Power Efficient Motion Estimation Search Strategy For Video Codecs

Yilma Taye *

Abstract

This paper investigates the power efficient motion estimation for real time video codecs. The purpose of this project is to examine fast motion estimation search strategies based on block matching techniques while maintaining acceptable output Video Quality.

The algorithems are implemented in C-language.

Various block matching techniques are discussed & implemented: Full search, Three Step Search, Four Step Search and Block Based Gradient Decent Diamond Search.

Comparisons are made between the optimal searching technique, i.e. exhaustive searching (full search), and the above mentioned techniques using the PSNR (Peak Signal to Noise Ratio) and the number of SAD (Sum of Absolute Difference) calculations.

All the simulation results are shown in detail.

Keywords

Motion Estimation, MPEG, Video Compression

1 Introduction

In many multimedia applications, video data is often compressed according to the MPEG (Motion Picture Experts Group) standard. There is considerable temporal redundancy in image sequences, and the MPEG codec uses motion estimation (ME) to eliminate the temporal redundancy of video frames. ME is the process of estimating the motion vectors during the encoding process. This ME is the most time consuming task. When using the exhaustive (full) search in the encoding of video sequences and is also the one using the most computational power (60 to 80%). Many algorithms have been studied in efforts to reduce the complexity and power. It is known that there is a trade-off between power and image quality. ME may be used in various situations and the relation between demands for power or image quality will depend on those circumstances. For example, when running ME with a commercial power source there are no constraints on power availability and we will give more weight to image quality. But when running ME with a battery, power considerations can be more important than considerations of image quality. For instance, PCs with video cameras are used widely and the power considerations will change whether using indoors or outdoors. Balancing the needs for saving power and the needs for image quality, we must insure a suitable behaviour for the motion estimator.

This experiment is aimed to investigate the algorithms that take the advantage of reducing the number of search locations in ME with acceptable image quality.

2 Block Based Motion Estimation

In order to form a prediction of the current frame to be encoded, it is necessary to select a reference frame on which to base the prediction. The reference frame used is typically the previous frame in the video sequence, although some video compression standards (e.g. MPEG-1, MPEG-2 & MPEG-4) allow for predictions based on future frames. The difference between the reference and current frame is assumed to be solely due to the presence of motion in the video scene. This scene motion could be as a result of camera motion (e.g. pan, zoom, tilt, etc.) when the scene was being filmed, or due to the movement of objects present in the scene (e.g. a moving car, a football player, etc.). The approach taken is to estimate the motion present between the reference and the current frame and to subsequently "undo" this estimated motion in order to create a prediction. One approach to estimate the motion between two frames would be to track the movement of each pixel in the image between the frames. However, such a process is computationally very expensive and in fact may produce unreliable results due to the presence of noise. A less computationally intensive approach is to estimate the motion of groups of pixels. The major video compression standards divide the current image into blocks of pixels and estimate the motion for each block separately. In estimating the motion of a particular block, what is actually sought is the best match for the block in the reference frame. For this reason, this approach to motion estimation is sometimes called block matching. The current block is moved around its coordinates in the reference frame and at each visited point the difference between it and the block under consideration in the reference frame is calculated using a matching criterion. The matching process is restricted to a particular search area within the reference

*Yilma Taye, Lecturer, Electrical Engineering Dept, Faculty of Engineering, Bahir Dar University, Ethiopia

frame. The search area should be large enough to capture the present motion. Thus, very active video sequences, such as television sports broadcasts, require a larger search area than less active video content, such as a "talking head" video conferencing sequence. Block-based motion estimation is illustrated in Figure 1 where the search area is of size s pixels in both the horizontal and vertical directions relative to the upper left corner of the block. Clearly, such an approach assumes that the motion in the scene is purely translational in nature. This is not true in general but the resulting simplification is again a good trade off between the prediction accuracy and the computational complexity. Once the best match is found, the displacement between the current block coordinates and the coordinates of the best match is found. This displacement is termed as a motion vector. In Figure 1 the motion vector is denoted (Vx,Vy), indicating that it is represented by two components - a displacement in the horizontal direction and a displacement in the vertical direction.

In the context of a coding scheme, the individual horizontal and vertical components of the motion vector of each block undergo separate entropy encoding and are transmitted to the decoder. In the decoder, the motion vector is first decoded by reversing the entropy encoding, and subsequently used to construct a prediction for the associated block. The prediction is also formed at the encoder and the prediction residual between the predicted and current block is encoded using the DCT-based approach.

Given a motion vector for every block, in an image, the prediction residual is defined as:

$$E(x,y) = I_{curr}(x,y) - I_{ref}(x+v_x, y+v_y) \qquad (1)$$

where I_{curr} is the current image and I_{ref} is the motion compensated prediction of I_{curr}(x,y)

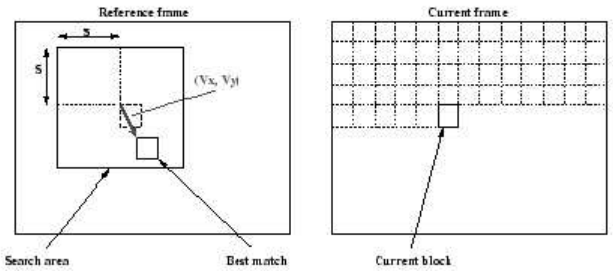

Figure 1: Block Matching

The search area is usually minimised to a certain area since it is predicted that an object can only move within a certain area between frames. There are a number of different possible matching criterion which can be used when calculating the difference between two blocks. Perhaps the popular one is the sum of the absolute differences(SAD). It is defined as:

$$\begin{aligned} \text{SAD} &= (B_{curr}, B_{ref}) \\ &= \Sigma_{i=1}^{16}\Sigma_{j=1}^{16} \mid B_{curr}(i,j) - B_{ref}(i,j) \mid \end{aligned}$$

Most of the motion estimation algorithms make the following assumptions in interframe coding:

1. Objects move in translation in a plane that is parallel to the camera plane, i.e. the effects of camera zoom, and object rotation are not considered.
2. Illumination is spatially and temporally uniform.
3. Occlusion of one object by another, and uncovered background are neglected.

3 Full Search (Exhaustive Search)

The most straightforward approach to searching is to simply search every position within the search area. Since the searching process is exhaustive, this guarantees that a minimum point in the matching function is found. However, the process is very computationally expensive.

4 Three Step Search (Logarithmic Search)

A logarithmic search is designed to reduce the number of positions, which need to be searched in order to locate a minimum in the matching function. The approach attempts to iteratively converge on the minimum point. The algorithm is outlined below where s denotes the search step size in pixels at each iteration of the algorithm.

1. Set s=2($\log_2$S-1). Denote the origin of the search by (x, y) and initialise it to (0,0).
2. Search the nine positions in the reference frame defined by s for the best match:

(x - s, y - s)	(x, y - s)	(x + s, y - s)
(x - s, y)	(x, y)	(x + s, y)
(x - s, y + s)	(x, y + s)	(x + s, y + s)

3. Set s=s/2. Set (x,y) to the best match position.
4. If s=0 the search is finished, else go to 2.

The most popular form of logarithmic search is when S = 8 so that, s_1=4, and s_2=2 and s_3=1. This approach is called the Three Step Search and is illustrated in Figure 2. In the Figure 2, initially the search is centred around the original location of the current block with a step size of 4. The

best match is found at position (4,-4) (relative to the search origin). In the second step, the search centers around this point with a step size of 2 and the best match is found at position(6,-6). Finally, in the third step, the search step size 1 and the best match can be assumed located at (7,-5). Thus, for this block the search ends.Using a logarithmic search algorithm is much more computationally efficient than performing a full search. However, fast algorithms for motion estimation such as this have a tendancy to get caught at local minima in the matching function. This is because the assumption underpinning such approaches is that the matching function increases monotonically as the search position moves away from the minimum value. This is not true in general. Due to the reduced number of positions searched, it is possible that an incorrect search direction will be found during the one of the steps. Since subsequent steps only search around this position the process will converge on this local minimum. If this happens, then the motion compensated prediction will yield a higher prediction error than that obtained with a full search algorithm.

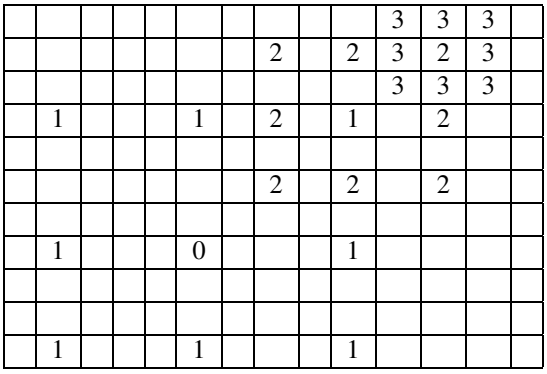

Figure 2: Three step search

5 Four Step Search

A Four step search algorithm was proposed to improve the efficiency of TSS especially for small motion vectors. For the first step, FSS checks 9 search positions with 1 pel distance in 5x5 search window. Then this 5x5 search window is moved in 2 pel steps in the direction towards best matched position. In the case of the center of the 5x5 search window being the best match, a local search at every integer -pel position in a 3x3 pel window is performed. In the worst case 4SS requires to check 27 matching positions, but the number of average search points lies between 17 and 22.

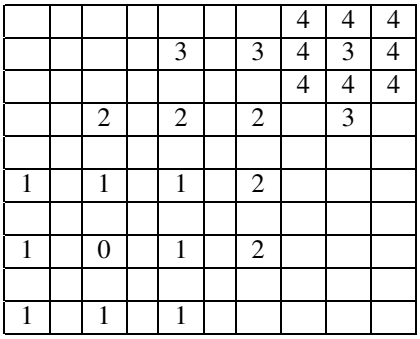

Figure 3: Four step search

6 Block Based Gradient Descent Diamond Search(GDDS)

The Diamond Search algorithm (DS) is mainly based on the assumption that motion vectors are in general center biased. The algorithm always starts searching from the center (0,0) of the search area,by examining nine(9) checking points as seen in figure 4 a. If the minimum is found at the centre, then four (4) additional checking points (figure 4 b) are examined and the search stops. Otherwise, depending on the position of the current minimum (figure 4 c and d) additional points will have to be examined. By considering the current minimum as the new center of this new large diamond that is created, the process iterates until the minimum is found to be in the center,where again the smaller diamond (the four additional checking points) are exmined.

The algorithm might actually have significant problems when coding non center-biased sequences.

7 Results

The coding performance is measured in decibels (dB) using the peak-Signal-to-Noise-ratio (PSNR) function and defined as:

$$PSNR = 20 * LOG_{10}(\frac{255}{\sqrt{MSE}}) \quad (2)$$

where MSE = Mean Square Error between the original image and the reconstructed one and is defined as:

$$MSE = \frac{1}{N * M} \sum_{i=1}^{N} \sum_{j=1}^{M} [I_{curr}(i,j) - I_{reconst}(i,j)]^2 \quad (3)$$

where I_{curr} is the original frame,and $I_{reconst}$ is the corresponding reconstructed frame of size N*M pixels,based on the obtained motion vectors. The error terms are not used in the frame reconstruction.

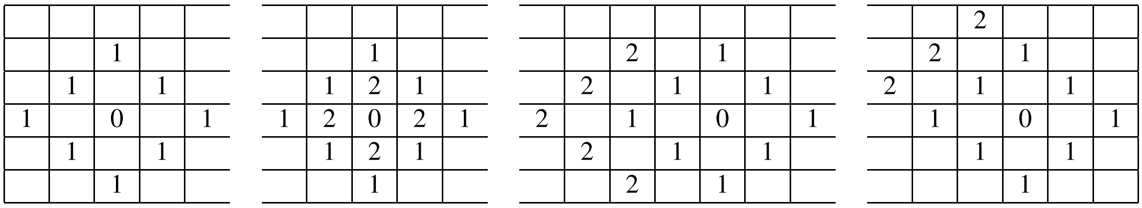

Figure 4: Gradient Descent Diamond Search a. b. c. & d.

The other criteria used to measure the coding performance is number of SAD calculations.

The simulations are carried out using cygwin bush compiler. The video test data sequences used are Foreman-qcif, & Coastguard-qcif (you can see the picture in the appendix).

Search Algorithm	Average Number of SADs Calculations	Average PSNR (dB)
Full Search	734416	30.9265
TSS	70727.88	28.1298
FSS	45619	27.966
GDDS	54752	28.117

Table 1: Summary of results obtained from coastguard test sequence

Table 1 shows average number of SADs calculations & the PSNR comparison among the algorithms. The results are exactly as expected with full search having the best performance in terms of PSNR, however, the others are performing almost as good. Looking at the PSNR values, there is a bigger reduction in the number of search locations for these algorithms: in particular, for GDDS and FSS. See Table 1above, which shows the average number of search positions for these algorithms.

Search Algorithm	Average Number of SADs Calculations	Average PSNR (dB)
Full Search	734416	32.818
TSS	222393.6	28.866
FSS	24981.75	29.327
GDDS	25981.75	27.856

Table 2: Summary of results obtained from foreman test sequence

Table 2 shows average number of SADs calculations & the comparison in terms of PSNR among the algorithms for the Foreman video sequence.
In general, the fast search algorithms saves more power since they reduce the computational complexity.

For any Block Based Motion Estimation, the algorithm should show a higher signal-to- noise ratio than when no motion compensation has been used. It is also expected to achieve the best quality from the full search than from any other search algorithm since it searches every pixel position for the best match within the search window. So all the algorithms implemented fall between these upper and lower limits as shown in Figure 5. The outcome of the calculation in this case is as expected from the theoretical considerations.

8 Conclusions

This paper looked at an existing algorithm designed to reduce the no of computations involved in different search strategies with reasonable value of PSNR. This task involved modelling and understanding the different strategies in detail. The results proved with expectations and the full details are given in the results section.
The results show that there is a very significant computational saving to be gained by using the fast search strategy. These fast search approaches implemented in this project exploit in various ways the assumption that the matching difference is monotonically increasing as a particular vector moves further away from the desired global minimum. However, theses methods run the risk of being trapped in local minima. One way to reduce the risk of being trapped in local minima is to use previous frames or blocks to provide a good initialisation of the search process. I would recommend from my observation for further investigation that there are fast search algorithms like Advanced Predictive Diamond Zonal Search (APDZS), Adaptive Search Length (ASL), Prediction Search algorithm (PSA) etc. Which not only can further decrease the computational complexity of the encoder but also more robust and can achieve better performance in terms of quality even for cases with large global motion.

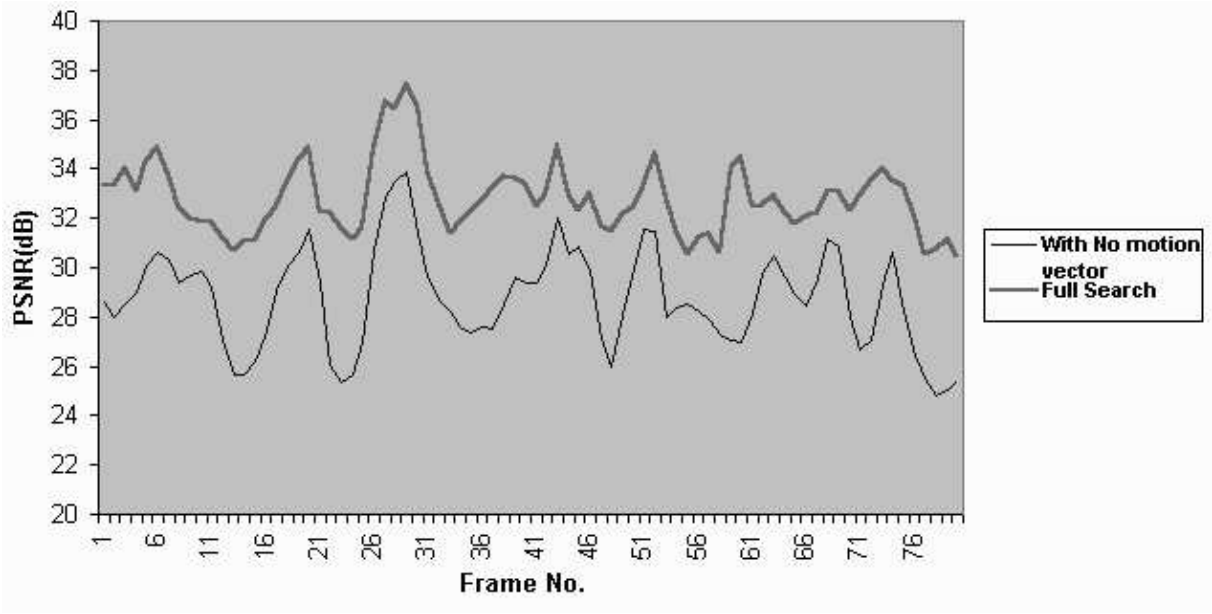

Figure 5: Simulation result for foreman with no motion compensation

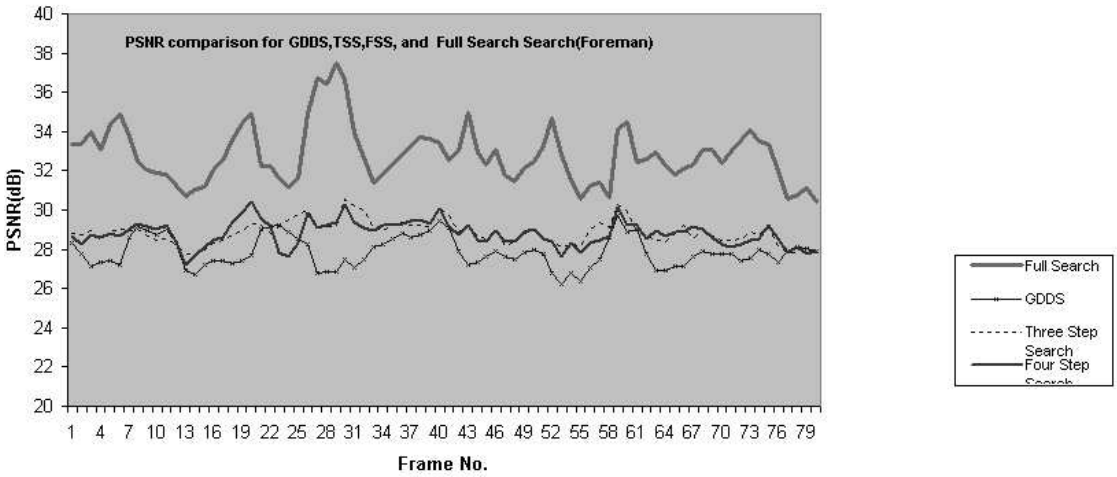

Figure 6: Simulation result for Foreman

Acknowledgement

The author is grateful to Dublin City University Instructors who boosted my knowledge, and to my wife Abeba for her continuous support and also to staff members of electrical Engineering department,Bahir Dar University, Ethiopia particularly Mr. Balakrishna singham for their great encouragement in my work.

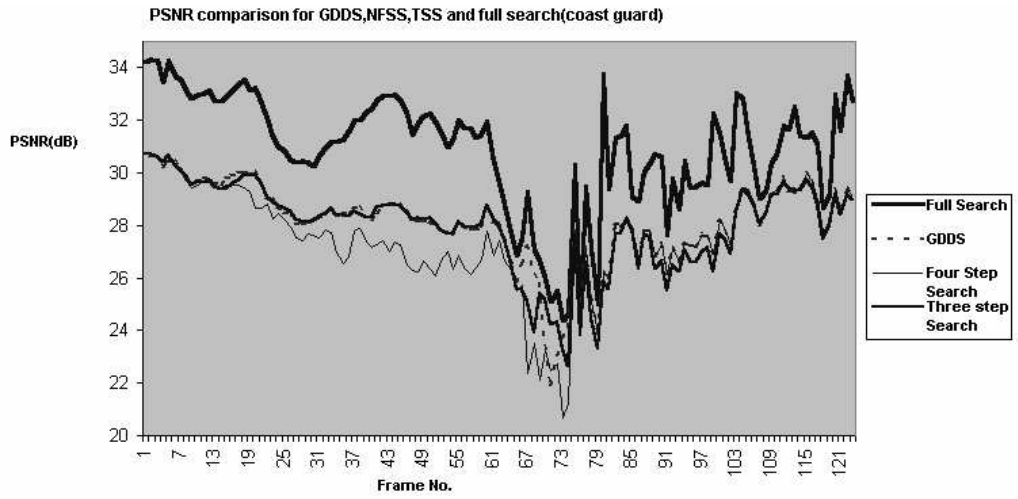

Figure 7: Simulation result for Coastguard

References

[1] *Motion Estimation Algorithms For Video Compression*, Borko Furht, Joshua Greenberg and Raymond Wastewater, Kluwer Academic Publishers.

[2] *MPEG-2*, John Watkinson, Focal Press.

[3] *Digital Video Processing*,A.Murat Tekalp, Prentice Hall.

[4] *The Lecture Notes For EE554 Course Module* Noel O'Connor and Noel Murphy, Dublin City University.

[5] *Principles of Digital Audio and Video* Arch C.Luther, Artech House

[6] *Algorithms,Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation*, Peter Kuhn, Kluwer Academic.

[7] *Digital Pictures Representation, Compression, and Standards*, Arun N.Netravali and Barry G.Haskell, Plenum Press.

[8] *A new diamond search algorithm for fast block matching*, Shan Zhu,Kai-Kuang Ma, submitted to IEEE Transactions on Image processing.

[9] *A novel Four-Step Search Algorithm for fast block matching*, Lai-Man Po, Wing-chung ma,IEEE Trans. on Circuits and systems for video-technology,vol. 7 no.6,Dec.1997,pp906-912.

[10] *Anew three-step search algorithm for block motion estimation*,R.Li,B.Zeng, M.L.Liu, IEEE Trans.on Circuits and systems for video-technology,vol. 4,no.4,Aug.1994,pp438-442.

Appendix

Figure 8: costguard.jpg

Figure 9: foreman.jpg