ADAPTIVE FAST SEARCH BLOCK MOTION ESTIMATION IN VIDEO COMPRESSION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

ELECTRONIC SYSTEMS AND COMMUNICATIONS

BY

DINESH KUMAR SINGH

ROLL NO. -212EE1214

Department of Electrical Engineering

National Institute of Technology, Rourkela-769008

2014

ADAPTIVE FAST SEARCH BLOCK MOTION ESTIMATION IN VIDEO COMPRESSION

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF TECHNOLOGY

IN

ELECTRONIC SYSTEMS AND COMMUNICATIONS

BY

DINESH KUMAR SINGH

ROLL NO. - 212EE1214

Under the Guidance of

PROF. DIPTI PATRA

Department of Electrical Engineering

National Institute of Technology, Rourkela-769008

2014

National Institute Of Technology, Rourkela

CERTIFICATE

This is to certify that the thesis entitled "**Adaptive Fast Search Block Motion Estimation In Video Compression"** submitted by **MR. DINESH KUMAR SINGH** in partial fulfillment of the requirements for the award of Master of Technology Degree in **Electrical Engineering** with specialization in **"ELECTRONIC SYSTEMS AND COMMUNICATION"** at National Institute of Technology, Rourkela is an authentic work carried out by him under my supervision and guidance.

To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Date: **Prof. Dipti Patra**

Department of Electrical Engineering

National Institute of Technology

Rourkela

ACKNOWLEDGEMENT

Firstly, I would like to thank Prof. Dipti Patra, Department of Electrical Engineering, National Institute of Technology, Rourkela as my guide, who encouraged and challenged me throughout my project work and guided me on writing conference papers, thesis and to put effort on our workings. She is a perfect motivator and I am glad to have guide like her.

After that, I would like to thank, Prof. Anup Kumar Panda, head of department, electrical engineering, National Institute of Technology, Rourkela and professor of my specialization Prof. Prasanna Kumar Sahu, Prof. Susmita Das, Dr. Supratim Gupta and Prof. K. Ratna Subhashini.

I would also like thanks my fellow lab mates in Image Processing & Computer Vision Lab: Guguloth Uma, Rohit Kumar, Smita Pradhan, Rajashree Nayak and Yogananda Patnaik for their help and support.

Dinesh Kumar Singh

Abstract

With the advancement of telecommunication technologies, such as internet, video conferencing and HDTV, we need an effective video compression technique. Motion estimation and motion compensation are the most complicated and time consuming part of any video coding technique. Motion estimation helps to reduce temporal redundancy that exists between successive video frames. The motion estimation part of any video codec should be such that, it can reduce computational complexity without having any effect on the quality of the video. The motion estimation process can be more efficient if we use spatial and temporal correlation between the blocks in a frame and between two consecutive frames.

In this thesis, a new search method for block motion estimation in video has been presented that uses neighbouring blocks of current macro-block and the block in the previous frame having the same coordinates as that of current macro-block for prediction of motion vectors. In the proposed method we use the motion vectors of neighbouring blocks that are more likely to be helpful in the prediction process. By using these motion vectors a search centre is located, around which a search window is placed. In this thesis, we have introduced Sorted Search Method (SSM) algorithm for motion estimation and compared the performance with existing Block Based Motion Estimation (BBME) techniques. Different sorting search methods have been developed by taking different neighbours around the current macro - block and their performances are compared.

Contents

List of figures

List of tables

CHAPTER 1

1. Introduction

1.1 Video Coding

Video coding is a very important application of video processing. It is applied to lessen the video sequence's data rate so that it is viable to send video in actual time through any channel. In addition to communication applications, video coding is too necessary for storage and recovery application, where different storage media have different capability and admission rates, thus demanding for varying amount of compression. Due to a wide range of data rates, different algorithms have been adopted. Video compression is a very important prerequisite for storage and low bandwidth transmission of digital video signal. The conception and selection of a video encoder is not solely depends on its power to compress information. There are many other issues that are considered such as algorithm complexity, computation time, bit rate versus distortion criteria, transmission channel characteristics.

1.1.1 Methods for video compression

The frame of a video have rectangular shapes, hence block based motion estimation is a good choice. However, there are different other methods. wavelet coding is an example of that, which consist in image encoding in JPEG2000 [1]. Wavelet compression based video coding [2] can provide better visual quality, but at cost of higher computational complexity.

MPEG-4 [3] is another method in which arbitrary shaped coding is utilized, this is founded on different moving objects whose motion is compounded to make a full frame. Thus, performance gets improved by predicting motion estimation more accurately, but computational complexity also increases.

1.1.2 Video encoder

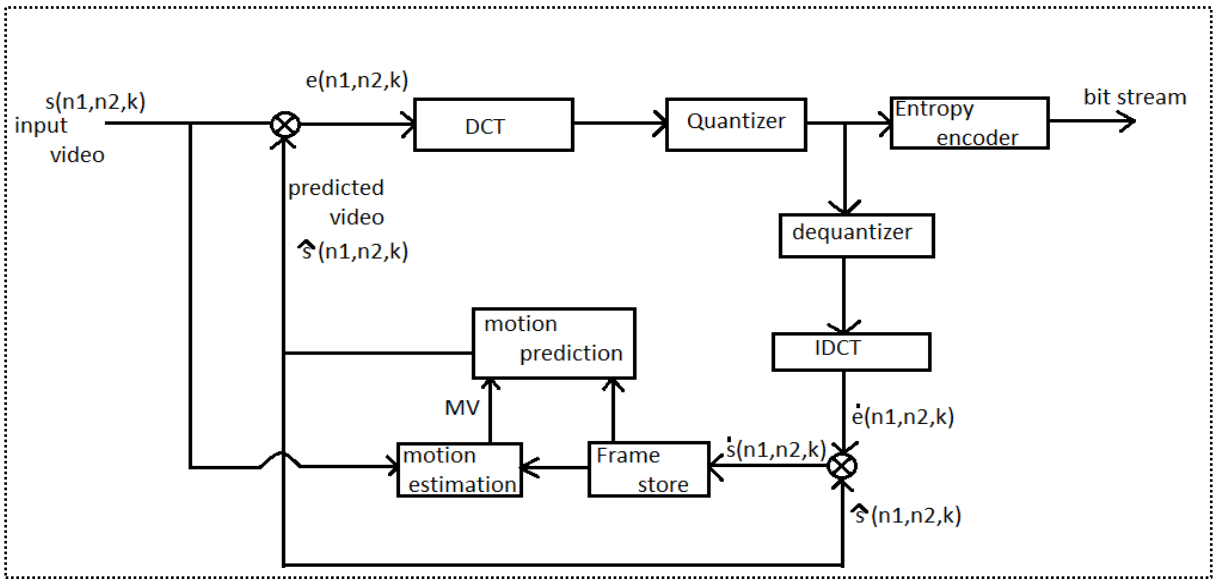


Figure1.1 Block diagram of video encoder

The data in a video signals are in three dimensions. For video encoding purposes, these dimensions can be modelled in spatial and temporal domain. Digital video compression technique minimizes information redundancy in each domain independently. In figure 1.1 a block diagram of basic video encoder is shown. There are two modes of operation: the interframe mode and the intraframe mode. In interframe video mode operation the feedback loop is applied to get a prediction error between the image blocks of the current frame and the current prediction frame. Motion estimation block creates the motion vector for each 8×8

block. The motion vectors and previously reconstructed frame are used to generate a prediction frame with the help of motion predictor block.

In intraframe mode of operation the system does not receive an input from the feedback loop. Input video frames (n1, n2, k) are spatially coded, and subsequently encoded by the entropy encoder to obtain bit stream which will be send to the decoder side. The first frame in the video sequence will be always intra-coded, and all subsequent frames will be inter-coded.

1.2 Motion estimation in video

Motion estimator block is the most popular block among all the blocks in a video encoder, it the most critical part of the encoder that affects the compression efficiency and video quality. In the last few decades, many algorithms have been made up to optimize the motion estimator part of the video codec. With improvements in video codec standards, there is a need of better motion estimation technique, then before, thus both hardware and software optimization must be continually improved to handle the increased complexity.

1.2.1 Block matching

Block matching is a process of finding a matching block from frame *i* in some other frame *j*. Block matching algorithms (BMA) make use of an evaluation metric to determine whether a given block in the *i* frame matches the search block in the *j* frame.

Several evaluation metrics can be applied.

- Mean square error (MSE)
- Mean absolute error (MAE)
- Sum of absolute error (SAE)

The size of the blocks used is generally of the order 16×16, 8×8 and sometimes-even 4×4.

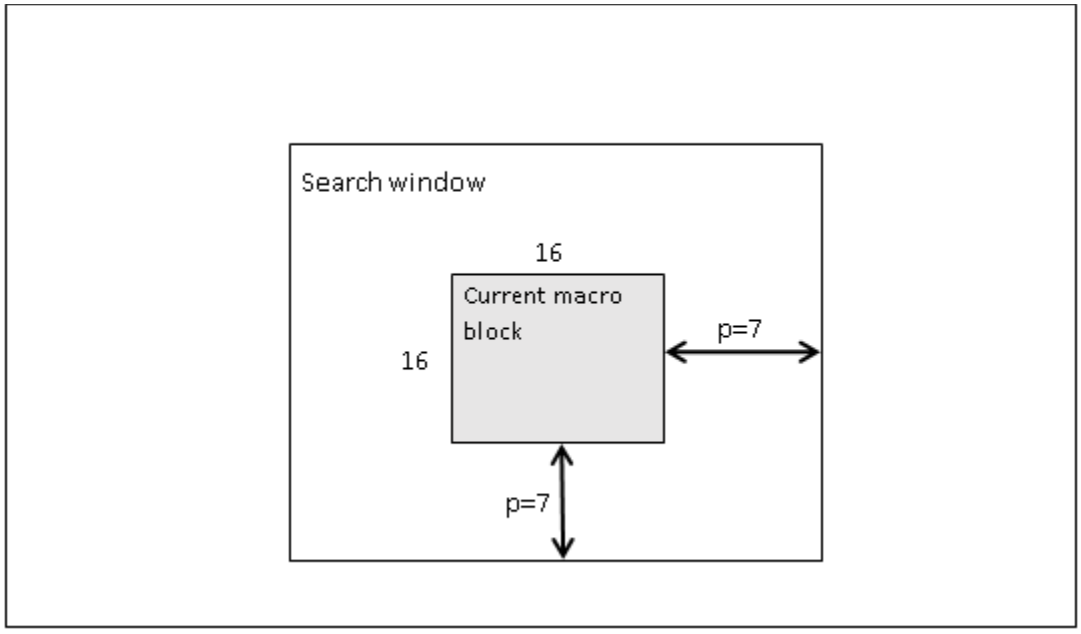


Figure 1.2 Process of block matching using p=7, current macro-block of size 16×16, and a search window

In figure 1.2, a search window is shown, in which a macro block of size 16×16 is chosen. The search area of the macro block is constrained upto seven pixels in all directions, The macro block moves within the search window and calculate cost function at each location. This p is known as the search parameter. The size of p depends on the size of the object, for a larger object we require a larger p, but as the size of search parameters increases the motion estimation process becomes computationally more expensive.

1.3 Motivation

In daily life everyone deals with different types of videos like television video signals, internet videos and many more. Since a natural video have a very huge size, so storage and transmission of video signals is a challenge. We need a video compression technique that can

compress our video without any loss of information. For that we will do video encoding. In the last few decades lots of research has been done to design an efficient video codec that can compress our information without any loss. There are different video codec developed so far, starting with H.261, MPEG-1, MPEG-2/ H.262, H.263, H.264, MPEG-4. The work is going on to develop MPEG-7 and MPEG-21, that uses meta data and video search.

1.3.1 Video Compression

The uncompressed raw video contains an immense quantity of data and our communication and memory capacities are costly and limited. For example [4], consider the amount of data required to represent a two-hour standard definition (SD) television movie using 720*480*24 bits pixel arrays. A digital movie or video is a sequence of video frames in which each frame is a full colour still image. Also video frames must be displayed sequentially at a rate of 30 fps (frame per second), therefore SD digital video must be accessed at

$$30\frac{frames}{sec}\ \text{x}\ (720\ \text{x}\ 480)\frac{pixels}{frame}\text{x}\ 3\frac{bytes}{pixel} = 31{,}104{,}000\ bytes/sec$$

and a two-hour movie consist of 224 GB (gigabytes) of data. Twenty seven 8.5 GB dual player DVDs are needed to store it. To put a two hour movie on a single DVD, each frame must be compressed. In two consecutive frames of a video sequence, there is a lot of redundancy both in spatial and temporal domain. We can accomplish a significant measure of compaction by reducing these redundancies.

1.4 Literature survey

Many algorithms for motion estimation have been offered in the past, depending upon the search type motion estimation can be split into two cases.

1) Block based motion estimation.

2) Pixel based motion estimation.

Iain E. G. Richradson [5]. Block based motion estimation uses Block Matching Algorithm (BMA) for finding the motion vector and it is easy to implement and is very efficient. In block based motion estimation each frame of a video sequence is divided into non overlapping blocks of sizes 16×16, 8×8 and even 4×4, then for each block in the current frame best match is searched in the reference frame, for each block a motion vector is calculated.

Yoa Wang, Joern Ostermann and ya –Qin Zang [6]. In pixel based motion estimation, motion vector for every pixel in the image is determined. But the problem is that, suppose one uses the constant intensity assumption, for every pixel in the current frame, there are many pixels in the reference frame that have exactly the same intensity. If one uses optical flow equation, the problem is again indeterminate, because there are two unknown and there is only one equation. To overcome this problem, there are a few approaches. We can add some smoothness constraints on the motion field, or one can assume the motion vectors in a neighbourhood surrounding each pixel are the same, and apply the constant intensity assumption or the optical flow equation over the entire neighbourhood.

In **Nasir D. Memon, Khalid Sayood [7],** they have researched to achieve lossless compression of video sequences. In their paper they used both spatial correlation and temporal correlation to design an design an adaptive video compression technique.

Avijit Kundu [8], has developed a modified diamond search algorithm for motion estimation, a reviewed different existing block matching techniques such as Full Search (FS) algorithm, Three Step Search (TSS), Adaptive Dual Cross Diamond Hexagonal Search.

Xiaolin Chen, Nishan Canagarajah [9], it describes a lossless video compression scheme that uses backward adaptive pixel- pixel based fast predictive motion estimation. This theme does not use block motion estimation scheme, it searches for the motion vector by searching pixel by pixels. This method gives good result as compared to Full Search (FS) in terms of computational speed.

1.5 Thesis objective

The aim of the thesis is as follows:

- To design a block matching technique that can adaptively search for a more accurate first search point to estimate the motion vector for video compression, which helps in performance improvement of motion estimation and significant reduction of total number of search points, used to establish the motion vector of current block.
- To develop a fast adaptive search block matching algorithm that utilizes both temporal and spatial correlated blocks to identify a better first point search, as the neighbouring blocks of the same frame are spatially correlated and temporally correlated with blocks having the same coordinates in the previous frame.

- Performance comparison of different block matching techniques, such as Full Search (FS), Three Step Search (TSS), Four Step Search (FSS), Diamond Search (DS) and Adaptive Rood Pattern Search (ARPS) with our proposed block matching algorithm.

1.6 Thesis contribution

- Study and comparison of different block matching techniques in terms of PSNR (dB) and computational time.
- An adaptive block matching technique, Sorted Search Method (SSM) algorithm has been developed for motion estimation.
- The different sorting search methods, i.e. sorted5, sorted4, sorted4a, sorted3, sorted3a and sorted3b are also developed and compared by taking different neighbourhood combinations.

CHAPTER 2

2. Background theory

2.1 Motion estimation

A video is formed when sequence images are passed at a rate of more than 30 frames per second. Motion estimation is a process of determining the motion vectors. Motion vector describes the motion of a block in the candidate frame with respect to the reference frame.

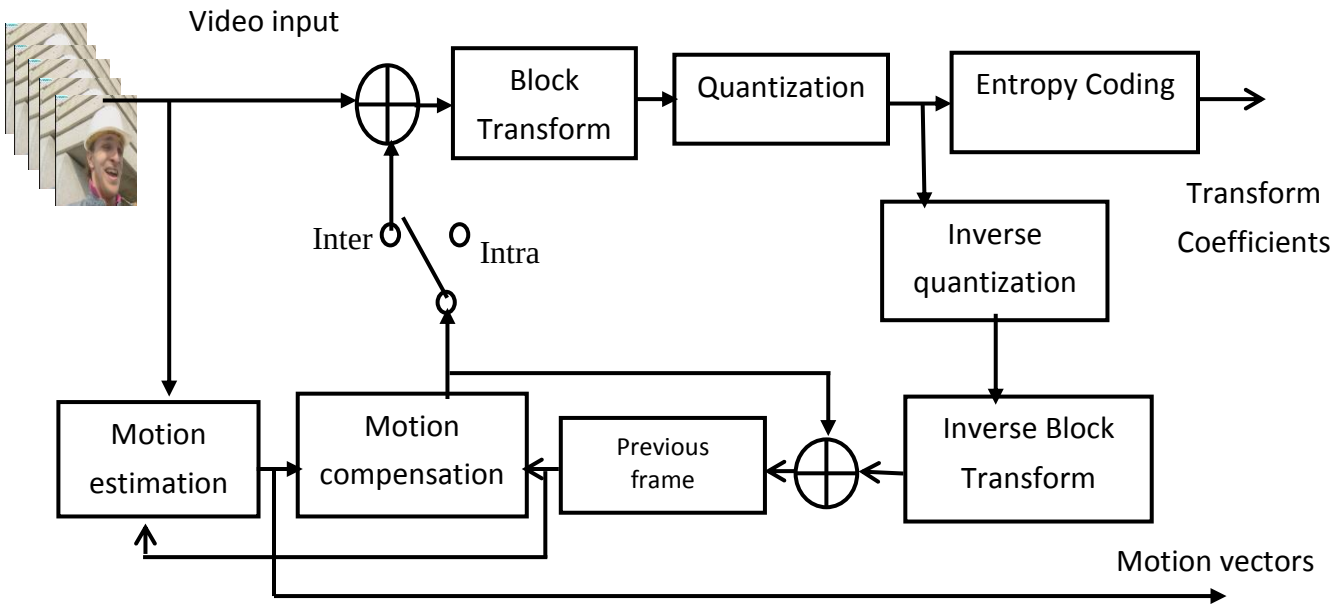


Figure 2.1 motion compensated video coding

In two consecutive frames, as the object moves, the information content of the frame changes. By using an appropriate model, the motion between two frames or the movement of objects in the frames can be estimated. This procedure of estimating motion in a frame with the aid of a reference frame (generally a past frame) is called motion estimation (ME) [10]. This model is then used by the encoder to predict the next frame using the past frame. This process is known as motion compensation (MC). In figure 2.1, block diagram of a motion compensated coding system is shown. Here both inter-frame and intra frame coding techniques are shown.

2.1.1 Backward motion estimation

Backward motion estimation, is most commonly used motion estimation technique. In backward motion estimation current frame is employed as the candidate frame, and in the past frame best match is searched that gives the motion vector. This method is also known as forward motion prediction.

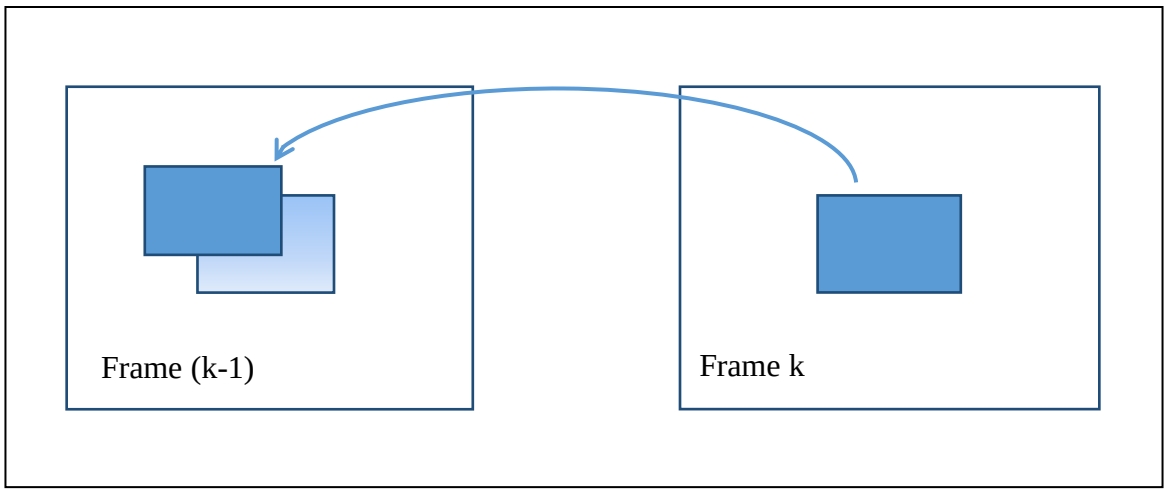


Figure 2.2 Backward motion estimation: frame (k-1) as past frame or reference frame and frame k as current frame

2.1.2 Forward motion estimation

It is exactly the reverse of backward motion estimation. In this the current frame is our candidate frame and future frame acts as our reference frame, which means current frame is predicted from a future frame. Or we can say search is forward. And this process is also known as backward motion prediction.

Predicting the current frame from a future frame seems very unusual. Actually, there is nothing unusual in it. It can be understood with a simple example, suppose we have a group of pictures (GOP), let a GOP consist of 13 frames, the first frame of a GOP is always intra coded, now we can predict subsequent frame using the first frame as the reference frame. Suppose instead of predicting the second frame we have predicted the fifth frame, now if we predict fourth frame using fifth frame as our reference frame, then this type of estimation is called forward motion estimation. Also, if second frame is estimated using both first frame and fifth frame as a reference frame, this type of estimation is known as bi-directional motion estimation.

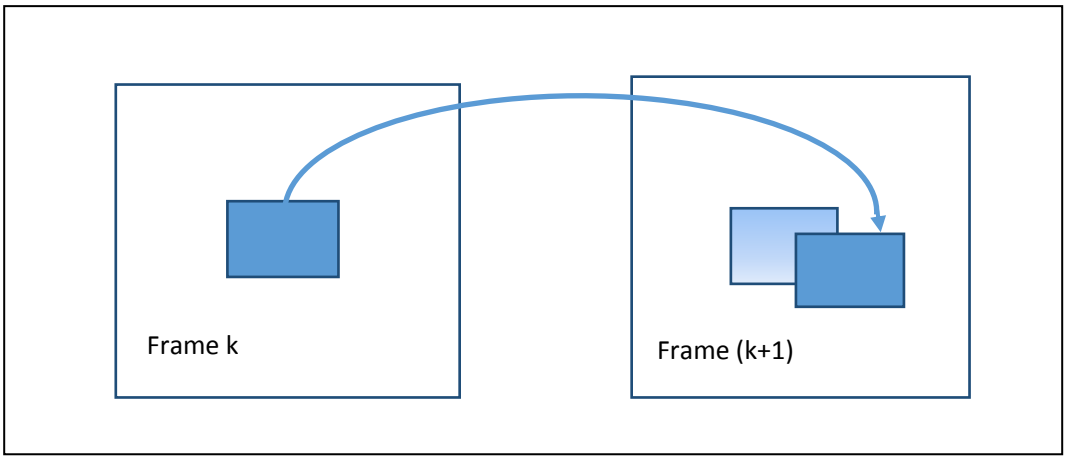


Figure 2.3 Forward motion estimation : frame (k+1) is future frame or reference frame and frame k as the current frame

In MPEG 1 and MPEG 2 standards we use Forward motion estimation (or backward motion compensation). These standards also support bi-directional motion estimation, in bi-directional motion estimation current frame is predicted with the help of a past frame (k-1) and a future frame (k+1).

2.2 Block Based motion estimation

Block based motion estimation (BBME) is the most popular matching technique used for motion estimation in video coding. Since frames of a video generally have a rectangular shape, it can easily be divided into blocks. MPEG is a standard body that decides the standard block size, that is used in the matching process and is taken as 24 × 24, 16 ×1 6, 8 × 8 depending upon the type of motion. MPEG-4 or H.264/AVC is a latest coding standard. It supports variable block sizes which can be 4 × 4, 8 × 8 and 16 × 16. Motion estimation predict future frame using current frame and generates motion vectors (MVs) as accurate as possible.

The quality of prediction depends on the size of the block used. For example the quality of prediction for a block size of 4 × 4 will be batter compared to a block size of 8 × 8 and 16 × 16 but at the same time number of search points associated also increases.

2.2.1 Motion estimation procedure

We get a picture residue and a set of motion vectors as a result after motion estimation. For each block (16x16, 8x8 or 4x4) in the current frame the following procedure is executed.

1. A search window is taken In the reference frame, for each block in the current frame. The size of the search window is two to three times the size of the macro block (16x16). As facts say the motion between two consecutive frames is statistically very

less, therefore the search range is confined to this area. Within a search window a best match is found using a search process. Matching process searches for the block in the search window that gives minimum distortion (that minimizes mean square error) with respect to the macro-block in the candidate frame. This process of searching best match block by block is known as BBME.

2. Once we got the best match, the motion vectors and residues between the reference block and current block are calculated. The operation of determining the motion vectors and residues is called motion compensation.
3. The motion vectors and residues of best match are encoded and transmitted to the decoder.
4. At decoder side, a reverse process is applied to rebuild the original image.

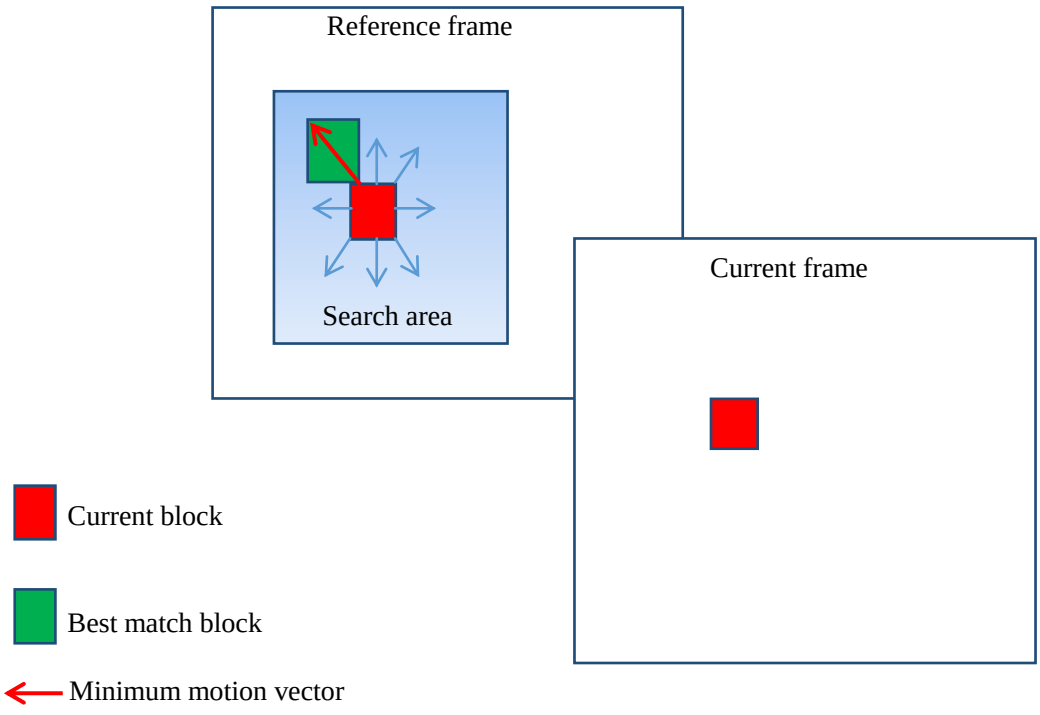


Figure 2.4 Motion estimation and motion vector

Above process is systematically explained in the figure 2.4. In latest video coding standards, the reference frame can be a future frame, a past frame or a combination of two or more previously coded frames. The desired accuracy depends on the number of reference frames. The accuracy of prediction depends upon the number of reference frames used, if the number of reference frame used for prediction is large, the prediction will be more accurate.

2.3 Different algorithms for Block Based Motion Estimation

Block Based Motion Estimation (BBME) algorithm is method of finding motion vectors that corresponds to the block showing minimum mean square error in the matching process. Different algorithms can generate different motion vectors. Full search gives best result among all the proposed algorithms. By using other algorithms we can get result near to optimal, but at very low computational cost with respect to full search. Here, we will study different BBME algorithms that are highly utilized in present coding standards.

2.3.1 Full search (FS)

within the search window, the FS algorithm gives a global optimal motion vector from all the candidate blocks. If the size of the search window is 48 × 48 pixels and the block size is 16 × 16 pixels, there will be total of 16 ×16 = 1024 search points that need to undergo mean absolute error (MAE) computation.

FS algorithm exhaustively searches all the search points within the search window, therefore it is very simple to implement but it has very high computational time. It can be hardware implemented by using multiple structures in parallel.

As shown in figure 2.5, a block of N × N pixels in the reference frame at the coordinate location (m,n), and consider a range of ±w pixel of the search window in both directions on

the reference frame. The candidate block is compared to a block of size N × N pixels for each of the $(2w + 1)^2$ search.

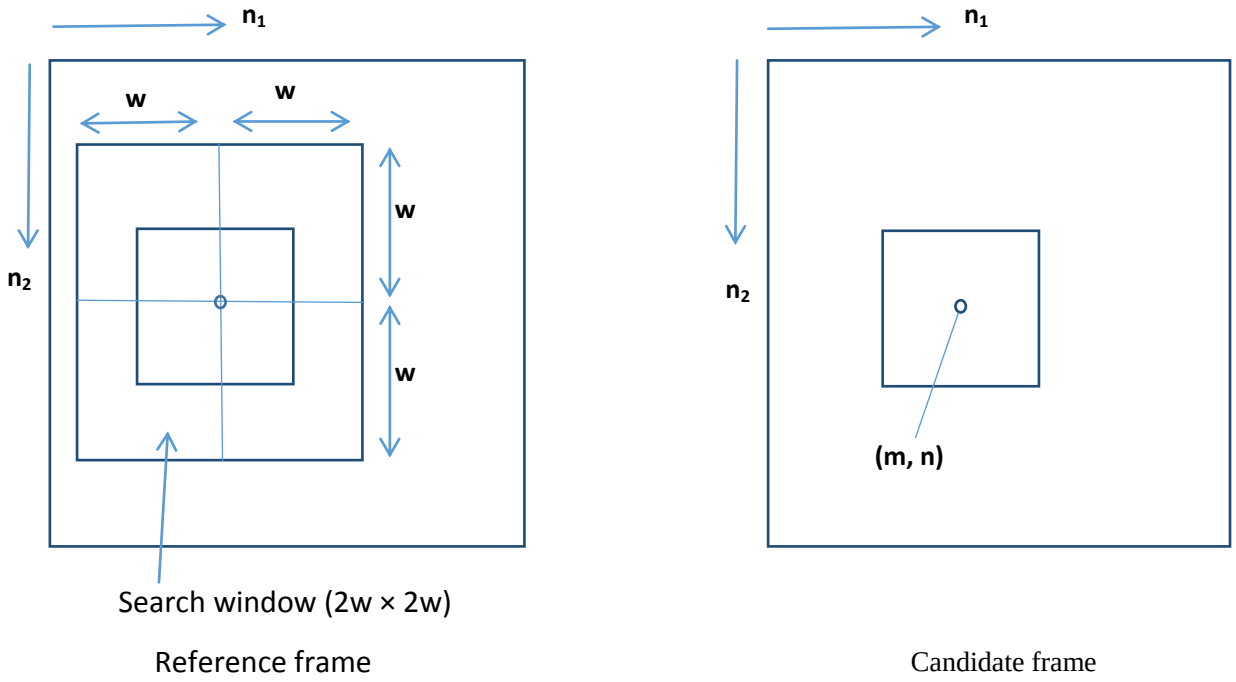


Figure 2.5 Full search motion estimation

2.3.2 Two-dimensional logarithmic search

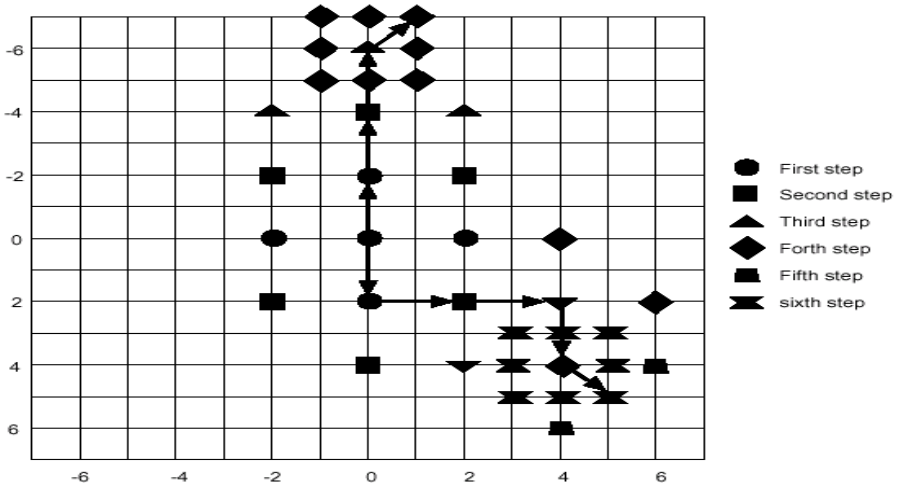


Figure 2.6 2D-logarithmic search algorithm

2D-log search algorithm was suggested by Jain et. al. in 1981. In this method we use a cross search pattern. Initially step size is taken as d/4. If the minimum SAE point is at center or at the boundary, in that case the step size is reduced to half of the current step size. Otherwise, the step size remains the constant. When the step size is equal to 1, all the point around the current minimum distortion point are searched. In figure 2.6 two different cases are shown. The top search path involves (5 +3 +3 +8) = 19 search points. The bottom search path involves (5+3+2+3+2+8) =23 search points.

2.3.3 Three step search (TSS)

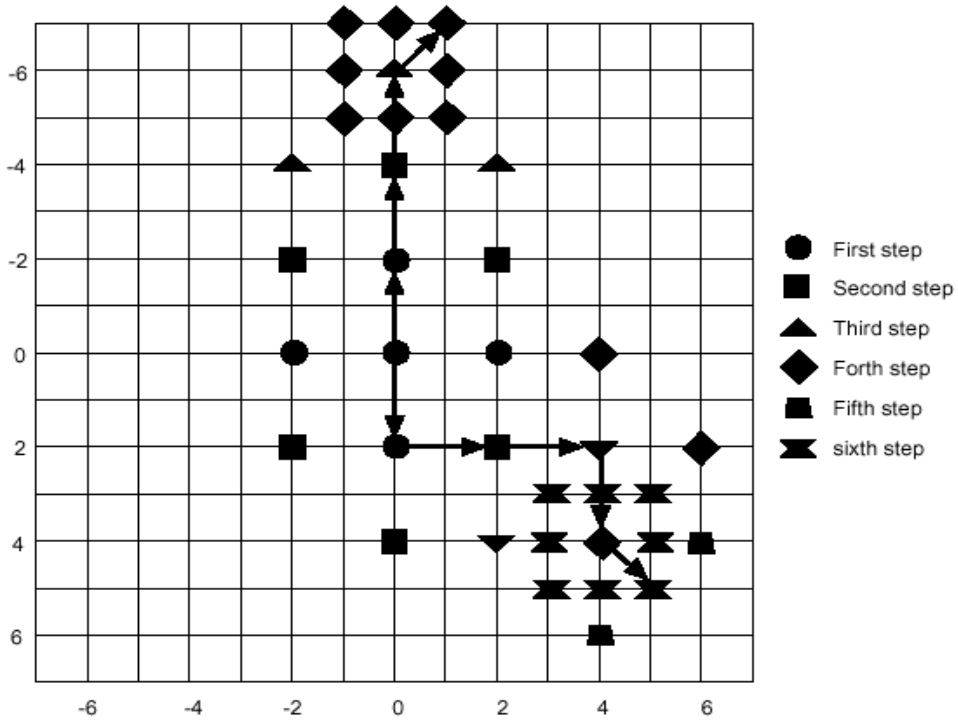


Figure 2.6 Three step search algorithm

The three-step search algorithm (TSS) was introduced by Koga et. al. in 1981 [11]. In this method step size is decremented to half after each step. If "d" is the maximum motion displacement, then the initial step size will be half of that, which is d/2.

At every step, 9 search points are matched among them the one showing minimum error is is selected as our search center for the next step. For example, if d = 7, the number of search points required is (9 + 8 + 8) = 25. For larger search window , 3SS can be easily extended to n-steps using the same searching strategy with the routine of such points required equals to [1 + 8 log2 (d + 1)].

2.3.4 Four step search (4SS)

The four-step search algorithm (4SS) was introduced by L.M. Po and W. C. Main in 1996 [12]. This algorithm is similar to three step search but uses a smaller initial step size. The initial step size for four step search algorithm is one fourth of the maximum motion displacement, which is (d/4). As the initial step size of four step search is smaller (d/4), it reaches the boundary of the search window in four steps for d=7. The four step search algorithm may complete its search in second or third step search that will save some computation time. Two search paths are shown for four step search in figure 2.6. These two paths are the worst case example of four step search, here for top-right, 27 searching points are required, and for bottom left part it takes 25 searching points.

For small motion two search paths of four step search can be analyzed. In best scenario four step search requires only 17 searching points if the best match of the first step is not at the corner otherwise it requires 20 searching points.as we can see there are just three or five search points required in second search step. Also, if minimum distortion point is at center we can jump directly to step four. In worst scenario the number of search points required are (18 log2 [(d+1)/4] + 9).

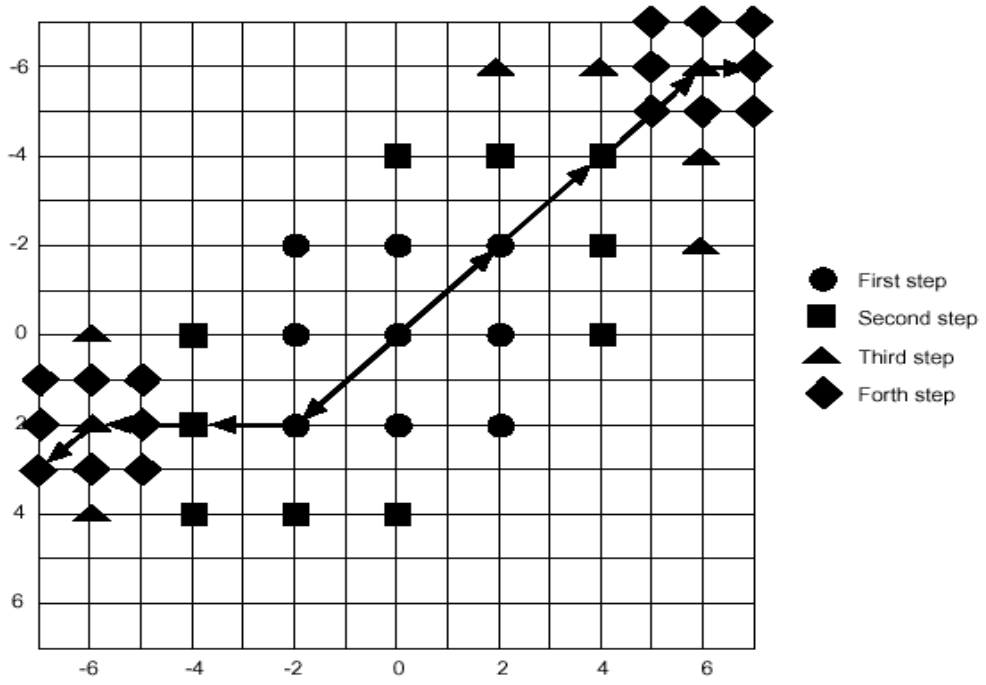


Figure 2.7 Four Step Search algorithm

2.3.5 Diamond search (DS)

diamond search and Four step search [13] are almost similar, except that the search pattern used in diamond search is of diamond pattern, and the number of search steps is not fixed like 3SS or 4SS. It uses two types of search patters, (1) SDSP, stands for Small Diamond Search Pattern and (2) LDSP, stands for Large Diamond Search Pattern, as it is shown in figure 2.7.

First step of a diamond search is always LDSP as in 4SS, and if the minimum distortion is at center, then we jump to step four which is SDSP. The subsequent steps are also similar and use LDSP. As we can see in the diagram, the number of search points for second and third steps are either 3 or 5, which depends upon the location of minimum distortion point of the last step. If the minimum distortion point is at the corner than there will be 5 search points

otherwise it will have 3 search points. If the best match of any of the step is at center then we directly jump to last step, which is SDSP. Diamond search gives very accurate motion vector. Thus it gives very good PSNR (close to full search), and at a less computational cost.

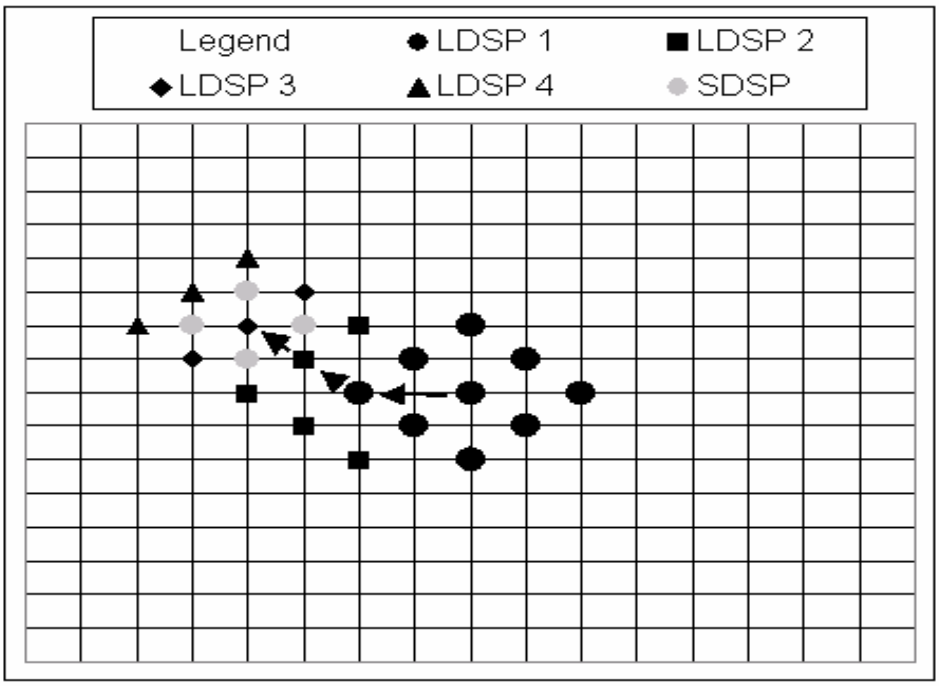


Figure 2.8 Diamond Search

2.3.6 Adaptive rood pattern search

In a frame, the motion vector of current block and its neighbouring blocks are generally coherent, that is, there is very high probability that motion vector of current macro-block and its neighbouring macro-block will have similar motion vector. Adaptive Rood Pattern Search [14] (ARPS) algorithm uses this property to start its search. To find the motion vector of current macro-block, the motion vector of neighbouring block just immediate left to the current macro-block is used, as shown in figure 2.8. As we seen from the figure the coordinates of predicted motion vector is (3,-2) that will be our first search point, and then

rood pattern distributed points will be checked where they are at a step size of S = Max (|X|, |Y|). X and Y are the x-coordinate and y-coordinate of the predicted motion vector. Using rood pattern search as the first step puts the search area in global minimum, where SDSP can be used for further matching. It reduces number of search points to a great extent with a PSNR comparable to FSA.

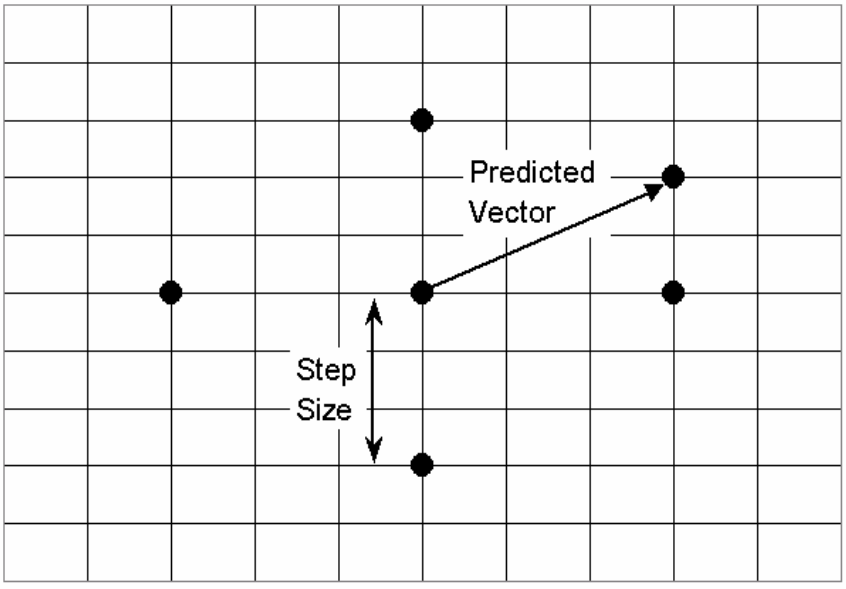


Figure 2.9 Adaptive Rood Pattern Search

CHAPTER 3

3. Adaptive block motion estimation

3.1 Introduction

In any type of block matching algorithm, the first step is dividing the video frame into non-overlapping blocks. After that, for each block in the candidate frame, a search window is designed in the reference frame. Then the process if block matching is done, that is by matching every block in candidate frame with reference frame, to obtain a motion vector for each block in the candidate frame.

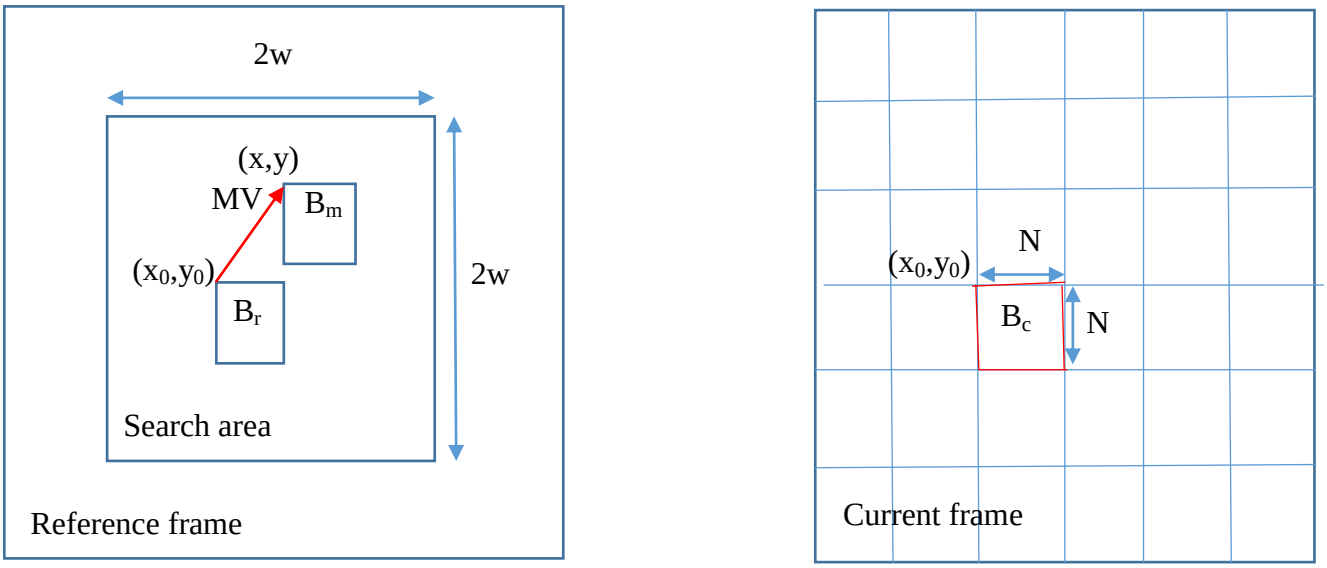


Figure 3.1 Block matching process showing current frame, reference frame and MV

In above figure 3.1, shows the process of block matching. Here, B_c, is a block in the current frame, which is to be matched in the reference frame, B_r is the block corresponding to B_c in the reference frame. The coordinates of B_r and B_c is same, which is (x_0,y_0). Around B_r we will design a search window, and within this search area we will search for the best match for B_c. suppose the best match for B_c is B_m whose motion vector is given by (x,y). The best match for B_c in the reference frame is one that minimizes the following matching criteria.

$$E(MV) = \left(\frac{1}{N^2}\right)^{\beta} \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} \left| I_c(x_0 + m, y_0 + m) - I_r(x + m, y + n) \right|^{\gamma}$$

Where $MV = (x - x_0, y - y_0)$ (3.1)

Where *E (MV)* is the distortion or error produced while matching block Bc in the current frame with the block in the reference frame whose motion vector is *MV*. Motion vector is defined as the displacement of the best match (B_m) with respect to B_r. In above equation 3.1 I_c and I_r, are the intensities of the pixels in the current and reference frame, and N is the dimension of the blocks (size of each block is $N \times N$). Here, value of β and γ will decide the type of matching criteria. For β = 0, γ = 1, the criteria is Sum of Absolute Error (SAE), for β = 1, γ = 1, it is Mean Absolute Error (MAE) and for β = 1, γ = 2 it is Mean Square Error (MSE). There are different motion estimation techniques to obtain MV, we have already discussed many block matching techniques, but all of these techniques have a fixed search pattern and are computationally expensive. We can save a lot of computation effort without losing the quality of the video by selecting an appropriate adaptive block matching technique which can take judgement on the motion vectors of neighbouring blocks that we already know. By using

the motion vectors of neighbouring blocks we can select an appropriate search centre from there we start our matching process that will save a lot of computation and time.

3.2 Correlation among motion vectors

In figure 3.2, we have shown the current block $B_{c,}$ and its neighboring blocks B_1, B_2, B_3 and B_4. There is high spatial correlation among these blocks for which different combinations of these blocks have been selected. Also block B_c is temporally correlated with B_5 as shown in figure 3.1. If we use MVs of these blocks ($B_{c,}$ B_r and blocks B_1 to B_4) to predict MV of current block, we can save a large amount of computation time. The MV of B_1 to B_4 and B_r or B_5 can be described as $MV_i = (MVX_i, MVY_i)$ where MVX_i and MVY_i are the displacements in the horizontal and vertical direction relative to its position in the reference frame.

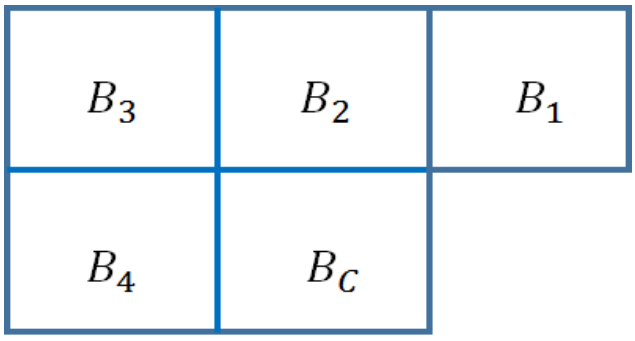


Figure 3.2 current block Bc and its neighbouring block B_1, B_2, B_3 and B_4

These six blocks mentioned above are compared with the current block using error function (shown in equation 3.1). After that each block is assigned with indices, the assignment of indices is based on the amount of distortion using equation 3.2. The block producing the least distortion is named as I_1, similarly the one producing maximum distortion

is named as I_6. For example, if the MV of B_2 results into minimum distortion then I_1=3 and if MV of B_4 results into maximum distortion then I_6 = 4, and so on.

$$I_k = \begin{cases} \underset{j}{arg\,min}\left\{E\left(MV_j\right)\right\} & k=1 \\ \underset{j \notin \left\{I_1,\ldots\ldots,I_{k-1}\right\}}{arg\,min}\left\{E\left(MV_j\right)\right\} & k \neq 1 \end{cases} \quad (3.2)$$

The event $A_{k,d} = \left\{\left\|MV_c - MV_{I_K}\right\| \leq d\sqrt{2}\right\}$ consists of the outcome that the distance between MV_c and MV_{I_K} vectors are less than or equal to $d\sqrt{2}$ pixels. Thus $p_{1,d} = P\left(A_{1,d}\right)$ is the probability that the distance between the MV_c and the vector that produces the minimum distortion is less than or equal to $d\sqrt{2}$ pixels. In other words it defines the probability of the point (MVX_C, MVY_C) falling within a circle of radius $d\sqrt{2}$ pixels centred at a point $\left(MVX_{I_1}, MVY_{I_1}\right)$. In general, we can define $P_{k,d}$ as shown in equation 3.3.

$$P_{k,d} = \begin{cases} P\left(A_1, d\right) & k=1 \\ P\left(A_{k,d} \left| \cap_{i-1}^{k-1} \overline{A}_{i,d} \right.\right) & k \neq 1 \end{cases} \quad (3.3)$$

Using equation 3.3, we have calculated $P_{k,d}$ for different values of k and d. The result shown in table 1 is performed on different video sequence like caltrain (CIF), foreman (CIF), Susie (SIF). It can be observed from the table that, with the higher value of "d", the probability of finding the accurate MV increases. We can also observe that, with increase in number of prediction vectors (k), for a search radius of $d\sqrt{2}$, has very little effect on the correct prediction of motion vector. From the above results and discussion it can be conclude that, predicting motion vector of a block in the candidate frame can be done using the motion

vectors of its neighboring blocks B_1, B_2, B_3 and B_4 (shown in figure 3.2), and a block in the reference frame B_r.

Table 3.1 Probability $P_{k,d}$ with different values of k and d

$P_{k,d}$	d=1	d=2	d=3	d=4	d=5
k=1	0.829	0.864	0.889	0.917	0.928
k=2	0.18	0.2	0.21	0.22	0.24
k=3	0.06	0.06	0.06	0.08	0.08
k=4	0.04	0.04	0.05	0.05	0.08
k=5	0.02	0.03	0.03	0.05	0.06
k=6	0.02	0.01	0.03	0.04	0.02

From the above table it is clear that even for small values of "d" the motion vector of a block is close to the MV of the neighboring block showing minimum distortion. In the above table d=1 corresponds to first row, and its shows the probability that the distance between actual motion vector of the block and the motion vector of the block showing least distortion, that is MV_{I_1} is less than or equal to $d\sqrt{2}$. Similarly for second row shows the probability that distance between actual motion vector and MV_{I_2} is less than or equal to $d\sqrt{2}$ if the distance between true motion vector is greater than$d\sqrt{2}$.

3.3 Main ideas for proposed algorithm

In the last chapter we have studied different block based search algorithms. Here an attempt has been made to improve the performance of the proposed algorithm in terms of computational complexity, computational time. We have developed an algorithm that can identify a more appropriate first search point instead of using a fixed search pattern. Starting the search from a more accurate first search point, more accurate motion vector can be predicted within less number of search points and it is as efficient as full search algorithm.

3.3.1 Consistent directivity of motion vector

It can be observed that in general videos the moving objects mainly possess translational motion [15]. We can use this as an advantage to predict first search point in our motion estimation process. If we are able to predict first search point near global minimum, the chance of predicting accurate motion vector increases. Prediction of first search point can be done with the help of motion vector from previous blocks. By using an appropriate first point search the performance of motion estimation can be increased at the same time number of search points required to find motion vector can be highly reduced.

The first search point is calculated using the motion vector of the block with same coordinates in the previous frames. From the motion vector of the same coordinate block of the previous frame, the direction of the current block will be decided. Using equation 3.4 and 3.5 angle of the motion vector can be calculated.

$$Radian = a\sin\left(\frac{|y|}{\sqrt{x^2+y^2}}\right) \qquad 3.4$$

$$MVAngle = radian \times 180 \quad 3.5$$

Where, x and y are the components ofmotion vector in horizontal and vertical directions respectively. The MVAngle calculated in equation 3.5 is not the final angle, the final motion vector angle is calculated using table 2.

Table 3.2 Calculation of final motion vector angle

Position of motion vector	**Final angle**
First aspect	$FMVAngle = MVAngle$
second aspect	$FMVAngle = 180 - MVAngle$
third aspect	$FMVAngle = 180 + MVAngle$
Forth aspect	$FMVAngle = 360 - MVAngle$

After calculating the angle of final motion vector angle (*FMVAngle*), the direction the motion vector is defined according to table 3. From table 4 direction of motion vector is defined, if the angle of motion vector of a block falls in a particular range, a certain direction is assigned to the motion vector of that block. For example if the angle is $22.5 \leq FMVAngle < 67.5$, “direction 2”, will be assigned. The result of above operation is shown in table 4, this operation is performed on 396 blocks of different video sequences, and it can be observed that for more than 80% times the direction of the current block is in the direction of the motion vector of the block having same coordinates in the previous frame. Hence, we can say, the use of the blocks in the previous frame can provide an accurate first search point.

Table 3.3 Relation between final angle of motion vector and direction of block

Angle of *FMVAngle* (in degree)	**Direction**
FMVAngle 0	0 Direction
$0 <$ *FMVAngle* < 22.5 and $337.5 \leq$ *FMVAngle* < 360	1 Direction
$22.5 \leq$ *FMVAngle* < 67.5	2 Direction
$67.5 \leq$ *FMVAngle* < 112.5	3 Direction
$112.5 \leq$ *FMVAngle* < 157.5	4 Direction
$157.5 \leq$ *FMVAngle* < 202.5	5 Direction
$202.5 \leq$ *FMVAngle* < 247.5	6 Direction
$247.5 \leq$ *FMVAngle* < 292.5	7 Direction
$292.5 \leq$ *FMVAngle* < 337.5	8 Direction

Table 3.4 Number of blocks with same direction at same coordinates in two successive frames

Image sequences	**Number of blocks**	**Percentage**
coastguard	354.0	89.39%
Foreman	306.9	77.50%
Mother	381.6	96.39%
News	344.4	86.97%
Stefan	352.9	89.12%
Table	366.5	92.95%
Average	**351.1**	**88.65%**

3.3.2 Center biased distribution of motion vector

In all the motion estimation technique discussed in chapter 2, searches the motion vector in a predefined search points that are uniformly distributed within search window. But this technique of block matching gives very poor results when we deal with the blocks showing very small motion. So the challenge here is to design a search technique that can appropriately search the motion vector even for the blocks having small motion.

As we know in a small region, error surface is monotonous for small neighborhood around global minimum. Hence the probability of finding accurate motion vector near global minimum is very high. The above statement can be expressed mathematically as an optimization problem represented by equation 3.6.

$$\vec{v} = \sum_{x_i \in w} \left\| s_i - x_i \right\| P\left(x_i\right) \qquad (3.6)$$

In most video sequences, images have centre-biased motion vector distribution. It has been observed that more than eighty percentages of the blocks are within in region 3×3 even for the blocks having fast motion. Hence if we choose an optimal search centre, the chances of predicting an accurate motion vector increases without involving much computation.

CHAPTER 4

4. Video encoding using proposed method

4.1 Sorted Search Method (SSM) algorithm

Based on the discussion in the last chapter, we have developed an algorithm that uses both temporal and spatial correlation. The neighboring blocks in a frame are spatially correlated and temporally correlated with the block having same coordinate in the previous frame.

Before introducing the algorithm here is a description of the parameters used: 'd' is the distance between any two pixels, for example, d=1 shows the distance between two consecutive pixels. 'k' is the number of blocks involved in sorted search algorithm [16]. The sorted search method (SSM) algorithm consists of following five steps:

1. We will start with the computation of Sum of Absolute Error (SAE), and then SAE is compared with a threshold (T), if the obtained value of SAE is less than T, then there is no motion in that block or we can say motion vector is zero.
2. SAE of all the neighbouring blocks (B_1, B_2, B_3 and B_4) and the block in the previous frame having same coordinates (B_5) are calculated.
3. The motion vectors of the blocks, mentioned in above step are arranged in an ascending order according to the SAEs produced.

4. A window with the size of (2^d) × (2^d) pixels is centred on ($MVXI_j$ $MVYI_j$), where $1 \leq j \leq k$. the block matching will be done within this search area. If the best match is at the centre of the search area, the motion vector of that block is taken as the motion vector of current block. Otherwise up to k vectors are examined in an ascending order one by one. If the minimum SAE of any block falls at the centre, then the motion vector of that block will be the final motion vector of the current block, otherwise next step is performed.

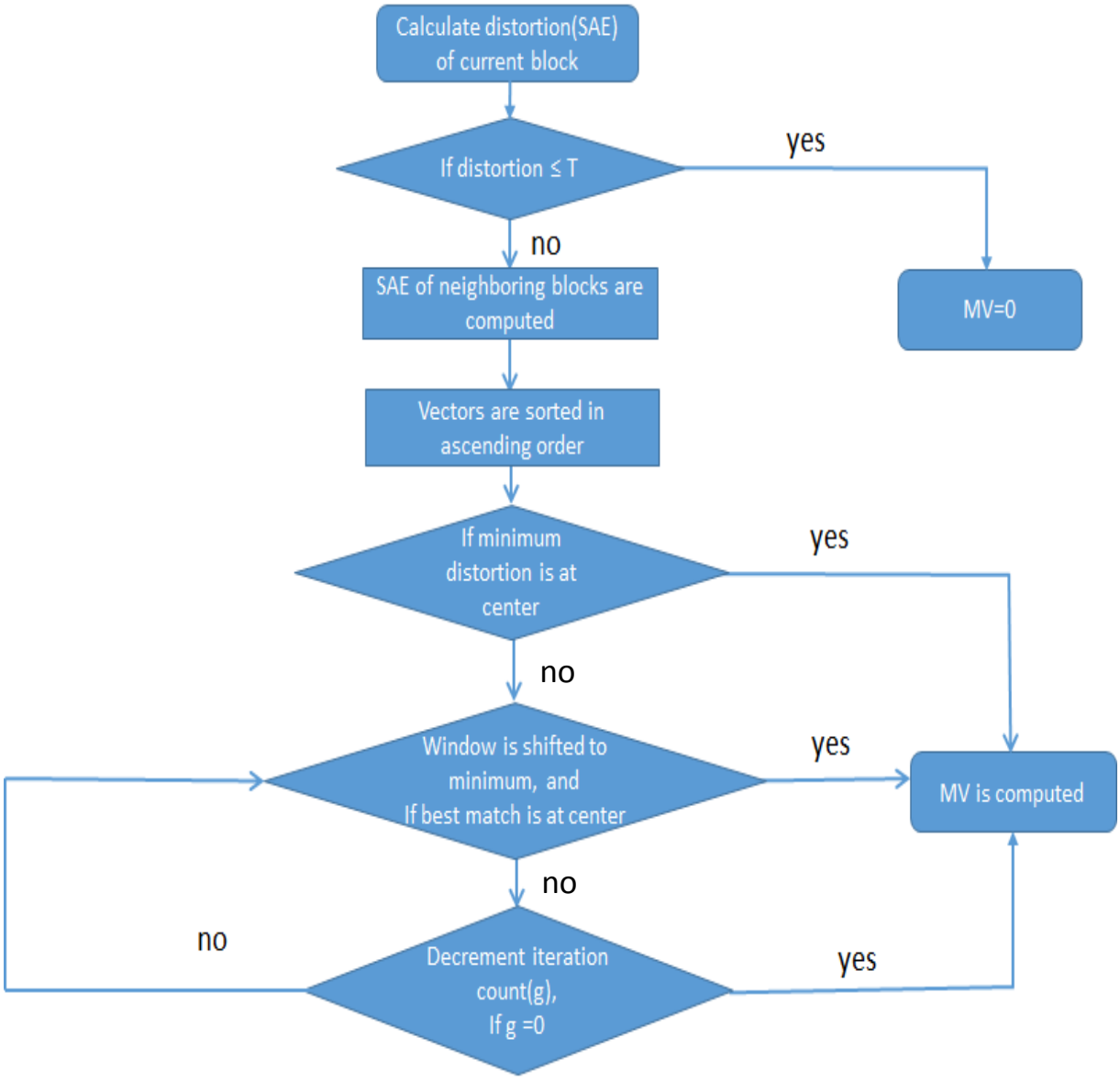
5. A search window of size (2^d) × (2^d) is placed on the obtained minimum distortion point of the step 4. The block matching is done for all the points in the search region. If the minimum SAE of this search falls at the centre then that will give the final motion vector. Otherwise to increase the accuracy of the motion vector estimation this step can be repeated for g times.

The parameters used in this algorithm are k, d and g. And the values of these parameters will decide the number of search point in motion estimation process. Equation 4.1 shows the maximum number of search points for sorted search algorithm.

$$NSP_{\max} = \mathrm{k} \times (2d+1)^2 + \mathrm{g} \times (3d^2 + 2d) + 6 - \mathrm{k} \qquad (4.1)$$

It can be seen from the above equation that as, k increases the number of search points linearly increases because as, k increases the search area also increases, since value of k is directly related to the number of blocks used for the prediction. Increase in d will results in quadratic increase in number of search points. But for smaller values of d, the difference between the

Algorithm Flow Chart



neighbouring motion vectors is very small, so it is easy for variable length encoder (VLC) to do a better compression. At the same time, the smaller number of bits is required for encoding of motion vectors and larger bandwidth can be allocated transmission or error storage, which result into better quality of reconstructed frames.

The result of SSM algorithm when performed on two consecutive frame of "foreman" video sequence is shown in figure 4.1 and figure 4.2. The motion vector field is also shown when the proposed algorithm is imposed for two consecutive frame of "foreman" video sequence.

Figure 4.1 Algorithm performed on two consecutive frames of a "foreman" video sequence

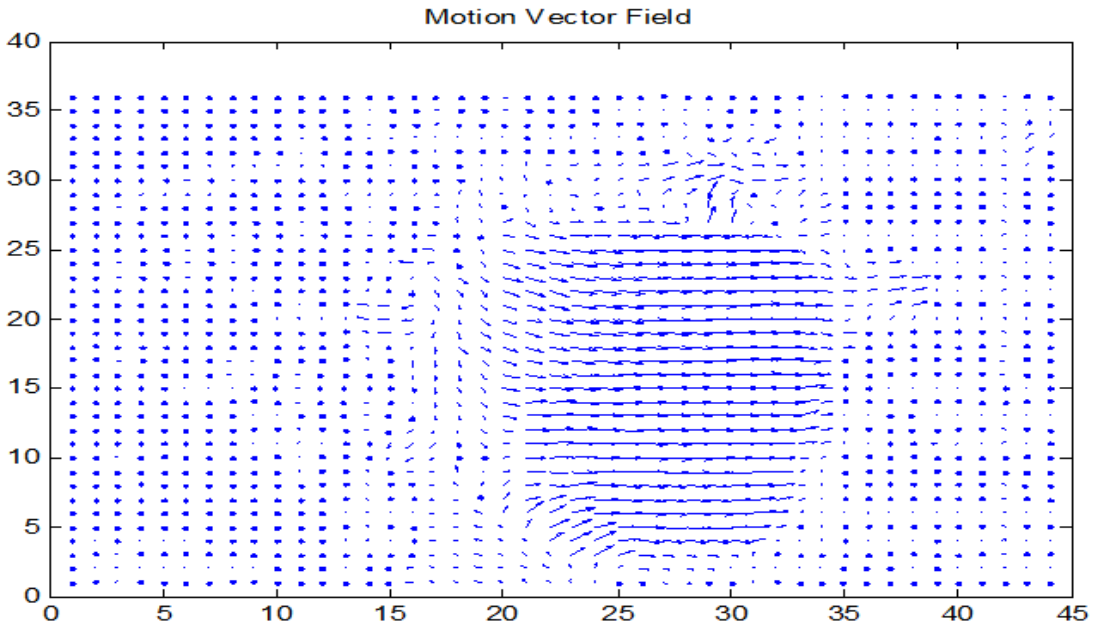


Figure 4.2 Motion vector obtained for two consecutive frames of a "foreman" video sequence

Peak signal to noise ratio (PSNR) is a major criterion for measuring the quality of reconstructed image. Higher value of PSNR means a better quality of reconstructed image. PSNR in terms of dB is defined as $PSNR(dB) = 20\log\left(255/\sqrt{MSE}\right)$. In Table 4, the comparison of FS algorithm and sorted search method (SSM) algorithm for different values of d is shown for four different video sequences. Note that this result is obtained for k=1 and g=0. Higher values of g and k results in implementation complexity, therefore these parameters will be kept as low as possible.

In table 5 number of search points (NSP) for SSM algorithm for five different video sequences are shown. As we can see in the table, the effect of increase in d does not have much impact on NSP. We will perform our SSM algorithm with k=1and g=0 because for these settings we get a fixed pattern and we does not have to shift our search window amount a non-cantered minimum point.

Table 4.1 PSNR (dB) comparison between FS and different values of d

	FS	*d=1*	*d=2*	*d=3*	*d=4*	*d=5*
Foreman	35.80	35.85	35.87	35.91	35.94	35.98
Caltrain	31.67	31.71	31.78	31.81	31.81	31.81
Container	32.77	32.88	32.88	32.88	32.88	32.88
Carphone	33.16	33.24	33.24	33.24	33.24	33.21
Average	**33.35**	**33.42**	**33.44**	**35.46**	**35.46**	**35.47**

Table 4.2 Number of search points comparison between FS and different values of d

	FS	*d=1*	*d=2*	*d=3*	*d=4*	*d=5*
Foreman	869.3	7.5	18.1	33.0	51.9	74.2
Caltrain	869.3	9.1	32.78	32.77	32.77	32.77
Container	869.3	5.6	33.96	33.96	33.96	33.96
Carphone	782.2	6.2	34.06	34.04	34.06	34.03

4.2 Comparison of SSM algorithm for different neighboring combination

In the last section we have discussed SSM algorithm for four neighbouring blocks (B_1, B_2, B_3 and B_4) and a block in the previous frame having same coordinates as that of current block (B_5). By forming a combination of different neighbours with current block B_C, six different cases can be formed as shown in figure 4.3. SSM algorithm has been applied to all six cases we have formulated. The SSM algorithm is named on the number of blocks we are using in the prediction process.

Case1. Current block B_C, and neighbouring bocks B_1 to B_4 are used in the process and this method is known as “sorted5” algorithm.

Case2. Current block B_C, and neighbouring bocks B_2, B_3 and B_4 are used in the process and this method is known as “sorted4” algorithm.

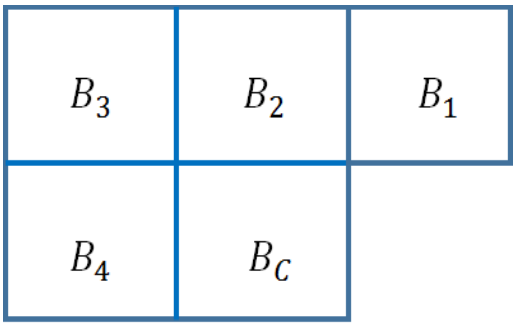
Case3. Current block B_C, and neighbouring bocks B_1, B_2 and B_3 are used in the process and this method is known as “sorted4a” algorithm.

Case4. Current block B_C, and neighbouring bocks B_2 and B_4 are used in the process and this method is known as “sorted3” algorithm.

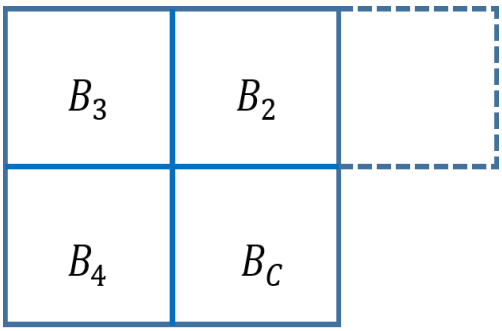
Case5. Current block B_C, and neighbouring bocks B_3 and B_4 are used in the process and this method is known as “sorted3a” algorithm.

Case6. Current block B_C, and neighbouring bocks B_2 and B_3 are used in the process and this method is known as “sorted3b” algorithm.

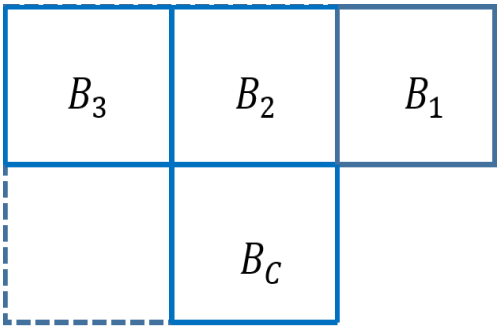
As the number of blocks increases in the sorted search method, the computation time increases. For example, in “sorted5” algorithm number of blocks involved in computation are five and in “sorted3a” algorithm, three blocks are involved in computation, as “sorted3a” algorithm is using less number of blocks for computation of motion vectors , therefore computation time involved in case of “sorted3a” algorithm will be less.



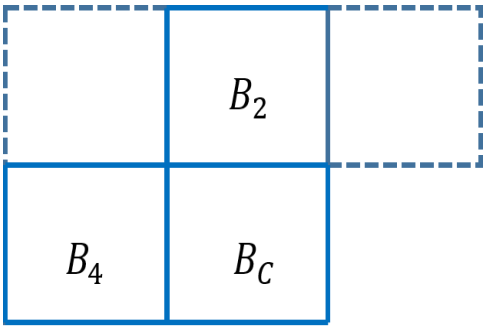
(a) Blocks used for sorted5 method.



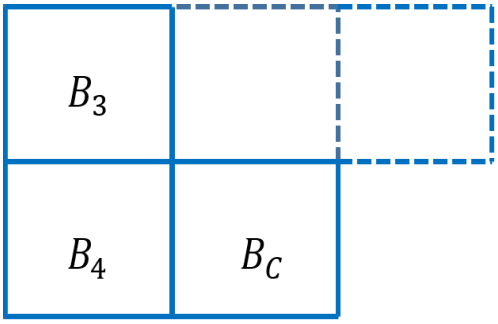
(b) Blocks used for sorted4 method.



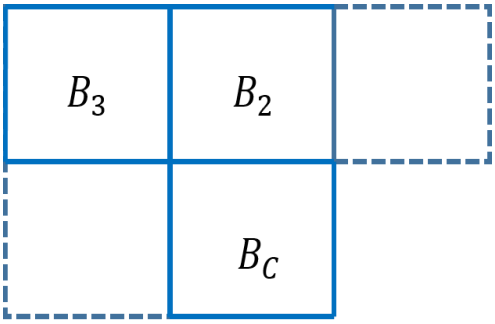
(c) Blocks used for sorted4a method.



(d) Blocks used for sorted3 method.



(e) Blocks used for sorted3a method.



(f) Blocks used for sorted3b method.

Figure 4.3 Different combination of current block "B_c" with its neighbouring blocks.

4.3 Results and analysis

In this section the implementation result of different sorted search method using different video sequences is shown. Every sorted algorithm will have different PSNR value and will have different computation time. The comparison of Sorted Search Method (SSM) algorithm with Full Search (FS), Adaptive Rood Pattern Search (ARPS), Diamond Search (DS) and Three Step Search (TSS) is shown in figure 4.4. The algorithm is performed on 31 consecutive frames of "caltrain" and "foreman" video sequences. As it can be seen from the figure 4.4, the PSNR (dB) for SSM algorithm is better than the existing fast block motion estimation techniques. This is because in SSM algorithm a better first search point is selected and at the

same time full search is used for block matching that will enhanced the quality of recovered image.

We have performed SSM algorithm on different video sequences and here we have shown the results of “foreman”, “salesman” and “city” video sequences. In table 4.3a comparison of PSNR (dB), Structural Similarity Index Measurement (SSIM) and computation time are shown for “foreman”, “salesman” and “city” video sequences respectively. The sorted algorithm having maximum PSNR and taking minimum computation time will be selected as our best SSM algorithm. After exhaustive study, it is observed that “sorted3a” algorithm outperformers the other existing search algorithms in terms of PSNR and computation time.

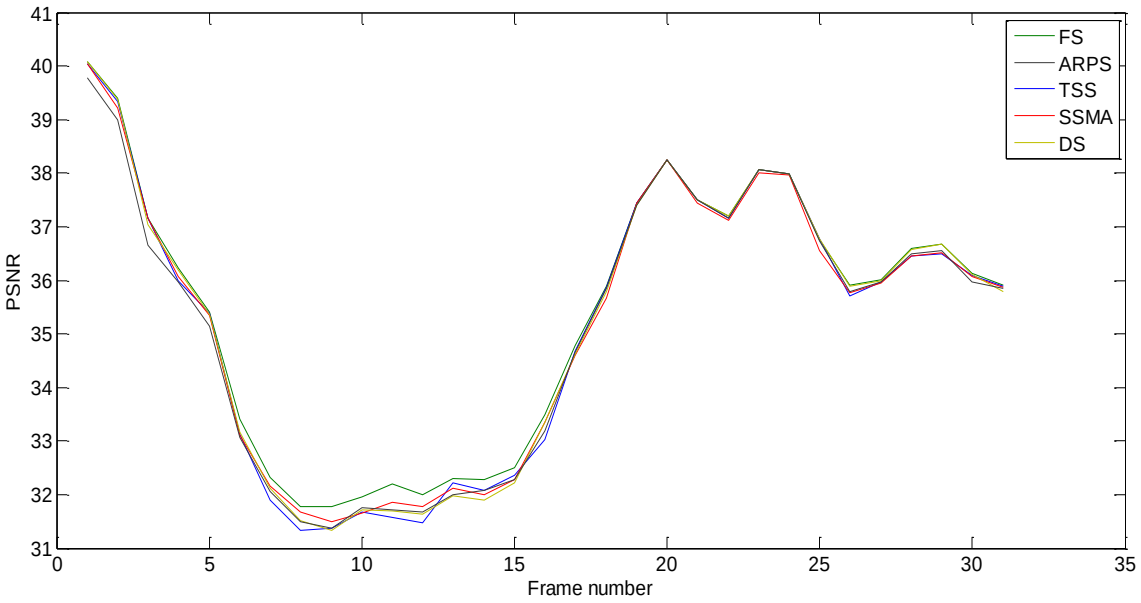


Figure 4.4 Comparison of PSNR (dB) among different algorithms for 31 consecutive frames of “foreman” video sequence.

TABLE 4.3

Comparison of different sorted search algorithms

for "city" video sequence

Method Used	PSNR (dB)	SSIM	Computation Time(sec.)
Sorted5	21.9946	0.9990	30.6911
Sorted4	21.7238	0.9984	30.4474
Sorted4a	21.7869	0.9988	30.1503
Sorted3	21.7646	0.9988	28.2671
Sorted3a	21.9614	0.9990	28.0503
Sorted3b	21.7677	0.9985	28.0961

CHAPTER 5

5. Conclusion and future scope

In this thesis, a new technique has been proposed for motion estimation in video compression that uses its neighboring blocks to predict the motion vector of the current block. A sorting search method algorithm has been introduced here. The different sorting search methods by taking different neighborhood combinations are also developed and compared. The results of the proposed algorithm was carried out on three different video sequences and the performance of of the algorithm is compared with FSA and different fast block motion estimation algorithms like, TSS, DS and ARPS. After exhaustive study, it is observed that sorted3a algorithm outperforms the other existing search algorithms in terms of PSNR and computation time. The advantage of sorted search method algorithm is its simplicity and predefined behavior.

The sorted search methods that uses blocks immediate left of the current macro-block that is sorted5, sortred4, sorted3, sorted3a gives better PSNR and SSIM, since the motion in a video are generally translational in nature. Also, the sorted method that uses fewer numbers of blocks in search prediction takes less computational time. From above discussion and obtained results we can say that sorted3a is a better candidate for block matching as it takes less computational time and gives good quality of reconstructed image.

- In future, hardware implementation of the proposed algorithm can be explored.

References:

[1] ISO/IEC15444, *Information technology-JPEG2000 image coding system*, 2000.

[2] W.Hsuand, H. Derin, "Three-dimensional subband coding of video," *Proc. Int. Conf. Acoustics, Speech, and SignalProcessing (ICASSP)*, pages 1100–1103, Apr. 1988.

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

[4] Rafael C. Gonzalez, Richard E. Woods, "*Digital Image Processing*", 2012.

[5] Iain E. G. Richardson, "*H.264 and MPEG-4 Video Compression Video Coding for Next generation Multimedia*," The Robert Gordon University, Aberdeen, UK, 2010.

[6] Yao Wang, Joern Ostermann, and Ya-Qin Zhang, "*Video Processing And Communications*," Printice hall signal processing series, 2002.

[7] Nasir D. Memon, Khalid Sayood, "Lossless Compression of Video Sequences*," IEEE transaction on Communications*, vol. 44, no. 10, 1996.

[8] Avijit Kundu, "Modified Block Matching Algorithm for Fast Block Motion Estimation," *International Conference on Signal and Image Processing*, 2010.

[9] Xiaolin Chen, Nishan Canagarajah, and Jose L. Nunez-Yanez "Backward Adaptive Pixel-based Fast Predictive Motion Estimation," *IEEE signal processing letters*, vol. 16, no. 5, 2009.

[10] T. Koga, K. Iinuna, A. Hirano, Y. Iijima, and T. Ishiguro "Motion Compensated Interframe Coding for Video Conferencing". *In Proc. of National Telecomm. Conf*, pp. G5.3.1–G5.3.5, New Orleans, Nov. 1981.

[11] J. R. Jain and A. K. Jain, "Displacement Measurement and its Application in Interframe Image Coding," *IEEE Trans. on Communication*, vol. 29, no. 12, pp.1799–1808, 1981.

[12] Lai-Man Po Wing-Chung Ma. "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation", *IEEE Trans. Circuits Syst. Video Technol*., vol. 6, no. 3, pp. 313-317, 1996.

[13] Shan Zhu, and Kai-Kuang Ma, " A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", *IEEE Trans. Image Processing*, vol 9, no. 2, pp. 287-290, 2000.

[14] Yao Nie, and Kai-Kuang Ma, "Adaptive Rood Pattern Search for Fast Block-Matching Motion Estimation", *IEEE Trans. Image Processing*, vol 11, no. 12, pp. 1442-1448, 2002.

[15] S.M.R. Soroushmehr, S. Samavi, S. Shirani,"Fast block motion estimation based on sorting of prediction vector," *Canadian Journal of Electrical and Computer Engineering*, vol. 35, no.1, pp. 25-32, 2010 .

[16] Jae-Yeal Nam, Jae-Soo Seo, Jin-Suk Kwak, Myoung-Ho Lee and Yeong Ho Ha "New Fast-Search Algorithm For Block Matching Motion Estimation Using Temporal And Spatial Correlation Of Motion Vector," *IEEE Transactions on Consumer Electronics*, vol. 46, no. 4, 2000.