FAST FULL SEARCH ALGORITHM FOR MOTION ESTIMSTION

BY

DHRUV DAVE

07MCE003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

MAY 2009

Fast Full Search Algorithm For Motion Estimation

Major Project

Submitted in partial fulfillment of the requirements

For the degree of

Master of Technology in Computer Science and Engineering

By

Dhruv P. Dave

07MCE003

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

AHMEDABAD-382481

May 2009

Certificate

This is to certify that the Major Project entitled “Fast full search algorithm for motion estimation” submitted by Dhruv Dave(07MCE003), towards the partial fulfillment of the requirements for the degree of Master of Technology in Computer Science and Engineering of Nirma University of Science and Technology, Ahmedabad is the record of work carried out by him under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven’t been submitted to any other university or institution for award of any degree or diploma.

Dr. S.N. Pradhan
Guide and Professor,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Prof. D. J. Patel
Professor and Head,
Department of Computer Engineering,
Institute of Technology,
Nirma University, Ahmedabad

Dr K Kotecha
Director,
Institute of Technology,
Nirma University, Ahmedabad

Abstract

Video compression is vital for efficient storage and transmission of digital signal. The hybrid video coding techniques based on predictive and transform coding are adopted by many video coding standards such as ISO MPEG-1/2 and ITU-T H.261/263.

Motion estimation and motion compensation is an predictive technique for exploiting the temporal redundancy between successive frames of video sequence. Block matching techniques are widely used motion estimation method to obtain the motion compensated prediction. By splitting each frame into macroblocks, motion vector of each macroblock is obtained by using block matching algorithm (or motion estimation algorithm). In order to get motion vector of each macroblock, the most obvious and simplistic method is full search algorithm. All possible displacements in the search window are evaluated using block-matching criteria (cost function). The advantage of full search is that we can find the absolute optimal solution. However, its high computational complexity makes it impossible for real-time implementation. Because the computational complexity of video compression, the compression efficiency and the compression quality is determined by the motion estimation algorithm, development of Fast Motion Estimation Algorithm for real-time application becomes compelling.

The computational complexity of a motion estimation technique can then be determined by three factors: 1. search algorithm. 2. cost function/evaluate function. 3. search range parameter p. Actually, we can reduce the complexity of the motion estimation algorithms by reducing the complexity of the applied search algorithm and/or the complexity of the selected cost function. An full search algorithm evaluates all the weights in the search window, and a more efficient, less complex search algorithm will decrease the search space.

We will identify and evaluate the recent and widely used fast estimation algorithms, especially in hybrid estimation algorithms: how they get the trade-off between video quality and compression efficiency? Based on this, we will propose our new motion estimation algorithm produced our new video encoder. To test its validation and its efficiency, we used the standard tested video sequences, which include three kinds of sequences: one is little motion sequences, second is moderate motion sequences, and third is the fast motion sequences.

Acknowledgements

I would like to thanks to Dr. S.N. Pradhan , PG Coordinator, Department of Computer Engineering, Institute of Technology, Nirma University, Ahmedabad for his valuable guidance and continual encouragement throughout the Major project. I heartily thankful to him for his time to time suggestion and the clarity of the concepts of the topic that helped me a lot during this study.

I like to give my special thanks to Prof. D.J.Patel, Head, Department of Computer Engineering, Institute of Technology, Nirma University, Ahmedabad for his continual kind words of encouragement and motivation throughout the Major Project. I am also thankful to Dr. K Kotecha, Director, Institute of Technology for his kind support in all respect during my study.

I am thankful to all faculty members of Department of Computer Engineering, Nirma University, Ahmedabad for their special attention and suggestions towards the project work. The blessings of God and my family members makes the way for completion of major project. I am very much grateful to them. The friends, who always bear and motivate me throughout this course, I am thankful to them.

Last, but not the least, no words are enough to acknowledge constant support and sacrifices of my family members because of whom I am able to complete the degree program successfully.

- Dave Dhruv p.

(07MCE003)

Contents

List of Tables

List of Figures

Abbreviations

TSS	Three Step Search
FSS	Four Step Search
DS	Diamond Search
ARS	Adaptive Root Search
MAD	Mean Absolute Difference
MSE	Mean Squared Error
SAD	Sum of Absolute Differences
PSNR	Peak-Signal-to-Noise-Ratio
LDSP	Large Diamond Search Pattern
SDSP	Small Diamond Search Pattern

Chapter 1

Introduction

The temporal prediction technique used in MPEG video is based on motion estimation. The basic premise of motion estimation is that in most cases, consecutive video frames will be similar except for changes induced by objects moving within the frames. In the trivial case of zero motion between frames and no other differences caused by noise, etc.., it is easy for the encoder to efficiently predict the current frame as a duplicate of the prediction frame. When this is done, the only information necessary to transmit to the decoder becomes the syntactic overhead necessary to reconstruct the picture from the original reference frame. When there is motion in the images, the situation is not as simple. Another goal of Motion Estimation is to reduce the total amount of bits required for transmission or storage of the frames of an image sequence. Motion compensated image sequence coding primarily focuses on the reduction of the high temporal correlation of the signal to be stored or transmitted. To that end, motion information has to be extracted from the sequence in order to relate locations in consecutive frames that correspond in gray level. This motion is represented by so called corresponding or displacement vectors.

Motion estimation techniques can be divided into four main groups.

- Gradient techniques
- Pel recursive techniques
- Block matching techniques
- Frequency based techniques

Gradient techniques have been developed for image sequence analysis applications. They solve the optical flow and results in a dense motion field. Both pel recursive and block matching techniques have been developed in the framework of image sequence coding. Pel recursive techniques can be considered as a subset of gradient techniques. However as they constitute as important contribution in the field of coding, they are considered as a separate group. Block matching techniques are based on the minimization of a disparity measure. They are most widely used in coding applications. In block matching techniques, a block in the current picture is matched with block of previous picture and hence motion is calculated. Finally, frequency-domain techniques are based on the relationship between transformed coefficients (e.g., Fourier transform) of a shifted image. However they lack widespread use, especially in the field of image sequence coding and are not popular.

1.1 Motivation

- Low latency capabilities and better quality for higher latency.
- Straightforward syntax specification that simplifies implementations.
- Communication and storage capabilities are limited and expensive.
- Motion estimation (ME) is an important part of many video processing tasks.
- ME main applications are video compression, sampling rate conversion, filtering etc..

1.2 Scope of Work

Implementation of following algorithms

- Three Step Search (TSS).
- Four Step Search Algorithms (FSS).
- Diamond Search Algorithms (DS).
- Adaptive Root Search Algorithms (ARSA)

And compare the result of average number of search and PSNR ratio of each algorithm. Next scope of my project is to find motion vector from two sequence of frame and predict next frame by using motion vectors and finding the motion.

1.3 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2, *Literature Survey* Describes the broad classification of motion estimation ,Motion Representation,motion estimation

Chapter 3, *Search Algorithms for Block motion estimation* Describe the all search algorithms for block motion estimation like three step search, four step search,full search,diamond search,adaptive root search algorithms etc..,and some implementation of that algorithms

Chapter 4, *Motion Vectors and Motion Compensation* Describe the motion vector and implementation of finding motion vectors,

Chapter 5, *Implementation, Results and Analysis* Describes in brief about motion and how motion can be taken out by using motion vectors,and some algorithm to find motion,and the comparison of all algorithms,Implementation of all algorithms.

Chapter 6,*Conclusion and Future Scope* Concluding remarks and scope for future work is presented.

Chapter 2

Literature Survey

2.1 General

2.1.1 Broad Classification of Motion Estimation

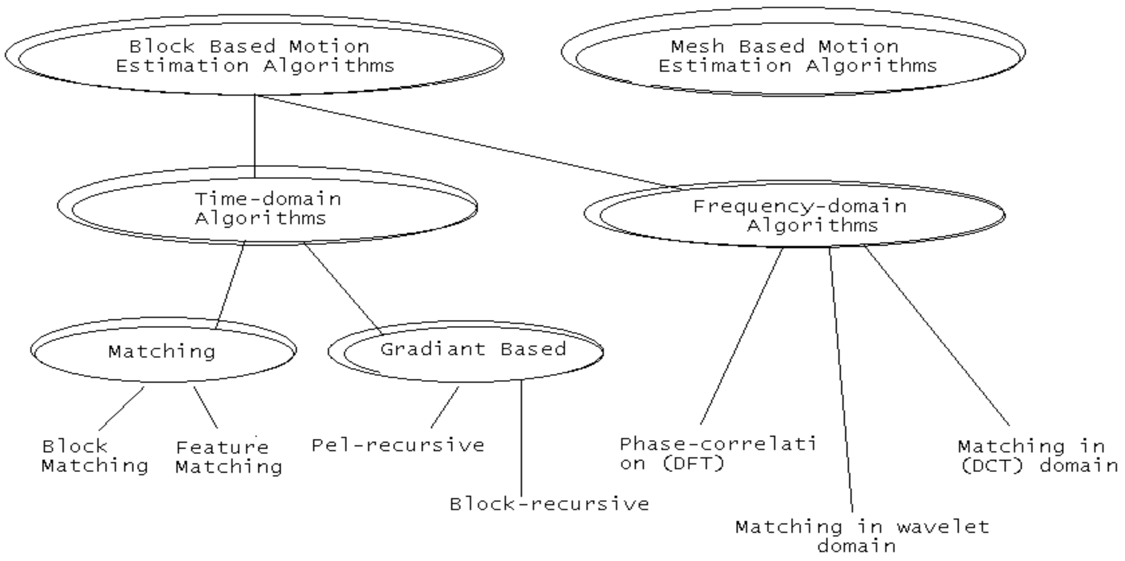

Figure 2.1: Broad Classification of motion estimation

- Mesh Based Motion Estimation Algorithm
- Block Based Motion Estimation Algorithm
  - Time Domain Algorithm
    - Matching Algorithms
      - Block Matching Algorithms
    - Gradient Based Algorithms
      - Feature Matching Algorithm
  - Frequency Domain Algorithm
    - Phase-Correlation
    - Matching in Domain
    - Matching in Wavelet

2.1.2 Motion Representation

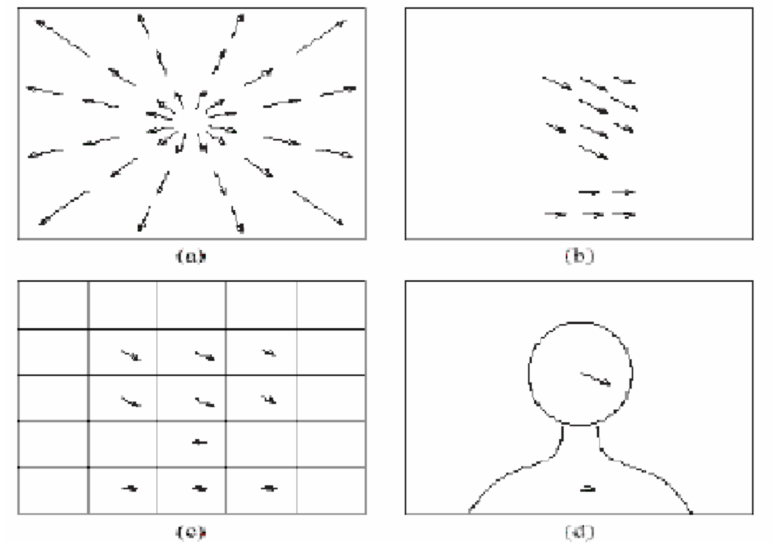

Figure 2.2: Motion Representation

- Global
  - Entire motion field is represented by a few global parameters (Global motion representation, Camera motion)
- Pixel-Based
  - One motion vector at each pixel with some smoothness constraint between adjacent motion vectors
  - Very time consuming
- Block-Based
  - Entire frame is divided into non-overlapping blocks then motion in each block is characterized by a few parameters
- Predictive Motion Estimation
  - Prediction of Motion Vectors is usually performed to gain an initial guess of next motion vector. This reduces the computational burden.

2.1.3 Motion Estimation

An MPEG video can be understood as a sequence of frames. Because two successive frames of a video sequence often have small differences (except in scene changes), the MPEG-standard offers a way of reducing this temporal redundancy. It uses three types of frames:

I-frames(intra), P-frames (predicted) and B-frames (bidirectional).

The I-frames are key-frames, which have no reference to other frames and their compression is not that high. The P-frames can be predicted from an earlier I-frame or P-frame. P-frames can not be reconstructed without their referencing frame, but they need less space than the I-frames, because only the differences are stored. The B-frames are a two directional version of the P-frame, referring to both directions (one forward frame and one backward frame). B-frames cannot be referenced by other P-frames or B-frames, because they are interpolated from forward and backward frames. P-frames and B-frames are called inter coded frames, whereas I-frames are known as intra coded frames [1].

The usage of the particular frame type defines the quality and the compression ratio of the compressed video. I-frames increase the quality (and size), whereas the usage of B-frames compresses better but also produces poorer quality. The distance between two I-frames can be seen as a measure for the quality of an MPEG-video. In practise following sequence showed to give good results for quality and compression level: IBBPBBPBBPBBIBBP.

The references between the different types of frames are realized by a process called motion estimation or motion compensation. The correlation between two frames in terms of motion is represented by a motion vector. The resulting frame correlation, and therefore the pixel arithmetic difference, strongly depends on how good

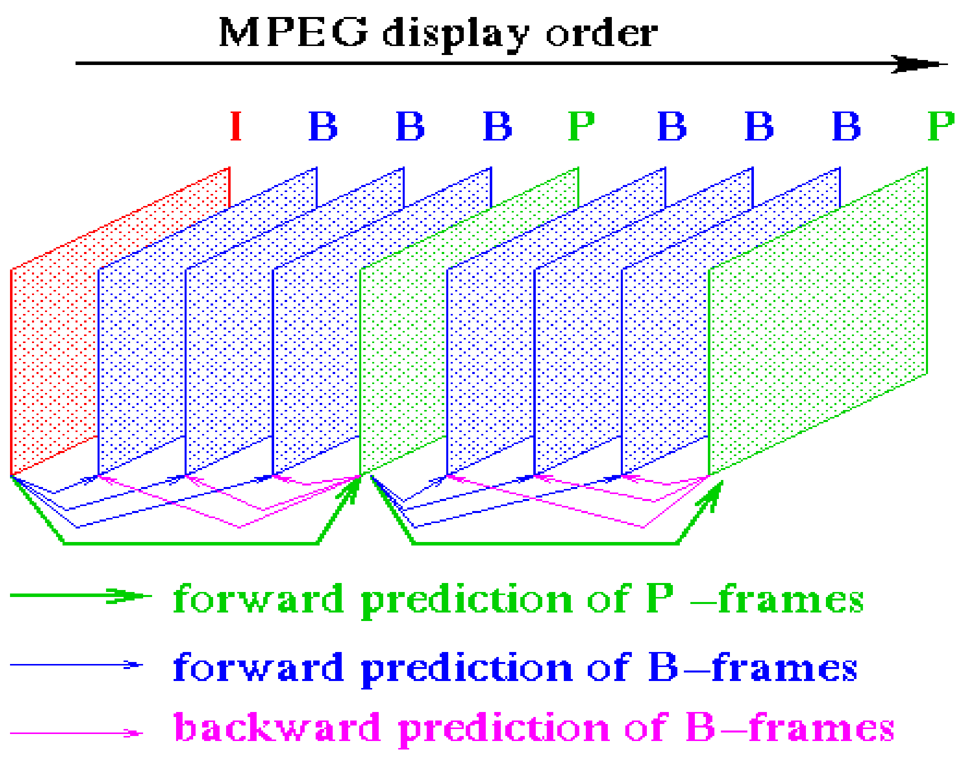
MPEG display order
I B B B P B B B P
forward prediction of P –frames
forward prediction of B–frames
backward prediction of B–frames

Figure 2.3: An MPEG frame sequence with two possible references: a P-frame referring to a I-frame and a B-frame referring to two P-frames.

the motion estimation algorithm is implemented. Good estimation results in higher compression ratios and better quality of the coded video sequence. However, motion estimation is a computational intensive operation, which is often not well suited for real time applications. Figure 2.4 shows the steps involved in motion estimation, which will be explained as follows[2]:

- Frame Segmentation

  The Actual frame is divided into non-overlapping blocks (macro blocks) usually 8x8 or 16x16 pixels. The smaller the block sizes are chosen, the more vectors need to be calculated; the block size therefore is a critical factor in terms of time performance, but also in terms of quality: if the blocks are too large, the motion matching is most likely less correlated. If the blocks are too small, it is

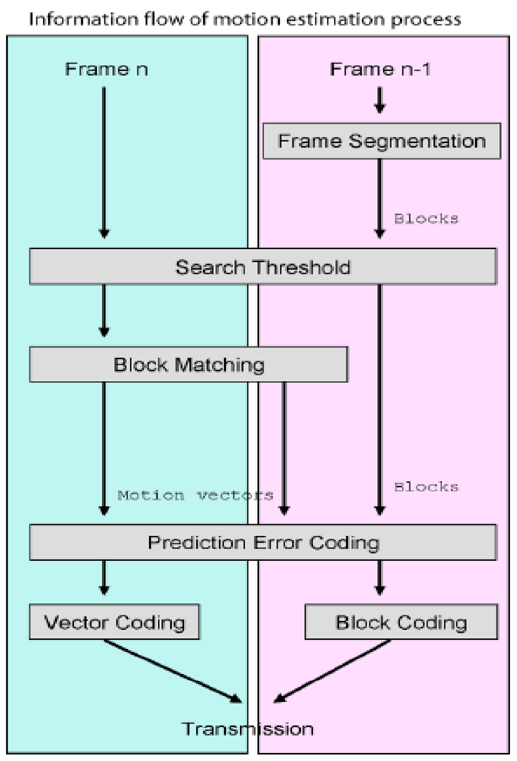

Figure 2.4: Schematic process of motion estimation

probably, that the algorithm will try to match noise. MPEG uses usually block sizes of 16x16 pixels.

- Search Threshold

  In order to minimize the number of expensive motion estimation calculations, they are only calculated if the difference between two blocks at the same position is higher than a threshold, otherwise the whole block is transmitted.

- Block Matching

  In general block matching tries, to stitch together an actual predicted frame by using snippets (blocks) from previous frames. The process of block matching is the most time consuming one during encoding. In order to find a matching block, each block of the current frame is compared with a past frame within a search area. Only the luminance information is used to compare the blocks, but obviously the color information will be included in the encoding. The search area is a critical factor for the quality of the matching. It is more likely that the algorithm finds a matching block, if it searches a larger area. Obviously the number of search operations increases quadratically, when extending the search area. Therefore too large search areas slow down the encoding process dramatically. To reduce these problems often rectangular search areas are used, which take into account, that horizontal movements are more likely than vertical ones. More details about block matching algorithms

- Prediction Error Coding

  Video motions are often more complex, and a simple shifting in 2D is not a perfectly suitable description of the motion in the actual scene, causing so called prediction errors. The MPEG stream contains a matrix for compensating this error. After prediction the, the predicted and the original frame are compared, and their differences are coded. Obviously less data is needed to store only the differences (yellow and black regions in figure 2.5)

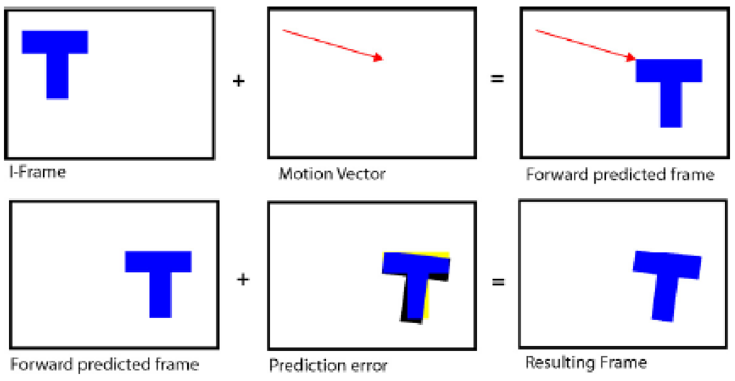

Figure 2.5: Schematic process of motion estimation[1]

- Vector Coding

  After determining the motion vectors and evaluating the correction, these can be compressed. Large parts of MPEG videos consist of B-frames and P-frames as seen before, and most of them have mainly stored motion vectors. Therefore an efficient compression of motion vector data, which has usually high correlation, is desired.

2.2 Motion Estimation Process

2.2.1 Block based matching method

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, standards bodies, e.g. MPEG, defines the standard block sizes for motion estimation. This can be 16 by 16, 8 by 8, 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 4 by 4, 8 by 8 and 16 by 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 and thus the accuracy.Figure 2.6 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[3].

2.2.2 Motion estimation procedure

After motion estimation, a picture residue and a set of motion vectors are produced. The following procedure is executed for each block (16x16, 8x8 or 4x4) in the current frame.

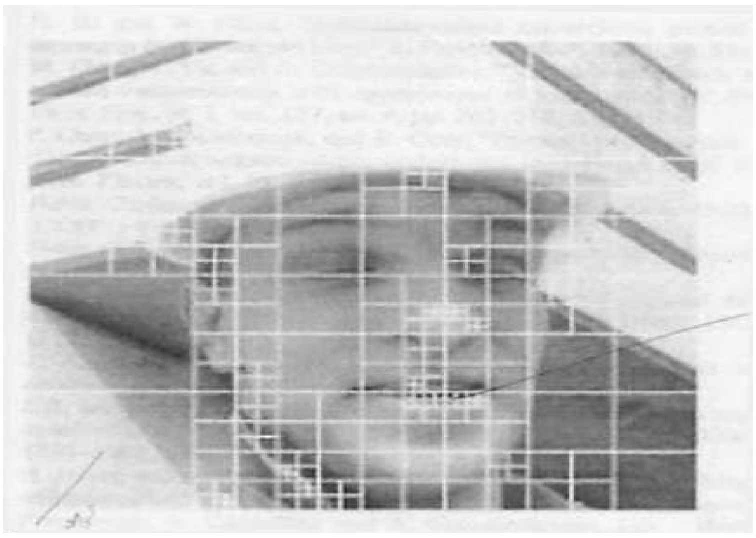

Figure 2.6: Selection of block sizes within a frame[3]

a. 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.

b. When the best match is found, 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.

c. The residues and motion vectors of best match are encoded by the transform unit and entropy unit and transmitted to the decoder side.

d. At decoder side, the process is reversed to reconstruct the original picture.

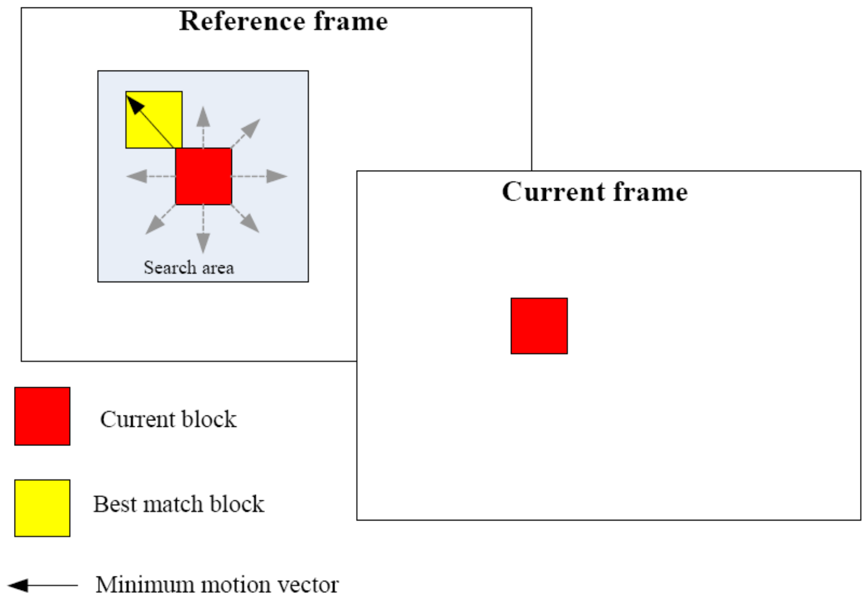

Figure 2.7: Motion estimation and motion vector

Figure 2.7 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. 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.

2.2.3 Motion vectors

To represent 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. The larger the search range, 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 2.7

Traditionally one motion vector is produced from each macro-block in the frame. MPEG-1 and MPEG-2 employ this property. Since 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 in one macro-block and they are passed to rate- distortion optimization to choose the best combination. This is known as mode selection.

2.2.4 Quality judgment

The quality of a video scene can be determined using both objective and subjective approaches. The most widely used objective measure is the Peak-Signal-to-Noise-Ratio (PSNR) which is defined as:

$$PSNR = 10\log_{10}[\frac{(peaktopeakvalueoforiginaldata)^2}{MSE}]$$

where the MSE is the mean square error of the decoded frame and the original frame. The peak value is 255 since the pixel value is 8 bits in size.

The higher the PSNR, the higher the quality of the encoding. The PSNR and bit-rate are usually conflicting, the most appropriate point being determined by the application. Although PSNR can objectively represent the quality of coding, it does not equal the subjective quality. Subjective quality is determined by a number of human testers and a conclusion is drawn based on their opinions. There exist cases for which high PSNR results in low subjective quality. However, in most cases, PSNR provides a good approximation to the subjective measure.

2.3 Study of H.264 Motion Estimation Engine v1.0

2.3.1 Introduction

The H.264 Motion Estimation Engine Version 1.0 is a fully functional netlist implemented on a Xilinx FPGA. The Motion Estimation core accepts input parameters and macro-blocks and generates output motion vectors and Sum of Absolute (SAD) values in accordance with the ITU-T Video Coding Experts Group (VCEG) together with the ISO/IEC Moving Picture Experts Group (MPEG) as the product of a collective partnership effort known as the Joint Video Team (JVT). The collaborative effort is also known as H.264/AVC/MPEG4 Part 10.

2.3.2 Applications

The H.264 Motion Estimation core can be utilized in H.264 video encoding applications where hardware acceleration is needed to achieve real time operation. Typical video applications are video surveillance, video conferencing and video broadcast.

2.3.3 Description

The H.264 Motion Estimation core computes the sum of absolute difference (SAD) for a set of 120 search locations within a 112 x 128 search window for 8 x 4 blocks. The search locations are determined by a set of 10 seeds that are provided by the user and the 4 x 3 region to the right and down from each seed. The core provides as output the 120 SAD calculated values and the motion vectors. In addition, the coded block pattern is computed for a macro-block and the best motion vector for each sub-block is provided The functional inputs and outputs to the H.264 Motion Estimation Core are:[4]

- Inputs:
  - New Macro-block
  - Parameters for the New Macro-block
    - Macro-block location (h, v)
    - Motion Vector predictors
      - Reference Frame pointer
      - Reference Frame
- Outputs:
  - Output Parameters
    - Best Motion Vector
    - Motion Vectors
    - SAD values
    - Coded block pattern

Figure 2.8 below is a diagram of the H.264 Motion Estimation architecture. Motion estimation requires the associated portion of the reconstructed frame to be available before any processing can start. The Reference frame is stored in the external memory and is accessible through an external memory controller. One port is reserved for writing the reconstructed frame, and the user is responsible for loading it prior to starting the motion estimation process. The second port is reserved for motion estimation reading the necessary data from the external memory.

The motion estimation is delivered as a netlist. The motion estimation core requires the user to provide a list of initial motion vectors where the searches will be performed. The motion estimation module is responsible for performing the searches around the provided location, as well as for the search area data management. The

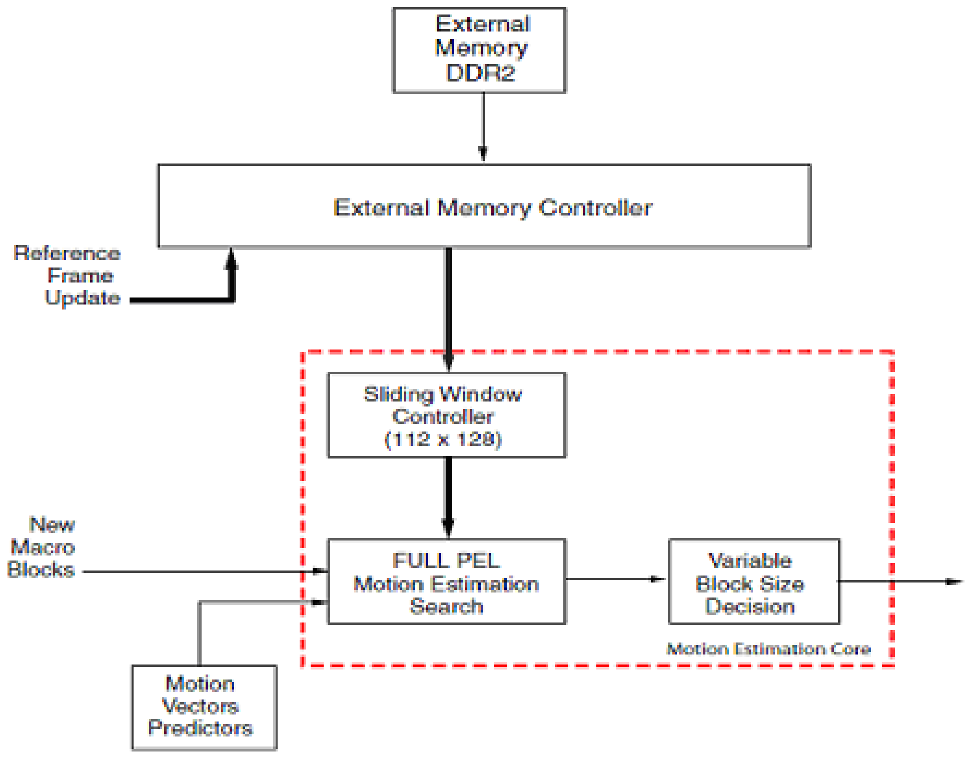
External Memory DDR2
External Memory Controller
Reference Frame Update
Sliding Window Controller (112 x 128)
New Macro Blocks
FULL PEL Motion Estimation Search
Variable Block Size Decision
Motion Estimation Core
Motion Vectors Predictors

Figure 2.8: H-264 Motion Estimation Architecture

motion estimation algorithm uses the input information such us macro-block location for processing the current macro-block.

Chapter 3

Search Algorithms For Block-matching

3.1 Introduction

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 (based on the previous frame) is coded and transmitted. The better the prediction, the smaller the error and hence the transmission bit rate. If a scene is still, then a good prediction for a particular pel in the current frame is the same pel in the previous frame and the error is zero. However, 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. The use of the knowledge of the displacement of an object in successive frames is called Motion Compensation. There are a large number of motion compensation algorithms for inter-frame predictive coding. In this study, however, we have focused only on one class of such algorithms, called the Block Matching Algorithms. These algorithms estimate the amount of motion on a block by block basis, i.e. for each block in the current frame, a block from the previous frame is found, that is said to match this block based on a certain criterion. There are a number of criteria to evaluate the "goodness" of a match and some of them are:

a. Cross Correlation Function

b. Pel Difference Classification (PDC)

c. Mean Absolute Difference

d. Mean Squared Difference

e. Integral Projection

Some of these criteria are simple to evaluate, while others are more involved. Different kinds of algorithms use different criteria for comparison of blocks. One of the first algorithms to be used for block based motion compensation is what is called the Full Search or the Exhaustive Search. In this, each block within a given search window is compared to the current block and the best match is obtained (based on one of the comparison criterion). Although, this algorithm is the best one in terms of the quality of the predicted image and the simplicity of the algorithm, it is very computationally intensive. With the realization that motion compensation is the most computationally intensive operation in the coding and transmitting of video streams, people started looking for more efficient algorithms. However, there is a trade-off between the efficiency of the algorithm and the quality of the prediction image. Keeping this trade-off in mind a lot of algorithms have been developed. These algorithms are called Sub-Optimal because although they are computationally more efficient than the Full search, they do not give as good a quality as it.

There are several approaches to reducing the computational complexity. For instance there are the Signature Based Algorithms that reduce the computation by using several stages, in each of which a different comparison criterion is used. In the first stage all the blocks are evaluated using a computationally simple criterion and then based on the results of this stage a subset of the candidates is picked for the next stage, where a more complex criterion is used. There are algorithms that exploit the limitations of the human observers. These algorithms reduce computational complexity by reducing the candidates that are chosen for the comparison, based on the knowledge that the human eyes cannot perceive fast motion with full resolution. So they use what is called a coarse quantization of vectors i.e. around the centre of the search area all blocks are evaluated as potential matches, while far from the centre only a subset of blocks is considered.

Some algorithms are based on the nature of the image data than the limitations of the human observers. It is believed by these algorithms that very good matches are likely to be found in the vicinity of reasonably good matches. Although this assumption might not be necessarily true, it is useful for reducing the computation as the search can be broken down into stages where the algorithm successively narrows down on the regions of good matches. There are a large number of algorithms that make this assumption and these may be classified as algorithms based on the Principle of Locality. One of the problems with these algorithms is that they can converge to a local minimum rather than to the global minimum. These algorithms can be modified by changing the manner in which the algorithm narrows down the search area. For instance the extent of reduction of the search area can be made a function of the two smallest distortions in the previous stage, rather than just the smallest distortion. Such algorithms are called the Dynamic search Window algorithms.

There is another class of algorithms, that seeks to exploit the natural spatial dependency (homogeneity) that exists in most images. Hence the motion vector for a block can be predicted based on the motion vectors of the blocks surrounding it. One can also exploit temporal dependency by trying to predict the motion vectors for the current block based on the motion vectors for the same block from the previous frame. There are other approaches to the problem like the Hierarchical motion vector estimation using the Mean Pyramid where the image is broken down into lower resolution components and the motion vectors for this lower resolution image are computed and propagated down the pyramidal structure of better and better resolution images.

On the whole there are a very large number of algorithms for block based motion compensation. This report includes a study of some of the main algorithms. We have tried to implement some of these and the results are discussed.

3.2 Block Matching Algorithms

The idea behind block matching is to divide the current frame into a matrix of 'macro blocks' that are then compared with corresponding block and its adjacent neighbors in the previous frame to create a vector that stipulates the movement of a macro block from one location to another in the previous frame. This movement calculated for all the macro blocks comprising a frame, constitutes the motion estimated in the current frame. The search area for a good macro block match is constrained up to p pixels on all fours sides of the corresponding macro block in previous frame. This 'p' is called as the search parameter. Larger motions require a larger p, and the larger the search parameter the more computationally expensive the process of motion estimation becomes. Usually the macro block is taken as a square of side 16 pixels, and the search parameter p is 7 pixels. The matching of one macro block with another is based on the output of a cost function. The macro block that results in the least cost is the one that matches the closest to current block. There are various cost functions, of which the most popular and less computationally expensive is Mean Absolute Difference (MAD). Another cost function is Mean Squared Error (MSE).Equations are given below.

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

where N is the side of the macro bock, C_{ij} and R_{ij} are the pixels being compared in current macro block and reference macro block, respectively.

Peak-Signal-to-Noise-Ratio (PSNR) is given by equation. It characterizes the motion compensated image that is created by using motion vectors and macro clocks from the reference frame.

$$PSNR = 10\log_{10}[\frac{(peaktopeakvalueoforiginaldata)^2}{MSE}]$$

3.3 Search Algorithms For Block-matching

3.3.1 The Full Search (FS) algorithm

In selecting a suitably matched block, the FS algorithm searches the entire search region for a block such that the BDM is a global minimum. If more than one block generates a minimum BDM, the FS algorithm selects the block whose motion vector has the smallest magnitude, in order to exploit the centre-biased motion-vector distribution characteristics of a real-world video sequence. To achieve this, checking points are used in a spiral trajectory starting at the centre of the search region. If the maximum displacement of a motion vector in both the horizontal and vertical directions is + d or -d pixels, the total number of search points used to locate the motion vector for each block can be as high as $(2d + 1)^2$. The spiral trajectory of the checking points used by the FS algorithm with the maximum displacement, d = 7, is shown in figure 3.1 below:

3.3.2 Three Step Search (TSS)

- Step 1: An initial step size is picked. Eight blocks at a distance of step size from the centre (around the centre block) are picked for comparison.

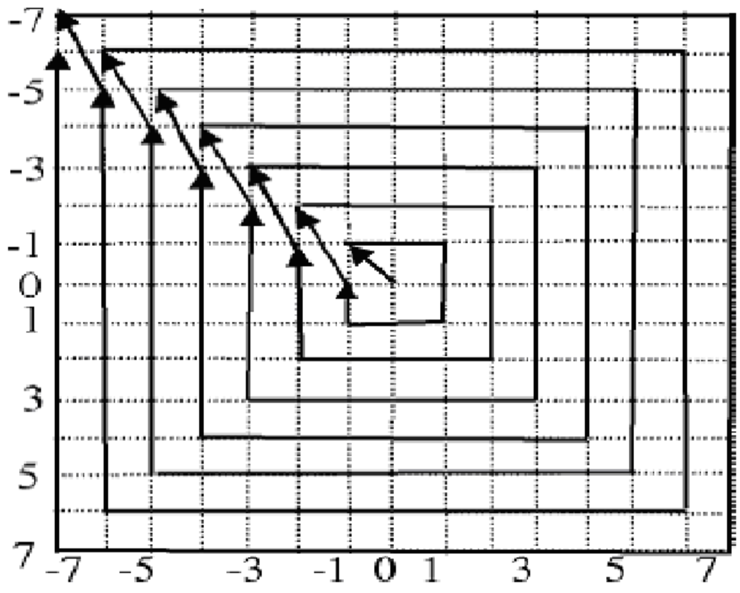

Figure 3.1: The spiral trajectory of the checking points in the FS algorithm

- Step 2: The step size is halved. The centre is moved to the point with the minimum distortion.
- Steps 1 and 2 are repeated till the step size becomes smaller than 1. A particular path for the convergence of this algorithm is shown below:

One problem that occurs with the Three Step Search is that it uses a uniformly allocated checking point pattern in the first step, which becomes inefficient for small motion estimation.[5]

3.3.3 Four Step Search (FSS)

The algorithm starts with a nine point comparison and then the other points for comparison are selected based on the following algorithm:[5]

- Step 1: Start with a step size of 2. Pick nine points around the search window centre. Calculate the distortion and find the point with the smallest distortion.

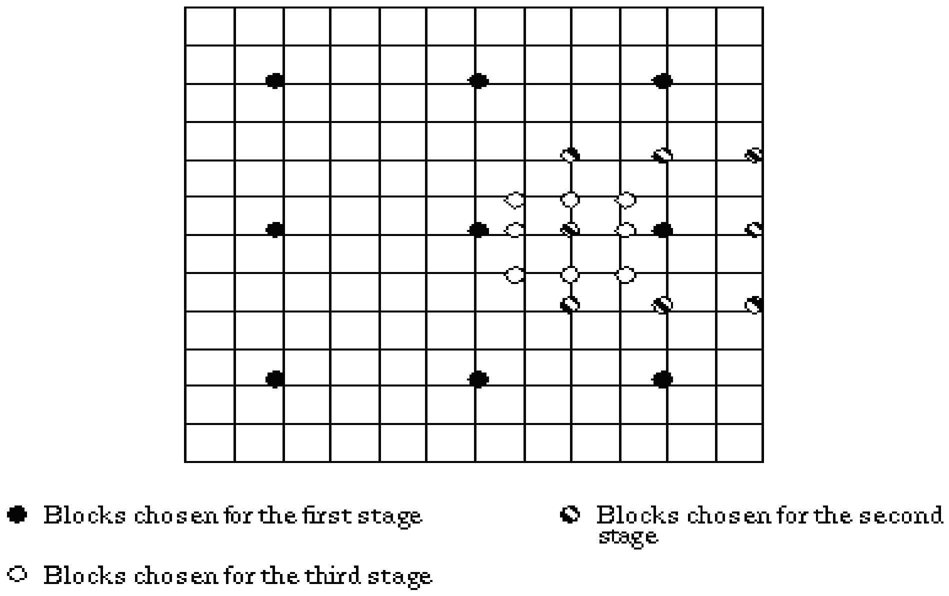
Blocks chosen for the first stage
Blocks chosen for the second stage
Blocks chosen for the third stage

Figure 3.2: Three Step Search Algorithm

If this point is found to be the centre of the searching area go to step 4, otherwise go to step 2.

- Step 2: Move the centre to the point with the smallest distortion. The step size is maintained at 2. The search pattern, however depends on the position of the previous minimum distortion.

  a. If the previous minimum point is located at the corner of the previous search area, five points are picked as shown in the figure 3.3.

  b. If the previous minimum distortion point is located at the middle of the horizontal or vertical axis of the previous search window, three additional checking points are picked. as shown in the figure 3.3. Locate the point with the minimum distortion. If this is at the center, go to step 4 otherwise go to step 3.

- Step 3 : The search pattern strategy is the same, however it will finally go to step 4.

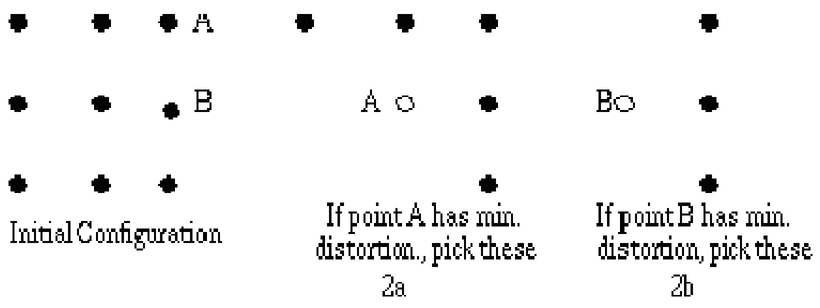
A
B
Initial Configuration
A
If point A has min. distortion., pick these
2a
B
If point B has min. distortion, pick these
2b

Figure 3.3: Four Step Search Algorithm-1

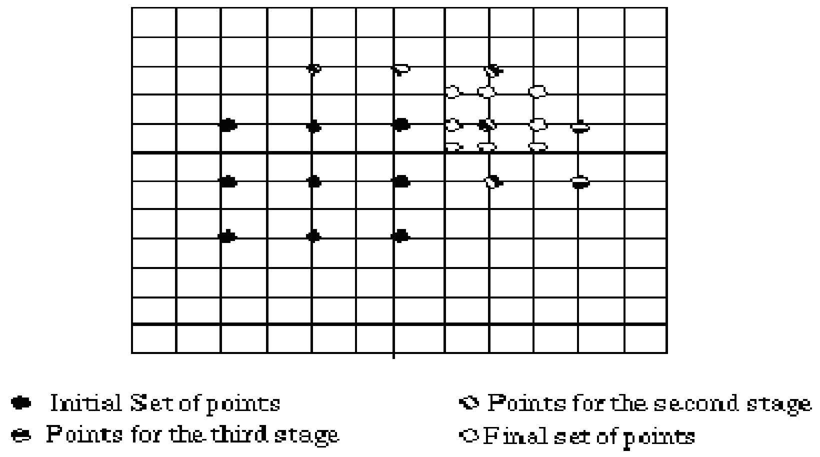

Figure 3.4: Four Step Search Algorithm-2

- Step 4: The step size is reduced to 1 and all nine points around the centre of the search are examined.

The computational complexity of the four step search is less than that of the three step search, while the performance in terms of quality is as good. It is also more robust than the three step search and it maintains its performance for image sequences with complex movements like camera zooming and fast motion. Hence it is a very attractive strategy for motion estimation

3.3.4 Binary Search Algorithm (BS)

This is also one of the algorithms that are very popular for motion estimation and in fact it is used for motion estimation by MPEG-Tool. The basic idea behind this algorithm is to divide the search window into a number of regions and do a full search only in one of these regions. It may be described as [5]:

- Step 1 : The MAD is evaluated on a grid of 9 pixels that include the centre, the four corners of the search window and four pels at the boundaries. The search window is divided into regions based on these points.
- Step 2: A full search is performed in the region corresponding to the point with the smallest MAD.

The convergence of the algorithm may be viewed in figure 3.5 below. The pels that lie between the dashed lines are never considered. Hence, although the Binary search requires fewer comparisons (the worst case scenario for this search window is 33 comparisons), it's performance is not very good because of this zone of pixels that are never considered.

3.3.5 Diamond Search Algorithm:

The diamond search is based on MV distribution of real world video sequences. It employs two search patterns in which the first pattern, called Large Diamond Search Pattern (LDSP) comprises nine checking points and form a diamond shape. The second pattern consists of five checking points make a Small Diamond Search Pattern (SDSP). The search starts with the LDSP and is used repeatedly until the minimum BDM point lies on the search center. The search pattern is then switched to SDSP. The position yielding minimum error point is taken as the final MV. The search process is shown in figure 3.6. DS is an outstanding algorithm adopted by MPEG-4 verification model (VM) due to its superiority to other methods in the class of fixed search pattern algorithms[6].

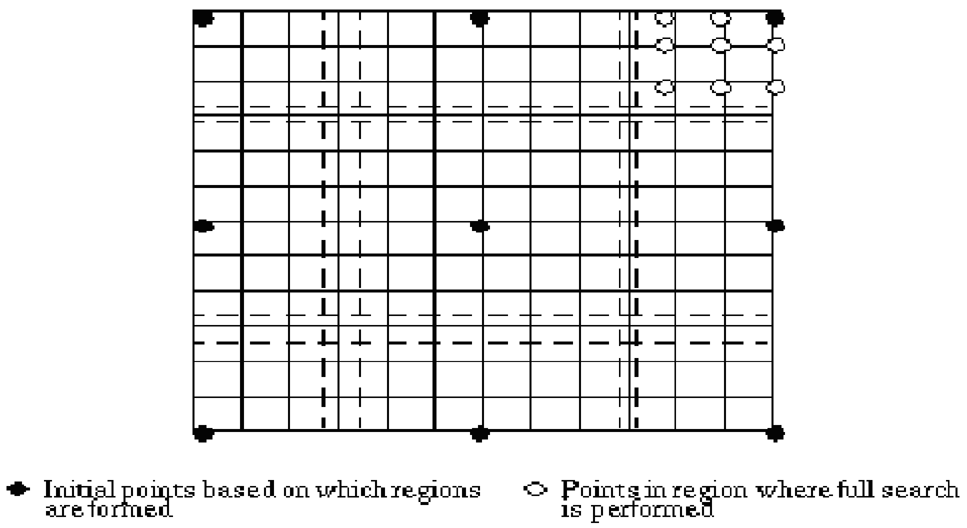
Initial points based on which regions are formed
Points in region where full search is performed

Figure 3.5: Binary Search Algorithm

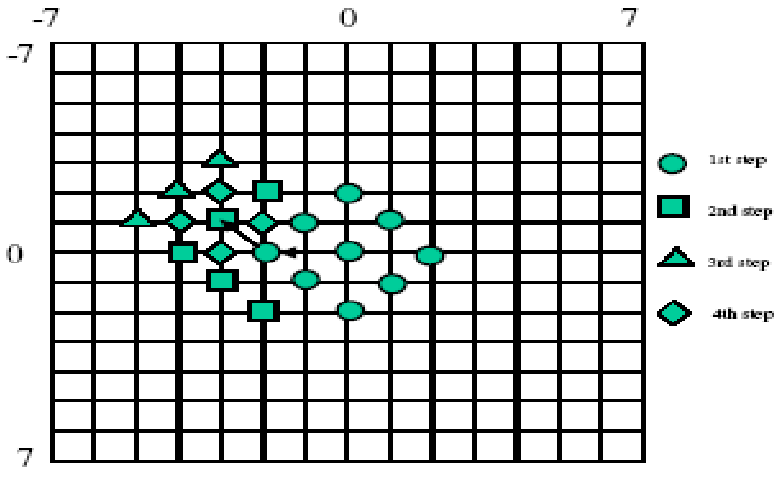

Figure 3.6: Diamond Search Algorithm

Chapter 4

Motion Vector and Motion Compensation

4.1 Description of a Motion Vector

MPEG achieves it its high compression rate by the use of motion estimation and compensation. MPEG takes advantage of the fact that from frame to frame there is very little change in the picture (usually only small movements). For this reason macro-block size areas can be compared between frames, and instead of encoding the whole macro-block again the difference between the two macro-blocks is encoded and transmitted.

Figure 4.1 demonstrates how forward motion compensation is achieved (backward compensation is done in the same way except a future frame in the display order is used as the reference frame.)

Macro-block ”x” is the macro-block we wish to encode, macro-block ”y” is its counterpart in the reference frame. A search is done around ”y” to find the best match for ”x”. This search is limited to a finite area, and even if there is a perfectly matching macro-block outside the search area, it will not be used. The displacement between the two macro-blocks gives the motion vector associated with ”x”[7].

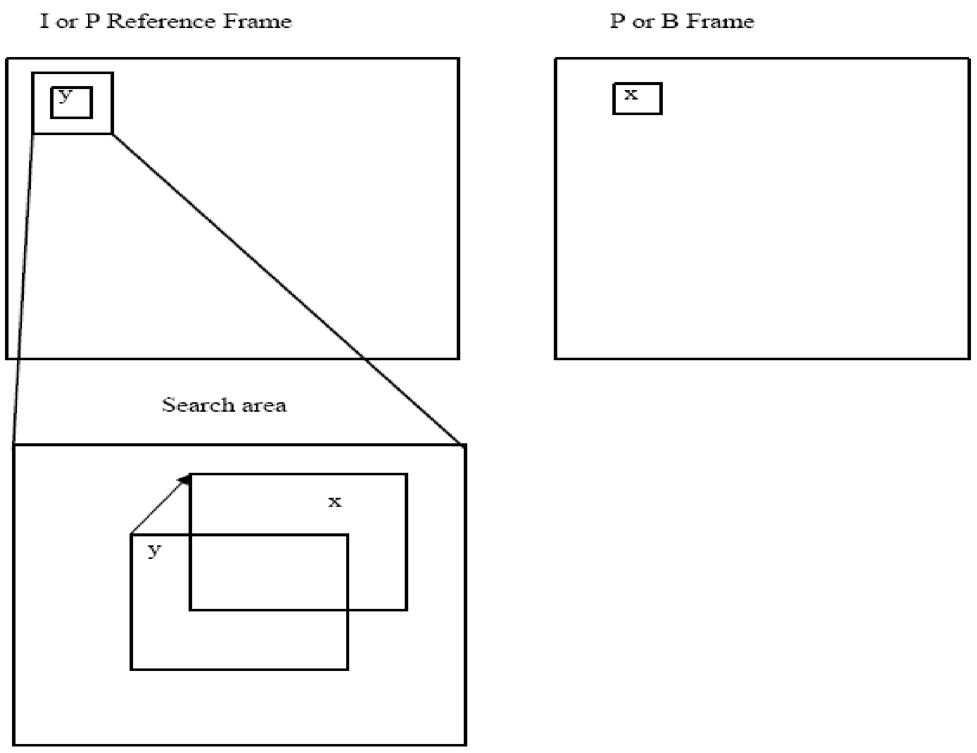

Figure 4.1: A forward predicted motion vector

4.2 Operation of the program

The motion vectors have to be stored in an order that will allow the motion from frame to frame to be calculated. First, the process of reordering the bit stream order to the display order is discussed. This is followed by a description of how selective vector storage allows this reordering[7].

4.2.1 Reordering the bit stream order to the display order

The frames do not come into the decoder in the same order as they are displayed. To reorder the frames to the display order the following procedure is used:

- If an I or P frame (lets call it ”1”) comes in it is put in a temporary storage future. I and P frames always come into the decoder before the B frames that reference them.
- ”1” is left in future until another I or P frame (”5”) comes in. The arrival of ”5” indicates it is ”1’s” turn in the display order. ”1” is taken out of future, put in the display order. ”5” is put in future until another I of P frame arrives.
- All B frames are immediately put in the display order.
- At the end whatever frame is left in future is taken out and put in the display order.

A typical bit stream is shown in figure 4.2, the display order number of each frame is also given. Note this process does not use the display order number. It is given to clarify what is happening.

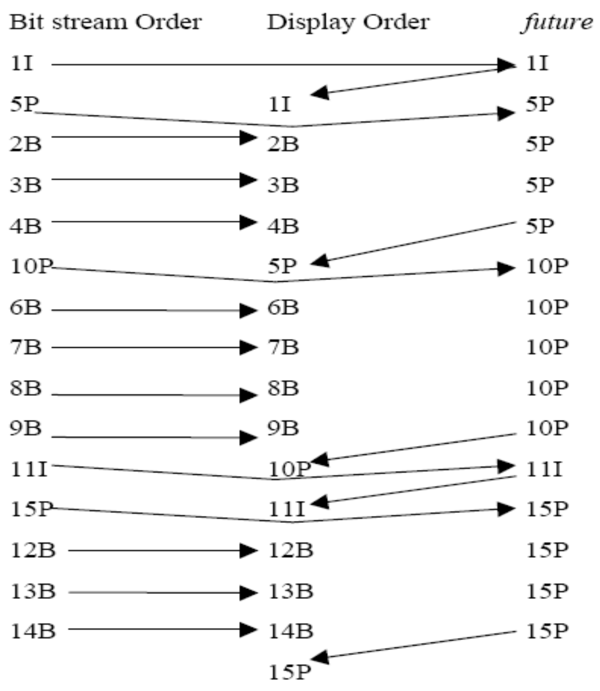

Figure 4.2: Converting from bit stream order to display order

4.2.2 Storing the motion vectors

For ease of handling, it was decided that the motion vectors should be stored in two dimensional arrays. The size of the array corresponds to the frame size (in macro-blocks). The position of the entry in the array corresponds to the macro-blocks position in the frame. There is a separate array for the two components of the vector, one for the right component and one for the left component. To allow the storage of all the vectors that may be present in a frame, four arrays have to be created. Two arrays are needed for the storage of the forward predicted vectors, and two for backward predicted vectors.

To find the motion from one frame to another, a record of the motion vectors in the previous frame has to be kept. This means four more arrays have to be created. Finally the motion vectors in a P frame have to be stored until it is the P frames turn in the display order. As a P frame can only have forward predicted vectors, only two arrays need to be created. The names of all the arrays used in this project are given below[7]:

Array Name	**Function Of Array**
FutureRight FutureDown	Store the motion vector in a p frame until It is the P frames turn in the display order
PresentForwardRight PresentForwardDown PresentBackwardRight PresentBackwardDown	Store the motion vectors of the present frame in the display order
PastForwardRight PastForwardDown PastBackwardRight PastBackwardDown	Stores the motion vectors of the previous frames in the display order

Table I: Array and Functions

4.2.3 Explanation of the Algorithm

If an I frame comes into the decoder, all the vectors in future are reset to zero (after the values that were in it are taken out and put in present), as an I frame has no motion vectors. If a P frame comes in all its vectors have to be stored in future (after the values that were in it are taken out and put in present). If a B frame comes in, all its vectors have to be stored in present. However present has two types of vector; presentForward and presentBackward. A diagram of where the vectors are stored is given in figure 4.3

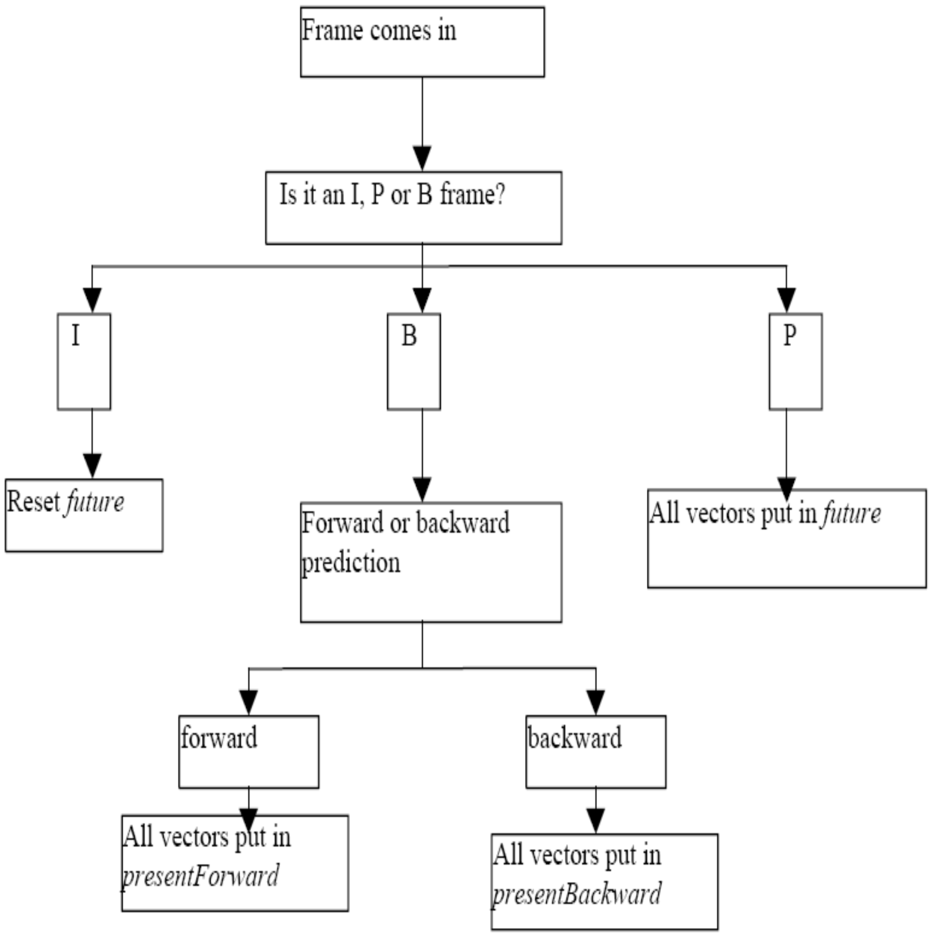
Frame comes in
Is it an I, P or B frame?
I
B
P
Reset *future*
Forward or backward prediction
All vectors put in *future*
forward
backward
All vectors put in *presentForward*
All vectors put in *presentBackward*

Figure 4.3: Diagram of where the motion vectors for the different frames are stored

4.2.4 Alterations made to the decoder

The processes of inputting the motion vectors into the correct arrays and reordering the frames into the display order were incorporated into the decoder. The end result was that the motion vectors for the present frame in display order are in presentForward and presentBackward. While the motion vectors for the previous frame in the display order are in pastForward and pastBackward. A flow chart of the program is given in figure 4.4.

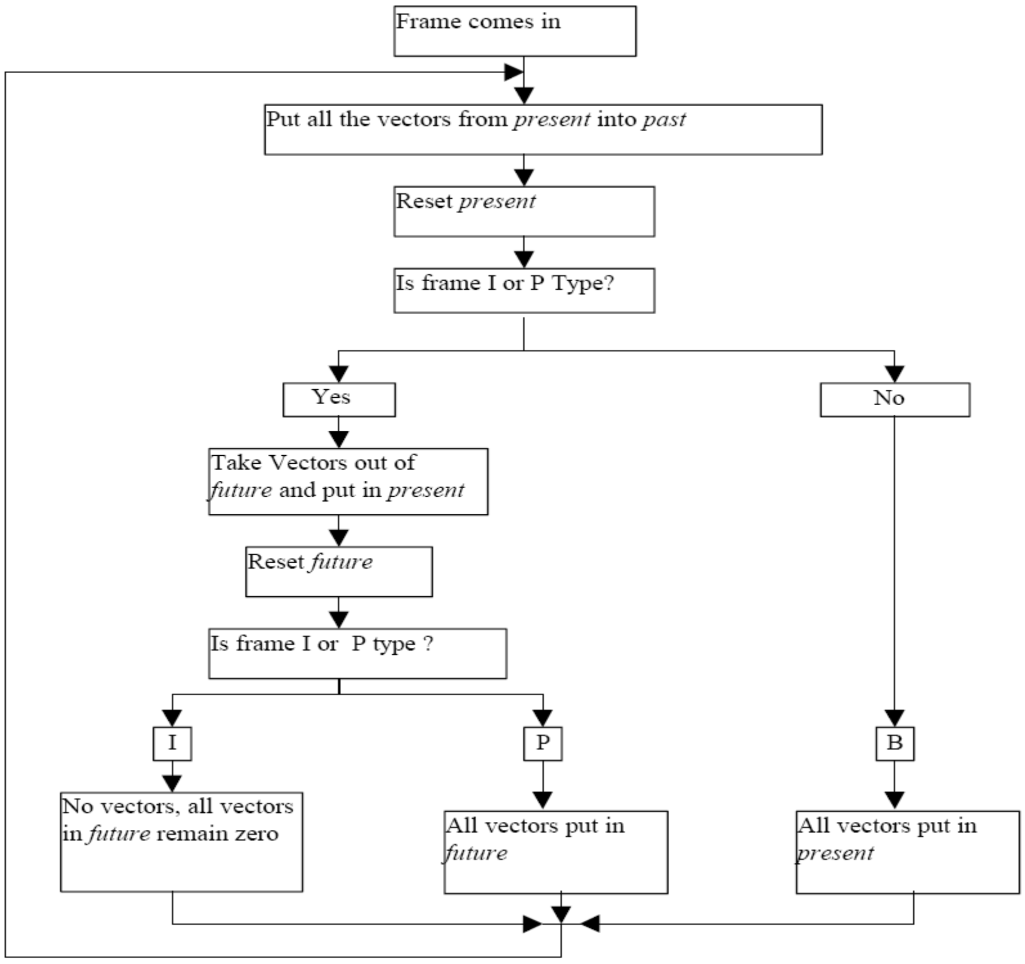
Frame comes in
Put all the vectors from *present* into *past*
Reset *present*
Is frame I or P Type?
Yes
No
Take Vectors out of *future* and put in *present*
Reset *future*
Is frame I or P type ?
I
P
B
No vectors, all vectors in *future* remain zero
All vectors put in *future*
All vectors put in *present*

Figure 4.4: Flow chart of the operational program

4.3 Finding the motion from frame to frame

To find the motion from frame to frame, the motion vectors in the present frame are subtracted from the vectors in the previous frame. However, depending on what type of frame (I, P, or B) is in present and past, not all of the arrays can be used. An explication of this is given below.

A vector defines a distance and a direction, it does not define a position. We have to know the vectors initial position (reference point) to find all the motion from frame to frame. Only vectors with the same reference point can be subtracted from each other. To illustrate, lets take the simple example of "x" moving across a portion of the screen as shown in Figure 4.5

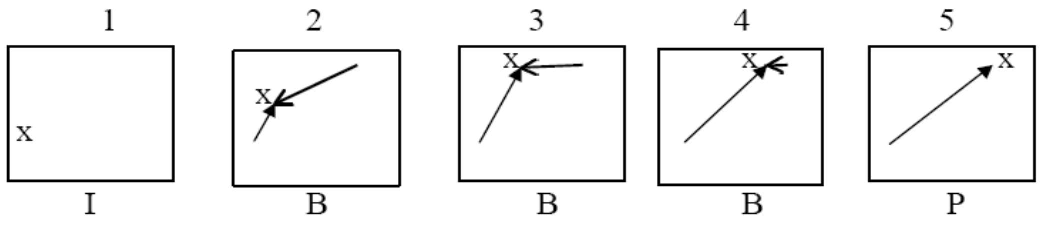

Figure 4.5: Motion vectors associated with a moving picture

The arrows represents a forward vector and represents a backward vector [Note a forward vector does not have to be pointing forward, and a backward vector pointing backward. It is just the naming convention for whither the reference frame in the past (forward) or future (backward)].

The values for the vectors are given below:

In the first frame there are no motion vectors

Frame 2:

forwardRight = 2; forwardDown =-3; forward=(2, -3)

backwardRight = -7; backwardDown = 5; bsckward=(-7, 4)

Frame 3:

forward = (4, -6)

backward = (-5, 1)

Frame 4:

forward = (7, -7)

backward = (-2, 0)

Frame 5:

forward = (9, -7)

Transition 1:

To find the motion in the transition from frame one to frame two, we can only use the forward vector. The backward vector has no reference in the I frame. The motion is just (2, -3)

Transition 2:

Here, the forward and backward vectors can be used as both forward vectors have the same reference point and both backward vectors have the same reference point.

presentForward -pastForward = forward motion

(4, -6) - (2, -3) = (2, -3)

presentBackward -pastBackward = backward motion

(-5, -1) - (-7, 4) =(2, -3)

To find the total motion avarage the two results motionRight = (2+2)/2 = 2

motionDown = (-3+-3)/2 = -3

Total motion = (2, -3)

Note in this example the forward motion will always equal the backward motion but this is not usually the case in video.

Transition 3:

forward (7, -7) - (4, -6) = (3, -1)

backward (-2, 0) - (-5, 1) = (3, -1)

Total motion (3, -1)

Transition 4:

Both the forward and backward vectors can be used here. Both forward vectors are referenced to the same point and, as the B frames backward vector is referenced to the P frame. The P frame is said to have a zero backward vector.

forward (9, -7) - (7, -7) = (2, 0)

backward (0, 0) - (-2, 0) = (2, 0)

Total motion (2, 0)

The motion for the sequence is: (2, -3), (2, -3), (3, -1), (2, 0)

4.3.1 Considerations that have to be taken into account-Frame level

Table 4.3.1 shows which types of vector can be subtracted depending on what type of frame is in past and present.

- **I Frame to B or P Frame:** When going from an I frame to a B or P frame only the forward motion vectors can be used. The P frame will only have forward vectors, the B frames backward vectors cant be used as they have no reference

Past	**Present**	**Vector types that can be subtracted**
I	B or P	Forward only
I	I	None
P	B or P	forward only
P	I	None
B	B or P	forward and backward
B	I	backward only

Table II: Vector types that can be used in the transition from frame to frame

in the I frame.

- **I Frame to I frame:** There are no vectors present in either frame.
- **P Frame to P or B frame:** None of the backward vectors in the B frame have a reference in the P frame. Therefore only forward vectors can be used.
- **P Frame to I Frame:** The forward vectors in the P frame do not have a reference in the I frame. No motion can be found.
- **B Frame to B or P Frame:** Both forward and backward vectors can be used as both have the same reference point from frame to frame.
- **B Frame to I Frame:** Only the backward vectors are referenced in the I frame.

4.3.2 Considerations that have to be taken into account-Macro-block level

All the different types of macro-block that can be present in a frame were described. Each macro-block in a B frame does not have both forward and backward vectors. Some macro-blocks will only have either a forward or backward vector. Other macro-blocks will have no vector at all, either because it is an Intra macro-block, or it is a skipped macro-block.This complicates the process of finding the motion from frame to frame even further.

It is not a simple matter of subtracting all the values in one array from all the values in its corresponding past array. A more accurate representation of ”x” moving across a portion of the screen may be as shown if Figure 4.6

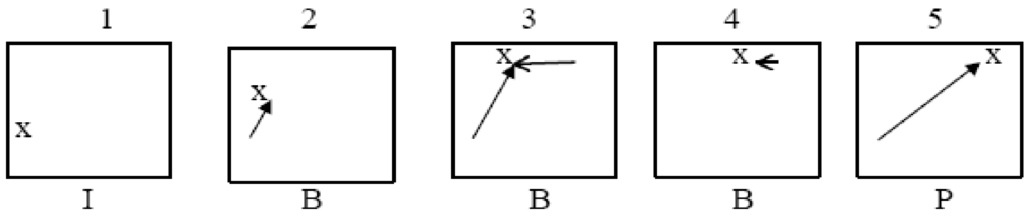

Figure 4.6: Realistic version of vectors associated with a moving picture

In this example the transition from frame 1 to frame 2 can be calculated as before. If the second transition is calculated as before we get:

forward motion:

(4, -6) - (2, -3) = (2, -3)

backward motion:

(-5, 1) - (0, 0) = (-5, 1)

Total motion = (-1.5, -1)

This result is incorrect.

To get the correct result, only the forward motion can be used. Similarly only the backward motion is used for the third transition. The motion for the final transition can not be found because there is only a backward vector in frame 4 and only a forward vector in frame 5. Only similar types of vector can be subtracted from each other.

Below are further rules to complement the rules that were established in table

- Only if there is a similar type of vector (forward, backward or both) present in both frames can the motion be found.

- A reference frame is said to have all vectors equal to (0, 0)
- If there is a skipped macro-block in the present P frame, there is zero motion for that transition.
- If there is a skipped macro-block in the previous P frame, the motion for that transition cant be calculated. An exception to this is if there is also a skipped macro-block in the present P frame in which case the motion will be zero.
- If there is an Intra macro-block in either the present or previous frame, the motion for that transition cant be calculated.

Chapter 5

Implementation, Results and Analysis

All algorithms are implemented in MATLAB and computes average number of search for macro-block, peak-signal-to-noise-ratio(PSNR) and motion vectors.
Implemented Results are shown below:

5.1 Average number of search for macroblock

Three step search(TSS), Four Step Search(FSS), Diamond Search(DS) and Adaptive Root Search(ARS) are implemented and the comparison of average number of search for macro-block computed by different algorithms is shown in figure 5.1

The graph clearly indicates that Adaptive root search has minimum average number of search for macro-block and three step search has maximum average number of search for macro-block. TSS take much time to search macro-block

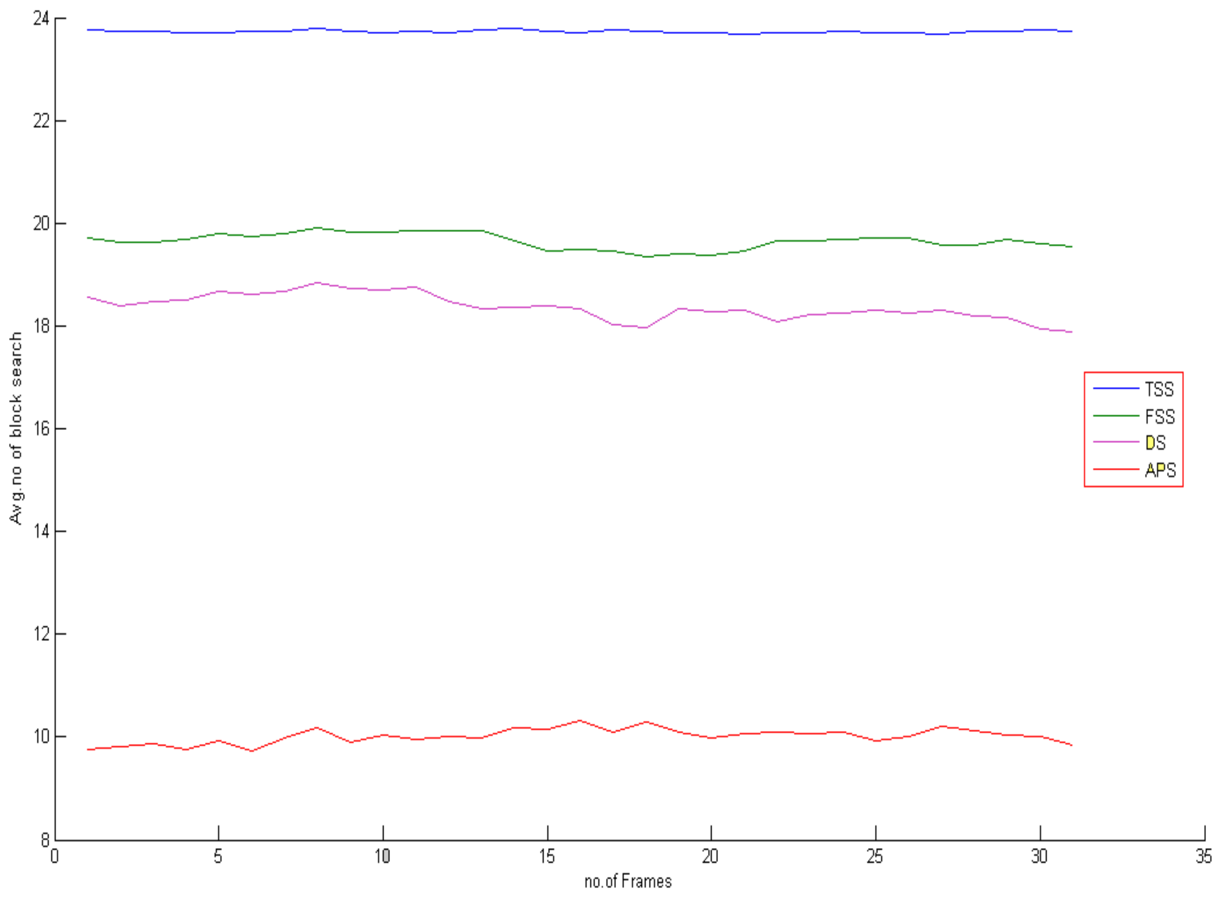

Figure 5.1: Average No.Of Search For macroblocks

Now in comparison with full search algorithm, TSS, FSS, DS and ARS are better in macroblock comparison and takes less time to search macro-blocks but full search algorithm find accurate macro-block than all others.

5.2 Peak-Signal-to-Noise-Ratio (PSNR)

PSNR ratio of all algorithms is shown in figure5.2 and from that graph it is clear that PSNR ratio of TSS is very less compared to all other algorithms and full search algorithm having very high PSNR ratio.

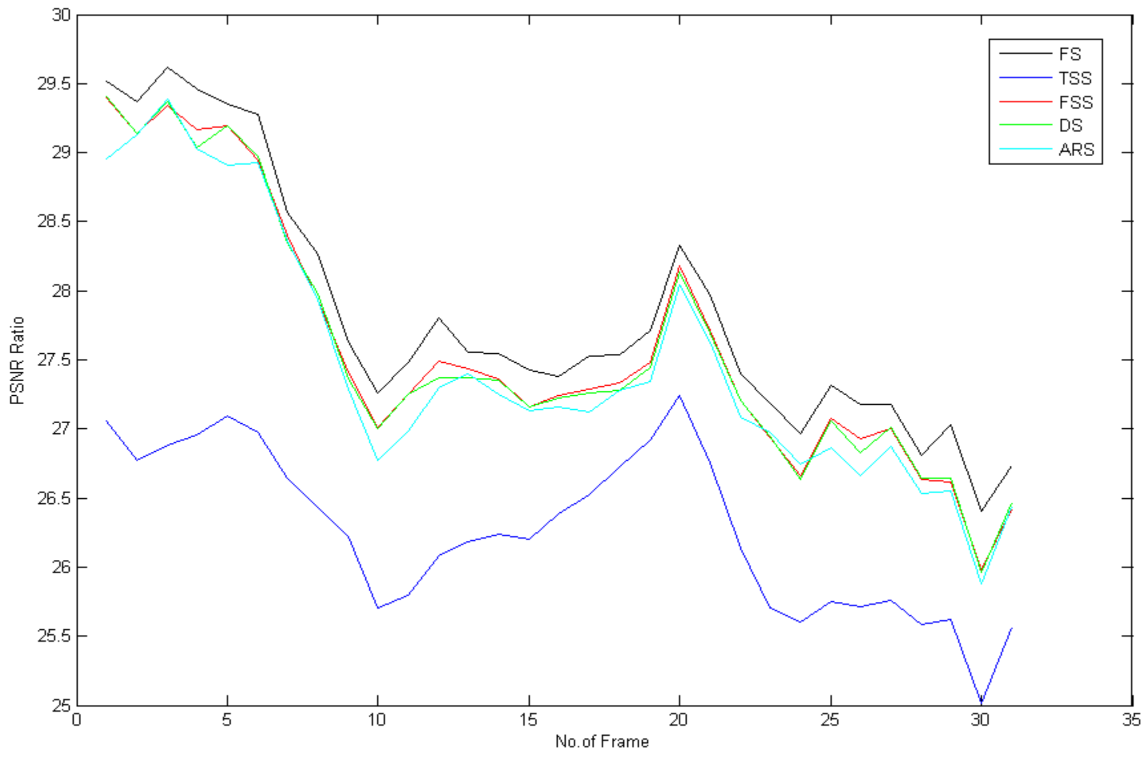

Figure 5.2: peak-signal-to-noise-ratio (PSNR)

5.3 Motion Vectors

To make sure the correct vectors are being given out they had to be plotted, this was done in MatLab. There is a function in MatLab called quiver that allows vectors to be plotted. To use the function first a grid of the screen has to be set up this is done with the command:

```
≫[x y] = meshgrid(1:1:22,1:1:18);
```

The 22 is the number of columns and 18 is the number of rows. The u and v values from the output file are then entered.

The command:

```
≫quiver(x,y,u,v);
```

plots the vectors.

If the vectors are all very small you can use:

```
≫quiver(x,y,u,v,s);
```

s scales the vectors to the size you want.

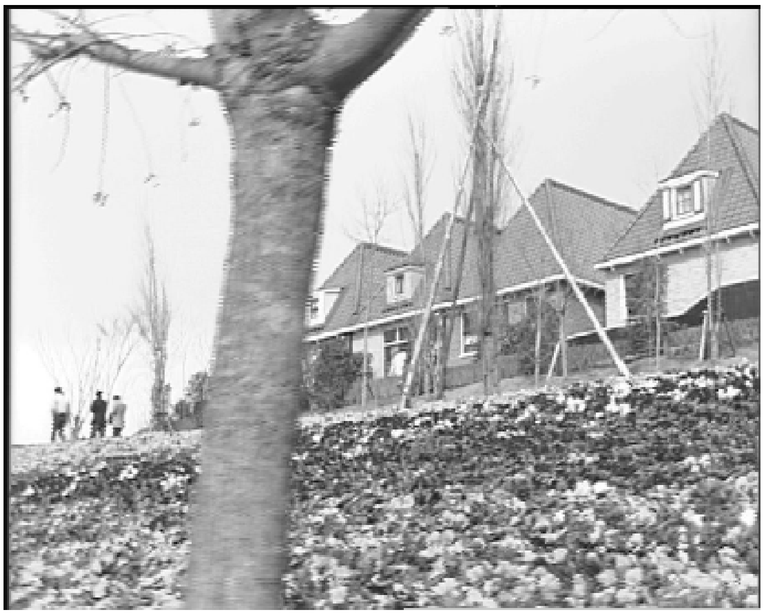

Figure 5.3: 1st Frame

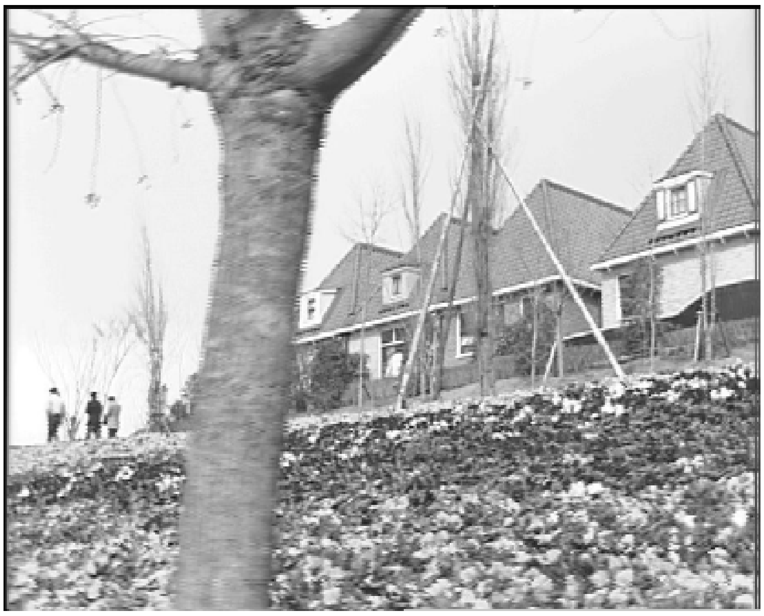

Figure 5.4: 2^{nd} Frame

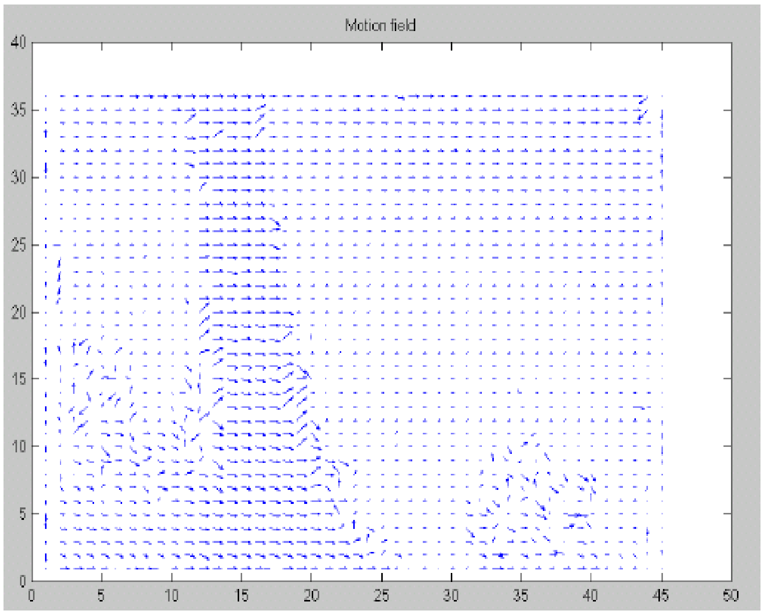

Figure 5.5: Motion Vector of that two frames

Figure5.3 and figure5.4 are two subsequent frames and figure5.5 is the Motion Vector of these two frames.

Now the next stage is to predict next frame using motion vector. Figure5.6 and figure5.7 are two subsequent frames and figure5.8 shows motion vector. figure5.9 shows estimated next frame by using motion vector of previous two images.

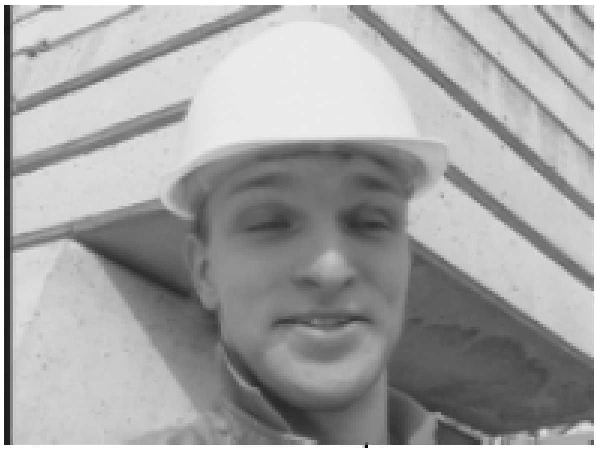

Figure 5.6: Foreman 1st Frame

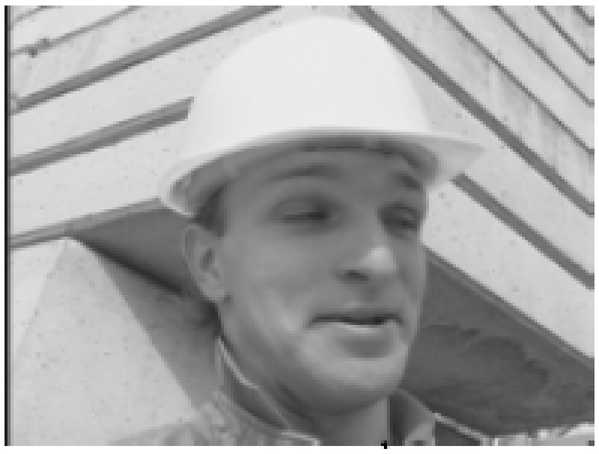

Figure 5.7: Foreman 2nd Frame

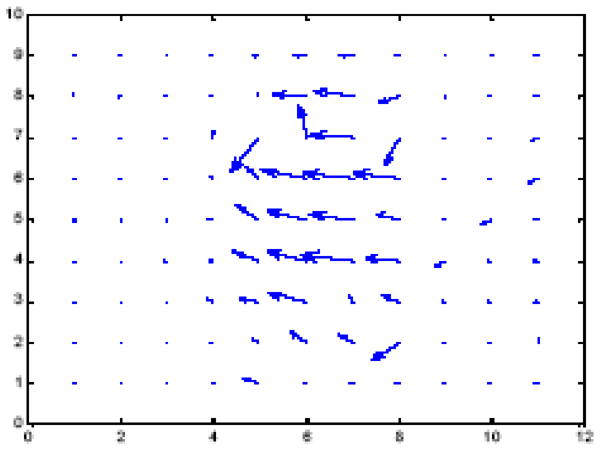

Figure 5.8: Motion Vectors

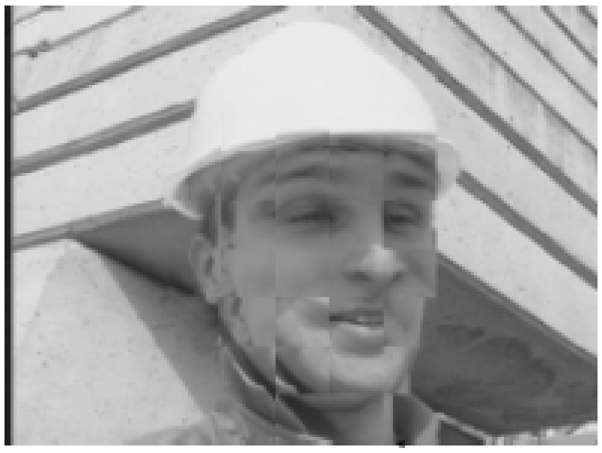

Figure 5.9: Estimated 3^{rd} Frame

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

This work has explored the theory of all motion estimation algorithms and examined basic features of motion estimation algorithms.

Even tough more commonly linked to lossy video compression, motion estimation is in fact a technique that goes beyond and allows for video processing and computational vision algorithms and applications. It allows a computer to detect movement as well as to perform comprehensive video sequence analysis, identifying scenes, and camera and object movements.Motion estimation is one technique that allows for a simple, yet effective, object identification scheme.

Five different algorithms for motion estimation are tested and compared Average number of search and PSNR ratio. It is concluded that fast full search(FS) motion estimation algorithm gives better result compared to all other algorithms but takes some what more time to search particular macro-blocks because it compare with each and every macro-blocks in search area.

6.2 Future Scope

In future this algorithm can be further improved by developing new search technique so that it can take less time to search macro-blocks and By changing some size of macro-blocks, quality of motion can also be further improved.

References

[1] D. Mitrovic, “Video compression,” tech. rep., University of Edinburgh.

[2] D. B. Brown, “Motion-based foreground segmentation,” tech. rep., Stanford University, March 13, 2000.

[3] L. M. Ho, “Variable block size motion estimation hardware for video encoders,” Master’s thesis, The Chinese University of Hong Kong, Nov 2006.

[4] *H.264 Motion Estimation Engine*, April 23,2008. XMP010.

[5] M. A. Deepak Turaga, “Search algorithms for block-matching in motion estimation,” mid-term project, 1998.

[6] “Block matching motion estimation algorithms,”

[7] J. Gilvarry, “Calculation of motion using motion vectors extracted from an mpeg stream,” technical report, School of Computer Applications Dublin City University, 20 September 1999.

Index