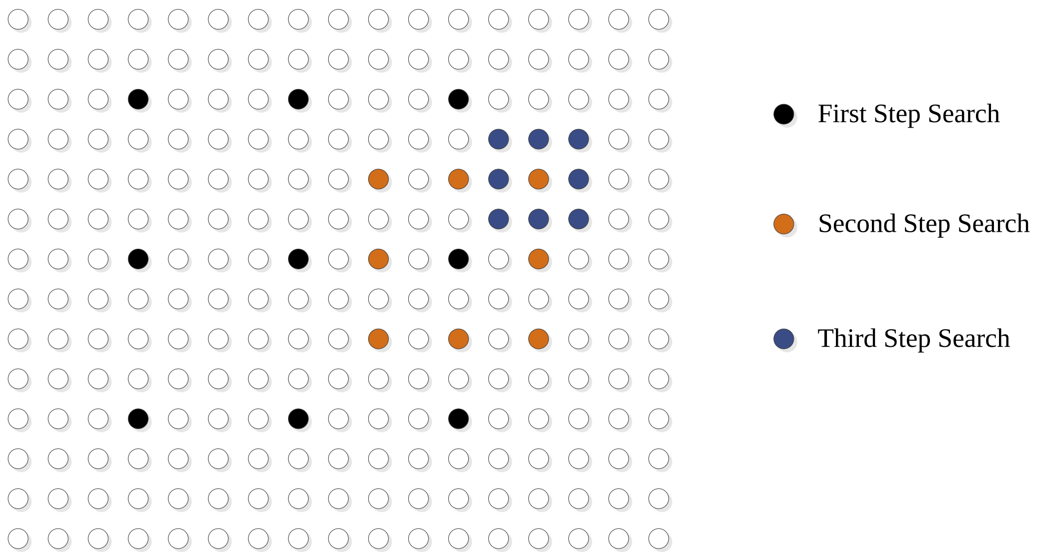


Figure 19 The Three Step Search Algorithm

The three step search (TSS) algorithm is implemented for different step size (s), and search parameter (p). However, this project has considered initial step size s=4, and search parameter p=7. TSS starts at center as its first match location. It moves to next match location in terms of +/- of the step size. At the end of the first search, TSS matches macro-block of current frame with macro-block of reference frame at nine different locations. The minimum value of cost function point would be proceeded further for the second search, and considered as best matching point. The step size is reduced to half of its current value. The second search also starts at center, and moves to the next search point in step of +/- s. The best matching point would be set as the third step's initial location, and the step size is reduced to half of current value. TSS starts moving to 9 locations to find the best match. TSS stores XY-coordinates of the best match location, and the XY-coordinates are used to represent motion vector for macro-block of the current frame.

Analysis of full search algorithm and TSS algorithm in term of total number of matching takes place in single macro-block. Exhaustive search compares a macro-block 255 times within reference frame, and TSS compares a macro-block 25 times within reference frame in case of search parameter equal to seven. The computational cost is reduced drastically in TSS than in full search. One problems associate with TSS is that TSS applies uniform searching pattern. Small motion detection is inefficient due to uniform searching pattern. TSS always searches in 25 steps irrespective of PSNR, and misses a small motion due to fixed search window. New step search algorithm is proposed to improve the remedy of TSS algorithm [11].

NTSS improves from TSS in two aspects. TSS searches for 9 points in first step in step size of 4, but NTSS adds 8 extra search points near the center of the search window at step size 1 shown in below figure. If the minimum cost occurred at center of search window, the search stops and the motion vector is stored. This enables fast searching in NTSS as compared to 25 searches in TSS. If the minimum cost occurs at one of eight inner first search points, it continues to the second search as shown the figure below. Figure 20 shows the new three step search process for motion estimation.

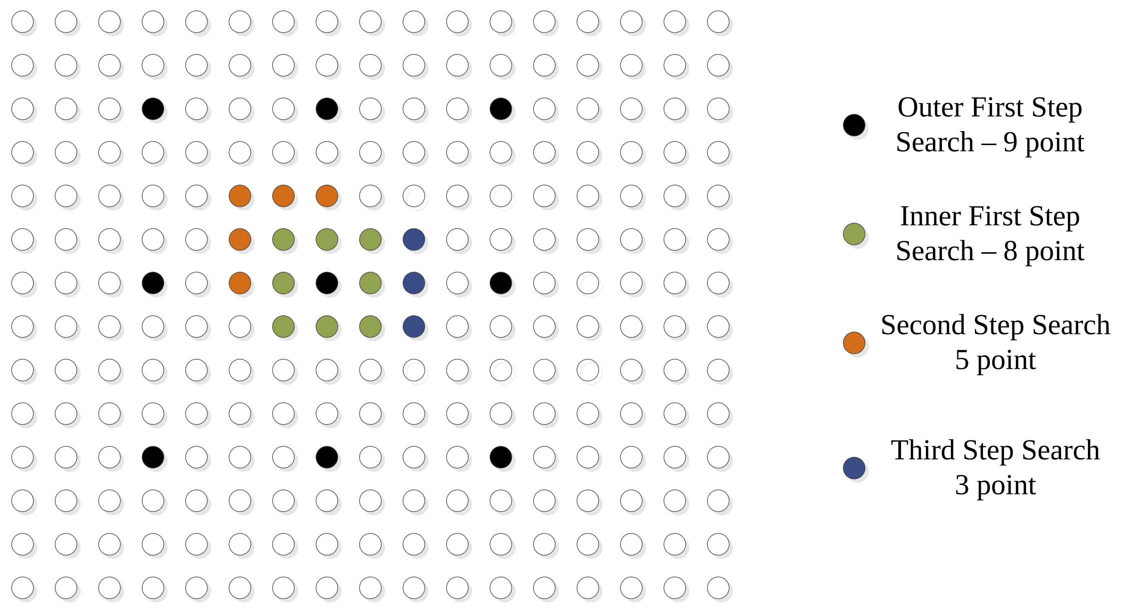

Outer First Step Search – 9 point
Inner First Step Search – 8 point
Second Step Search 5 point
Third Step Search 3 point

Figure 20 The New Three Step Search motion estimation algorithm

Notice that NTSS is center biased matching so it searches until it finds minimum cost at center. Considering the best-case scenario for NTSS is 17 steps matching, and the worst case is 33 steps matching if cost function is not at center. The table 1 shows a the comparison of simulation results of exhaustive search, three step search, and new three step search algorithm in terms of number of block matching done for a macro-block. Tennis and football videos are considered as inputs.

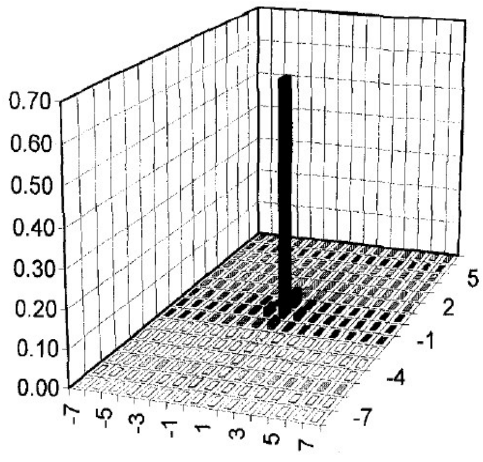
Table 1 The comparison of motion search algorithms

	Full Search	Three Step Search	New Three Step Search
Football	152	25	19
Tennis	175	25	22

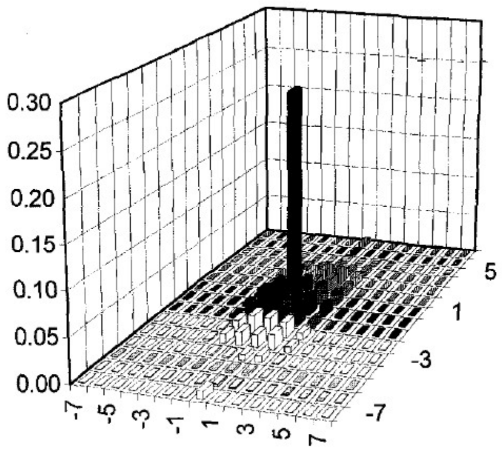
Table 1 shows the number of computational steps is much less in TSS and NTSS compared to full search. The fast search algorithms compromise in image quality in order to improve computation.

3.2.3 Novel Four Step Search Algorithm

NTSS modifies TSS by assuming center biased matching, and halfway stop instead of going through all search steps in case of stationary or quasi-stationary image. Those assumptions are made based on fact that real world images are gentler, smoother, and have slow motion. Figure 21 shows motion vector distribution in football and tennis based on observation [12]. 4SS also implements center biased matching pattern. It starts searching at center at nine points with initial step size equal to 2, instead of initial step size equal to 4 in 3SS. A 4SS initial search window 5x5 is compared with 3SS initial search window 9x9. If minimum cost found at center of the block, 4SS will jump to the forth step. If the minimum cost located at one of eight neighbor points, 4SS will go to the second or the third step. 4SS maintains step size equal to 2 and searches in 5x5 window in the second or the third step. In the fourth step, step size is reduced to 1, and the search window is set to 3x3. It matches nine points in the final step, and the minimum cost point would be considered as the matching macro-block.



Football



Tennis

Figure 21 The motion vector distribution in football, and tennis frames [12]

The 4SS algorithm is given below.

Step 1: Search for minimum cost at nine points in 5x5 window located at the center of the search area. If the minimum cost is located at the center of the search window, jump to step 4; else go to step 2.

Step 2: maintains search window of 5x5. If the minimum cost is found at the center of search window, jump to step 4. If the minimum cost locates other five searched points, go to step 3. The match pattern depends on the position of the previously matched points. If the previous minimum cost located at the corner of the search window, five point search pattern will be used which is shown in figure 22 (b). If the previous minimum cost is located at the center of the vertical or the horizontal axis of search window, three point search pattern will be used which is shown in figure 22 (c).

Step 3: Repeat process mentioned in step-2. Maintains search window of 5x5 and maintain search pattern mentioned in step 2.

Step 4: Reduce search window size to 3x3, and step size to 1 as shown in figure 22 (d). The minimum cost point would be considered as motion vector among nine point search pattern.

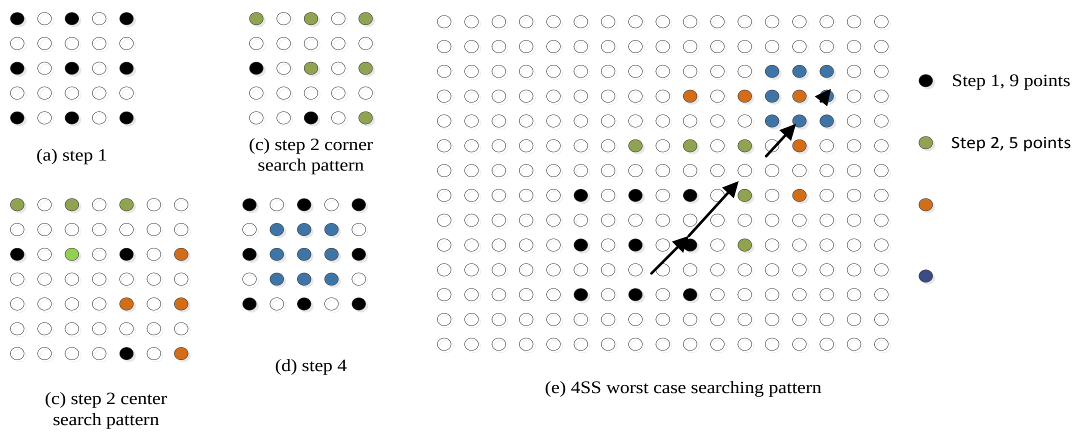
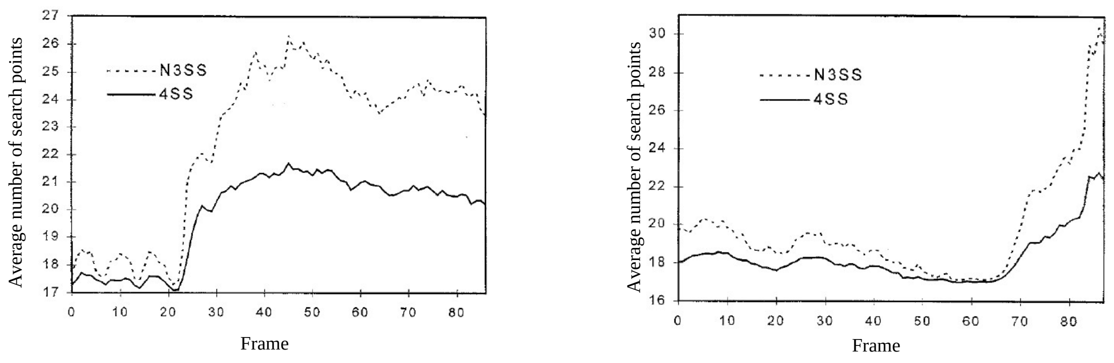

(a) step 1
(c) step 2 corner search pattern
(c) step 2 center search pattern
(d) step 4
(e) 4SS worst case searching pattern
Step 1, 9 points
Step 2, 5 points

Figure 22 The 4SS algorithmic search pattern (a) step 1 search pattern, (b) step 2 additional 5 points search pattern, (c) step 2-additional 3 points search pattern, (d) final step search pattern, and (e) worst case scenario for 4SS algorithm

From above algorithmic step, we can estimate the best case matching steps and the worst case matching steps. In the best case of 4SS, the matching point could get from step 1 and step 4. The total number of searching points is 9+8=17 for best case. The 4SS follow all steps to get minimum cost point in the worst-case condition. The total number of matching require in worst case is 9+5+5+8=27 steps. Figure 22 illustrates the 4SS algorithmic search pattern and the worst-case scenario.

Table 2 The number of search point per macro-block for FS, TSS, NTSS, and 4SS

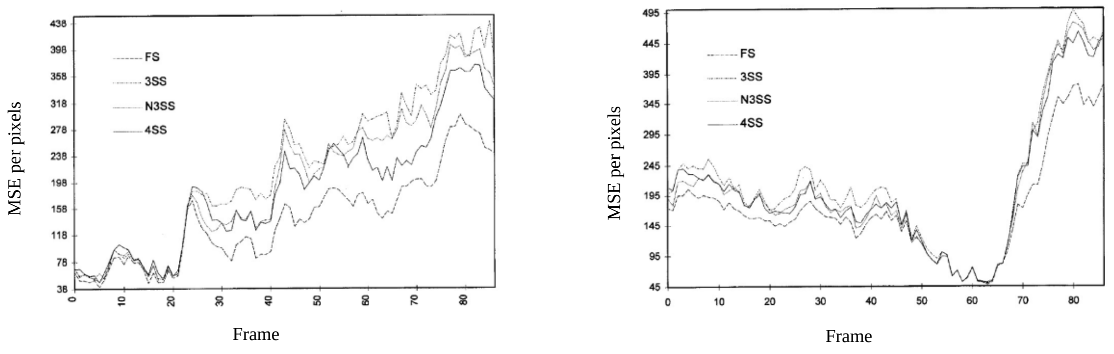
	Full Search	Three Step Search	New Three Step Search	Novel Four Step Search
Football	152	25	19	18
Tennis	175	25	22	20



(a) Tennis Sequence

(b) Football Sequence

Figure 23 The comparison of NTSS and NFSS considering average search points per macro-block in video frames for (a) tennis sequence, (b) football sequence [12]



(a) Tennis Sequence (b) Football Sequence

Figure 24 The comparison of NFSS, NTSS, TSS, and FS for videos of (a) tennis, (b) football [12]

Table 2, shows the comparison of average number of search point require in discussed motion estimation algorithm. Table 2 shows the 4SS algorithm has minimum number of comparison per motion vector. Figure 23 shows average number of search point verses frame for NTSS, and NFSS algorithms for (a) tennis, (b) football. The figure 23 indicates the number of search point is always less than 25 in NFSS algorithm and is significantly low compared to NTSS algorithm in the highly active areas. Figure 24 shows comparison of MSE and the frame for all motion estimation algorithms. The figure 24 indicates the FS has very low pixel error value compared to fast motion search algorithms. The NFSS algorithm is compromised with the pixel values to reduce the computational cost in the video codec. The NFSS has higher pixel error compared to the FS, but lower pixel error compared to other fast search algorithms. Indeed, the figure 23, figure 24, and table 3.2 indicates that the NFSS algorithm improves the performance of video codec compared to the TSS, the FS and the NTSS, without compromising image quality significantly in fast motion area[12].

Chapter 4

MOTION COMPENSATION

Motion estimation and motion compensation are playing important role in the H.264 video codec. Motion compensation is considered as module of the prediction process. Motion estimation and motion compensation techniques are employed to remove temporal redundancy between consecutive frames of the video data. An encoded frame has enough amount of temporal redundancy after applying motion estimation. The observation indicates that either a camera or an object is moving in the moving picture. The difference between consecutive frames of video data should result into motion of camera or motion of object in frames. The results of motion estimation are motion vectors, but this output does not match with desired output. Motion vector indicates the displacement of macro-block of the current frame in the reference frame. Motion estimation cannot remove temporal redundancy from consecutive frames of the video data. The H.264 has introduced motion compensation block to remove complete temporal redundancy.

Motion compensation regenerates a reference frame. The regenerated reference frame is more likely the current frame, because the reference frame is generated using a motion vector of the current frame and a macro-block of the reference frame. The regenerated reference frame is also known as a compensated frame. The difference of the current frame and the compensated frame is known as a residual frame. The residual frame contains only a camera motion or an object motion in frame. Motion compensation transmits the predicted error, which has much less correlation with the original frame. The predicted error can be coded at lower bit rate. The same prediction is generated at the receiver side, and is combined with the received error to regenerate the current frame [13]. Motion compensation enables low bit rates and low bandwidth features of H.264 video codec.

The motion compensation can be implemented by fixed block size, variable block size, overlapped block size, and half pixel and quarter pixel. The current frame is divided into a number of non-overlapped macro-block. The motion vectors are estimated for all generated macro-blocks by motion estimation. The motion vectors are extra information, and require lots of memory on a storage space. In fixed block size, the current frame is encoded as conventional way. The variable block size utilizes block-matching compensation, with ability to choose dynamic block size for video encoder. Thus, VBMC (variable block motion compensation) requires a less number of bits to represent a motion vector by merging the macro-blocks into a large block. Due to non-overlapped block matching, the discontinuities are introduced at edges of the block. The overlapped block size avoids the discontinuities, which is induced in case of non-overlapped block size. The OBMC (overlapped block motion compensation) block size is twice big in each dimension than non-overlapped block size. It is overlapping eight macro-blocks surrounded the original macro-block. The OBMC helps to improve quality of the reconstructed frame. The half-pixel and quarter-pixel motion compensation have ability to calculate sub-pixel values. The sub-pixels are interpolated based on full-pixel values and full motion vectors by utilizing bi-cubic or bilinear 2D filtering. It treads off between high cost and complexity to accuracy. This chapter will discuss about FSBM (fixed size block matching) and VSBM.

4.1 Fixed Block Motion Compensation

The fixed block motion compensation is the conventional technique for motion compensation. The motion vectors and the reference frame inputs to FBMC module. FBMC generates the compensated current frame based on the given input, and additional parameter such as a macro-block size, and a search area. The first motion vector is applied as an input to FBMC. The FBMC moves into XY direction based on the motion vector. Then the macro-block is picked up based on XY-coordinates, and is placed into the first location of a blank frame. The FBMC moves to the next motion vector. These steps are repeated until FBMC walks through all the motion vectors. As a result, the compensated frame, and motion vectors without any modifications are outputted at the end of motion compensation. Figure 25 shows comparison of macro-block in FBMC and VBMC.

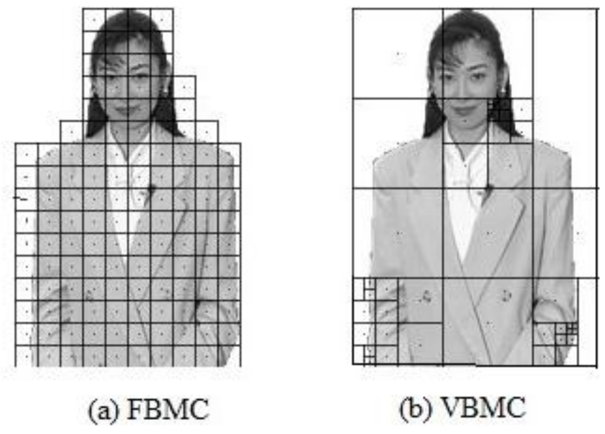


Figure 25 The comparison of the macro-block size in FBMC and VBMC

4.2 Variable Block Motion Compensation

The VBMC merges normal size of a macro-block into a block of varying size, and the VBMC considers the motion in each macro-block whether to merge them into a larger block. The FBMC has many disadvantages. The main disadvantages are that the encoded frame loses a motion inside of the macro-block due to fixed block size, and all motion vectors have to send over the network even though most of information is same in a frame. Those disadvantages can be improved by VBMC technique. The VBMC has block of varying size, so it can capture a small motion by dividing a block into a sub-blocks. The VBMC can avoid the transmission of all motion vectors by merging blocks into a larger block, and sends only single motion vector of large block. The proposed motion compensation methodology is based on a bottom-up merging process. It is implemented by merging macro-blocks into large rectangular region under preset threshold value. [14]

The proposed technique utilizes the fact that motion vectors of neighboring macro-blocks have high correlation among them. In most of a video data, the background either moves slowly or stands still. in the video pictures. The motion vectors representing the still background; can be merged before transmission. The main idea is to combine macro-blocks of still background into a single macro-block of still background. The storage space, bit-rate and bandwidth utilization can be reduced significantly by applying proposed bottom-up merging algorithm. The proposed scheme is very effective in moving picture, whose background and objects are moving in the same direction. The algorithmic details are discussed below.

The first step of motion compensation based on the bottom-up merging procedure is to find out motion vector of current frame by any kind of motion estimation algorithm. Once, the motion vectors are available to motion compensation module, then the bottom-up merging process is implemented in two steps. Firstly, the VBMC merges macro-block into the bigger block, and secondly, it selects one of the merged rectangular blocks among multiple rectangular regions [14]. Figure 26 shows an example of the bottom up merging procedure.

The VBMC is encoding a frame in raster scan order. It starts merging at each macro-block. The decision of merging should be taken after evaluating a difference of root macro-block and next macro-blocks in raster scan in line. If the difference of two macro-blocks is less than the adjustable threshold, the two macro-blocks are eligible to be merged into a region. The merged macro-block is known as the matched macro-block.

The first block is known as root block, because it starts merging process. This evolution process continues until the first mismatch occurs in the raster scan order. The evolution equation is shown below.

$$\sqrt{\{ (MV_{x,root} - MV_{x,\ next})^2 + (MV_{y,root} - MV_{y,\ next})^2 \}} \leq \text{Threshold} \ldots\ldots\ldots\ldots\ldots \quad (4.1)$$

Where, $(MV_{x,root,} MV_{y,root})$ is the motion vector of the root macro-block, and $(MV_{x,next,} MV_{y,next})$ is motion vector of the next macro-block

If the differences of the root block and the next block are bigger than the adjustable threshold in row, the next macro-block cannot be merged into the root block region. The macro-blocks, which stop the merging process are known as mismatched block. The distance between the root block and the mismatched block in the same row is defined as the maximum matching width for next row scan in line. The merging process is jumped to the next row, after the mismatched block is encountered in that row to continue merging process. The merging process should not exceed by the maximum matching width found in the above row. The whole merging process will stop when mismatched block appears in the same column of the root macro-block. If the first macro-block next to the root block in the row and the column is mismatched, the root block is encoded in the conventional way of motion compensation.

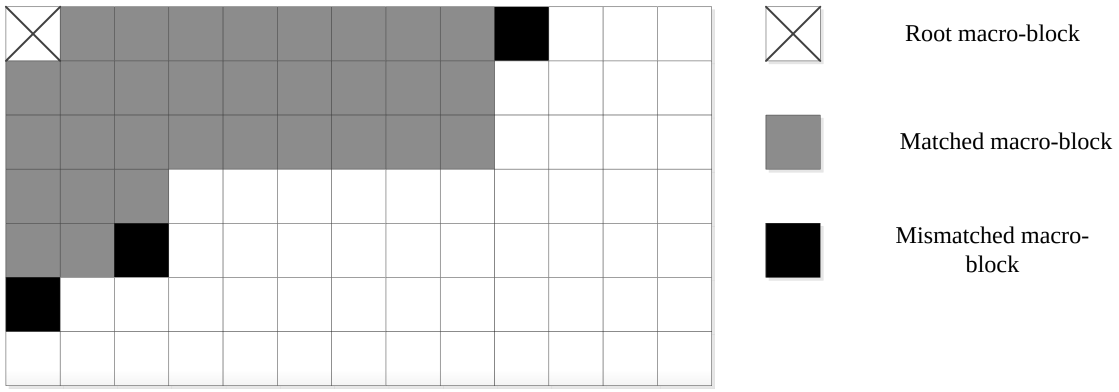


Figure 26 The example of the macro-block merging procedure

Figure 26 shows an example of the merging process. The merging process starts at the first block shown in the above figure with cross. The merging process is extended all the way to the right until the first mismatch is occurred in row. The first mismatch is occurred at the 10th macro-block so the matching width is defined as 9 macro-block wide for next scan row in line. The next scan row is the second row in the above figure. The merging process jumps to the second row for merging macro-blocks into a region. The merging process should not exceed the matching width. If all macro-blocks of the matched width of the current row are matched, the merging procedure jumps to next scan row in line. The next scan row is the third row in our example. If the mismatch occurs before the 10th macro-block in the current scan row, the new matching width is defined by the new mismatched macro-block in the current row. This new matching width is less than the original matching width, and is applied to the next scan row in line. In the above figure, the fourth row has new location of the mismatched macro-block; so the mismatched block of the fourth row is defined a new matching width for the fifth scan row in line. The mismatch occurs in the same column of the root block in the sixth row, so the whole merging procedure is stopped. The merging process moves to the next step of the bottom-up merging motion-compensation.

The selection of region is the second step of the bottom-up block merging process. The candidate regions are varying in size and shape so it is very inefficient coding technique to store information regarding all the regions [14]. The merged region is divided into multiple rectangular regions. The rectangular region, which has the highest number of macro-blocks than other the rectangular regions, is selected for encoding. The rest of the candidate regions are erased. The merging process continues at the first unmerged macro-block. Figure 27 shows an example of division of merged block into candidate region.

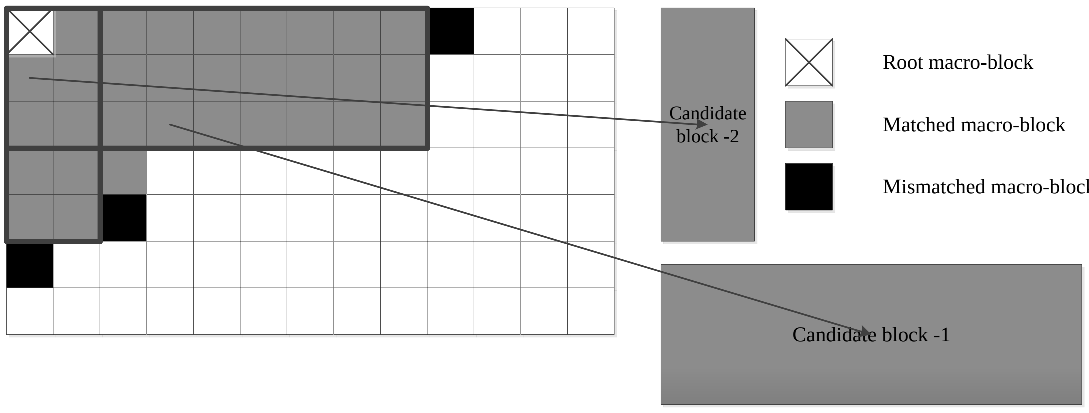


Figure 27 The example of division of merged block into candidate rectangular region

The figure 27 shows the two rectangular candidate region for the merging process. The candidate region-1 contains 27 macro-blocks, and the candidate region-2 contains 10 macro-blocks. The candidate region-1 is selected to merge. The upper left coordinates of the root macro-block, the lower right coordinates of merged region, and the number of macro-block into merged region is recorded for decoding process. This process is completed by repeating merging process. There are only merged and unmerged types of data are outputted in the end of the whole bottom-up merging process [14]. The decoding process is implemented by unmerging macro-blocks from merged regions in raster order. The unmerging process of the merged macro-blocks is executed using recorded upper left coordinates, lower right coordinates, and number of macro-blocks. The unmerged macro-blocks are compensated in raster scan manner, and the process of unmerged macro-blocks skips the region of merged macro-blocks.

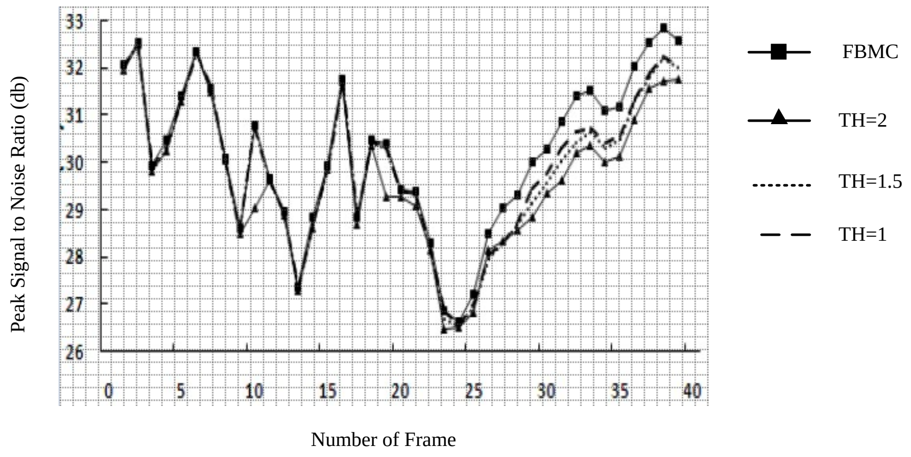


Figure 28 The comparison VBMC with different threshold and FBMC [14]

The aim of the motion vector coding using bottom-up merging procedure is to achieve higher compression by merging consecutive macro-blocks with insignificant motion vector difference. The caltrain video has characteristic that the train and the background are moving in the same direction. Most of the background information is same in the football, and the tennis video sequence. These characteristics of video sequence enable higher compression using VBMC than conventional block motion compensation. Figure 28 shows the graphical representation of FBMC, VBMC with threshold value 1, 1.5, & 2 for tennis video sequence. The PSNR is reduced because of higher threshold value as shown in above figure. As we keep on increasing the threshold value of the bottom-up merging process, the quality of image will be reduced significantly. The practical threshold value for the merging process should be nearly 1 to 1.5 to maintain the image quality, and to achieve higher compression based on the figure 28. This motion compensation technique offers good trade-off between quality of image and storage area.

Chapter 5

IMAGE COMPRESSION AND DECOMPRESSION MODEL

The chapter 2 described a basic overview of image compression model. This chapter will discuss image compression and decompression in depth. The residual image proceeds further for additional compression after the spatial and the temporal redundancy have been removed from the image shown in figure 6. This process is called an image compression model. Figure 29 gives the high-level block diagram of the image compression model.

The main candidates of image compression models are DCT block, quantization, and entropy coder. The incoming frame is divided into an 8-by-8 block size, and each block is passes through the image compression block. First, an 8-by-8 block goes through the Discrete Cosine Transform (DCT). The original signal is mapped into a frequency domain. It works on pure real numbers, and performs pure mathematical functions on the original signal.

An 8-by-8 matrix proceeds to the quantization block after DCT. It is a process of division of the DCT co-efficient by the quantization matrix. In simple words, the DCT co-efficient is divided by some integer value to achieve a high compression. The quantized block is converted into a serial stream using zigzag or alternation methodology.

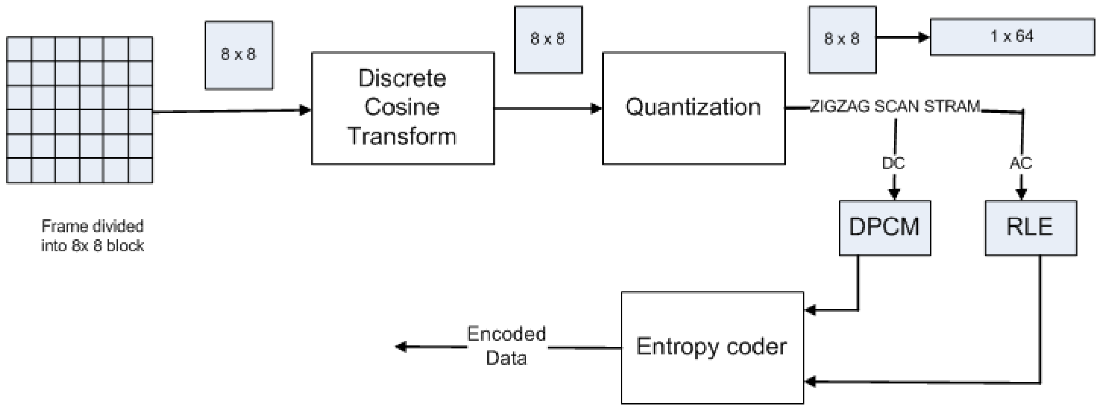


Figure 29 Block diagram of image compression model

This serial stream is passed through the intermediate block known as the differential pulse coding modulation (DPCM) and the run length coding (RLE). The DC component runs through the DPCM, while the AC component runs through the RLE block. The output of those blocks is applied to the entropy coder. The entropy coder is applied to the signal in order to represent the information as compactly as possible.

5.1 Discrete Cosine Transform

DCT is performing a math function on the original matrix. It converts a signal into the frequency domain matrix. It represents components in terms of the sum of the cosine function [5]. The incoming frame is divided into an 8x8 block, and the DCT is applied on each block in the frame from left to right and top to bottom. The first step is to generate a DCT matrix with the given equation. Table 3 shows the generated 8x8 DCT matrix using given the equation.

$$A(\mathrm{i},\mathrm{j}) = \mathrm{Ci}\cos\frac{(2j+1)i\pi}{2N}\text{……………..(5.1)}$$

Where,

$$Ci = \sqrt{\frac{1}{N}}\ (i = 0)$$

$$Ci = \sqrt{\frac{2}{N}}\ (i > 0)$$

Table 3 An 8-by-8 discrete cosine transform matrix

0.353553	0.353553	0.353553	0.353553	0.353553	0.353553	0.353553	0.353553
0.490393	0.415818	0.277992	0.097887	-0.097106	-0.277329	-0.415375	-0.49024
0.461978	0.191618	-0.190882	-0.461673	-0.462282	-0.192353	0.190145	0.461366
0.414818	-0.097106	-0.490246	-0.278653	0.276667	0.49071	0.099448	-0.41448
0.353694	-0.353131	-0.354256	0.352567	0.354819	-0.352001	-0.355378	0.351435
0.277992	-0.490246	0.096324	0.4167	-0.414486	-0.100228	0.491013	-0.27467
0.191618	-0.462282	0.461366	-0.189409	-0.193822	0.463187	-0.46044	0.187195
0.097887	-0.278653	0.4167	-0.490862	0.489771	-0.413593	0.274008	-0.09241

Figure 30 is a graphical representation of table 3. The top-left corner has lowest frequency component, and bottom-right corner has highest frequency component. In the figure 30, the frequency component is increased from left to right, and top to bottom.

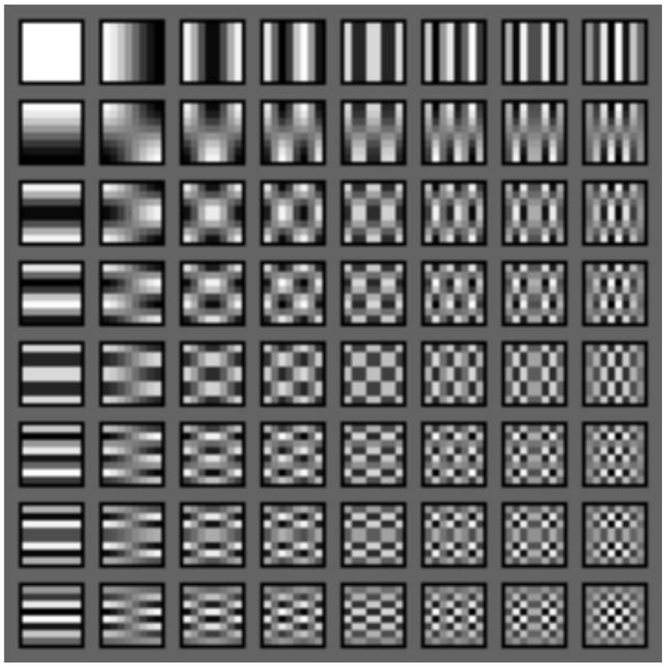


Figure 30 An 8x8 DCT matrix element is represented in frequency domain.

The two-dimensional (2D) 8-point DCT initially assumes that an 8x8 input matrix is only the rows of pixels, and the 1D DCT is applied to each of the rows. Again, the 1D DCT is applied to each of the columns of the resultant matrix. At the end, the original matrix is transformed into a frequency domain matrix. The 2D DCT is illustrated using the following example. Table 4 is shown an original matrix on which 2D 8 point DCT will be applied.

Table 4 The original pixel values of picture before DCT

125	131	127	125	126	122	121	122

123	127	131	133	132	127	126	127
123	125	131	131	130	129	126	128
128	123	128	122	121	121	120	121
127	122	124	122	121	122	119	123
126	123	128	124	123	124	123	126
121	126	130	13d3	131	131	129	129
121	126	130	134	132	130	130	130

The DCT is designed to work on real values ranging from -127 to 128. An Image frame has color pixel, and the color-pixel value varies from 0 to 256. A 128 is subtracted from the original matrix to map color-pixel value -127 to +128. The resultant matrix is shown in table 5 after subtraction.

Table 5 The resultant matrix after subtract integer number 128 from each pixel

-3	3	-1	-3	-2	-6	-7	-6
-5	-1	3	5	4	-1	-2	-1
-5	-3	3	3	2	1	-2	0
0	-5	0	-6	-7	-7	-8	-7
-1	-6	-4	-6	-7	-6	-9	-5
-2	-5	0	-4	-5	-4	-5	-2

-7	-2	2	5	3	3	1	1
-7	-2	2	6	4	2	2	2

The DCT output can be obtained by matrix multiplication. Let the table 5 matrix is called M, the table 3 matrix is called T, and inverse of table 3 is called T_inv. A matrix M is multiplied by the DCT matrix-T in first multiplication, which transposes a row of the matrix-M. The resultant matrix of a first multiplication is multiplied with the matrix-T_inv in the second multiplication. The multiplication transposes a column of resultant matrix of the second multiplication. It generates final resultant matrix, shown in table 6. The output of DCT block is applied to the quantization block.

Table 6 An output matrix from 2-D 8 point DCT block

1009	2	-9	-7	1	-1	4	1
-4	8	1	0	0	-3	-1	1
14	-5	-8	-3	-1	-3	-6	-2
-10	6	7	2	0	-2	0	0
-10	6	6	2	0	-1	0	0
-1	2	1	0	-2	0	0	0
-1	0	2	0	-1	0	1	0
2	-3	-2	1	0	-1	-1	-2

5.2 Quantization

An 8-by-8 DCT block is ready for the next level of compression, which is known as quantization. It is a process of transformation of continuous set of values to the finite set of small values. When the quantization is performed on the original signal, the part of information is lost from the original signal. It means higher the compression but lower the quality of image, and vice versa.

A block size of 8x8 quantization matrix is created using the popular formula { 1 + (i+j) * quality }. The variable of quantization equation sets a tradeoff between the compression and the image quality. The quantization matrix is duplicated in a step of 8-by-8 block in the size of original frame. The original signal is divided by quantization matrix, and the resultant matrix is rounded to nearer integer value, which is known as quantize matrix.

The principle of quantization is worked on the fact that the human visual system is more sensitive to a lower frequency signal than a higher frequency signal. The goal of quantization is to apply higher compression on high frequency signal and less compression on low frequency signal. The quantization is illustrated with the following example. The table 6 is inputted to a quantization block. It is considered as the original matrix for quantization block. The table 7 is illustrated the quantization matrix by setting quality=2 in this chapter.

Table 7 The quantization matrix using 1+(i+j)*quality, quality=2

5	7	9	11	13	15	17	5
7	9	11	13	15	17	19	7
9	11	13	15	17	19	21	9
11	13	15	17	19	21	23	11
13	15	17	19	21	23	25	13
15	17	19	21	23	25	27	15
17	19	21	23	25	27	29	17
19	21	23	25	27	29	31	19

Table 8 The resultant matrix after quantization

202	0	-1	-1	0	0	0	0
-1	1	0	0	0	0	0	0
2	0	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

The high frequency component is rounded to nearest integer and most of them are zero after quantization. Table 8 shows the quantized matrix. When the de-quantization is applied to the received signal, the de-quantized signal losses partial information of the original received signal. Those errors are increased because of inverse DCT on the de-quantize matrix. Due to de-quantization, and inverse-DCT, the image quality degrades compare to original image quality. A quantize matrix is inputted to a entropy coder.

5.3 Entropy Coder

The quantize matrix m-by-n is converted to 1-by-(m*n) vector before applying to the entropy coder block. The matrix conversion is possible using multiple different methods, but two of them are presented here. The zigzag scan and the alternative scan methods are shown in figure 31.

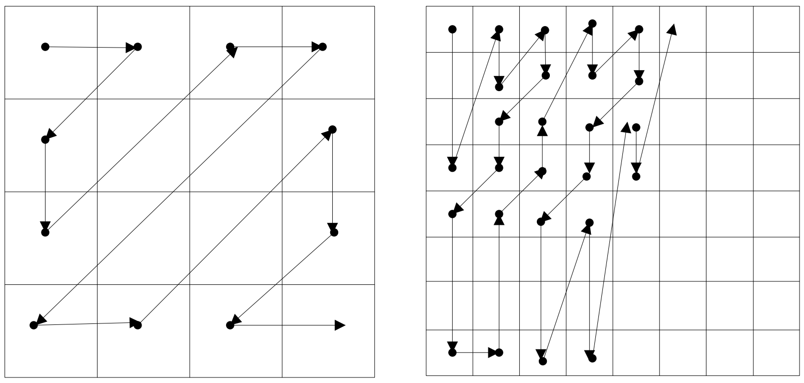


Figure 31 Zigzag Scan, and Alternative Scan

The differential pulse code modulation (DPCM), and the run length coder (RLE) are considered as a part of entropy coder. DPCM is used to process DC Components before passing through entropy coder. DC component has still higher values, but these values are very close to the previously coded DC values. The difference of the current value and the previously coded value is outputted from DPCM for the final compression.

The final step of the image compression model is an entropy coder. The simplest entropy coder is known as a run length coding, but the huffman coder and the arithmetic coder is used to achieve higher compression. The entropy coder is used to compress information without any loss of information. The H.264 uses content based adaptive variable length coding, and it is processed in major five steps

1. Encode the number of coefficient and the trailing ones
2. Encode the sign of each trailing ones
3. Encoded the level of non-zero coefficient
4. Encode the total number of zeros before the last coefficient
5. Encode the location of encoded zeros

5.3.1 Run Length Coder

An image serial sequence is represented in form of tuple by RLE. The tuple has two fields; one represents a symbol and other represents a length. Let us consider the example of sequence 44555567777. This sequence is represented in tuples as (4,2), (5,3), (6,1), (7,4). A biggest down fall of RLE is that it sometimes ends up with large data size instead of a compressed data because of large but not repetitive symbol in the sequence [6].

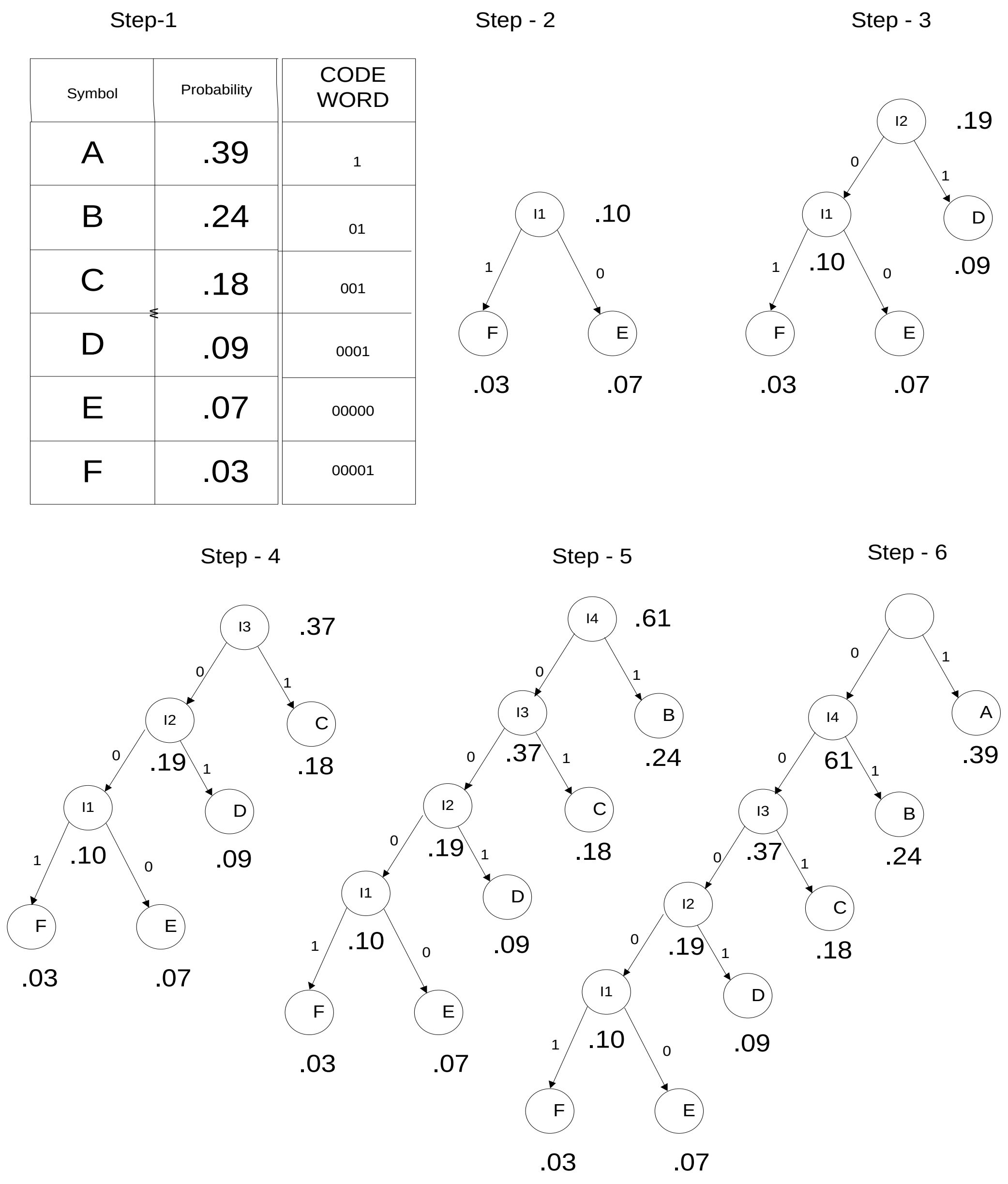
The quantize DCT matrix converted into a serial stream with zigzag scan. The long zero streams after non-zero coefficient are noticed in the serial stream. The long zero chain after non-zero number can be removed by RLE, and it saves length of chain.

5.3.2 Huffman Entropy Coder

Huffman entropy coder is known as a variable length coder, and very popular and lossless method to achieve additional compression. The variable length coder means each symbols has different codeword length. It is worked on probability of symbol of the given sequence. It uses shortest length codeword for high probability symbol, and largest length codeword for low probability symbol. Huffman coder works on the following properties.

1. Different codeword must represent different symbol.
2. Define codeword provides enough information for symbol, and do not require any other information than codeword.
3. Different codeword might have equal length, but their final bits are not same.
4. Length of higher probability symbols has short or equal length codeword than the lower probability symbols.

The given example clears idea of the huffman coding. In the first step, it creates the symbol table and arranges each of the symbols based on the highest probability to the lowest probability. In the next step, the lowest probability symbol is combined and lowest probability symbol is assigned a value one, and zero to second lowest probability of symbol. These steps are repetitively performed until the whole tree is created. Figure 32 is explained the Huffman coded tree.



Step-1

Symbol	Probability	CODE WORD
A	.39	1
B	.24	01
C	.18	001
D	.09	0001
E	.07	00000
F	.03	00001

Figure 32 An Example of the Huffman code

5.4 Image Decompression Model

The image decompression is a reverse process of image compression. The transmitted signal is received at the receiver side. The received signal is passed through entropy coder, de-quantization, and inverse-DCT sub-blocks of the image decompression. The entropy decoder converts the codewords into its respective symbols. Those symbols are converted in the serial stream of 1-by-(m*n), The serial stream is converted into a matrix, which is known as the received quantized-DCT matrix of m-by-n form.

The output of entropy decoder is applied to a de-quantization block of decoder. The de-quantization is done with multiplication of received quantized-DCT matrix and quantized co-efficient matrix. The resultant matrix is shown in the table 9 after de-quantization.

Table 9 The de-quantized resultant matrix, quality=2

1010	0	-9	-11	0	0	0	0
-7	9	0	0	0	0	0	0
18	0	-13	0	0	0	0	0
-11	0	0	0	0	0	0	0
-13	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

0	0	0	0	0	0	0	0

The resultant matrix of de-quantized block is applied to the inverse-DCT block. The reconstructed matrix from an inverse-DCT and de-quantization should be same based on theoretical condition, but the table 9 is different from the table 6. It means that the reconstructed signal is loses partial information in quantization/de-quantization process. This error will be increased after application of inverse DCT on de-quantized matrix. Table 10 shows the resultant matrix after inverse DCT. The inverse DCT reconstructs the original residual signal from its DCT coefficient. The output from IDCT is the residual frame.

Table 10 The resultant matrix after inverse DCT

121	125	130	130	127	123	121	120
126	129	132	133	130	127	125	126
127	129	131	130	127	125	126	128
123	124	124	122	120	120	122	125
120	121	122	121	119	119	121	124
122	125	127	127	125	124	126	128
124	127	132	133	132	130	130	131
122	127	133	135	134	131	130	130

The H.264 is the block-based video codec standard, so it reduces the reconstructed image quality compare with the original image. The reconstructed image filters out the pixels value of an image to improve the image quality. Many proposed filtered is implemented to enhance the image quality and to reduce the blocking artifact. The de-blocking filter is implemented to remove blocking artifact in this project.

5.5 De-blocking Filter

The H.264 is the widely accepted standard for video compression. It is the block based video codec standard. When H.264 decompresses the received image, the blocking artifact can be observed in the reconstructed image. Due to block based DCT and quantization, the frequency components are changed abruptly near the block boundaries. The block boundaries can be identified by HVS in the uncompressed image and video. It is known as a blocking artifact. Filter is used to enhance the image quality, remove blur, and increase PSNR. The popular de-blocking filter is implemented in H.264 video codec standard.

Multiple approaches have been offered to filter out pixel values and smoothing out artifacts in reconstructed image. The implementation can be applied either on DCT co-efficient or on pixel value of the reconstructed image, but the averaging values of boundary pixel is utilized by most of the filter implementation. The filter takes the benefit of the fact that any pixel value and its neighboring pixel value are very nearer to each other, but it is not true for the boundary pixels of the image. Boundary pixels have very low inter pixel correlation. The reconstructed image is turned into blurred image because of low inter pixel correlation near boundaries. The implementation steps of de-blocking filter will be discussed below [15].

1. The reconstructed current image is applied to de-blocking filter and reads the image into a 256x256 matrix. Consider a 8x8 macro-block size
2. Apply 2D-DCT and quantization on the matrix. The quantization matrix is generated with the quality factor set to 25 as discussed earlier in the section 5.2.
3. Read the DC sub-band from a 256x256 quantization matrix into a 32x32 matrix.
4. Apply filtering on the DC sub-band, and filter DC sub-band by 3x3 mask neighborhood of sub-band pixel.
5. Replace the original pixel value with the filtered pixel value if the filtered pixel value falls within original value of quantization range
6. Discard the filtered pixel value if it falls outside of quantization range.
7. Repeat filtering process and move the 3x3-window to next block in line.
8. Replace the original DCT coefficient with the new DC sub-band.
9. Apply inverse-DCT on updated DCT coefficient to reconstruct final image.

Each of the pixels in DC sub-band is filtered by filtering process mentioned in the step 4. The pixels in the 3x3 block are segmented and each of the segments is processed independently using value averaging process. The segments are classified as a class zero segment and class one segment. Each of the pixels of 3x3 mask is categorized into one of the segment based on the following condition,

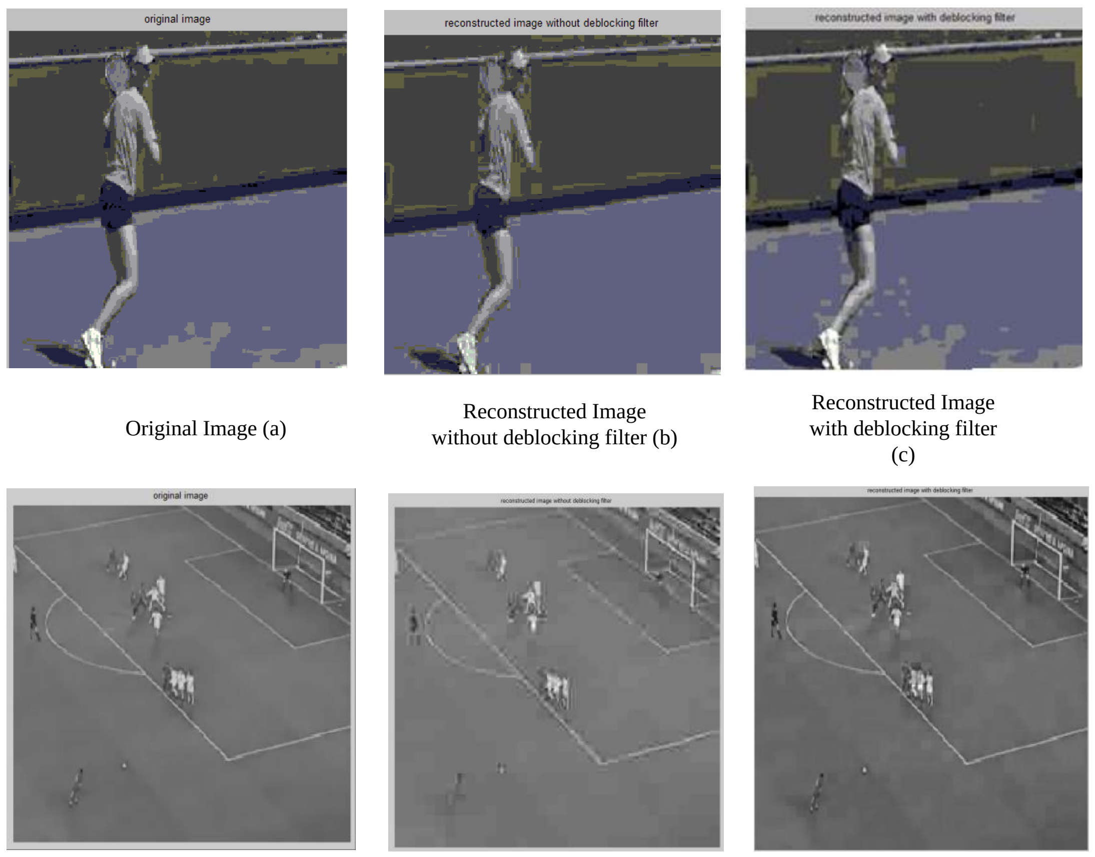
$$segment = \begin{cases} 0 & \text{if pixel} < \text{mask average;} \\ 1 & \text{else.} \end{cases} \qquad \ldots\ldots\ldots\ldots (5.2)$$

The average value of pixels is calculated and known as a filtered pixel value. The condition of replacement of original pixels is mentioned in step 5.

The PSNR is calculated for the reconstructed image with and without using de-blocking filter. The simulation results are shown in figure 33. The original image is shown in figure 33 (a), and the reconstructed image without de-blocking filter is shown in figure 33 (b). The humans are important information in original image. The important information is distorted in reconstructed image without de-blocking filter. The reconstructed image with de-blocking filter is shown in figure 33 (c). The reconstructed image with de-blocking filter is more closely matched to the original image. The tennis and football video are shown in the below figure. The table 11 shows the improvement in PSNR of the reconstructed image with filter compared to the reconstructed image without filter.

Table 11 The comparison of PSNR of reconstructed image with filtered and without filtered

Reconstructed image	Without filter	With filter
PSNR(db)	39.007	40.335


original image
reconstructed image without deblocking filter
reconstructed image with deblocking filter
original image
reconstructed image without deblocking filter
reconstructed image with deblocking filter

Original Image (a)

Reconstructed Image without deblocking filter (b)

Reconstructed Image with deblocking filter (c)

Figure 33 The original image, the reconstructed image without filtering, the reconstructed image with filtering for tennis and football sequence

Chapter 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

H.264 video codec is the most effective video compression standard developed in video industries. H.264 uses more accurate predication algorithms and motion compensation techniques to achieve better compression and same quality video at the low bit rate, without compromising the image quality. The H.264 encoder and decoder is implemented and simulated in the MATLAB, and the simulation results gives an idea about improvement made in H.264 standard. The implementation details show us the complexity involved in the real time application and hardware implementation. The focus of this project is motion estimation and motion compensation techniques. The image compression techniques and de-blocking filter are also important in the video compression techniques, but motion estimation and motion compensation are the most computationally expensive and time consuming process in whole video compression process. The high definition video has very high temporal redundancy in the consecutive frames. This temporal redundancy can be removed with fast motion search algorithms, and motion compensation. This process keeps only required information in the frames and forwards to other video compression blocks. The entropy coder outputs low bit rate information compared to the previously developed standards with same video quality.

The different algorithms are discussed for inter frame predication in this project, and compared those algorithms based on the search pattern, number of computation step, and peak signal to noise ratio. The novel four step search algorithm requires minimum computation step to search motion vector compared to other discussed algorithms. The reconstructed video quality in NFSS is matched to the reconstructed video quality in exhaustive search.

The computational cost can be reduced utilizing parallel processing technique in the hardware implementation. To enable a parallel processing, the macro blocks are processed on dedicated and special processor, so the all motion vectors will be outputted for an individual frame.at the same time. This technique definitely reduces the time and computation complexity, but also increases the requirement of hardware resources.

6.2 Future Work

This project has discussed multiple H.264 motion estimation algorithms, motion and compensation algorithms. H.264 algorithms are block based algorithm. The motion is considered constant within block, which is main pitfall of block based algorithm. H.264 video codec has been applied in diverse application such as high definition DVD, digital video broadcasting including HDTV, YouTube video storage, 3G mobile, apple face time application. Those applications require very high accuracy with low latency. The accuracy can be achieved by developing faster motion search algorithm with less computational expensive, and half-pixel & quarter pixel motion compensation algorithms. Those future algorithms generate more accurate motion vector with less computation cost, and calculate the inter pixel value to enable low bit rate transmission.

APPENDIX

Acronyms

ITU-T - International Telecommunication Union–Telecommunication standardization sector.

ISO - International Organization for Standardization.

AVC – Advanced Video Codec.

VCEG - Video Coding Experts Group.

MPEG - Moving Picture Experts Group.

IEC - International Electro-technical Commission.

CODEC – COmpression and DECompression.

MATLAB – MATrix LABoratory.

DPCM – Differential Pulse Code Modulation.

RLE – Run Length Coder.

PSNR – Peak Signal to Nose Ratio.

MAD – Mean Absolute Difference.

MSE – Mean Square Error.

SAE – Sum of Absolute Error.

SSE – Sum of Square Error.

DCT – Discrete Cosine Transform.

FS – Full Search.

ES – Exhaustive search.

TSS – Three Step Search.

3SS – Three Step Search.

NTSS – New Three Step Search.

NFSS – Novel Four Step Search.

4SS – Four Step Search.

FBMC – Fixed Block Motion Compensation.

VBMC – Variable Block Motion Compensation.

OBMC – Overlapped Block Motion Compensation.

HVS – Human Visual System.

BIBLIOGRAPHY

[1] G. J. Sullivan, P. Topiwala, and A. Luthra, "The H.264/AVC Advanced Video Coding Standard: Overview and Introduction to the Fidelity Range Extensions", Presented at the SPIE Conference on Applications of Digital Image Processing XXVII Special Session on Advances in the New Emerging Standard: H.264/AVC, August 2004.

[2] http://www.drtonygeorge.com/video_codec.htm

[3] Thomas Wiegand, Gary J. Sullivan, "Overview of the H.264/AVC Video Coding Standard", IEEE Trans. on Circuits and Systems for Video Technology, vol. 13, no. 7, pp. 560–576, July 2003.

[4] H. Jozawa, K. Kamikura, H. Kotera, H.Watanabe, "Two-Stage Motion Compensation Using Adaptive Global MC and Local Affine MC", IEEE Trans. on Circuits and Systems for Video Technology, vol. 7, no. 1, pp. 75–85, February 1997.

[5] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete Cosine Transform", IEEE Trans. Computers, 90-93, Jan 1974

[6] http://www.cs.uga.edu/~sadiq/pdf/vasim.pdf

[7] Soon-kak Kwon, A. Tamhankar, K.R. Rao, "Special Issue on Emerging H.264/AVC video coding standrad" J. Visual Communication and Image Representation, vol 17.

[8] http://en.wikipedia.org/wiki/Video_compression

[9] K. Peter, "Algorithms,complexity analysis and VLSI architectures for MPEG-4 motion estimation", Kluwer Academic Publishers, Boston, 1999.

[10] Jianhua Lu and Ming L. Liou, "A Simple and Efficient Search Algorithm for Block-Matching Motion Estimation", IEEE Trans. on Circuits and Systems for Video Technology, vol. 7, no.21, pp. 429-433, April 1997.

[11] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation" IEEE Transactions on Circuits and Systems Video Technology,vol. 4, pp. 438–443, Aug. 1994.

[12] P. Lai-Man and M. Wing-chung, “A novel four-step search algorithm for fast block motion estimation”, IEEE Transactions On Circuits and Systems for Video Technology, vol. 6, pp. 313-317, Jun. 1996.

[13] M.H. Chan,.B. Yu, A.G. Constantinides, “Variable size block matching motion compensation with applications to video coding”, IEE Proceeding, Vol. 137, Pt. I , No. 4, pp 205-212, August 1990

[14] Shou-Yi Tseng, Che-How Mak, “A Motion Vector Coding Scheme Based on Bottom-up Merging Procedure”, IEEE Transaction on advances in multimedia, pp 125-129, July 2009

[15] L. Kizewski, Student Thesis on “Image Deblocking Using Local Segmentation”, School of computer science and software engineering Monash University, November 2004

[16] http://www.vcodex.com/files/H264_intrapred_wp.pdf