Evaluation of different block matching algorithms to motion estimation

M. R. KHAMMAR

Electrical and Electronic Department, Faculty of Engineering, University Putra Malaysia, 43400 UPM Serdang, Selangor
Khammar_m@yahoo.com

ABSTRACT

Motion estimation is an essential problem in digital video processing and computer vision. It prepares information needed for video analysis, such as video compression, video restoration, object tracking, and etc. Block matching techniques are the most popular and efficient of the various motion estimation techniques. This paper is a review of the block matching algorithms used for motion estimation. Four methods have been discussed and implemented range from the very basic exhaustive search to fast adaptive algorithms, and they are evaluated and compared together in terms of PSNR and search points per macro block for different block size and search area.

Keyword: motion estimation, block matching, full search, diamond search, adaptive rood pattern search

1. Introduction:

Video signal processing has become an extremely attractive area of research based on the recent advances of technology and its demand. Therefore, it spread in many applications such as, online monitoring of assembly processes, robot navigation and inspection, multimedia broadcasting, medical, remote sensing, and military. The processing of image sequences embodying motion of some objects is an valuable and challengeable section of video signal processing. It is well known that the issue of analyzing the motion of natural moving objects in a general scene is a tremendously difficult one. over the past, a considerable number of studies have been conducted on the motion estimation techniques. Motion estimation (ME) removes temporal redundancy within frames and thus provides coding systems with high compression ratio. Since ME package is usually the most computationally intensive part in a video encoder, efficient implementation of ME is quite essential and can affect the final result in any application.

In general, there are several approaches to motion estimation, and these can be divided as follows: region matching, gradient based methods and transform methods[1]. many efforts have been done and also many approaches have been proposed. Therefore, it is essential to review and compare them in terms of assessing advantages and disadvantages. It is not the role of this paper to cover the evaluation of the whole methods but we want to make a review and comparison in some region matching methods. Block matching approach is mostly selected as the ME package in video coding applications and is also implemented in most existing video coding standards because of its simplicity and good performance. The most important steps in block matching as follows[2]:

Step1: Subdivide every frame into square blocks such as 16x16.
Step2: Find one displacement vector for each block. (motion vector(MV))
Step3: Within a search range, find a best „match“ that minimizes an error measure.
Step4: Intelligent search strategies can reduce computation

There are many matching criteria such as MSE (mean square error) or Mean absolute difference( MAD) which is the most popular matching criterion and it is suitable to implement it for hardware realizations. The searching strategy is another important factor which is determine the efficiency and computational complexity and also time consuming of difference matching algorithms.

In this paper, the foundation of exhaustive search( or full search method)(ES), four step search(4SS), Diamond search(DS), and Adaptive rood pattern search(ARPS) discuss and the most properties of them compare together and their behavior evaluated based on implementation and measure on some image sequences. Therefore, section 2, explains principle of block matching algorithms and then clarify each above selected methods, section 3, illustrates the experimental results and finally, in section 4, conclusion are shown.

2. Fundamental of motion estimation algorithms

Motion estimation (MS) is the essential task in most of video processing applications. MS techniques have been implemented for the reduction of temporal redundancies. In fact, a high degree of redundancy exists between successive frames of image sequences because the number of capture images per second are considerable with the movement of objects in the given scene and, it is possible to achieve high compression ratio in video compression applications if we be able to detect and remove redundancy effectively. A natural way to exploit redundancy between frames is for current frame t determine predicted frame t from the frame t-1 or from frame t+1. motion estimation and motion compensation are used to predict frame t to be coded between
successive frames. Motion compensation works by estimating motion between two image frames. The motion is described by motion filed of motion vectors. Consequently, the prediction error is transmitted instead of the frame itself. This is the general fundamental of video coding process in order to get compression.

Figure 1: The general video coding encoder includes motion estimation block

Figure 1 shows block diagram of a general video coding system. On the other hand, motion estimation is useful step in many other applications such as object tracing, activities recognition, video restoration, and so on. In many available blotches and scratches detection methods, it is necessary to follow objects in consecutive frames in order to realize the corrupted area in each frame. For example in SDI_a algorithm,

\[ E_b = I_n(x) - I_{n-1}(x + d_{n-1}(x)) \]
\[ E_f = I_n(x) - I_{n+1}(x + d_{n+1}(x)) \]

\( n \) refers to current frame and \( I \) refers to intensity of a special pixel in frame \( n \), \( d \) is displaced pixel difference (DPD)

\[
b_{SDI_a} = \begin{cases} 1 & \text{for} \ (|E_b| > E_f) \text{AND} \ (|E_f| > E_b) \\ 0 & \text{otherwise} \end{cases}
\]

and \( E_t \) is user defined threshold, based on motion estimation we localize the specific pixel in three consecutive frames \( n-1, n, n+1 \), then, \( E_b \) and \( E_f \) calculate and according to the condition, it is possible to evaluate whether that pixel is corrupted or not. The performance of above algorithm to detect blotches is completely related to motion estimation. Any errors in motion estimation causes many false alarm in final result of detection procedure. It is necessary to highlight again which motion estimation is essential in many applications and the final output are affected seriously by it. Therefore, the evaluation of different available algorithms is an essential task in order to increase the accuracy and efficiency.

Block matching motion estimation

The main idea related to the block matching motion estimation is that, there is a high correlation between each pixel and its surrounding in a frame. Therefore, it is not necessary to assign motion vector to each pixel. It is enough to identify one motion vector per a block of pixels. In a typical frame work to block matching motion estimation, a frame is divided into blocks of \( nxn \). then, for the maximum motion displacement of \( P \) pixels per frame, the current block of pixels is matched against a corresponding block at the same coordinates but in the previous frame, within the square window of width/height \( n+2P \). The best match on the basis of a matching criterion yields the displacement.
There are some essential questions in this field, what are the affect of size, n in final result? And how the PSNR will vary regards to variation of n? How many criteria can be define to evaluate each candidate block and get the correct motion vector with high accuracy? How many ways are available to perform search strategy in order to find the best matching block? Does the algorithm guarantee to give the minimum error? Or minimum computational complexity? Fig. 2 shows a given block and its neighbors and also the procedure for seeking the best candidate block with minimum error. There are a space with P pixels around the block which is defined to search in the assumption area in order to find the best matching block based on the any types of criteria like MSE, or MAD. It is spread to four side of macro block, p is 7 here, but it can get different values and each one has its properties which we will cover in next section.

The bigger value for n means that the number of total blocks which need to process in each frame are decreased and for this reason, it is clear that the computational complexity will reduce, however, finding the high similarity and correlation in more pixels exists in the larger size of block is rarely happen so it means that it is difficult to find a good match block for the given block in its surrounding and hence, the PSNR will decrease based on increasing the size of n. In next section, we have shown some experimental results to show it.

In general, the search area for a good macro block match is constrained up to P pixels on all four sides of the corresponding macro block in previous frame. This *P* is defined as the search parameter. Large motions need a large P, and the larger the search parameter the more computationally expensive the process of motion estimation becomes. As it is shown, P is 7 in Fig. 2. In terms of block matching, two important subject are considerable, searching strategy and matching criteria.

**matching criteria**: Indeed, block matching is a subset of image matching and can be consider from a view perspective. In many image processing tasks, sometimes it is essential to examine two images or two portions of images on a pixel by pixel basis. These two images or two image regions can be choose from a spatial image sequence. The aim of the examination is to determine the similarity between the two images and two portions of images.

The similarity measure or correlation measure, is a key element in the matching process. On the other hand, instead of finding the maximum similarity, or correlation, an equivalent yet more computationally efficient way of block matching is to find the minimum dissimilarity, or matching error. In the literature, there are several types of matching criteria, such as the mean square error (MSE) illustrate by equation 1, and mean absolute difference (MAD) illustrate by equation 2, which are used most of the time. It is noted that the sum of square difference (SSD), or the sum of squared error (SSE), is essentially the same as MSE.

$$MSE = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} (C_{ij} - R_{ij})^2$$  \hspace{1cm} 1

$$MAD = \frac{1}{n^2} \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |C_{ij} - R_{ij}|$$  \hspace{1cm} 2

Where n is the size of the macro block, $C_{ij}$ and $R_{ij}$ are the pixels being compared in current macro block and reference macro block, respectively. And peak signal to noise ratio (PSNR) illustrate by equation 3 determine the motion compensated image which is produced based on motion vectors and macro blocks from the reference frame.

$$PSNR = 10 \log_{10} \left[ \frac{255^2}{MSE} \right]$$  \hspace{1cm} 3

**searching procedures**: searching strategy is another important subject to deal with the block matching. There are several methods available which are discussed below:

**full search (or exhaustive search) strategy**: this algorithm is the most computationally expensive block matching algorithm but it finds the best possible match and gives the highest PSNR in comparison with other methods. The algorithm measure the cost function at each possible location in the search area. There are a total $(2P+1) \times (2P+1)$ positions (Fig. 2) that need to be examined. The maximum similarity or minimum dissimilarity gives the best match. It also delivers good accuracy in searching for the best match. But because of a large amount of computation is involved, it is not suitable in real time video coding. The obvious drawback of this method is that the larger the search area gets the more computations it requires.

**Four step search strategy**: FS matches all possible displaced candidate blocks within the search area in the reference frame, in order to get the block with the minimum errors. Huge computation is essential in FS implementation and this is a big limitation to apply it to real time encoder. Many efforts have been done to develop the method, in order to keep the PSNR as big as possible and at some time reduce the computational complexity such as the three step search (3SS)[3], the 2-D logarithm search[4], the conjugate directional search[5], the cross search algorithm[6], etc. The four step search (4SS) algorithm has been proposed in [7].
has the better performance than well-known three step search and has similar performance to the new three step search (N3SS) in terms of motion compensation errors. In addition, the 4SS also reduces the worst-case computational requirements compared with N3SS.

Similar to N3SS, 4SS[7] also employs center biased searching and has a halfway stop provision. 4SS sets a fixed pattern size of \( S = 2 \) for the first step, the value of search parameter \( p \) is not important. Thus it looks at 9 locations in a 5x5 window. If the least weight is found at the center of search window the search transfers to fourth step. If the least weight is at one of the eight locations except the center, then we make it the search origin and move to the second step. The search window is still maintained as 5x5 pixels wide. Depending on where the least weight location was, we might end up checking weights at 3 locations or 5 locations. The patterns are shown in Fig 3. Once again if the least weight location is at the center of the 5x5 search window we transfer to fourth step or else we move on to third step. The third is exactly the same as the second step. The fourth step the window size is dropped to 3x3, it means that, \( S = 1 \). The location with the least weight is the best matching macro block and the motion vector is set to point \( s \) that location. This search algorithm has the best case of 17 or \((9+8)\) checking points and worst case of 27 or \((9+5+5+8)\) checking points.

![Figure 3: Diamond Search procedure. This figure shows the large diamond search pattern and the small diamond search pattern. It also shows an example path to motion vector \((-4,-2)\) in five search steps four times of LDSP and one time of SDSP.](image)

**Figure 4: four step search procedure. the motion vector is \((3,-7)\)**

**Diamond search (DS) strategy:** DS[8] algorithm is exactly the same as 4SS, but the search point pattern is changed from a square to a diamond, and there is no limit on the number of steps that the algorithm can take. DS uses two different types of fixed patterns, one is Large Diamond Search Pattern (LDSP) and the other is Small Diamond Search Pattern (SDSP). These two patterns and the DS procedure are illustrated in Fig 9. Just like in FSS, the first step uses LDSP and if the least weight is at the center location we jump to fourth step. The consequent steps, except the last step, are also similar and use LDSP, but the number of points where cost function is checked are either 3 or 5 and are illustrated in second and third steps of procedure shown in Fig 4. The last step uses SDSP around the new search origin and the location with the least weight is the best match. As the search pattern is neither too small nor too big and the fact that there is no limit to the number of steps, this algorithm can find global minimum very accurately. The end result should see a PSNR close to that of ES while computational expense should be significantly less.

![Figure 5: Adaptive Road Pattern: The predicted motion vector is \((3,-2)\), and the step size \(S = \text{Max}(|3|,|-2|) = 3\).](image)
Adaptive rood pattern search (ARPS) strategy: ARPS[9] algorithm makes use of the fact that the general motion in a frame is usually coherent, it means that, if the macro blocks around the current macro block moved in a particular direction then there is a high probability that the current macro block will also have a similar motion vector. This algorithm uses the motion vector of the macro block to its immediate left to predict its own motion vector. An example is shown in Fig. 5.

The predicted motion vector points to (3, -2). In addition to checking the location pointed by the predicted motion vector, it also checks at a rood pattern distributed points, as shown in Fig 5, where they are at a step size of $S = \text{Max}(|X|, |Y|)$. X and Y are the x-coordinate and y-coordinate of the predicted motion vector. This rood pattern search is always the first step. It directly puts the search in an area where there is a high probability of finding a good matching block. The point that has the least weight becomes the origin for subsequent search steps, and the search pattern is changed to SDSP. The procedure keeps on doing SDSP until least weighted point is found to be at the center of the SDSP. A further small improvement in the algorithm can be to check for Zero Motion Prejudgment, using which the search is stopped half way if the least weighted point is already at the center of the rood pattern. The main advantage of this algorithm over DS is if the predicted motion vector is (0, 0), it does not waste computational time in doing LDSP, it rather directly starts using SDSP. Furthermore, if the predicted motion vector is far away from the center, then again ARPS save on computations by directly jumping to that vicinity and using SDSP, whereas DS takes its time doing LDSP. Care has to be taken to not repeat the computations at points that were checked earlier. Pay attention also needs to be taken when the predicted motion vector turns out to match one of the rood pattern location. We have to avoid double computation at that point. For macro blocks in the first column of the frame, rood pattern step size is fixed at 2 pixels.

3. Experimental result
In order to evaluate four above algorithms, they have been implemented in MATLAB and apply to three different image sequences, ‘CARPHONE’, ‘FOOTBALL’, ‘MOBILE_CALENDAR’. figures 6, 7, 8 show the result, a plot of the average number of searches required per macro block for each clip and based on three different methods are presented, it is clear that the value is 225 for FS and we can not draw it in these plots. And also the PSNR comparison of the compensated images generated using the algorithms are shown.

Figure 6: Search points per macro block while computing the PSNR performance of Fast Block Matching Algorithms (left) and PSNR performance of Fast Block Matching Algorithms CARPHONE Sequence was used with a frame distance of 2.

Figure 7: Search points per macro block while computing the PSNR performance of Fast Block Matching Algorithms (left) and PSNR performance of Fast Block Matching Algorithms FOOTBALL Sequence was used with a frame distance of 2.
As shown by figures in previous page, 4SS, DS and ARPS come pretty close to the PSNR results of FS. While the FS takes on an average around 205 searches per macro block, DS and 4SS drop that number by more than an order of magnitude. ARPS further drops by a factor of 2 compared to DS. Although PSNR performance of 4SS, DS, and ARPS is relatively the same, ARPS takes a factor of 2 less computations and hence is the best of the fast block matching algorithms studied in this paper.

Conclusion
In this paper, we review four different block matching algorithms in order to compare them together. They are FS, 4SS, DS, and ARPS. FS gives the best PSNR and guarantee the minimum errors but the computational expensive is a big limitation so, on the other hand, the ARPS are shown its PSNR close to FS and also the computational complexity is reduced. Therefore, in four considerable algorithms ARPS is the best one to implement and save time processing and get the better PSNR.

References