FAST BLOCK MATCHING MOTION ESTIMATION ALGORITHMS FOR VIDEO COMPRESSION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

ELECTRONICS AND INSTRUMENTATION

BY

B KASI VISWANATHA REDDY

ROLL NO- 211EC3317

Department of Electronics and Communication Engineering

National Institute of Technology Rourkela-769008

2013

FAST BLOCK MATCHING MOTION ESTIMATION ALGORITHMS FOR VIDEO COMPRESSION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

ELECTRONICS AND INSTRUMENTATION

BY

B KASI VISWANATHA REDDY

ROLL NO -211EC3317

UNDER THE SUPERVISION OF

PROF. SUKADEV MEHER

Department of Electronics and Communication Engineering

National Institute of Technology Rourkela-769008

2013

Department of Electronics & Communication Engineering

National Institute of Technology Rourkela

Date: 31-05-2013

CERTIFICATE

This is to certify that the thesis titled, **"Fast Block Matching Motion Estimation Algorithms For Video Compression"** submitted by Mr. **B KASI VISWANATHA REDDY** in partial fulfillment of the requirements for the award of Master of Technology Degree in Electronics and Communication Engineering with specialization in "**Electronics and Instrumentation**" during session 2012-2013 at the National Institute of Technology, Rourkela is a bona fide research work carried under my supervision.

Prof. SukadevMeher

ACKNOWLEDGEMENTS

This project is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me.

I would like to extend my gratitude and my sincere thanks to my honorable, esteemed supervisor **Prof. SUKADEV MEHER**. He is not only a great teacher/professor with deep vision but also and most importantly a kind person. I sincerely thank for his exemplary guidance and encouragement. His trust and support inspired me in the most important moments of making right decisions and I am glad to work with him. My special thank goes to **Prof. Sukadev Meher** Head of the Department of Electronics and Communication Engineering, NIT, Rourkela, for providing us with best facilities in the Department and his timely suggestions.

I want to thank all my teachers **Prof. S.K. Patra**, **Prof. K.K. Mahapatra**, **Prof. U.C. Pati**, **Prof. T.K. Dan**, **Prof. S.K .DAS** and **Prof. L. P. ROY** for providing a solid background for my studies and research thereafter. I would fail in my duty if I don't think the PhD Scholar **Mr. Deepak Singh** without whom the work would not have progressed. They have been great sources of inspiration to me and I thank them from the bottom of my heart.

I would like to thank all my friends and especially my classmates for all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I have enjoyed their companionship so much during my stay at NIT, Rourkela. I would like to thank all those who made my stay in Rourkela an unforgettable and rewarding experience.

Last but not least I would like to thank my parents, who taught me the value of hard work by their own example. They rendered me enormous support during the whole tenure of my stay in NIT Rourkela.

B. KASI VISWANATHA REDDY

CONTENTS

Abstract

As the telecommunication technology grows in the modern era from internet to video conferencing, Video compression has become an avoidable feature in information broadcast and also in the entertainment media. In this thesis we compared a different block matching motion estimation algorithms to find the motion estimation with a rapid growth of multimedia information; when transmitting a large amount of data video coding standards have become crucial. Motion estimation ascertain to be the key to splendid performance in video coding by recover the temporal redundancy effectively between adjacent frames, so it has been widely used to popular video compression coding standards such as MPEG-2, MPEG-4 and recent video coding standards H.264 of video data for storage and transmission.

So Based on the study of motion vector distribution from several commonly used test image sequences, a three step diamond search [TSDS]algorithm for fast block matching motion estimation is proposed in this paper .The performance of this algorithm is compared with other existing algorithms of basic full search [FS], three step search [TSS] and diamond search [DS] by means of error metrics and no of search points in this the simulation results shows that the proposed three step diamond search algorithm achieves close performance with that of diamond search [DS] and uses less no of search points than the three step search[TSS]. When compared with original diamond search [DS] algorithm, this algorithm requires less computation time and gives an improved performance.

List of Figures

List of Tables

Chapter 1

1. Thesis Overview

1.1 Thesis Motion

Video compression is the field in electrical engineering and computer science that deals with representation of video data, for storage and/or transmission, for both analog and digital video. Video coding is often considered to be only for natural video, it can also be applied to synthetic (computer generated) video, i.e. graphics. Many representations take advantage of features of the Human Visual System to achieve an efficient representation. The biggest challenge is to reduce the size of the video data using video compression. For this reason the terms ''video coding'' and often used interchangeably by those who don't know the difference. The search for efficient video compression techniques dominated much of the research activity for video coding since the early 1980s, the major milestone was H.261, from which JPEG adopted the idea of using the DCT; since the many other advancements have been made to algorithms such as motion estimation. Since approximately 2000 the focus has been more on Meta data and video search, resulting in MPEG-7 and MPEG-21.

Video Compression

The main problem with the uncompressed (raw) video is it contain immense amount of data and hence communication and storage capabilities are limited and are expensive. For example, if we consider a HDTV video signal with 720×2180 pixels/frame with progressive scanning at 60 frames/sec, then the transmitter must be able to send

$$(\frac{720 \times 1280\, pixels}{frame})(\frac{60\, frames}{\sec})(\frac{3 colours}{pixels})(\frac{8 bits}{colour}) = 1.3 Gb/s \text{ ---------- (1)}$$

But the available HDTV channel bandwidth is around 20 Mb/s [1], i.e., it require compression by a factor of 70. A DVD (Digital Versatile Disk) can only store a few seconds of raw video at

Frame rate and television quality resolution.

Achieving Compression

Video compression can be achieved by exploiting the similarities or redundancies and irrelevancy that exist in a typical video signal. The redundancy in a video signal is based on two principles. The first is the spatial redundancy that exists in each frame. The second is the difference between the corresponding frames. This is called temporal redundancy [2]. This temporal redundancy can be eliminated by using motion estimation and compensation procedure. The remaining goals of video compression are to reduce the irrelevancy in the video signal is relatively straight forward. The identification and reduction redundancy in video signal is the straightforward, in this what is perceptually relevant and what is not is very difficult and there. This operation can be done by using appropriate models of the Human Vision System.

In video successive frames may contain the same objects (still or moving). In inter frame coding motion estimation and compensation has become powerful techniques to eliminate the temporal redundancy due high correlation between consecutive frames. In video scenes, motion can be complex combination of translation of rotation. Such motion is difficult to estimate and may require large amount of processing. The translational motion is easily estimated and has been used successfully for motion compensated coding.

Chapter 1

Different search algorithms are used to estimate motion between frames. When motion estimation is performed by an MPEG-2 encoder it groups pixels into 16×16 macro blocks. MPEG-4 AVC encoders can divide these macro blocks into partitions as well as 4×4, and even of variable size within the same Macro block. Partitions allow for more accuracy in motion estimation because areas with high motion can be isolated from those with less movement.

1.2 Literature Review

Most of the researchers have proposed many algorithms for Motion Estimation. Generally, the motion estimation search types are divided into two types these are

1). Pixel based motion estimation. 2).Block based motion estimation

The pixel based motion estimation approach seeks to determine motion vector for every pixel in the image. This is also referred to as the optical flow method, which works on the fundamental assumption of brightness constancy that is the intensity of a pixel remains constant, when it is displaced. However, no unique match for a pixel in the reference frame is found in the direction normal to the intensity gradient [3]. It is for this reason that an additional constraint is also introduced in terms of the smoothness displacement vector in the neighborhood. The smoothness constraint makes the algorithm interactive and requires excessively large computation time, making it unsuitable for practical and real time implementation.

An alternate and faster approach is the block based motion estimation. In this method, the candidate frame is divided into non overlapping blocks (of size 16 × 16, 8 × 8 or even 4 × 4 pixels in the recent standards) and for each such candidate block, the best motion vector is determined in the reference frame. Here, a single motion vector is computed for the entire block, whereby we make an inherent assumption that the entire block undergoes translational

Motion. This assumption is reasonably valid, except for the object boundaries and smaller block size leads to better motion compensation and motion estimation.

Block based motion estimation is accepted in all the video coding standards proposed till date. It is easy to implement in hardware and real time motion estimation and prediction is possible.

Many studies in literature use different block matching motion estimation algorithms in these algorithms full search gives minimum error when compared to the all block matching algorithms this is the basic search but it takes maximum computational complexity. So that in literature survey we find so many block matching algorithms in this block matching motion estimation we have to remember two main things the first one is the type of search pattern this one is the most important one because when the object moving we have to follow the certain pattern then only we can get the minim number of search points for block matching [4].The second one is meaning absolute difference so when the search pattern is very close to objects then we can get the minimum error.

The block size is the one of the important parameter in block matching algorithm. If the block size is smaller, it achieves better prediction quality. This is due to a number of reasons. A smaller block size reduces the effect of the accuracy problem. In other words, with a smaller block size, there is less possibility that the block will contain different objects moving in different directions.

In literature review I observed different searching algorithms in all these if the error minimizes the number of search points increasing If search points decreases the error increases so that by observing these algorithms proposed one fast block matching algorithm this given best search accuracy and minim error. Finally the performance of the proposed algorithm is evaluated in terms of completeness and correctness.

1.3 Scope of this Project

The main theme of this thesis is to study different implementation methods for motion estimation in the latest and future video coding standards. The reference coding standard in this thesis is H.264/AVC or MPEG-4 part 10. All the work stated in this thesis assumes the definition given in H.264/AVC standard. For this we are doing temporal redundancy for this different block matching algorithms are proposed in all these the number of search points and minimum motion vector are main important factors in existed algorithms poor performance giving because of search points increased and not getting the closer performance.

To improve this performance we proposed one based on those algorithms, for searching best motion vector it will takes the less number of search points when compared to existing algorithms to get this advantage we can followed the certain known pattern that is called three step diamond search algorithm this algorithm simulation results are compared with the existing algorithms it given a better results and the video coding standards also preferring the block matching process only because of simplicity so now a day's most of video coding applications using block based matching only, because of pixel based search pattern takes maximum search window size. If the window size is large the computational complexity is also high that is why the block based search pattern using.

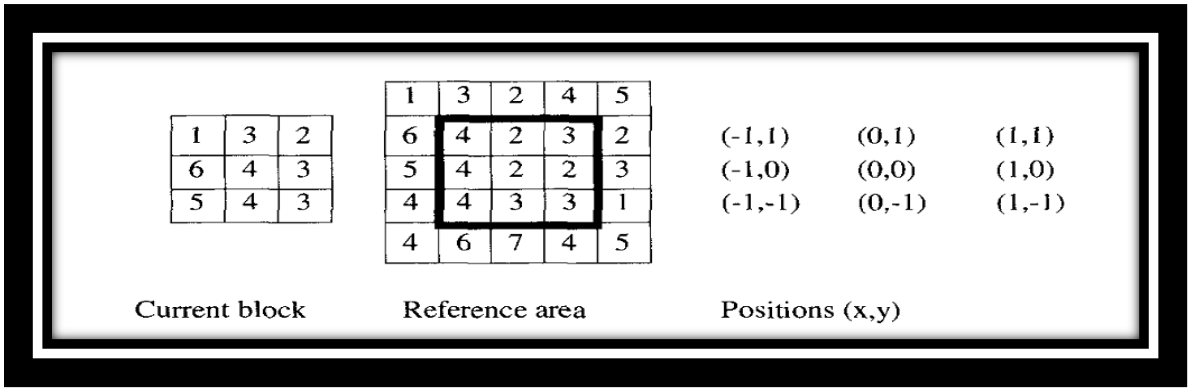
Chapter 1

1.4 Introduction to block matching Algorithms

The motion estimation is the most demanding in among all the blocks in a video coder [8]. It is also critical part that affects the video quality and compression efficiency. Till now many algorithms and architectures have been proposed to optimize this process. With advancement of video codec standards, the requirements of motion estimation have increased and thus both the software and hardware optimization must be continuously improved to cope with the increased complexity.

Block Matching

In all video coding standards motion compensation and estimation carried out on 8x8 or 16X16 blocks in the current frame. Motion Estimation of complete blocks is known as block matching. Each block of luminance samples in the current frame, the motion estimation algorithms searches a neighboring area of the reference frame 16x16 areas the best match is the one with minimises the energy of the difference between the current 16X16 block and the matching 16x16 area. The area in which the search is carried out may be centred around the position of the current 16X16 block because.



1.5 Thesis Outline

The Outline of this thesis is as follows.

Chapter 2 Focused on the fundamentals of video compression and requirements of block matching.

Chapter 3 Focused on fundamental concepts on motion estimation and explains about forward motion estimation and back ward motion estimation. Explains the role of motion estimation and the under lying algorithms.

Chapter 4 Describes the existed different block matching algorithms for motion estimation and draw backs of that algorithms number of search point's error metrics.

Chapter 5 Describes the proposed three step diamond search algorithm for fast block matching motion estimation algorithms number of search points of view and error metrics points of view got a good result when compared to existing techniques. It gives the better performance when compared to the other existing algorithms.

Chapter 6 gives the conclusion and future scope for work.

Chapter 2

2. Fundamentals of Video Compression

2.1 Introduction

Digital video coding has gradually increased in importance since the 90s when MPEG-1 first emerged. Is has had large impact on video delivery, storage and presentation compared to analogue video. This eliminates the need of high band width as required in analogue video delivery. With this important characteristic, many application areas have emerged. For example, set top box video playback using compact disk, video conferencing over IP networks, P2P video delivery, mobile TV broadcasting etc.

A major problem in a video is the high requirement for bandwidth. A typical system needs to send dozens of individual frames over second to create an illusion of a moving picture. For this reason, several standards to create an illusion of a moving picture. For this reason, several standards for compression of the video have been developed. Each individual frame is coded so that redundancy is removed. Furthermore, between consecutive frames, a great deal of redundancy is removed with a motion compensation system. Representing video material in a digital form requires a large number of bits. The volume of data generated by digitizing a video signal is too large for most storage and transmission systems (despite the continual increase in storage capacity and transmission 'bandwidth'). This means the compression is essential for most digital video applications.

A digital format for video is described by ITU-R 601 standard this is roughly equivalent to analogue television, in terms of spatial resolution and frame rate. One channel of ITU-R 601 television, broadcast in uncompressed digital form, requires a transmission bit rate of 216Mbps. At this bit rate, a 4.7 Gbyte DVD could store just87 seconds of uncompressed video.

Chapter 2

Table 2.1.1 shows the uncompressed bit rates of several popular video formats. From this table it can be seen that even QCIF at 15 frames per second (i.e. relatively low quality video, suitable for video telephony) requires 4.6Mbps for transmission or storage. Table2.1.2 lists typical capacities of popular storage media and transmission networks.

Format	Luminance Resolution	Chrominance Resolution	Frames per Second	Bits per second (uncompressed)
ITU-R 601	858×525	429×525	30	216 Mbps
CIF	352×288	176×144	30	36.5 Mbps
QCIF	176×144	88×72	15	4.6 Mbps

Table 2.1.1 Un-compressed bit rates

Media/Network	Capacity
Ethernet LAN (10 Mbps)	Max. 10 Mbps/ Typical 1-2 Mbps
ADSL	Typical 1-2 Mbps (downstream)
ISDN-2	128 KBPS
V.90 MODEM	56 kbps downstream/33 kbps upstream
DVD-5	4.7 Gbytes
CD-ROM	640 Mbytes

Table 2.1.2 typical transmission/storage capacities

2.2 Need of Compression

The following statement (or something similar) has been made many times over the 20-year history of image and video compression: 'Video compression will become redundant very soon, once transmission and storage capacities have increased to a sufficient level to cope with uncompressed video.' It is true that both storage and transmission capacities

Chapter 2

Continue to increase. However, an efficient and well-designed video compression system gives very significant performance advantages for visual communications at both low and high transmission bandwidths. At low bandwidths, compression enables applications that would not otherwise be possible, such as basic-quality video telephony over a standard telephone connection. At high bandwidths, compression can support a much higher visual quality. For example, a 4.7 Gbyte DVD can store approximately 2 hours of uncompressed QCIF video (at 15 frames per second) or 2 hours of compressed ITU-R 601 video (at 30 frames per second). Most users would prefer to see 'television-quality' video with smooth motion rather than 'postage-stamp' video with jerky motion. Video CODEC s will therefore remain a important part of the emerging multimedia industry for the foreseeable future, allowing designers to make the most efficient use of available transmission or storage capacity. In this chapter we introduce the basic components of an image or video compression system. We then describe the main functional blocks of an image encoded decoder (CODEC) and a video CODEC.

2.3 Video Standards

Both terminals in the figure 2.3.1 need to use a video decoder that is capable of decoding the video stream produced by the other terminal. Since there are endless ways to compress and encode data, and many terminal vendors which each may have a unique idea of data compression, common standards are needed, that rigidly define how the video is coded in the transmission channel.

There are mainly two standard series in series in common use, both having several versions. International Telecommunications Union (ITU) started developing Recommendation H.26 lin1984, and the effort was finished in 1990 when it was approved. The standard is aimed

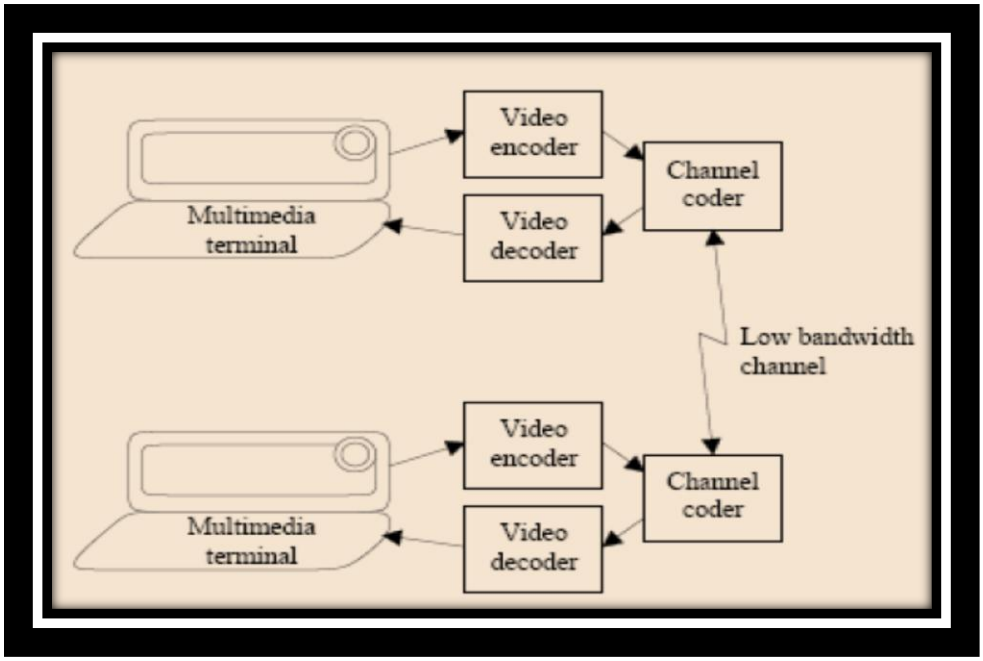


Figure 2.3.1 Wireless Video Conferencing applications

MPEG-1 is a video compression standard developed in joint operation by international standards Organization (ISO) and International Electro- Technical Commission (IEC) .The system development was started in 1988 and finished in 1990, and it was accepted as standard in 1992. MPEG-1 can be used at higher bit rates than H.261, at about 1.5 megabits per second, which is suitable for storing the compressed video stream on compact disks or for using with interactive multimedia systems [6]. The standard covers also audio associated with a video.

For motion estimation, MPEG-1 uses the same block size as H.261, 16 × 16 pixels, but in addition to backward compensation, MPEG can also apply bidirectional motion compression.

Chapter 2

A revised standard, MPEG-2, was approved in 1994. Its target is at higher bit rate than MPEG-1, from 2 to 30 megabits per second, where applications may be digital television or video services through a fast computer network. The latest ISO/IEC video coding standards MPEG-4, which was approved in the beginning of 1999. It is targeted at very low bit rates (832 kbps) suitable for e.g. mobile video phones. MPEG-4 can be also used with higher bit rates, up to 4 mbps.

2.4 Fundamentals of Lossy Video Compression

The video encoding system consists of two kinds of compression units, namely lossy and lossless In digital video, lossy compression is often employed to ensure good compression performance. Although the quality is inevitably degraded, there is minimal impact on perceived quality since only the high-frequency component is eliminated and human eyes are not sensitive to these components. Lossless data compression is a class of data compression algorithms in which the exact original data can be reconstructed. In contrast, lossy data compression must introduce some data loss during compression.

The lossy compression unit contains temporal and spatial redundancy compressors. The statistical compressor in the last stage is a lossless one. The video coder contains five stages in total [7], handling different kinds of compression. In the following subsection each compression unit is discussed. Figure 2.2 shows the block diagram of a hybrid video encoder with the three compressor types described [8].

2.5 Representation of Colors

We need at least three basic colors: red, blue and green, to display a true color picture. As a result, each pixel consists of at least three distinct channels of information. Common color representations methods include RGB and YUV [7]. RGB is a representation that includes three basic colors together with brightness information. It is a common method for monitor displays. YUV is a representation which divides the color space into luminance (brightness) and

Chrominance (color). The 3 original colors can be derived from YUV data. Since humans are less sensitive to color than luminance, color information can be suppressed compared to luminance data without significant quality loss. For example, in4:2:0 YUV, we have four times the luminance information than chrominance. Other representations such as 4:2:2 and 4:4:4depend on the sampling frequencies. In the video compression, YUV representation is a better method to specify a pixel since it more closely tracks human perception and can enhance compression efficiency.

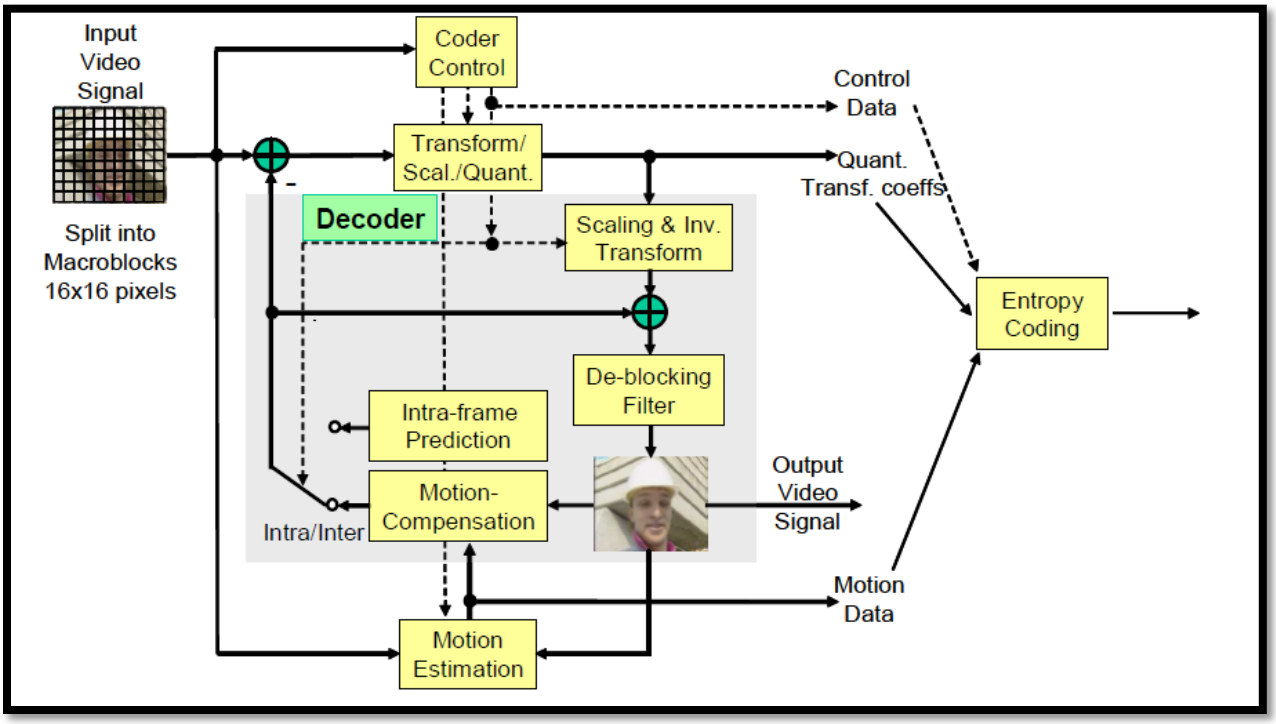


Figure 2.5.1 Hybrid Video Coder H.264/ AVC

Chapter 2

2.6 Video Compression Methods

Since a video scene is rectangular, block-based coding is a suitable choice of processing element. However, other compression methods also exist. One example is wavelet coding which consist in image encoding in JPEG2000 [8]. In three dimensional version of wavelet coding for video compression was suggested [9] which gives better visual quality but the computation complexity is much higher. Another method is called arbitrary shaped coding which is employed in MPEG-4 [11]. This is based on different moving objects whose motion is combined to form a complete frame. The performance of this one improved by accurate prediction in motion estimation. As a tradeoff, it has increased complexity.

Chapter 3

3. Fundamentals of Motion Estimation

3.1 Motion Estimation

A video sequence can be considered to be a discredited three dimensional projection of the real four-dimensional continuous space time. The objects in the real world may move, rotate, or deform. The movements cannot be observed directly, but instead the light reflected from the object surfaces and projected onto an image. The light can be moving, and the reflected back light varies depending on the angle between a surface and alight source. There may be objects occluding the light rays and casting shadows. The objects may be transparent (so that several independent motions could be observed at the same location of an image) or there might be fog, rain or snow blurring the observed image. The discretization causes noise into the video sequence, from which the video encoder makes its motion estimations. There may also be noises in the image capture device (such as a video camera) or in the electrical transmission lines. A perfect motion model would take all the factors into account and find the motion that has the maximum likelihood from the observed video sequence. The current frame and reference frame difference can observe in the figure 3.1.1diagram.

Changes between frames are mainly due to the movement of objects. Using a model of the motion of objects between frames, the encoder estimates the motion that occurred between the reference frame and the current frame. This process is called motion estimation (ME) [12]. The encoder then uses this motion model and information to move the contents of the reference frame to provide a better prediction of the current frame. This process is known as motion compensation (MC), and the prediction so produced is called the motion-compensated prediction (MCP) or the displaced-frame (DF) [13]. In this case, the coded prediction error signal is called the displaced-frame difference (DFD).A block diagram of a motion compensated coding system is illustrated in Figure 3.1.2 This is the most commonly used inter frame coding method.

Chapter 3

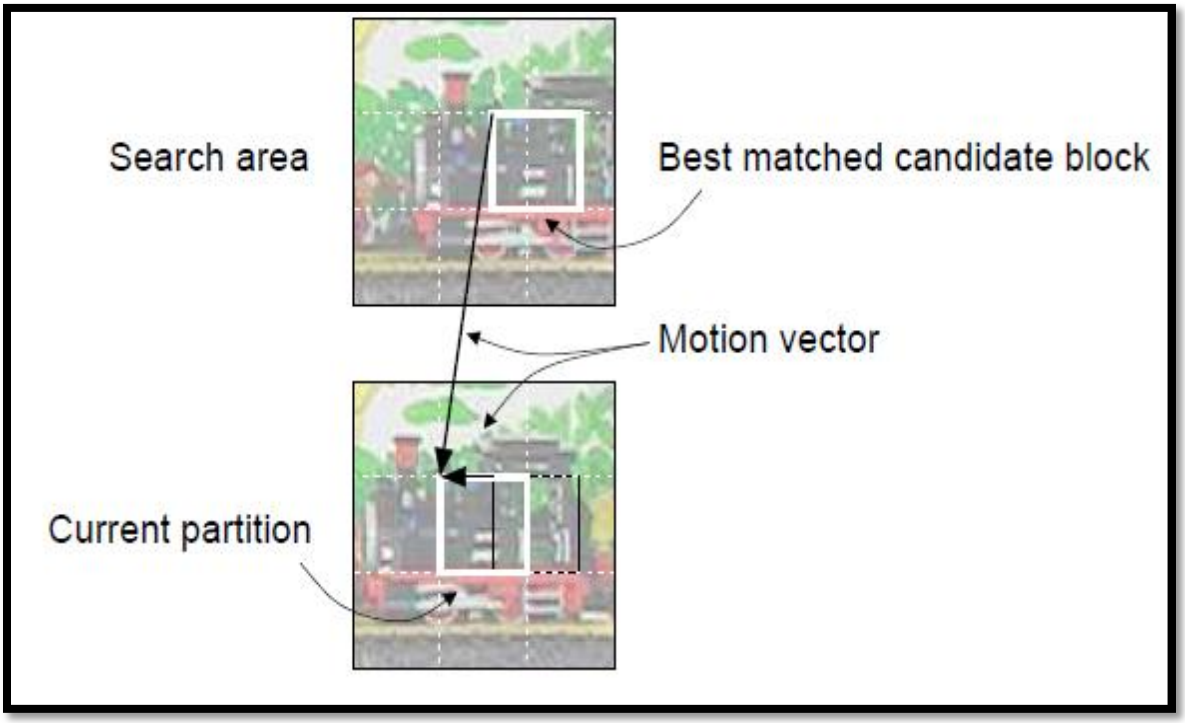


Figure 3.1.1 motion estimation detector

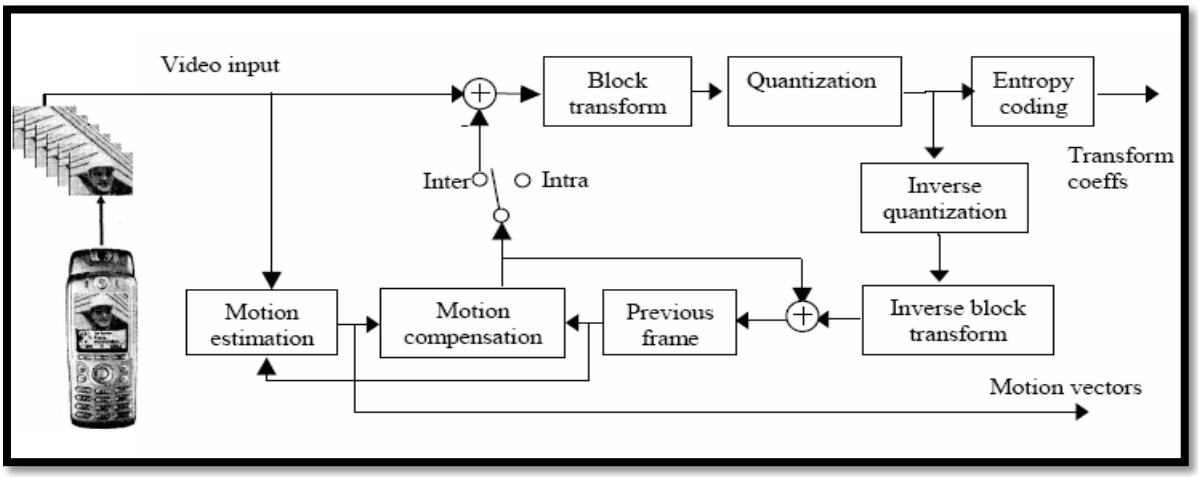


Figure 3.1.2 Motion compensated video coding

The reference frame employed for ME can occur temporally before or after the current frame. The two cases are known as forward prediction and backward prediction, respectively. The prediction can be observed in figure 3.1.3. In bidirectional prediction, however, two reference frames (one each for forward and backward prediction) are employed and the two predictions are interpolated (the resulting predicted frame is called B-frame). The most commonly used ME method is the block matching motion estimation (BMME) algorithm.

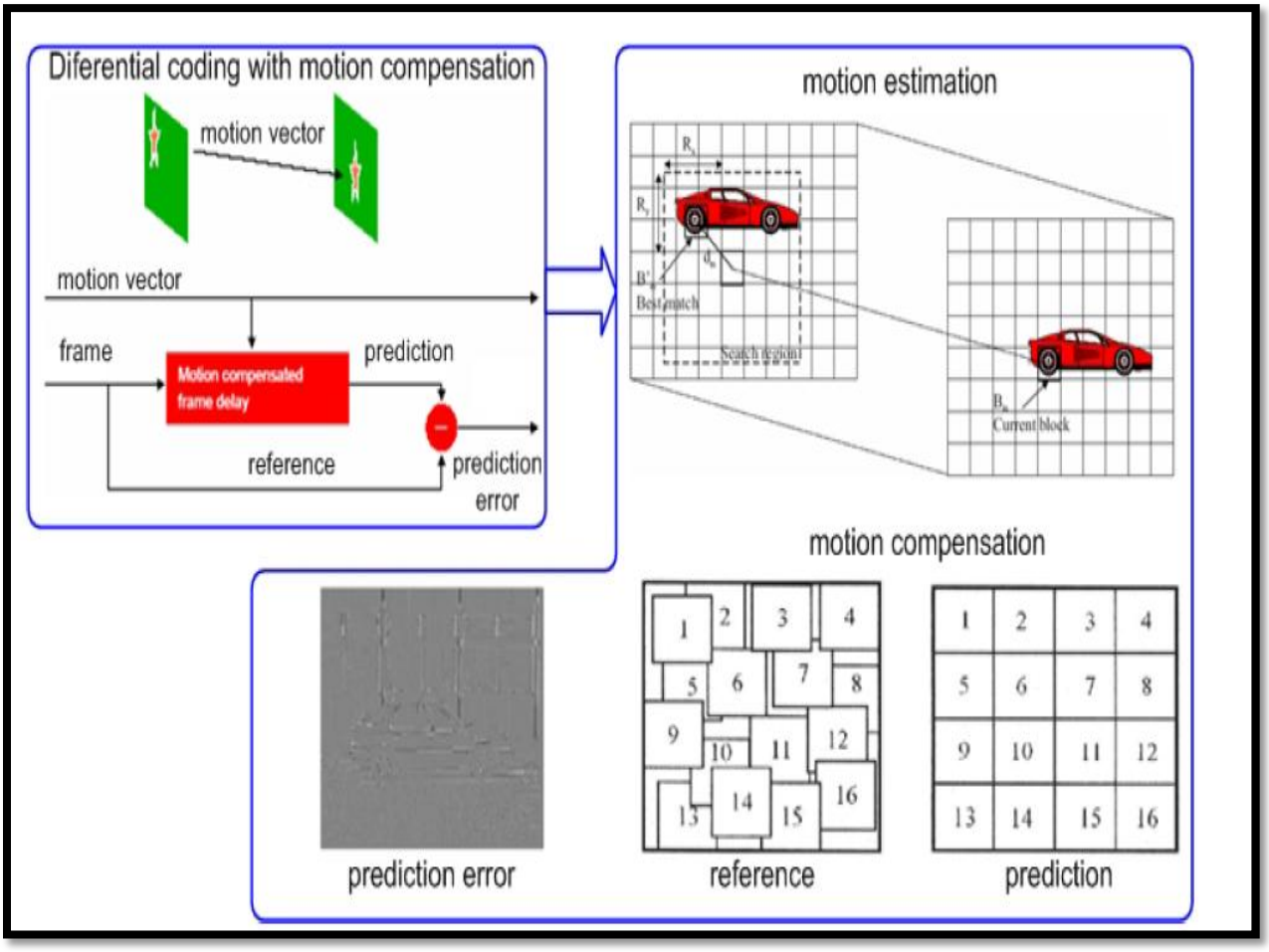


Figure 3.1.3 Predictive sources coding with motion compensation

3.2 Motion Estimation Procedure

After completion of motion estimation the residue of picture and motion vectors are predicted. This procedure is executed for each block (16x16, 8x8 or 4x4) in the current frame.

1. For the reference frame, a search area is defined for each block in the current frame. The search area is typically sized at 2 to 3 times the macro block size (16x16). Using the fact that the motion between consecutive frames is statistically small, the search range is confined to this area. After the search process, a 'best' match will be found within the area. The 'best' matching usually means having lowest energy in the sum of residual formed by subtracting the candidate block in search region from the current block located in current frame. The process of finding best match block by block is called block-based motion estimation.
2. After finding the best match, the motion vectors and residues between the current block and reference block are computed. The process of getting the residues and motion vectors is known as motion compensation.
3. The residues and motion vectors of best match are encoded by the transform unit and entropy unit and transmitted to the decoder side.
4. At decoder side, the process is reversed to reconstruct the original picture.

Figure 3.5 shows an illustration of the above procedure. In modern video coding standards, the reference frame can be a previous frame, a future frame or a combination of two or more Previously coded frames [14]. The number of reference frames needed depends on the required accuracy. The more reference frames referenced by current block, the more accurate the prediction is.

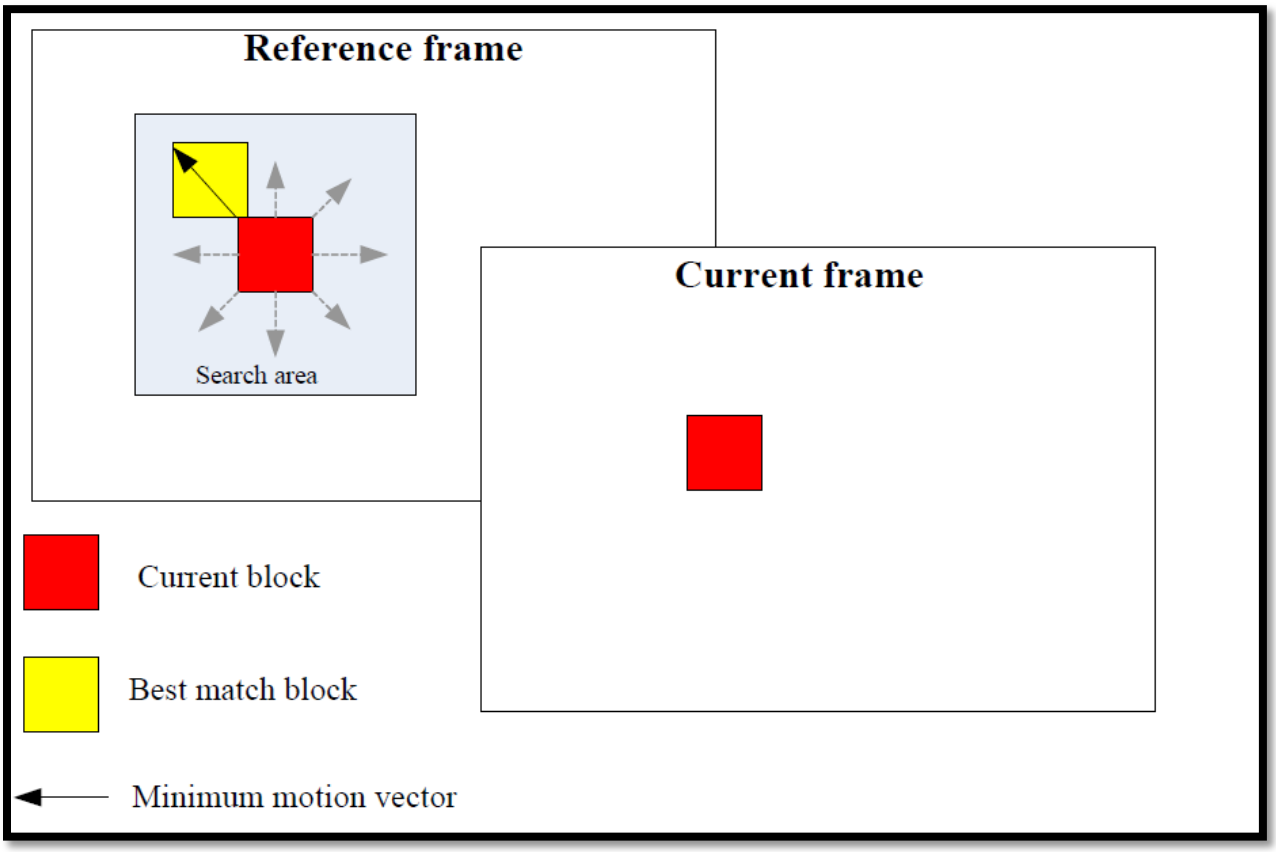


Figure 3.2.1 Motion Estimation and Motion Vector

3.3 Motion Vectors

To find the motion of each block, a motion vector is defined as the relative displacement between the current candidate block and the best matching block within the search window in the reference frame. It is a directional pair representing the displacement in horizontal (x-axis) direction and vertical (y-axis direction). The maximum value of motion vector is determined by the search range. If the search range is more, the more bits needed to code the motion vector. Designers need to make tradeoffs between these two conflicting parameters. The motion vector is illustrated in figure 3.2.1

Each macro block in the frame motion vector is produced. MPEG-1 and MPEG-2 employ this property. The introduction of variable block size motion estimation in MPEG-4 and H.264/AVC, one macro block can produce more than one motion vector due to the existence of different kinds of sub blocks. In H.264, 41 motion vectors should be produced [15] in one macro block and they are passed to rate distortion optimization to choose the best combination. This is known as mode selection.

3.4 Prediction of Video codec

A video signal consists of a sequence of individual frames. Each frame may be compressed individually using an image CODEC as described above: this is described as intra-frame coding, where each frame is 'intra' coded without any reference to other frames. However, better compression performance may be achieved by exploiting the temporal redundancy in a video sequence (the similarities between successive video frames). This may be achieved by adding a 'front end' to the image CODEC, with two main functions:

These are (1) The Prediction (2) Compensation

Prediction: The prediction of the current frame find based on the one or more previously transmitted frames.

Compensation: The prediction is subtracted from the current frame to produce a 'residual frame'.

The residual frame is then processed using an 'image CODEC'. The key to this approach is the prediction function: if the prediction is accurate, the residual frame will contain little data and will hence be compressed to a very small size by the image CODEC. In order to decode the frame, the decoder must 'reverse' the compensation process, adding the prediction to the decoded residual frame (reconstruction) (Figure 3.15)[16]. This is inter frame coding: frames are coded based on some relationship with other video frames, i.e. coding exploits the interdependencies of video frames.

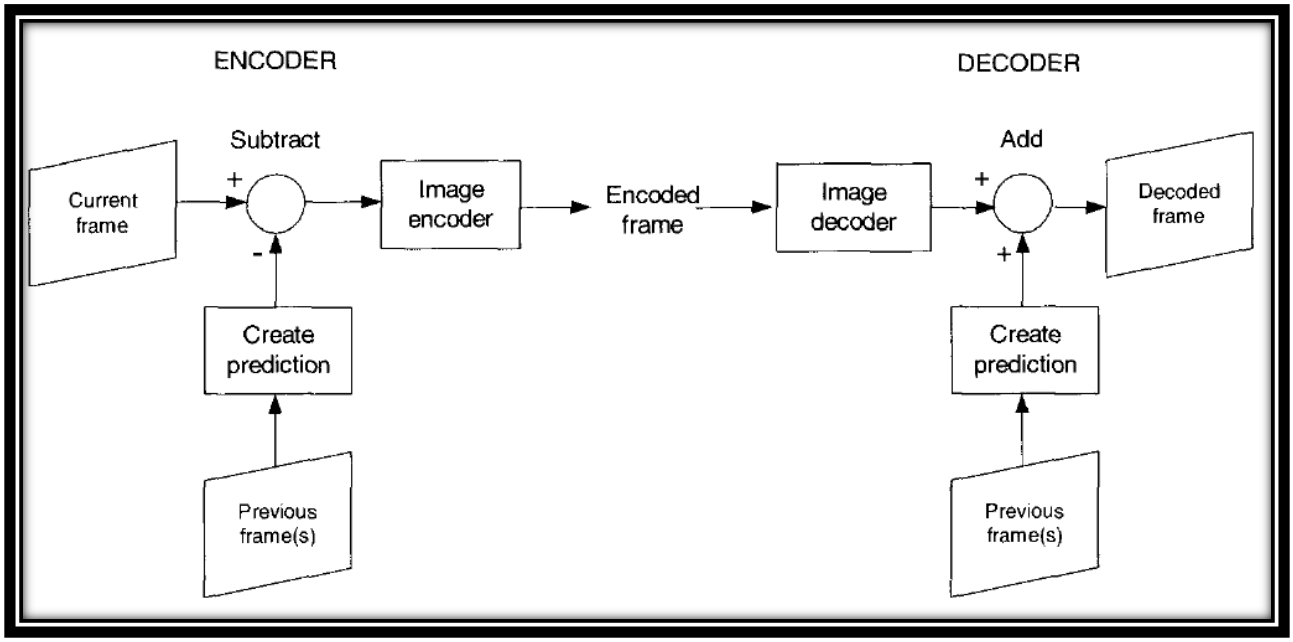


Figure 3.4.1 Video CODEC with prediction

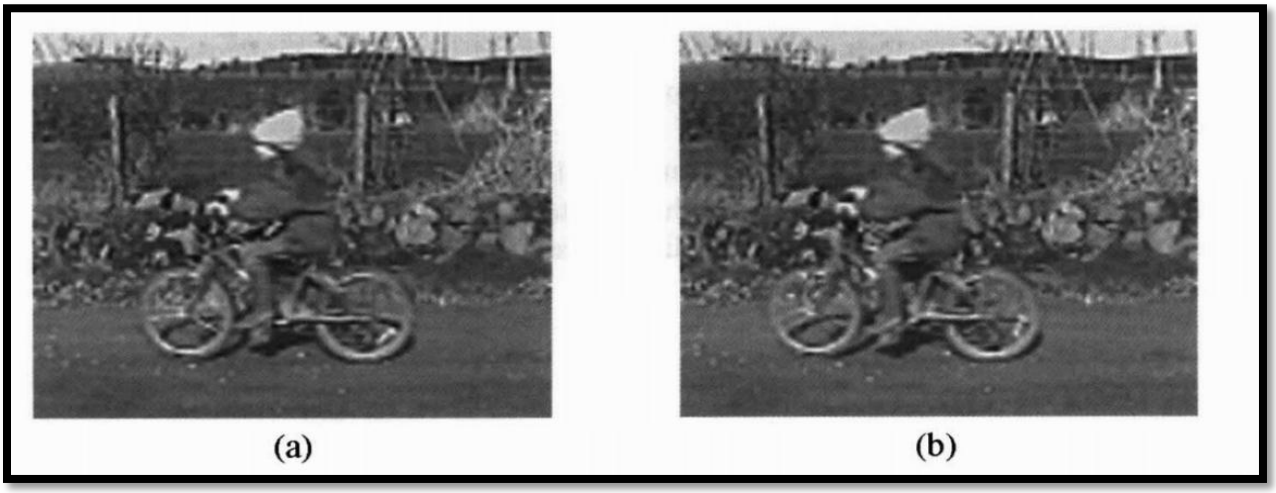


Figure 3.4.1(a) Current Frame

Figure 3.4.1(b) Previous Frame

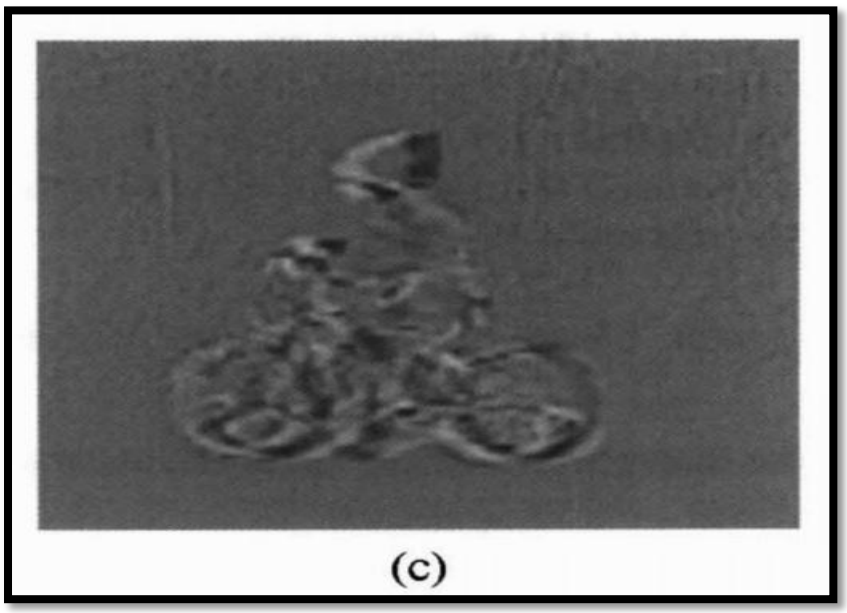


Figure 3.4.1(c) Residual frame

3.5 Frame differencing

The simplest predictor is just the previous transmitted frame. The above figure shows the residual frame produced by subtracting the previous frame from the current frame in a video sequence. Mid-grey areas of the residual frame contain zero data: light and dark areas indicate positive and negative residual data respectively [17]. It is clear that much of the residual data is zero: hence, compression efficiency can be improved by compressing the residual frame rather than the current frame.

Encoder Input	Encoder Prediction	Encoder output Decoder input	Decoder Prediction	Decoder Output
Original frame	Zero	Compressed Frame 1	Zero	Decoded frame 1
Original frame	Original frame 1	Compressed Residual frame 2	Decoded frame 1	Decoded frame 2
Original frame	Original frame 2	Compressed Residual frame 2	Decoded frame 2	Decoded frame 3

Table 3.5.1 Prediction drift

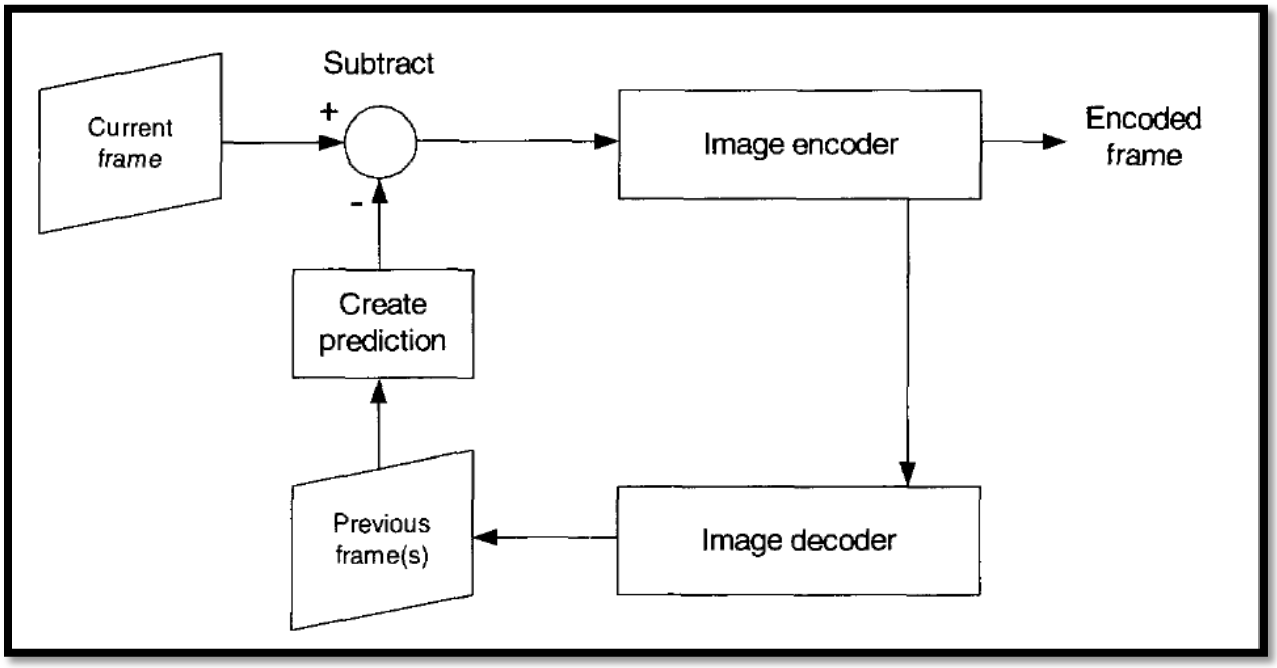


Figure 3.5.2 Encoder with Decoding Loop

The decoder faces a potential problem that can be illustrated as follows. Table 3.1 shows the sequence of operations required to encode and decode a series of video frames using frame differencing. For the first frame the encoder and decoder use no prediction. The problem starts with frame 2: the encoder uses the original frame 1 as a prediction and encodes the resulting residual. However, the decoder only has the decoded frame 1 available to form the prediction. Because the coding process is lossy, there is a difference between the decoded and original frame 1 which leads to a small error in the prediction of frame 2 at the decoder. This error will build up with each successive frame and the encoder and decoder predictors will rapidly 'drift' apart, leading to a significant drop in decoded quality. The solution to this problem is for the encoder to use a decoded frame to form the prediction. Hence the encoder in the above example decodes (or reconstructs) frame 1 to form a prediction for frame 2. The encoder and decoder use the same prediction and drift should be reduced or removed. Figure 3.7 shows the complete encoder which now includes a decoding 'loop' in order to reconstruct its prediction reference. The reconstructed (or 'reference') frame is stored in the encoder and in the decoder to form the prediction for the next coded frame.

3.6 Motion Compensated Prediction

Frame differencing gives better compression performance than the intra frame coding when successive frames are very similar, but does not perform well when there is a significant change between the previous and current frames. Such changes are usually due to movement in the video scene and a significantly better prediction can be achieved by estimating this movement and compensating for it.

The below figure 3.6.1 shows a video CODEC that uses motion-compensated prediction. Two new steps are required in the encoder:

(1) Motion estimation: a region of the current frame (often a rectangular block of luminance samples) is compared with neighbouring regions of the previous reconstructed frame.

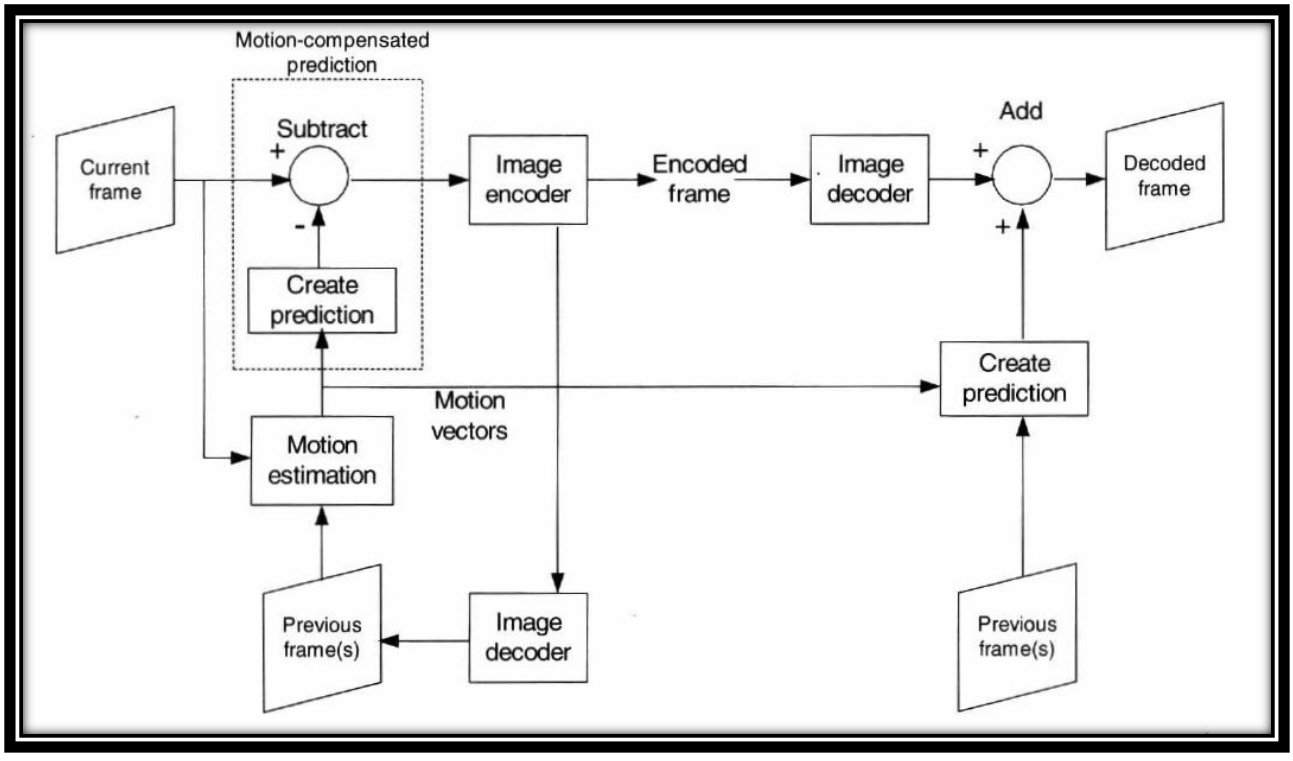


Figure 3.6.1 Video CODEC with Motion Estimation and Compensation

Motion estimator attempts to find the 'best match', i.e. the neighboring block in the reference frame that gives the smallest residual block.

Motion compensation: The matching region or block from the reference frame identified by

The motion estimator) is subtracted from the current region or block.

Here the decoder carries out the same motion compensation to re construct the current frame. This operation means the encoder has to transmit the location of the 'best' matching blocks to the decoder (typically in the form of a set of motion vectors)[18]. The below figure shows a residual frame produced by subtracting a motion compensated version of the previous frame from the current frame. The residual frame clearly contains less data than the residual in Figure 3.6.2this improvement in compression does not come without a price: motion estimation can be very computationally intensive.

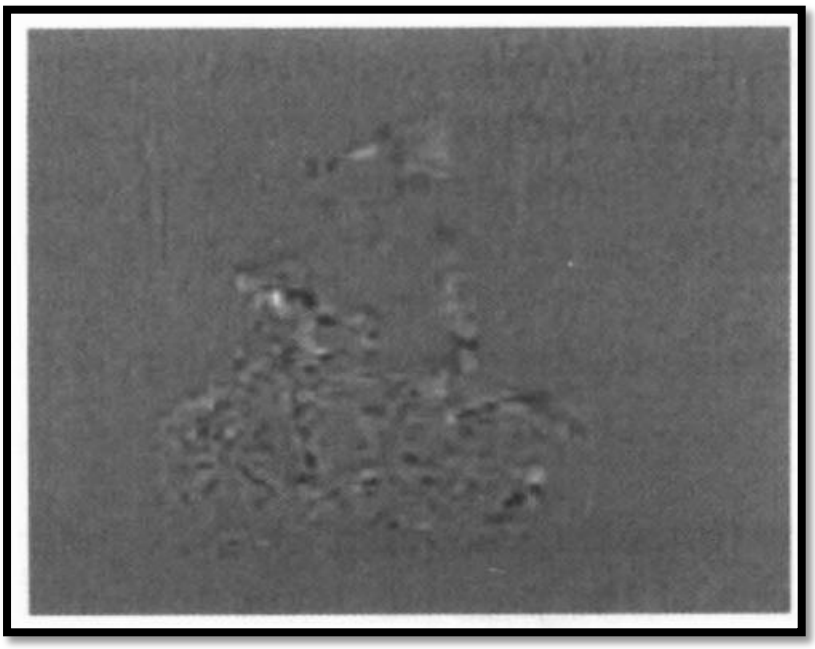


Figure 3.6.2 Residual frame (MAD)

Design of a motion estimation algorithm can have a dramatic effect on the compression performance and computational complexity of a video CODEC.

3.7 Block Matching Algorithm

Figure 3.10 illustrates a process of block-matching algorithm. In a typical block matching Algorithm, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. Each luminance block in the present frame is matched against candidate blocks in a search area on the reference frame. These candidate blocks are just the displaced versions of original block. The best candidate block is found and its displacement (motion vector) is recorded. In a typical inter frame coder; the input frame is subtracted from the prediction of the reference frame. Consequently the motion vector and the resulting error can be transmitted instead of the original luminance block; thus inter frame redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to their constructed reference frames.

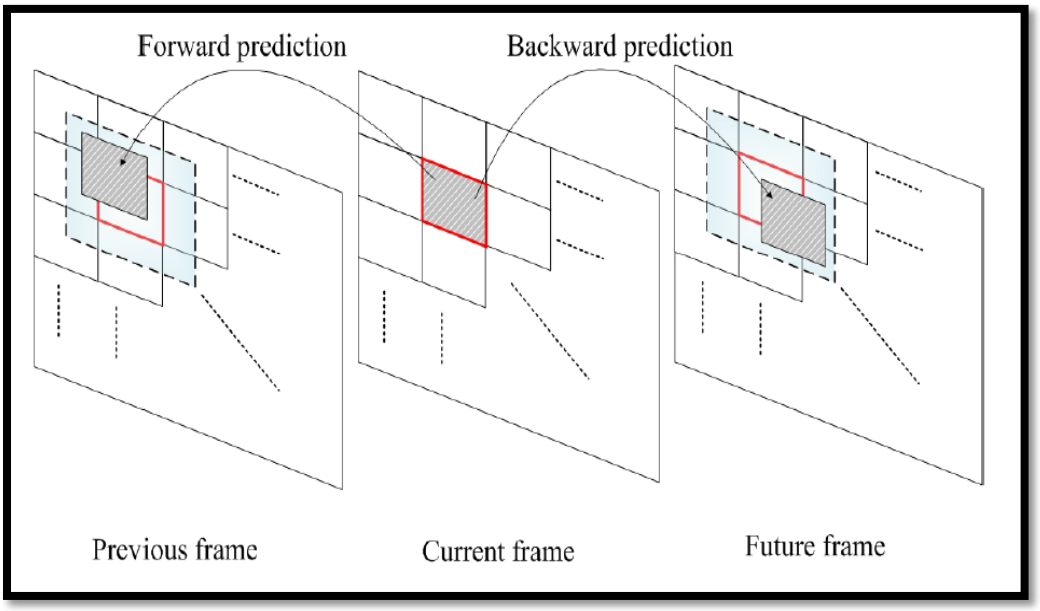


Figure 3.7.1 Illustration of Motion Estimation Process

This algorithm is based on a translational model of the motion of objects between frames. It also assumes that all pels within a block undergo the same translational movement. There are many other ME methods, but BMME is normally preferred due to its simplicity and good compromise between prediction quality and motion overhead. This assumption is not strictly valid, since we capture 3-D scenes through the camera and objects do have more degrees of freedom than just the translational one. However, the assumptions are still reasonable, considering the practical movements of the objects over one frame and this makes our computations much simpler.

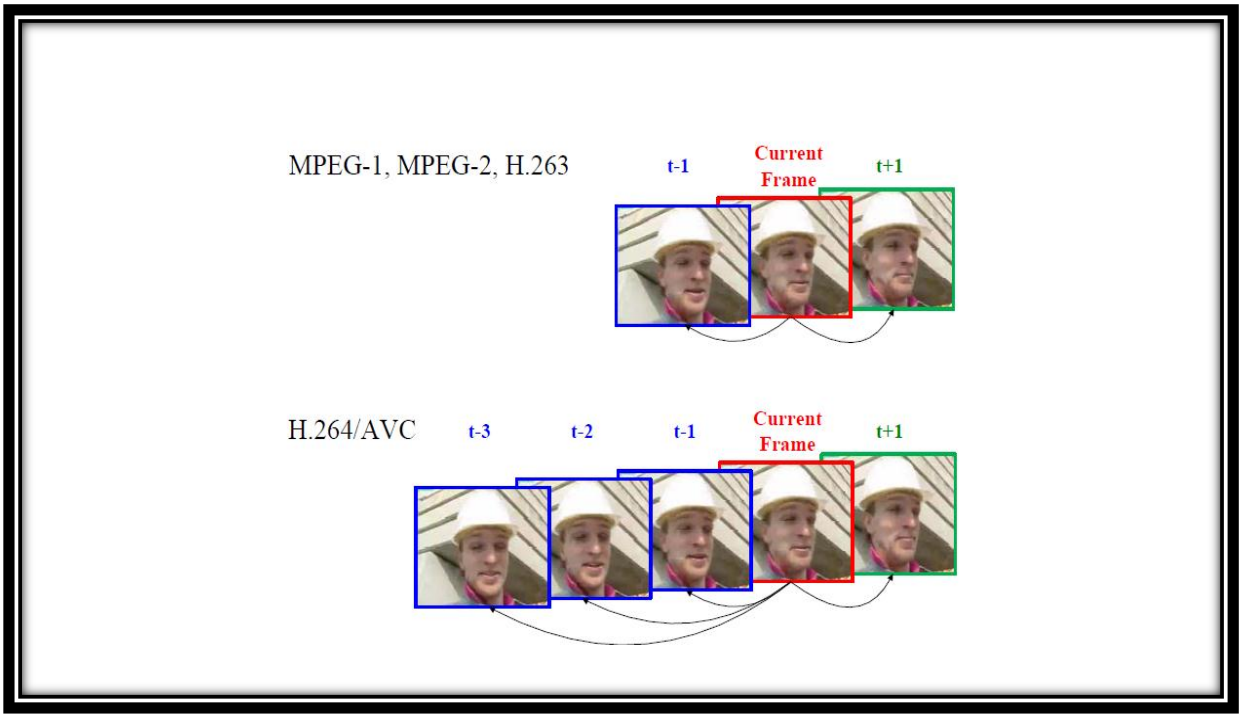


Figure 3.7.2Illustrations of Multiple Reference Frames

There are many other approaches to motion estimation, some using the frequency or wavelet domains, and designers have considered scope to invent new methods since this process does not need to be specified in coding standards. The standards need only specify how the motion vectors should be interpreted by the decoder. Block Matching (BM) is the most common

Method of motion estimation. Typically each macro block (16×16 pels) in the new frame is compared with shifted regions of the same size from the previous decoded frame, and the shift which results in the minimum error is selected as the best motion vector for that macro block. The motion compensated prediction frame is then formed from all the shifted regions from the previous decoded frame [19].

Backward Motion Estimation

The motion estimation generally considered as backward motion estimation, since the current frame is considered as the candidate frame and the reference frame on which the motion vectors are searched is a fast frame that is the search is back word. Back word motion estimation leads to forward motion prediction.

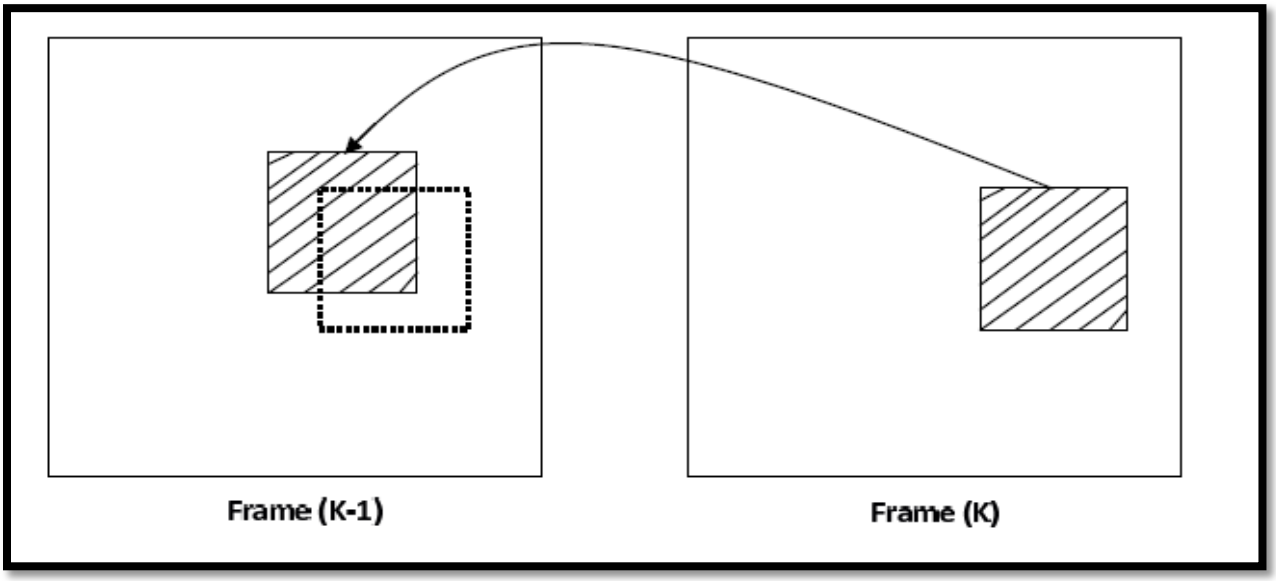


Figure 3.7.3 Back word motion estimation with current frame as k and frame (k-1) as the reference frame

Forward Motion Estimation

It is just the opposite of backward motion estimation. Here, the search for motion vectors is carried out on a frame that appears later than the candidates frame in temporal ordering. In other words, the search is “forward”. Forward motion estimation leads to backward motion prediction. It may appear that forward motion estimation is unusual, since

One requires future frames to predict the candidate frame. However, this is not unusual, since the candidate frame, for which the motion vector is being sought is not necessarily the current, that is the most recent frame. It is possible to store more than one frame and use one of the past frames as a candidate frame that uses another frame, appearing later in the temporal order as a reference.

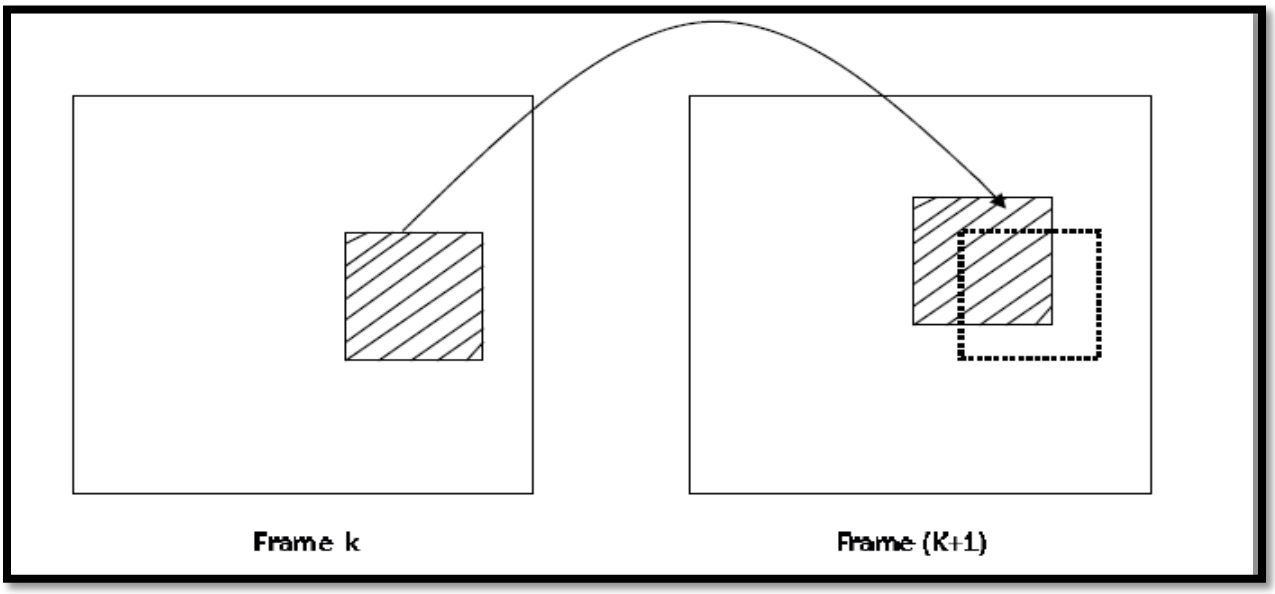


Figure 3.7.4 Forward motion estimation with current frame as k and frame (k+1) as the reference frame

3.8 Matching Criteria for Motion Estimation

Inter frame predictive coding is used to eliminate the large amount of temporal and spatial redundancy that exists in video sequences and helps in compressing them. In conventional predictive coding the difference between the current frame and the predicted frame is coded and transmitted. The better the prediction, the smaller the error and hence the transmission bit rate when there is motion in a sequence, then a pel on the same part of the moving object is a better prediction for the current pel. There are a number of criteria to evaluate the “goodness” of a match.

Chapter 3

Three popular matching criteria used for block-based motion estimation are

1. Mean Square Error (MSE)
2. Sum of Absolute Difference (SAD)
3. Matching Pel Count (MPC)

To implement the block motion estimation, the candidate video frame is partitioned into a set of non-overlapping blocks and the motion vector is to be determined for each such candidate block with respect to the reference. For each of these criteria, square block of size $N \times N$ pixels is considered. The intensity value of the pixel at coordinate (n_1, n_2) in the frame k is given by, $S(n_1, n_2, k)$ where $(0 \le n_1, n_2 \le N-1)$. the frame k is referred to as the candidate frame and the block of pixels defined is the candidates block.

1. Mean Square Error (MSE) Criterion

Considering (k-l) as the past references frame *l>0 for* backward motion estimation, the mean square error of a block of pixels computed at a displacement (I ,j) in the reference frame is given by

$$MSE(i, j) = \frac{1}{N^2} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [s(n_1, n_2, k) - s(n_1 + i, n_2 + j, k - l)]^2 \text{ ---------- (1)}$$

Consider a block of pixels of size $N \times N$ in the reference frame, at a displacement of, where I and j are integers with respect to the candidate block position. The MSE is computed for each displacement position (I, j), within a specified search range in the reference image and the displacement that gives the minimum value of MSE is the displacement vector which is more commonly known as motion vector and is given by

$$[d_1, d_2] = \arg\min[MSE(i, j)] \text{ ---------- (2)}$$

The *MSE* criterion defined in equation (1) requires computation of N^2 subtractions, N^2 multiplications (squaring) and (N^2-1) additions for each candidate block at each search position. This is computationally costly and a simpler matching criterion, as defined below is often preferred over the *MSE* criterion.

2. Sum of Absolute Difference (SAD) Criterion

Like the MSE criterion, the sum of absolute difference (SAD) too makes the error values as positive, but instead of summing up the squared differences, the absolute differences are summed up. The SAD measure at displacement (I,j) is defined as

$$SAD(i,j)=\frac{1}{N^2}\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1}[s(n_1,n_2,k)-s(n_1+i,n_2+j,k-l)] \text{ ---------- (3)}$$

The motion vector is determined in a manner similar to that for *MAD as*

$$[d_1,d_2]=\arg\ \min[\text{SAD(i,j)}] \text{ ---------- (4)}$$

The SAD criterion shown in equation (3) requires N^2 computations of subtractions with absolute values and additions N^2 for each candidate block at each search position. The absence of multiplications makes this criterion computationally more attractive and facilitates easier hardware implementation.

3. Matching Pixel Count (MPC) Criterion

In this criterion, the pixels of the candidates block B are compared with the corresponding pixels in the block with displacement (I, j), in the reference frame and those which are less than a specified threshold, i.e., closely matched are counted. The count for matching and the displacement (I, j), for which the count is maximum correspond to the motion vector. We define a binary valued function $count(n_1,n_2)\forall(n_1,n_2)\in B$ as

$$count(n_1,n_2)=\begin{cases}1 & \text{if } |s(n_1,n_2,k)-s(n_1+i,n_2+j,k-l)\le\theta \\ 0 & \text{otherwise}\end{cases} \text{ ---------- (5)}$$

Where, θ is a pre-determined threshold? The matching pel count (MPC) at displacement (i,j) is defined as the accumulated value of matched pixels as given by

$$MPC(i,j) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} [count(n_1, n_2)] \text{ ---------- (6)}$$

$$[d_1, d_2] = \arg\max[MPC(i,j)] \text{ ---------- (7)}$$

Block Size

Another important parameter of the BMA is the block size. If the block size is smaller, it achieves better prediction quality. This is due to a number of reasons. A smaller block size reduces the effect of the accuracy problem. In other words, with a smaller block size, there is less possibility that the block will contain different objects moving in different directions. In addition, a smaller block size provides a better piecewise translational approximation to non-translational motion. Since a smaller block size means that there are more blocks (and consequently more motion vectors) per frame, this improved prediction quality comes at the expense of a larger motion overhead. Most video coding standards use a block size of 16×16 and 8×8 on a Macro Block (MB) basis.

Motion compensation for each 16×16 macro block can be performed using a number of different block sizes and shapes. The original luminance component of each macro block (16×16) may be split into 4 kinds of size:$16\times16, 16\times8, 8\times16,\ 8\times8$, as shown in figure (a). Each of the sub-divided regions is a block partition. If the 8×8mode is chosen, each of the four8×8, $8\times4, 4\times8, 4\times4$ as shown in figure (b). These partitions and sub-partitions compose to a large number of possible combinations:

Macro block types

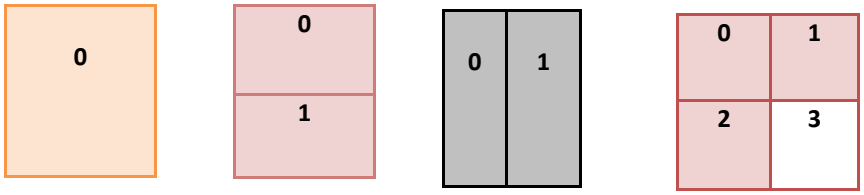


Figure (a) 16×16 16×8 8×16 8×8

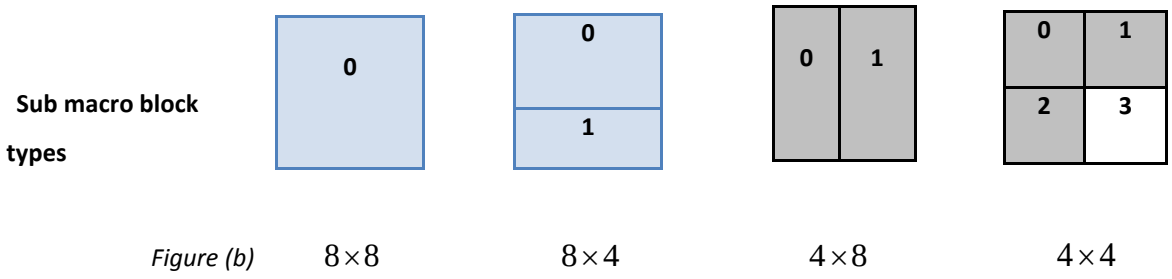


Figure (a) and (b) Macro block Partition and sub portion

A separate motion vector is required for each partition or sub-partition. Each of the motion vectors must be coded and transmitted, besides, the choice of partition must be encoded in the bit stream. If we use smaller partition size (e.g. 8×8, 4×4 etc.), we can get smaller prediction error, but we must consume some a lot of bits to store motion vector. Otherwise, if we choice larger partition size (e.g. 16×16, 16×8, 8×16), we can save a lot of bits, but the prediction error will be large. In general, a large partition size is appropriate for homogeneous areas of the frame and a small partition size may be beneficial for detailed areas. Usually, using seven different block sizes can translate into bit rate savings of more than 15% as compared to using only a 16×16 block size.

Search Range

The maximum allowed motion displacement d_m, also known as the search range, has a direct impact on both the computational complexity and the prediction quality of the BMA. A small d_m results in poor compensation for fast moving areas and consequently poor prediction quality. A large d_m on the other hand, results in better prediction quality but leads to an increase in the computational complexity (since there are $(2d_m+1)^2$ possible blocks to be matched in the search window). A larger d_m can also result in longer motion vectors and consequently a slight increase in motion overhead [20]. In general, a maximum allowed

Displacement of $d_m = \pm 15$ pels is sufficient for low bit rate applications. MPEG standard uses a maximum displacement of about ± 15 pels, although this range can optionally be doubled with the unrestricted motion vector mode.

Search Accuracy

Initially, the BMA was designed to estimate motion displacements with full pel accuracy. Clearly, this limits the performance of the algorithm, since in reality the motion of objects is completely unrelated to the sampling grid. A number of workers in the field have proposed to extend the BMA to sub pixel accuracy. For example, Ericsson demonstrated that a prediction gain of about 2db can be obtained by moving from full pel to 1x8-pel accuracy. Grid presented an elegant theoretical analysis of motion compensating prediction with sub pixel accuracy. He termed the resulting prediction gain the accuracy effect. He also showed that there is a "critical accuracy" beyond which the possibility of further improving prediction is very small. He concluded that with block sizes of 16×16 , quarter-pixel accuracy is desirable for broadcast TV signals, whereas half pixel accuracy appears to be sufficient for videophone signals. Today, most video coding standards adopt sub pel accuracy in its half pel form. In fact, it has been shown that most of the performance gain of H.263 over H.261 can be attributed to the move from full pel to half pel accuracy.

It should be pointed out, however, that the improved prediction quality of sub pel accuracy comes at the expense of a significant increase computational complexity. This increase is due to two reasons. First, the reference frame intensities have to be interpolated at sub pel locations. Second, there are now more possible candidate blocks within the search window. For example, when moving from full pel to half pel accuracy, the number of candidate blocks in the search window increases from $(2d_m + 1)^2$ to $(4\mathrm{d}_m + 1)^2$. To alleviate this complexity, most video codec's implement sub pel accuracy as a post processing stage, where first a full pel motion

Vector is obtained, usually using full search, and then this vector is refined to subpel accuracy using a limited search. This provides a large saving in computational complexity and at the same time maintains the improved prediction quality.

Basic Approaches to Motion Estimation

There exist two basic approaches to motion estimation

1. Pixel based motion estimation

2. Block based motion estimation.

The pixel based motion estimation approach seeks to determine motion vectors for every pixel in the image. This is also referred to as the optical flow method, works on the fundamental assumption of brightness constancy that is the intensity of a pixel remains constant, when it is displaced. However, no unique match for a pixel in the reference frame is found in the direction normal to the intensity gradient. It is for this reason that an additional constraint is also introduced in terms of the smoothness of velocity (or displacement) vectors in the neighborhood. The smoothness constraint makes the algorithm interactive and requires excessively large computation time, making it unsuitable for practical and real time implementation.

An alternative and faster approach is the block based motion estimation. In this method, the candidate frame is divided into non-overlapping blocks (of size 16×16, or 8×8 or even 4×4 pixels in the recent standards) and for each such candidate block, the best motion vector is determined in the reference frame. Here, a single motion vector is computed for the entire block, whereby we make an inherent assumption that the entire block undergoes translational motion. This assumption is reasonably valid, except for the object boundaries and smaller block size leads to better motion estimation and compression.

Chapter 3

Block based motion estimation is accepted in all the video coding standards proposed till date. It is easy to implement in hardware and real time motion estimation and prediction is possible. The block based motion estimation shown in figure 3.8.1.

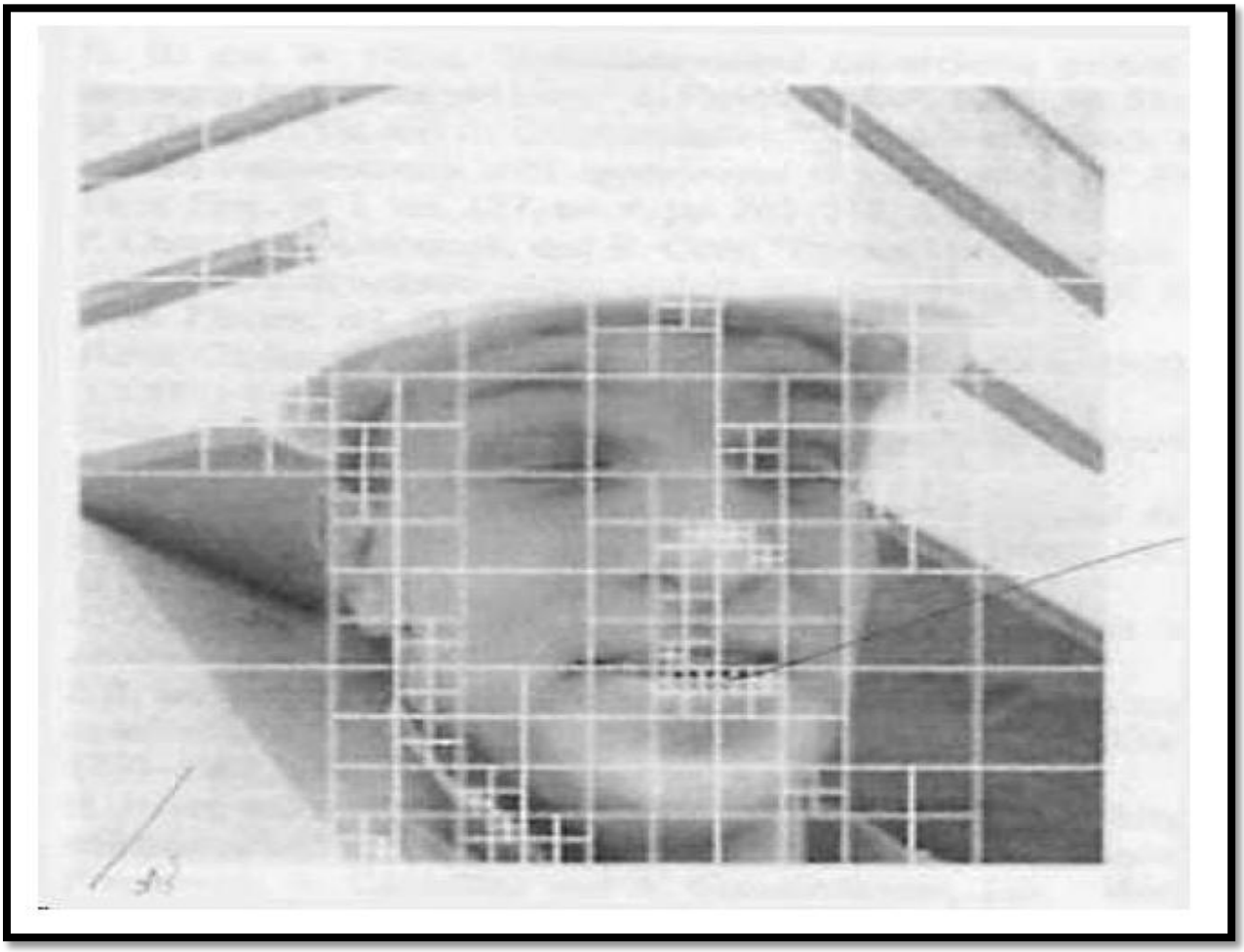


Figure 3.8.1 Selection of Block sizes with in a frame

Chapter 3

Block-based matching method is the most widely used motion estimation method for video coding since pictures are normally rectangular in shape and block-division can be easily done. Usually, a standards body, e.g. MPEG, defines the standard block sizes for motion estimation. This can be 16 x16, 8x8, etc., depending on the target application of the video codec. In the

Latest codec standards such as MPEG-4 or H.264/AVC, variable block sizes are supported which can be 4x4, 8x8 and 16by 16. The goal of motion estimation is to predict the next frame from the current frame by associating the motion vector to picture macro blocks as accurately as possible. The block size determines the quality of prediction [21] and thus the accuracy. Figure 3.13 shows the distribution of block sizes within a picture. It is easy to see that the detailed region is associated with small blocks whereas the large uniform region is associated with large blocks.

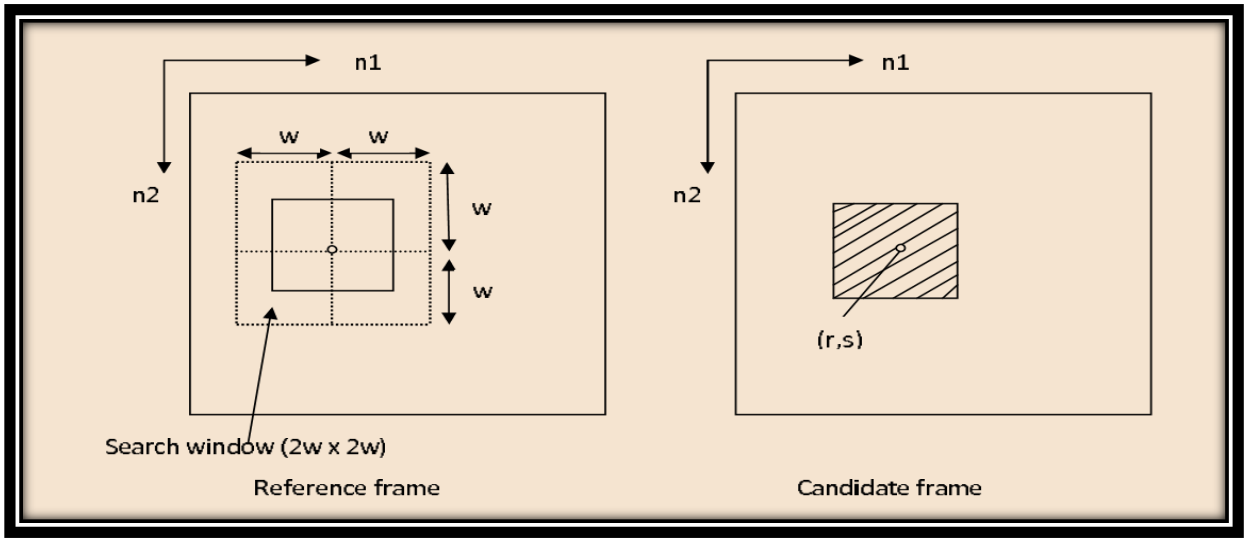
Chapter 4

4. Block based Matching Algorithms for Motion Estimation

Block based matching algorithms are processes for finding minimum motion vectors in the motion estimation process. Different kinds of algorithms could give a different motion vector. Among all algorithms proposed, only full search gives optimal result within the search range. Other algorithms will give near to optimal results but significant lower complexity by reducing the number of search points. In this section, we describe several block matching algorithms which are commonly used in modern coding standards.

4.1 Full Search Algorithm

The full search algorithm finds a global optimal motion vector from the entire candidate blocks within the search window. If our search window is 48X48 pixels and the block size is 16X16, there will be 16X16 = 1024 candidates that need to undergo MAD computation. FS exhaustively searches all candidates until a minimum motion vector is found and it is the algorithm with the simplest data flow and control. It is suitable for hardware implementation since the high computational complexity can be overcome by parallelism and high bandwidth be overcome by systolic architectures.



Reference frame Candidate frame

Figure 4.1.1 Full search motion estimation

Chapter 4

In full search motion estimation entire frame as a search window size.Consider a block of N × N pixels from the candidates from at the coordinate position (r, s) as shown and then consider a search window having a range ±w in both directions in the reference frame, as shown. For each of the (2w+1) ^2 search positions (including the current row and current column of the reference frame), the candidate block is compared with a block of size N × N pixels according to one of the best matching criteria is discussed and the best matching block, along with the motion vector is determined only after all the (2w + 1) ^2 search positions are exhaustively explored. The FSBM is optimal in the sense that if the search range is correctly defined, it is guaranteed to determine the best matching position. However, it is highly computational intensive. For each matching position, we require O (N^2) computations (additions subtractions multiplications etc) and the search positions, the number of computations for each matching criteria given in table 4.1.1

Method	**Computations**		
	Addition/ Subtraction	**Multiplication**	**Comparison**
MSE	$(2N^2-1)(2W+1)^2$	$N^2(2W+1)^2$	$(2W+1)^2$
MAD	$(2N^2-1)(2W+1)^2$	---------	$(2W+1)^2$
MPC	$(2N^2-1)(2W+1)^2$	--------	$N^2(2W+1)^2$

Table 4.1 Computational Complexity of FSBM

FSBM require large number of computations. If w=7 pixels, measuring that the best matching position exist within a displacement of ±7 pixels from the current block position, it require 15 × 15 = 225 search positions. For real time implementation, quick and efficient search

Strategies were explored. It may be noted that such quick search techniques do not make exhaustive with in the search area and can be best be suboptimal. A residual frame found by full search algorithm using the same frames as in the figure 4.1.2 is given in figure 4.3. However, the process is very computationally expensive. Therefore, the searching process is generally made within a particular search area which is called as fast search. There are number of techniques for determining the search area in the reference frame. Some of these algorithms.

Figures 4.1.2(a) Current frame *Figure 4.1.2(b) previous frame*

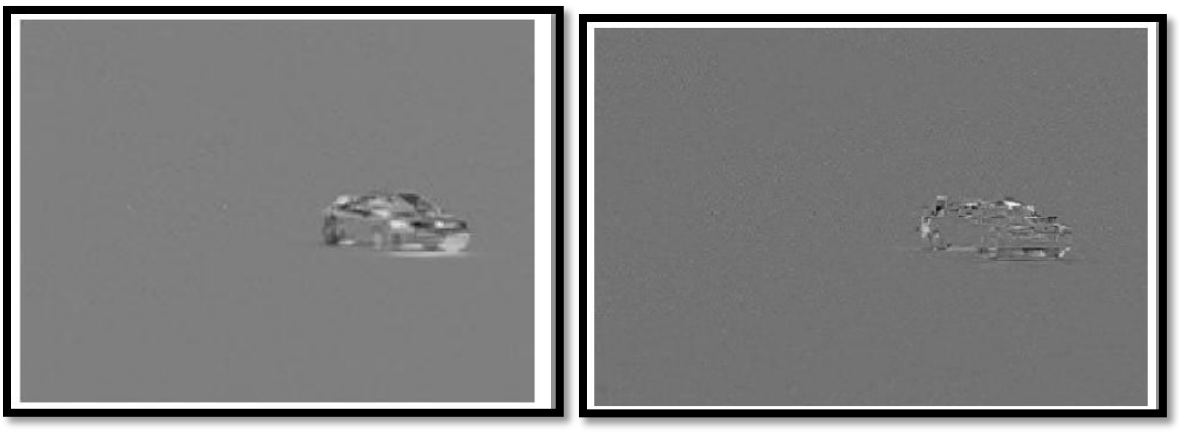


Figure 4.1.2(c) Residual frame no motion *Figure 4.1.2(d) Residual frame after motion compensation*

For software implementations, full search is often ignored [2] as contemporary microprocessors cannot handle full search with acceptable performance, especially for real time applications. The number of search locations to be examined by full search is directly proportional to the square of the search range r.

This section offers review of several well-known motion estimation search algorithms that have been frequently used in video coding. Previous works on motion estimation can be categorized into two groups one is search point reduction (SR) and the other is calculation reduction (CR) of SAD in the following these two groups are discussed in details.

4.2 Three Step Search

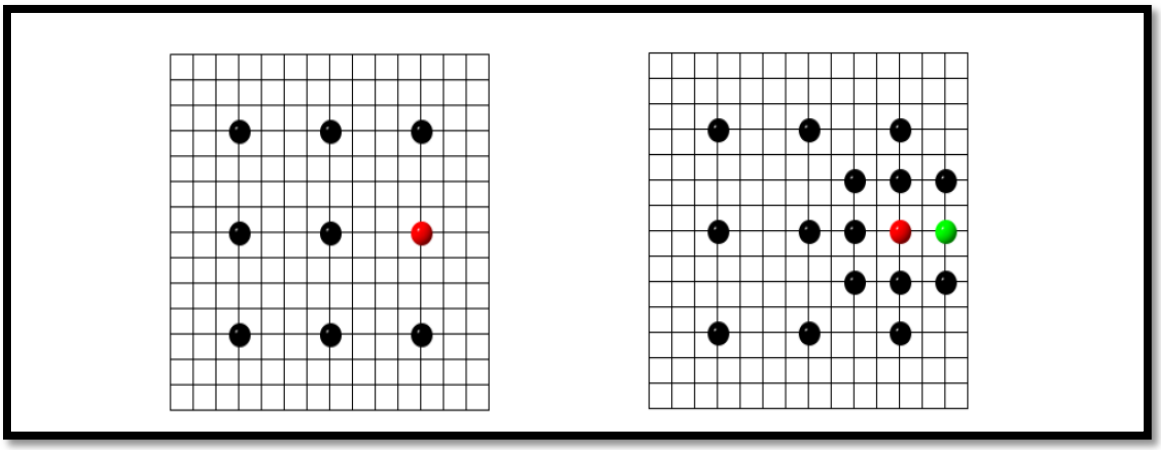
The Three Step Search algorithm is the first fast BMA without exhaustive search. TSS has two important concepts for motion estimation in terms of fixed search pattern and limited search step. Most of the latter works inherit these two concepts to develop their algorithms [23]. It adopts three steps to find the most similar MB with in the search window of the reference frame.

For a search window of +/- (2N - 1) pixels, the TSS algorithm is as follows:

1. Search location (0, 0).
2. Set S = 2N-’ (the step size).
3. Search eight locations +/-S pixels around location (0, 0).
4. From the nine locations searched so far, pick the location with the smallest MAD and make this the new search origin.
5. Set S = S/2.
6. Repeat stages 3-5 until S = 1.

The above figure illustrates the procedure for a search window of +/-7 (i.e. N = 3). The first ‘step’ involves searching location (0, 0) and eight locations +/-4 pixels around the origin. The second ‘step’ searches +/-2 pixels around the best match from the first step (highlighted in bold) and the third step searches +/-l pixels around the best match from the second step (again

Highlighted). The best match from this third step is chosen as the result of the search algorithm. With a search window of +/-7, three repetitions (steps) are required to find the best match. A total of (9+8+8) = 25 search comparisons are required for the TSS, compared with (15 x 15) = 225 comparisons for the equivalent full search. In general, (8N+ 1) comparisons are required for a search area of +/- (2N - 1) pixels.



Figures 4.2.1 (a) The first step of TSS *Figure 4.2.1 (b) The second step of TSS*

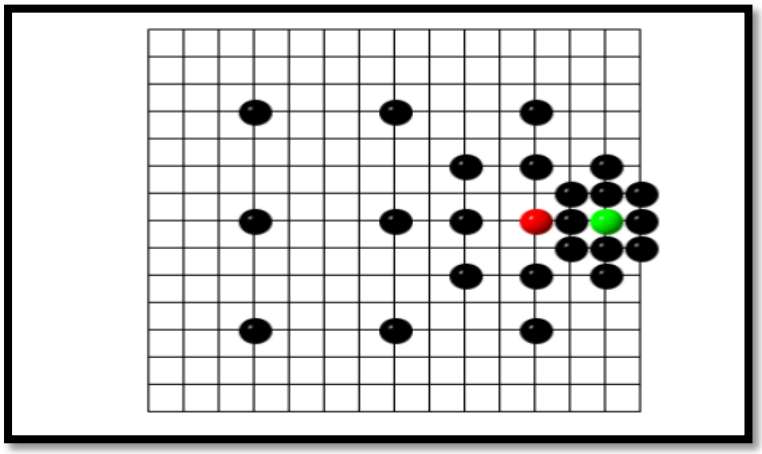


Figure 4.2.1 (c) The Third step of TSS

Advantage: It will give good result when compared to all searching algorithms.

- Disadvantage: It will take large computations (Number search points are more).

Another way of search pattern is shown in figure 4.2.2 (a)

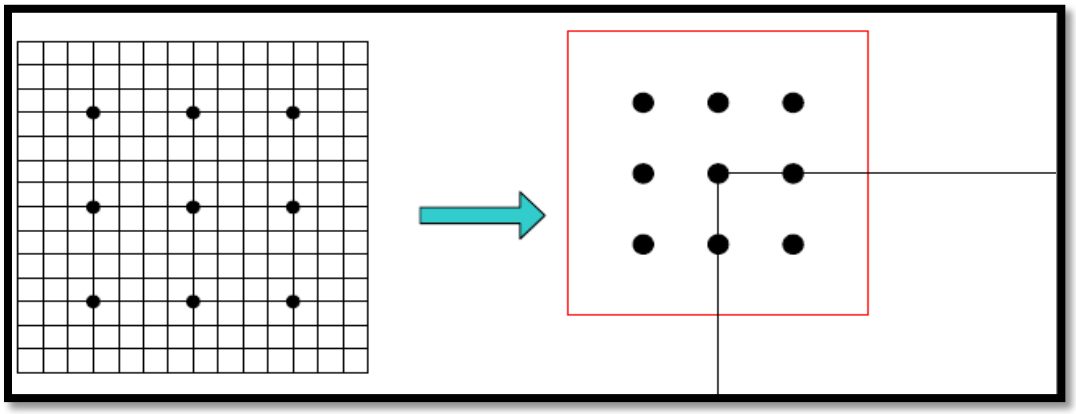


Figure 4.2.2 (a) three step search pattern

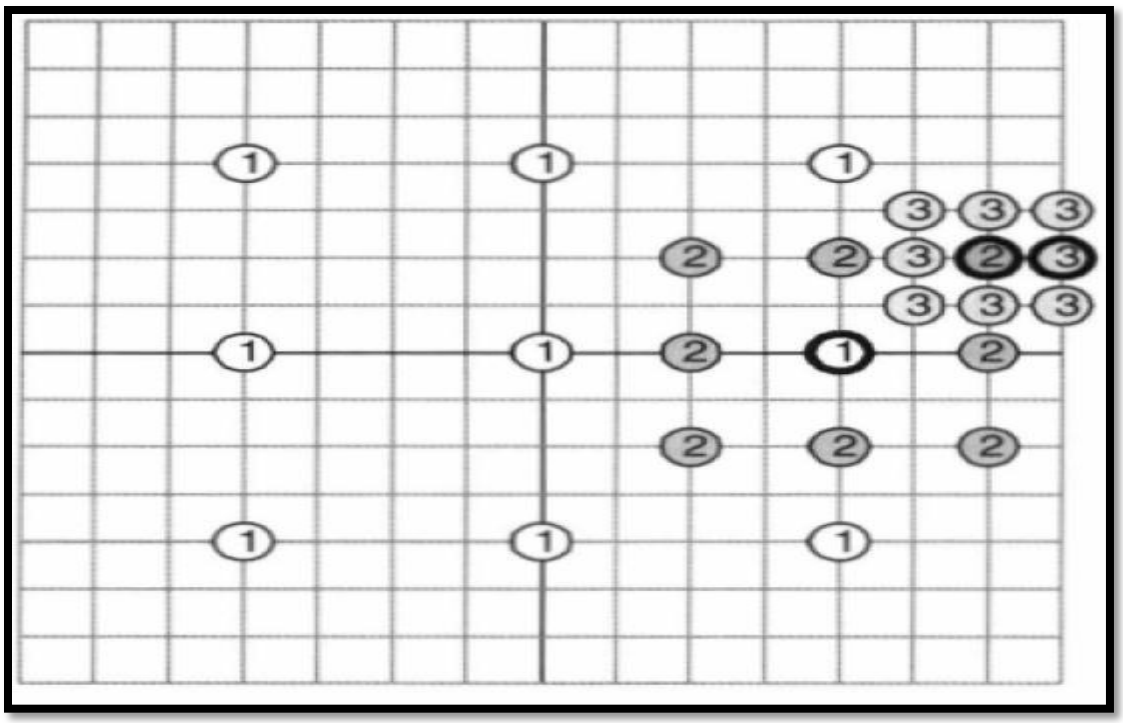


Figure 4.2.2 (b) three step search pattern

The Three Step Search algorithm is first fast search algorithm and it gives the better motion vector but this one also takes the large complex computations due to this we are moving to another searching algorithm.

Disadvantage: TSS may not find the global minimum (or maximum) of the matching criterion, instead it may find only a local minimum and this reduces the quality.

4.3 New Three Step Search

The observation that the MVs of the frame with slow motion are almost found near the center of the search window, the NTSS algorithm [23] proposed in1998 sues a center biased checking point pattern and a half way stop technique for stationary or quasi stationary MBs to improve the performance of the TSS. The NTSS algorithm first checks the seventeenth points shown in fig 4.4(a). If the center point has the minimum MAD, the search is will stop. If the minimum MAD occurs at one of the eight neighbors of the center point, the NTSS algorithm checks five corner or three edge points shown in figure 4.5 (b) or figure 4.5 (c), and then the search stops. Otherwise, the search steps of NTSS are similar to that of TSS as shown in figure 4.5 (b) or figure 4.5 (c). As NTSS constructs a center biased search pattern by adding 8 extra checking points around the center of search window, it can efficiently find the motion vector of the frame with slow motion. Meanwhile, a halfway stop scheme to keep the NTSS algorithm compatible to TSS in terms of computing complexity was used. As a result, NTSS always gets smaller motion compensation errors than TSS.

Advantage: A NTSS algorithm always gets smaller motion compensation errors than TSS.

Disadvantage: When compared to the TSS algorithm the NTSS have complex computations.

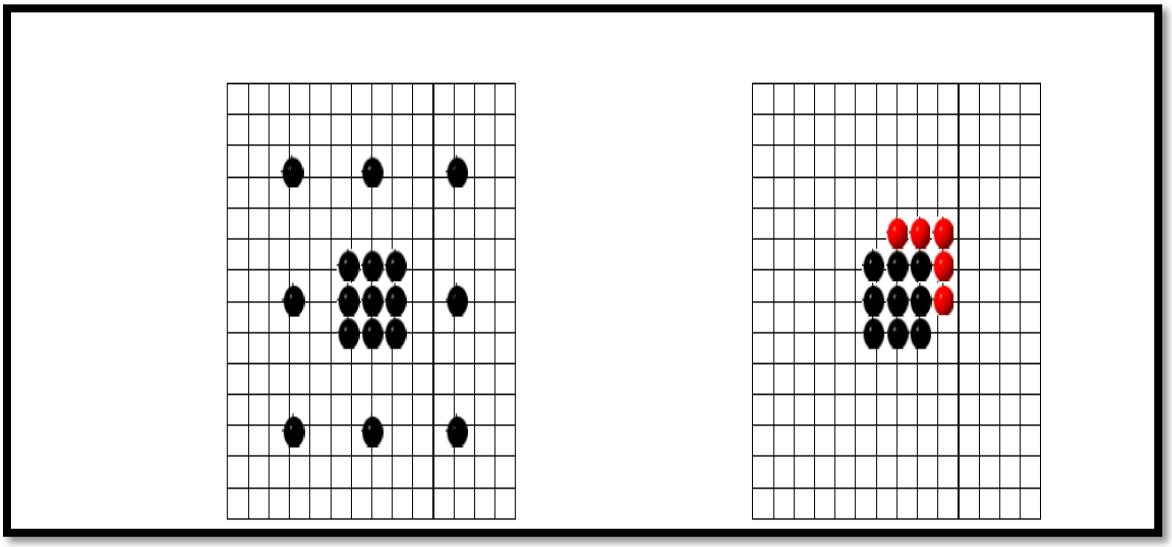


Figure 4.3.1 (a) NTSS Search Pattern Figure 4.3.1 (b) Corner Points

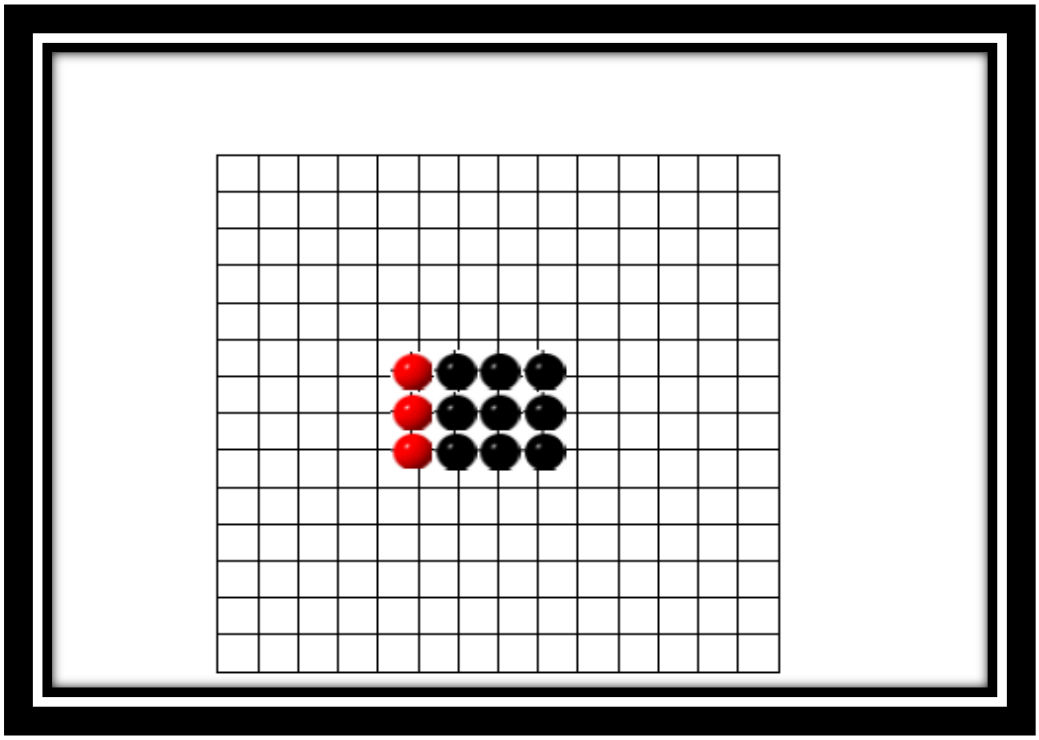


Figure 4.3.1 (c) Edge Points

NTSS algorithm also complex computations are high to reduce this draw back we are going to four step search algorithm.

4.4 Four Step Search Algorithm

The search shape of the Four Step Search (FSS) is similar to Three Step Search (TSS) [24] which is proposed in 2001, but different interval among search points in figure 4.6(a)-4.6(b). Initially, the FSS tests nine candidate points and the search center is then moved forward to the best matching point. If the best matching point could locates at the corner or edge, five and three additional checking points are required for the next step with the same step size shown in figure 4.6(c)-4.6(d),respectively. If the best matching point locates at the center of the search pattern, the step size is halved shown in figure 4.6(d). FSS process ends while the minimum MAD of the nine candidates in the fourth step is found and the optimal MV is obtained. FSS algorithm outperforms TSS and has similar performance compared to the NTSS in terms of MAD but with lower computing complexity. FSS is more robust than TSS and NTSS because FSS is suitable for video that contains complex movements such as camera zooming and fast motion. In addition, FSS is also easily implemented in hardware due to its regularity and simplicity.

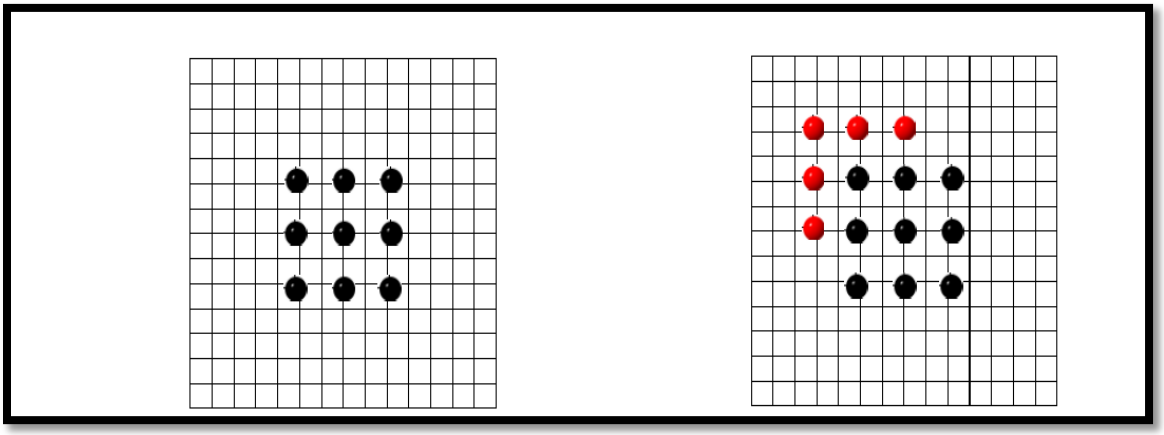


Figure 4.4.1(a) FSS Pattern

Figure 4.4.1(b) Corner Points of FSS Pattern

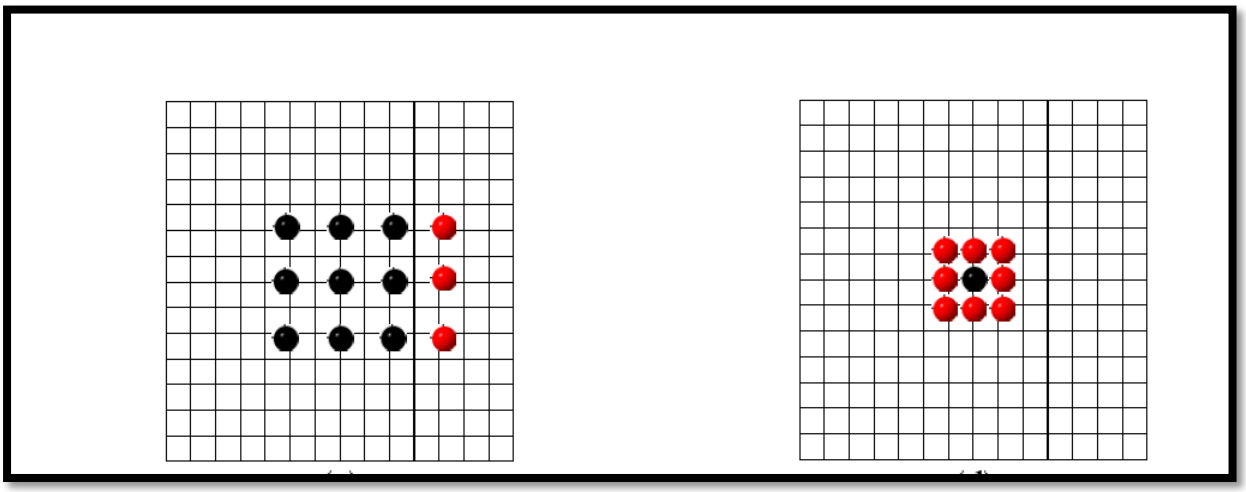


Figure 4.4.1(c) Corner Points of FSS

Figure 4.4.1 (d) Center points of FSS

4.5 Logarithmic Search

This algorithm was introduced at the same time that the Three Step Search was introduced and is closely related to it. This algorithm is known as "Logarithmic Search" because the multi-stage search is accomplished by successively reducing the search area during each stage until the search area is trivially small. Although this algorithm requires more steps than the Three Step Search, it can be more accurate, especially when the search window is large. The algorithm may be described as follows;

(1) Determine an initial step size 'N'. Search the block in the origin (0, 0) and the four blocks at a distance of 'N' pixels away from the origin both in the vertical and horizontal directions. The five positions from a '+' sign.

(2) If the position of the best match is at the Centre, set N=N/2. If one of the other four positions is the best match, then it becomes the Centre and step 1 is repeated.

(3) When the step size becomes 1, all the 9 blocks around the Centre are chosen for the search and the best among them is picked as the best matching block.

Here one particular search path for the logarithmic search algorithm is shown in the figure 4.7

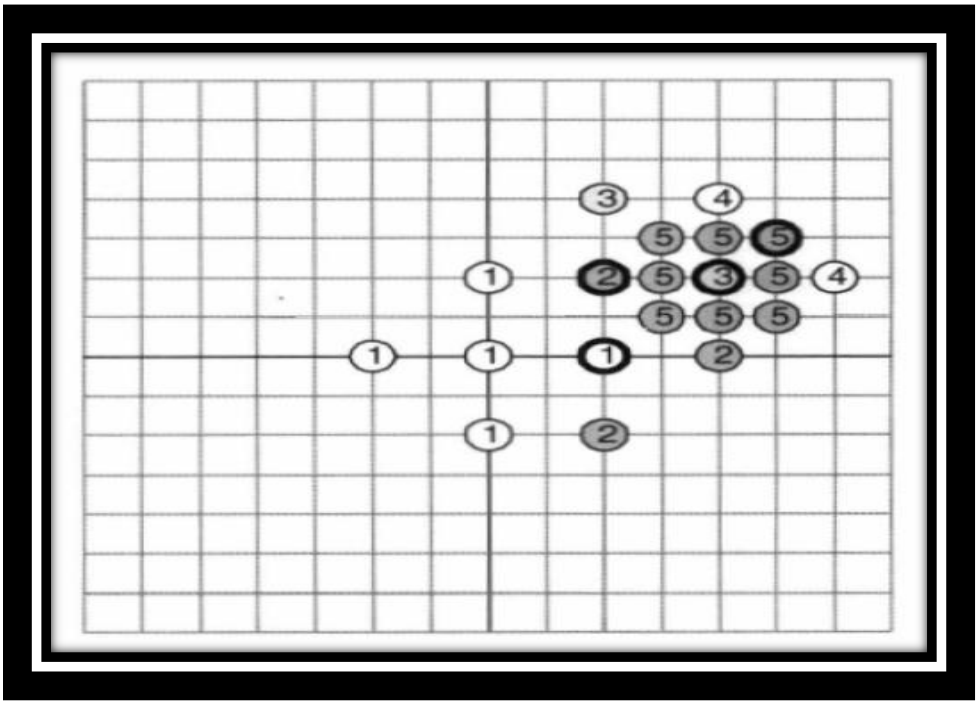


Figure 4.5.1 Logarithmic Search

4.6 Cross Search

This algorithm is similar to the three step search except that five points are compared at each step (forming an X) instead of nine.

(1) Search the location (0,0).

(2) Search four locations at +/- S, forming an 'X' shape.(where S=2N-' as per the TSS).

(3) Set the new origin to be the best match of the five locations tested.

(4) If S>1 then S=S/2 and go to stage 2; otherwise go to stage 5.

(5) If the best match is at the top left or bottom right of the 'X' evaluate four more points in an 'X' at a distance of of +/-l; otherwise (best match is at the top right or bottom left) evaluate four more points in a ' + ' at a distance of +/-l.

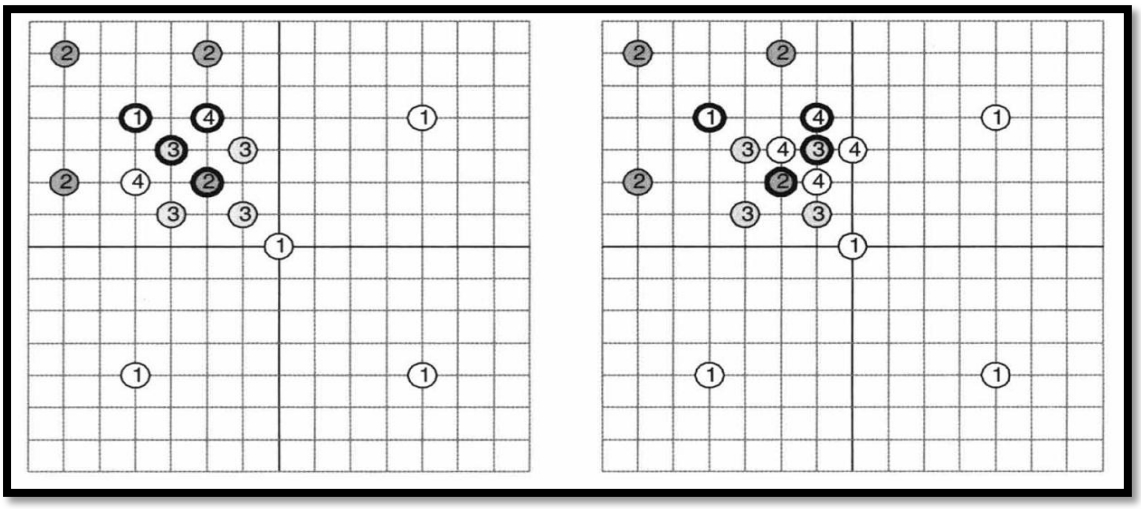


Figure 4.6.1 Cross Search Pattern

So from 90s onwards different types of block matching algorithms are proposed to find motion estimation. Observation of these algorithms I conclude two main things these are the number of search points are required to find the best motion vector should be minimum and the minimum displaced frame difference (DFD) error should be minimum (MAD). On the basis of these algorithms I proposed one block matching algorithm this algorithm simulated results are compared with different algorithms it gives the minimum error and search points are minimum.

Comparison of Motion Estimation Algorithms

Wide range of algorithms available for block based motion estimation can make it difficult to choose between them. In this the number of criteria that may help in the choice ***(1) Matching performance:*** How effective is the algorithm at minimizing the residual block***(2) Rate distortion performance*** : How well does the complete CODEC perform at various compressed bit rates***(3)Complexity***: Number of operations required to complete the matching process***(4)Scalability***: Does the algorithm perform equally well for large and small search window***(5)Implementation:*** Is the algorithm suitable for software or hardware implementation for the chosen platform (or) architecture.

Chapter 5

5.1 Three Step Diamond Search Algorithm

5.1 Introduction

Video coding is an important process in many multimedia applications. In addition to spatial redundancy, temporal redundancy plays an important role in the transmission of video frames .An effective and popular method to reduce the temporal redundancy called block matching (BMME), motion estimation is the technique used to reduce the temporal redundancy. It uses the correlation between the successive frames to predict the content of frames it has been widely adopted in various video coding standards such as H.261, H.263, MPEG-1, MPEG-2, andMPEG-4 and in any motion compensated video coding technique. Therefore fast and accurate block-based search technique is highly desirable to assure much reduced processing delay while maintaining good reconstructed image quality. In the motion estimation process the frame is divided into number of non-overlapping areas known as macro blocks. Each macro block can be with a standard size of 16×16. The difference between the current frame and the predicted contents is calculated in motion estimation. In addition to motion estimation, some additional information's are also needed to indicate any changes in the prediction process this is known as motion compensation.

By exhaustively testing all the candidate blocks with in the search window, full search algorithm gives the global optimum solution (i.e., the minimum matching error point over the search window) to the motion estimation, while a substantial amount of computational load is demanded. To overcome this drawback, many fast block matching algorithms (BMA's) have been developed. For example (TSS), new three step search (NTSS), four step search (4SS), diamond search (DS), etc. These fast BMA's exploit different search patterns and search strategies for finding the optimum motion vector with drastically reduced number of search points as compared with FS algorithm. In TSS ,NTSS, and 4SS algorithms square shaped search patterns of different sizes are employed in TSS a large search pattern with size of 9 × 9 and sparse checking points as exploited in the first step of TSS is most likely to mislead the search path to wrong direction and hence misses the optimum point. In other ways the diamond search algorithm maintains the diamond shape search pattern; it gives faster search pattern and minimum absolute difference when compared with TSS, NTSS, DS and 4SS. This inspires us to investigate why DS pattern can yield speed improvement over some square

Shaped search patterns and what the mechanism behind is. as a result we try to reduce the step size it gives the better performance than the DS and TSS, based on these search patterns we propose a three step diamond search algorithm that can achieve a substantial speed improvement. The search speed and the performance of an algorithm are determined by the shape and size of the search patterns. The TSS and NTSS algorithms are using a squared shape pattern, whereas the diamond search algorithm uses a Diamond shape. This diamond search (DS) algorithm uses unrestricted center biased searching concept and so it is computationally inefficient. In this paper, a three step diamond search algorithm is proposed to attain a computationally efficient search with a reasonable distortion performance gives the minimum number of search points and minimum error when compared to the TSS and DS.

5.2 Diamond Search Algorithm

The shape of the search pattern has an impact on the performance of the algorithm. Fast block matching algorithm such as TSS, NTSS are having a square shape search pattern and provides reasonable performance. The distribution of global minimum points is centered at the center of search window [22].The diamond search provides a better performance than the TSS, NTSS algorithms. The diamond search (DS) algorithm uses a diamond shape pattern with nine search points, four points located at the corners and another four points located at the midpoint of the edges of the diamond shape. This algorithm uses an unrestricted center biased searching process. The diamond search employs a large diamond search pattern (LDSP) [fig 5.2.1(a)] and small diamond search pattern (SDSP) [fig5.2.2 (b)].

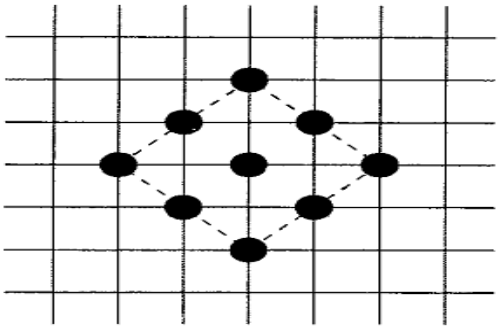


Figure5.2.1 (a) Large Diamond Search Pattern

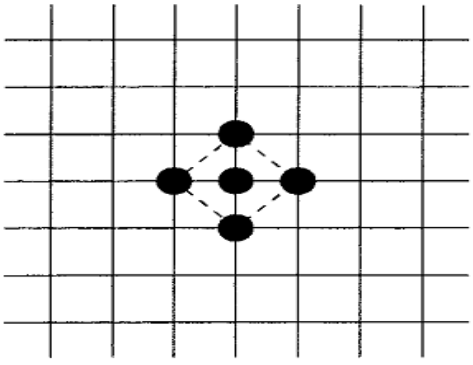


Figure 5.2.1 (b) Small Diamond Search

As some of the search points in the newly formed LDSP are overlapping, only the non over lapping points need to be evaluated. This greatly reduces the number of search points compared to other existing fast search algorithms. Therefore the search pattern uses five search points in the new LDSP if the MBD point is the corner point [fig 2(a)] and the three search points if the MBD point is at the edge of the pattern [fig5.1.2(c)] . The LDSP is used until the center point becomes the MBD point. Once the minimum block distortion (MBD) point is at the center, the search is switched to SDSP which uses four checking points [fig 5.1.2(d)].

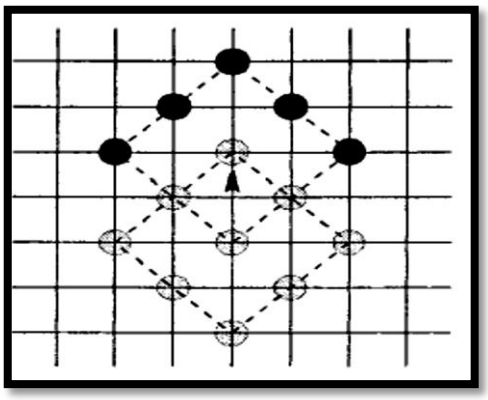


Figure 5.2 .1(c) Corner Point

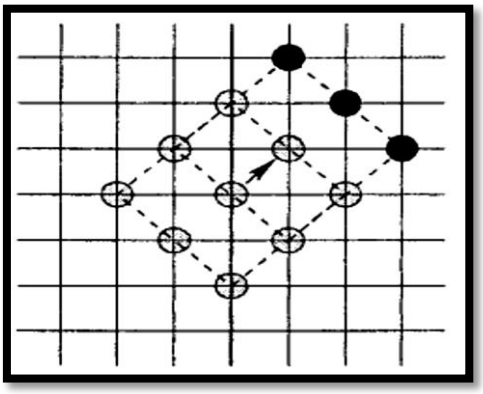


Figure 5.2.1(d) The edge point.

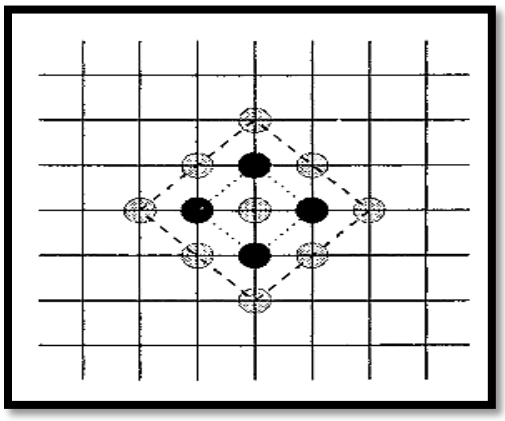


Figure 5.2.1(e) The centre point

Fig. The above figures represent the different search patterns over lapping in LDSP in this the fig 5.2(a) represent the large diamond search pattern when the minimum block distortion point occurred in center the large diamond change to small diamond search pattern the fig 5.2(a represent the corner search points the fig 5.2(b) represent the edge search point and center point. The solid block dots are the new checking points where the computation of block-distortion measurement is required for the current search step.

The minimum block distortion point (MBD) thus obtained will give the motion vector. The DS algorithmreduces the susceptibility of getting stuck at local minima due to its compact shape and relatively large step size in the horizontal and vertical direction. Thus the diamond search algorithm gives a faster processing and similar distortion performance with the other fast searching algorithms. The increase in number of steps leads to more number of search points which has an effect on the speed of the algorithms. This algorithm gives the less complexity when compared to the previous algorithms. A three step diamond search (TSDS) is proposed to overcome this disadvantage.

5.3 **Three Step DS Algorithm**

The proposed TSDS algorithm uses the same type of patterns used in DS algorithm with a reduction in a step size. Based on the location of the MBD point, the number of checking points to be used in the successive steps varies. The search pattern shows that the figures 5.1.3[a-d].

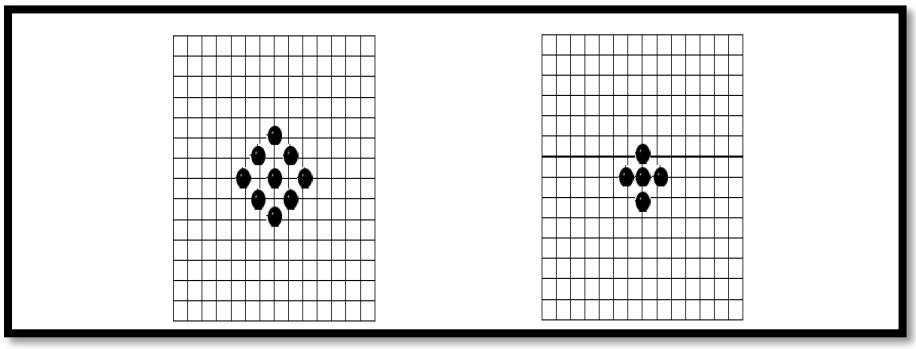


Figure 5.3.1(a) LDSPFigure5.3.1 (b) SDSP

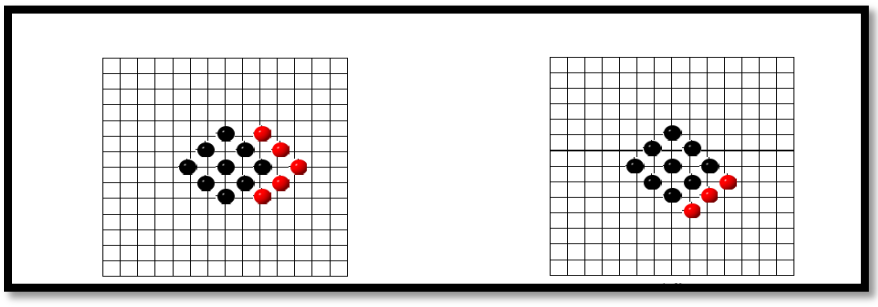


Figure5.3.1(c) Vertex point of LDSP Figure5.3.1 (d) Face point of LDSP

The number of searching steps is reduced to three and the SDSP search is reached at the third step regardless of the location of the MBD point. The LDSP is repeatedly used until the centre point becomes the MBD point. The algorithm for this TSDS algorithm is summarized as follows.The number of searching steps is reduced to three and the SDSP search is reached at the third step regardless of the location of the MBD point[32]. The LDSP is repeatedly used until

Thus the compact configuration and reduced number of search points provide an improved performance than the other existing algorithms.The algorithm for this TSDS algorithm is summarized as follows.the centre point becomes the MBD point[31]. Thus the compact configuration and reduced number of search points provide an improved performance than the other existing algorithms given below. The algorithm for this TSDS algorithm is summarized as follows.

(a)Algorithm:

Step 1: Initial LDSP is centered at the origin of the search window. Now, test each point in the search pattern. If the MBD point is the center point go to step3. Otherwise go to step2.

Step 2: Form a new LDSP with the MBD point as the center point. If the new MBD point is at the center position, go to step3. Otherwise repeat this step for one more time.

Step 3: Form the SDSP with previous MBD point as the center point. The new MBD point obtained in this step becomes the final solution i.e., the motion vector (x, y). The number of search points depends on the location of MBD point also determines the search direction.

TABLEI 5.3.2

Image sequences used for simulation result

Image sequences	Frame size	Length
CALTRAIN	400×512	32
FOREMAN	176×144	55
TENNIS	240×352	90
CRAFTER	256×256	28

The image sequence used for simulation in this the number of search points and MAD for each macro block wise finding and depending on sequence length so in the proposed method we find the MAD for "**caltrain**" sequence this is shown in the figure and flow chart also shows that the search steps and error calculation.

Flowchart

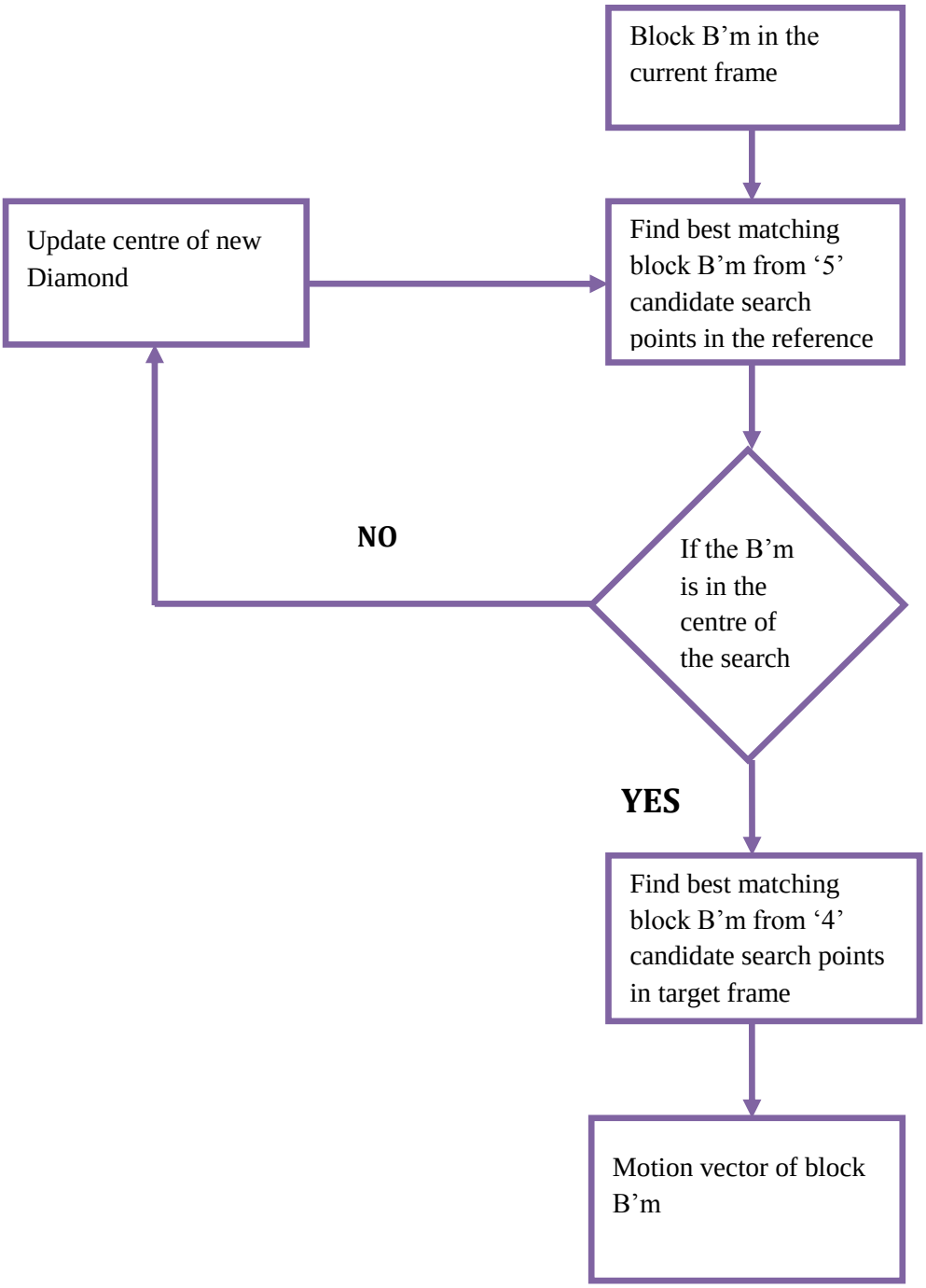

Block B'm in the current frame
Find best matching block B'm from '5' candidate search points in the reference
If the B'm is in the centre of the search
NO
Update centre of new Diamond
YES
Find best matching block B'm from '4' candidate search points in target frame
Motion vector of block B'm

Table5.3.3

Calculation of Search Points and Search Steps

Search Alg.	Search points [MIN]-[MAX]	No of search steps [MIN]------ -[MAX]	Formulae
TSS	17 ---- 33	1 3	9+8n
DS	13 ---- 22	0 3	9+3n+4
FS	225-- 225	1 1	(2q+1)^2
TSDS	11 ---- 17	1 3	5+3n+3

Theoretical calculations to find the number of search points and number of search steps. FS takes enter frame as search window size 15×15 so that it will take the large computational complexity (n represents the number of search steps)

5.4. Simulation Results and Comparison

In our simulation results, the block size is fixed at 16×16. To make a consistent comparison, block matching is conducted within a search window size of 15×15 is used for experimentation of this algorithm and the center point of initial LDSP is at the origin of the search window. The performances of this algorithm are evaluated by error metrics, such as the number of search points and mean absolute difference (MAD). The performance analysis has been done for different sequences like '' ***Caltrain***'' ''***Foreman***'' and ''***Tennis***'' sequences and the mean absolute difference (MAD) values and search points are presented in table I ii iii IV V VI. Shows that the theoretical calculations of number of search points and search steps. The result shows that, it gives a better performance compared with the existing [TSS] and [DS] algorithms with a reduction in step size also. The search is confined within the search window and the reduction in number of steps results in reduction in computational complexity. Easy way and good pattern of this algorithm provides good implementation. The criterion used for the distortion measurement is Sum of Absolute Difference (SAD), which gives the MBD point for the motion vector calculation. The pels are arranged in such a way that two in horizontal direction and in vertical direction, and one in each diagonal direction. This makes the algorithm to reach a global minimum point. The maximum number of search points used is 23 whereas the TSS uses 25 search points. It achieves a close MSE performance with the DS, TSS, and NTSS algorithm for the image sequences with small motion as well as large motion content.

Table II 5.4.1 (a)
Average MAD per Pixel for different sequences

Algorithm	Caltrain	Foreman	Tennis	Crafter
TSDS	53.6678	160.2307	149.5830	50.2650
DS	54.1875	164.7500	156.5000	53.6193
TSS	64.1091	217.2374	208.1700	69.3177
FS	50.2344	141.2005	132.9361	48.7834

Table II5.4.2 (b)

MAD Comparison of TSDS, DS, TSS and FS "Caltrain" Sequence Length is 32 Frames

Algorithm											
FS	*37.55*	*29.92*	*35.84*	*38.37*	*40.89*	*38.00*	*39.01*	*44.40*	*47.72*	*49.25*	*50.93*
	51.45	*54.58*	*55.70*	*56.53*	*56.21*	*51.50*	*49.40*	*47.00*	*47.75*	*48.5*	*48.71*
	50.68	*53.99*	*60.73*	*60.20*	*56.36*	*58.76*	*63.00*	*64.58*	*64.52*	*65.21*	*64.99*
DS	*38.5*	*36.00*	*36.50*	39.02	*42.20*	*39.32*	*40.00*	*45.50*	*50.20*	*52.00*	*54.00*
	55.21	*58.21*	*60.21*	*62.24*	*61.50*	*57.21*	*54.32*	*52.50*	*52.32*	*53.21*	*53.10*
	55.00	*59.00*	*65.00*	64.30	*62.21*	*62.50*	*68.21*	*70.77*	*69.70*	*69.23*	*69.78*
TSS	*40.00*	*39.00*	*39.50*	*43.00*	*46.00*	*50.50*	*52.00*	*52.50*	*59.00*	*60.00*	*64.21*
	66.21	*68.23*	*67.50*	*65.21*	*63.11*	*61.50*	*59.00*	*63.21*	*63.56*	*64.00*	*68.12*
	73.00	*73.50*	*70.00*	*69.12*	*68.97*	*70.10*	*76.00*	*75.10*	*73.23*	*74.68*	*75.23*
TSDS	*37.5*	*30.00*	*36.17*	*38.80*	*41.80*	*38.60*	*39.51*	*45.00*	*49.71*	*51.52*	*53.92*
	54.45	*57.58*	*59.70*	*61.53*	*61.24*	*56.53*	*53.42*	*52.23*	*51.70*	*52.53*	*51.73*
	52.54	*51.74*	*52.53*	*52.73*	*54.67*	*58.99*	*64.73*	*64.20*	*62.00*	*62.36*	*69.50*

TABLE III 5.4.3(c)
Average number of search points per macro block

Algorithm	Caltrain	Foreman	Tennis	Crafter
TSDS	53.6678	160.2307	149.5830	50.2650
DS	54.1875	164.7500	156.5000	53.6193
TSS	64.1091	217.2374	208.1700	69.3177
FS	225	225	225	225

TABLE IV 5..4.4(d)

Number of Search Points comparison of TSDS, DS, TSS and FS to Caltrain Sequence

Algorithm											
TSDS	15.6	15.7	15.5	15.5	15.7	15.8	15.70	15.8	15.9	15.19	15.50
	15.0	15.9	16.1	16.3	16.4	16.4	16.50	16.4	16.4	16.33	16.00
	15.9	15.8	15.6	15.6	15.6	15.5	15.40	15.5	15.2	15.52	15.88
DS	15.8	15.9	15.8	15.0	15.8	15.9	16.00	16.1	16.2	16.14	16.10
	16.2	16.3	16.5	16.8	16.7	16.8	16.60	16.7	16.8	16.71	16.85
	16.7	16.6	16.4	16.2	16.1	16.0	15.70	15.7	15.7	15.85	15.75
TSS	20.0	20.2	20.2	20.2	20.2	20.3	20.35	20.3	20.3	20.48	20.38
	20.4	20.3	20.3	20.3	20.0	19.8	19.60	19.5	19.8	19.93	20.00
	20.0	20.1	20.2	20.3	20.3	20.2	20.29	20.2	19.8	20.32	20.17

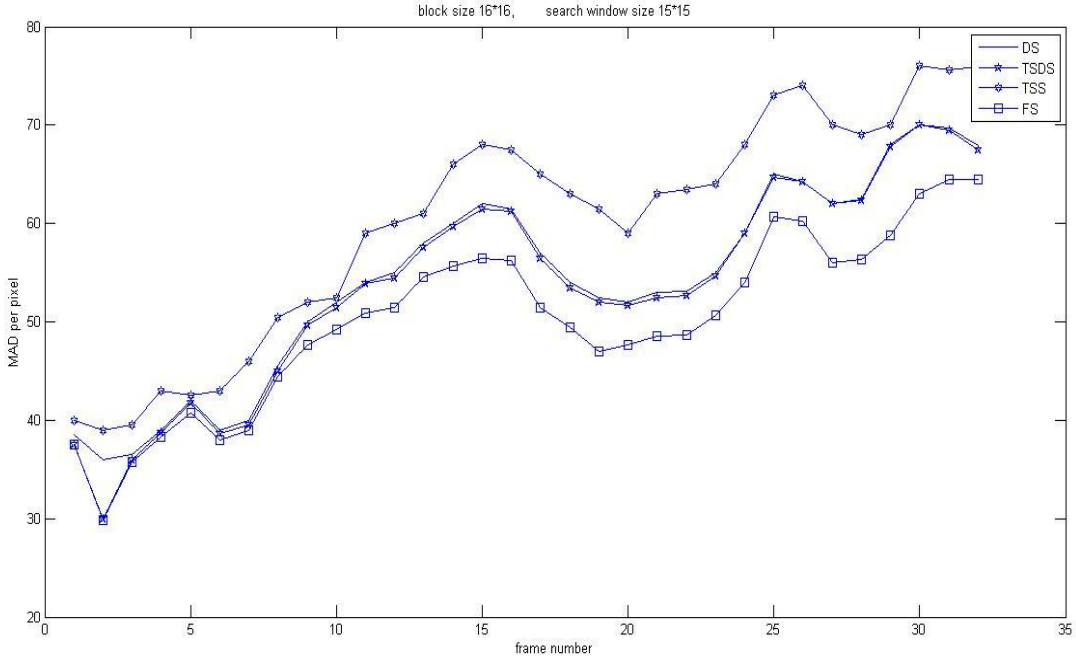


Figure 5.1.1 MAD comparison of DS, TSDS, TSS, FS for Caltrain Sequence

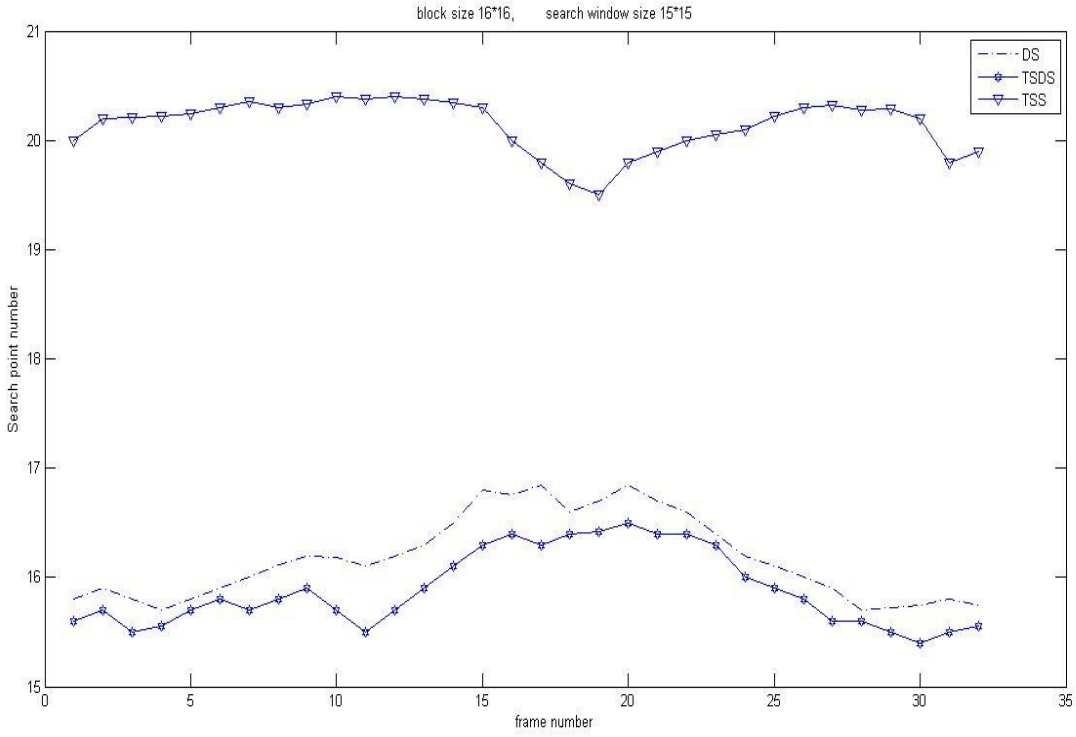


Fig. 5.1.2 Comparison ofAvg number of search points applying DS, TSDS, TSS for Caltrain Sequence

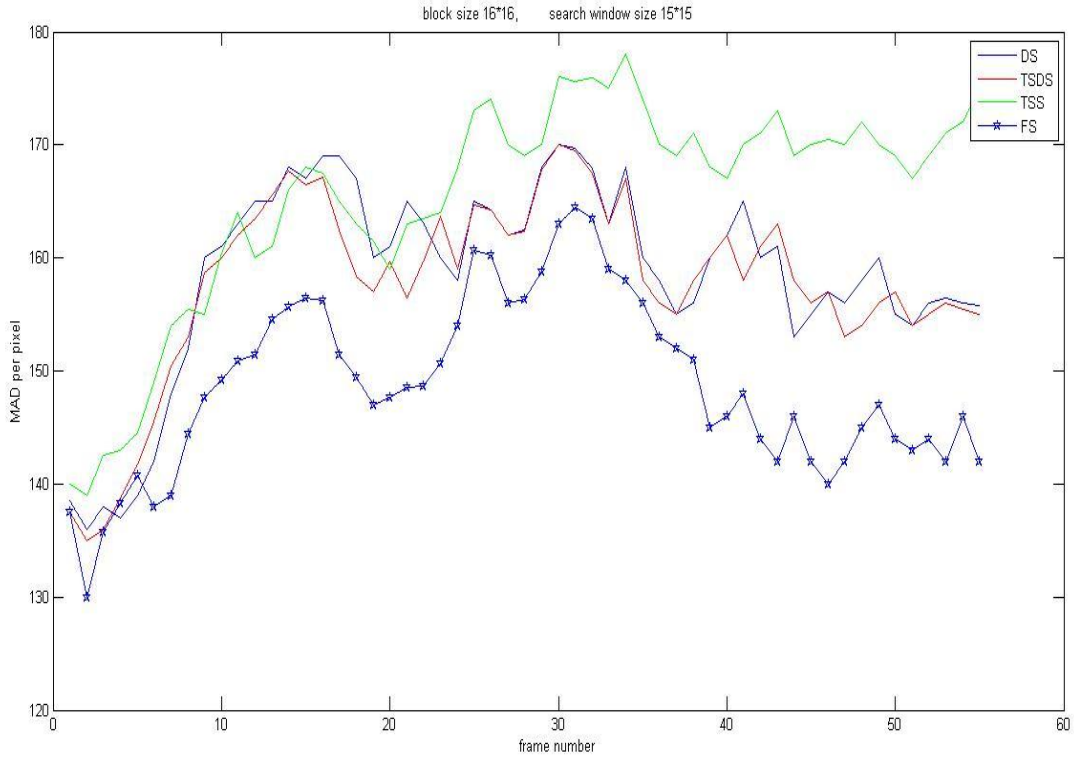


Fig.5.1.3 MAD Comparison of DS,TSDS,TSS and FS for foreman Sequence

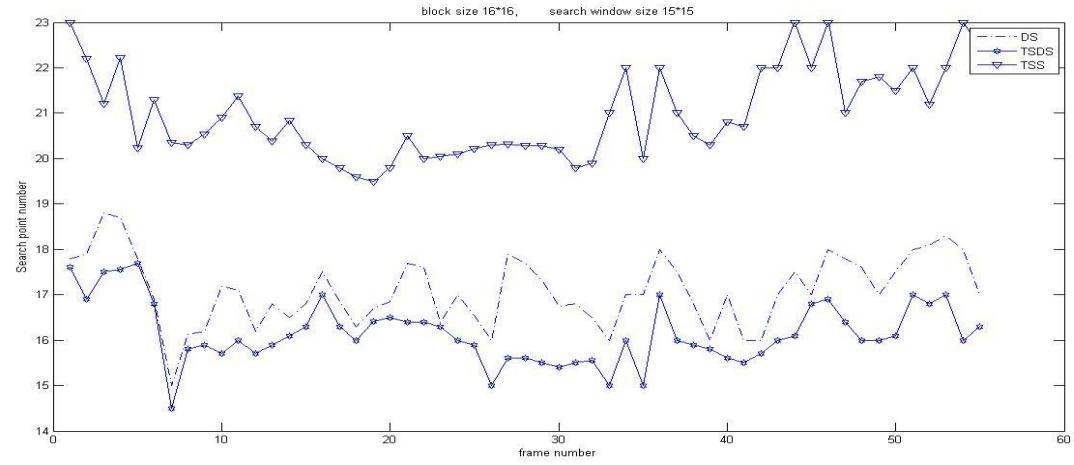


Fig 5.1.4 Comparison of Avg number of search points applying DS,TSDS and TSS to foreman

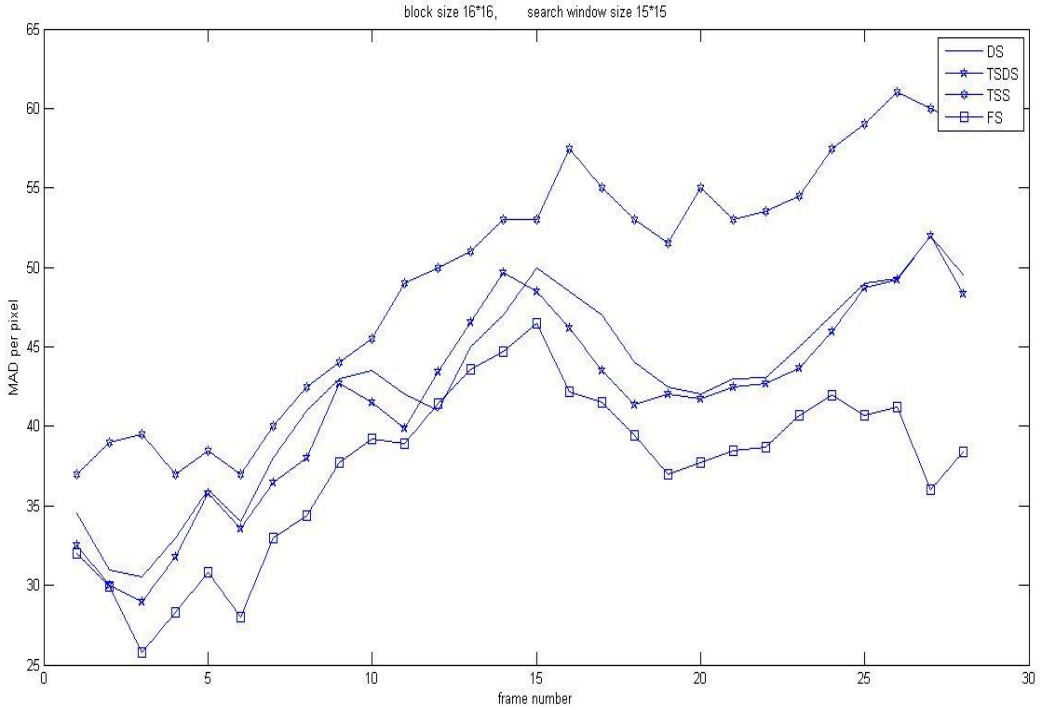


Fig.5.1.5 MAD Comparison of DS, TSDS, TSS, and FS for Crafter Sequence

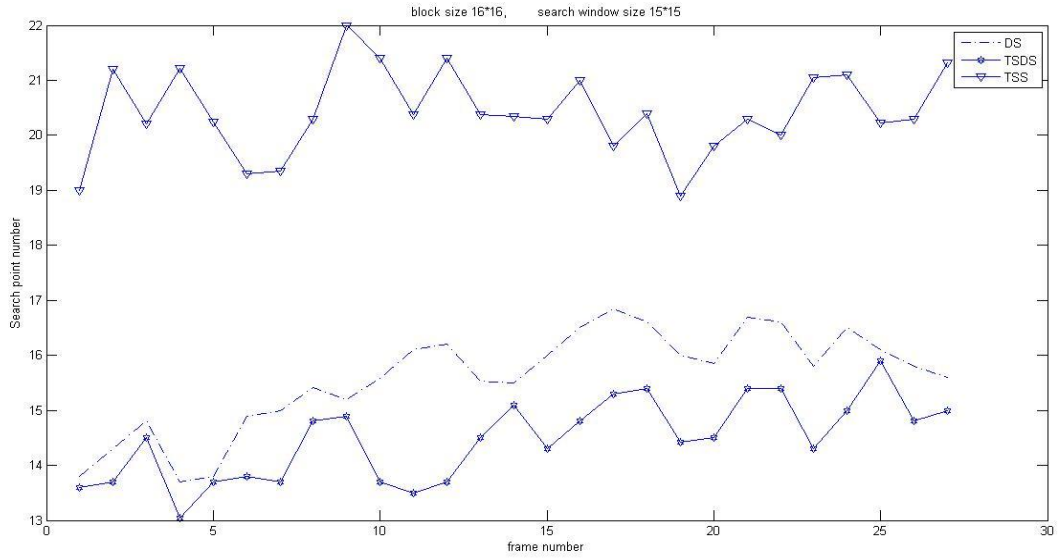


Fig.5.1.6 Comparison of Avg number of search points applying DS,TSDS and TSS toCrafter

6. Conclusion and Future Work

6.1 Conclusion

Because of the Internet is more and more universal and the technology of multimedia has been progressed, the communication of the image data is a part in life. In order to employ effect in a limit transmission bandwidth, to convey the most, high quality user information. It is necessary to have more advanced compression method in image and data. In video compression motion estimation is the most computational part and it takes the 70% to 90% complexity Motion estimation and compensation techniques, which can eliminate temporal redundancy between adjacent frames effectively, have been widely applied to popular video compression coding standards such as MPEG-2, MPEG-4 and H.264. Full search Motion Estimation algorithm is not fir for real time applications because of its unacceptable computational cost. Bidirectional ME forms a major noise in video sequences, prediction of missing data in video sequence, so the idea is a Fast Motion Estimation algorithm to improve the operation.

In this thesis, to speed up the search, a three step diamond search algorithm based different evolutionary computing techniques is proposed. The proposed bidirectional algorithm is giving less prediction error and the number of search points per each frame is less. The proposed algorithm simulation results shows minimum error and less number of search points when compared to the existing algorithms.

6.2 Future Work

These searching algorithms are limited by search speed and pattern so that very quick moments it can't find exact motion vector. So we can try different search patterns and In future other evolutionary computing techniques also can be tried for the better results. Three important factors Block size, search area, matching criteria can be varied such as Variable block size, large search area for complex motions and small search area for low complex motions and bidirectional motion estimation also we will try to implement new techniques which will further reduce the complexity of MPEG video coding.

Publications

[1] B KasiViswanath Reddy and SukadevMeher, “Three Step Diamond Search Algorithm for Fast Block Matching Motion Estimation” accepted in **ITSI International Conference onICEECP’2013** Goa Campus india.

[2] B K Viswanath Reddy and SukadevMeher, “Development of Motion Estimation Algorithms for Video Compression” has been selected for **IJAEEE Journal Publication 2013 march 22.**

REFERENCES

[1]. Y.Wang, J.Ostermann and Y.Q.Zhang. Video Processing and Communications.Tsinghua University Press and Prentice Hall, Beijing, China, 2002.

[2]. ISO/IEC 11 172-2, 'Information technology-coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s-part 2: Video', 1993 [MPEGl Video].

[3].GorpuniPawankumarandGanapatiPanda 'development of motion estimation and compensation algorithms for video compression' in digital storage media, India 2009.

[4].B.Liu and A.Zaccarin, New fast algorithms for the estimation of block motion vectors, IEEE Trans. Circuits Syst. Video technology, Vol.3, pp.440–445,Dec 1995.

[5].Tekalp Murat A. Digital video processing, Prentice Hall, PTR, 1995.

[6].I. E. G. Richardson. H.264 and MPEG-4 video compression. Wiley, Chichester,England, 2003.

[7].I. E. G. Richardson. H.264 and MPEG-4 Video Compression.John Wiley Publisher, 2003.

[8]. B. Parhami. Computer Arithmetic: Algorithms and Hardware Designs. Oxford University Press, 2000.

[9].ISO/IEC15444. Information technology - JPEG2000 imagecoding system. 2000.

[10].W.Hsuand H. Derin. Three-dimensional subbandcodingof video.Proc. Int. Conf. Acoustics, Speech, and SignalProcessing (ICASSP), pages 1100–1103, Apr. 1988.

[11].ISO/IEC14496-2. Amendment 1, Information technology coding of audio-visual objects Part 2: Visual. 2001.

[12].J.R.Jain and A.K.Jain, "Displacement measurement and its application in inter frame image coding," IEEE Trans. Commun., vol.COM-29,PP. 1799–1808, Dec.1981.

[13].S.kappagantula and K.R.Rao, "Motion compensated interframe image Prediction",IEEE Trans. Commun.vol,COM 33,PP, 1011–l015,Sept, 1985.

[14]. Iain E. G. RichardsonCopyright q 2002 John Wiley & Sons, Ltd ISBNs: 0-471-48553-5 (Hardback); 0-470-84783-2 (Electronic)

[15].I. E. G. Richardson. H.264 and MPEG-4 Video Compression. John Wiley Publisher, 2003.

[17].*"Information Technology—Generic Coding of Audio-Visual Objects"Part 2: Visual*, ISO/IEC 14 496-2 (MPEG-4 Video), 1999.

[18]. 4. ITU-T Q6/SG16 VCEG-MO8, 'Objective coding performance of [H.26L] TML 5.9 and H.263+',March 200 1.

[19]. S.Kappagantula and K.R.Rao, "Motion compensated interframe image Prediction",IEEE Trans. Commun.vol,COM 33,PP, 1011–l015,Sept, 1985.

[20].B.Liu and A.Zaccarin, New fast algorithms for the estimation of block motion vectors, IEEE Trans. Circuits Syst. Video technology, Vol.3, pp.440–445,Dec 1995.

[21].E. M. Fakhouri. Variable block-size motion estimation. citeseer.ist.psu.edu/fakhouri97vari able.html.

[22]. S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block matchingmotion estimation," in Proc. 1997 Int. Conf. Information Communication and SignalProcessing (ICICS), vol. 1, Sept. 9-12, 1997,pp.292-296.

[23].T.Koga,K.Iinuma,A.Hirano,y.lijima, and T.Ishiguro,"Motion compensated interframe coding for Video Conferencing", in proceedings of NTC81,G5.3.1-G5.3.5,1981.

[24].R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm forblock motion estimation," IEEE Trans. CircuitsSyst. Video Technol., vol. 4, pp. 438–442, Aug. 1994.

[25].L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits SystVideo Technol.,vol. 6, pp. 313–317, June 1996.

[26]. F. A. Kamangar and K. A. Rao, 'Fast algorithms for the 2-D discrete cosine transform*IE*'*E*, *E Trans. Computers,* 31(9), September 1982.

[27]. M. Ghanbari, "The cross-search algorithm for motion estimation," IEEE Trans.Commun., vol. 38, pp. 950953, July 1990.

[28]. ISO/IEC 138 18-3, 'Information technology: generic coding of moving pictures and associated au information: Audio', 1995 [MPEG2 Audio].

[29]. B KasiViswanath Reddy and SukadevMeher , " Three Step Diamond Search Algorithm for Fast Block Matching Motion Estimation" accepted in **ITSI International Conference onICEECP'2013** Goa Campus india.

[30]. S. Tsekeridou and I. Pitas, "PEG-2 error concealment based on block-matching principles', IEEE Trans. Circuits and Systems for Video Technology, June 2000.

[31] J. Xin, M.T. Sun, and V. Hsu, "Diversity-based Fast Block Motion Estimation," *Proc. IEEE International Conference on Multimedia & Expo (ICME), Baltimore, MD, USA, July 2003.*

[32]. B.K Viswanath Reddy and SukadevMeher, "Development of Motion Estimation Algorithms for Video Compression" has been selected for **IJAEEE Journal Publication 2013 march 22.**