Analysis of Block Matching Algorithms with Fast Computational and Winner-update Strategies

Ibrahim Nahhas and Martin Drahansky

Faculty of Information Technology, Brno University of Technology
Bozetechova 2, 612 66, Brno, Czech Republic

{inahhas, drahan}@fit.vutbr.cz

Abstract

Block matching for motion estimation has been widely used in video compression for efficient transmission and storage of video bit stream by reducing the temporal redundancy existing in a video sequence. The motion estimation is a process to predict the motion between two successive frames. This paper is primarily a review of the block matching algorithms using fast computational and winner-update strategies. The paper describes and analyses different types of block matching algorithms, namely Full Search (FS), Fast Computational of Full Search (FCFS), Three Step Search (TSS), New Three Step Search (NTSS), Three Step Search with Winner Update Strategy (WinUpTSS), Four Step Search (FSS) and Diamond Search (DS) algorithms.

Keywords: *block matching, motion estimation, FS, FCFS, TSS, NTSS, 4SS, DS*

1. Introduction

Image and video compression continues to be a very active field of research and development for more than 20 years. Many different systems and algorithms for compression have been proposed, developed and standardized. Video compression algorithms operate on removing redundancy in temporal, spatial and/or frequency domains. Figure 1 shows a simple diagram of video encoder, which consists of three main functional units: a temporal model, a spatial model and an entropy encoder [1].

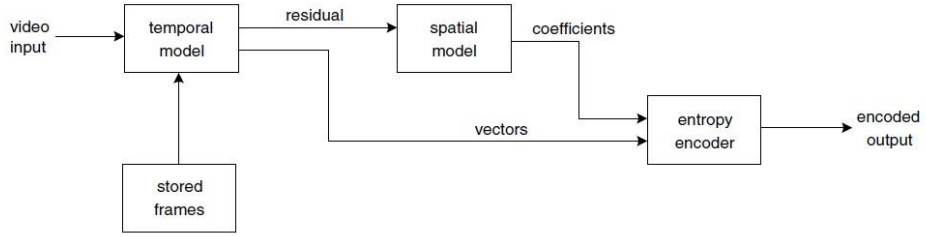

Figure 1. Block Diagram of General Video Encoder

In the figure, the input to the temporal model is an uncompressed video sequence. The main task of the temporal model is to reduce redundancy between transmitted frames by estimating the motion between successive frames. Changes between video frames may be caused by object motion, camera motion and lighting changes. It is possible to estimate the trajectory of each pixel between successive video frames. This will produce a field of pixel trajectories known as the optical flow. However, doing so is not considered as a practical method of mo-

tion compensation for several reasons. For example, the accurate calculation of optical flow is very computationally intensive and it would be necessary to send the optical flow vector for every pixel to the decoder. For a practical method of motion estimation, a frame is divided into non-overlapped, equally spaced, fixed size, small rectangular sections, called "blocks". All the pixels inside one block will have the same motion vector to compensate the movement inside the block in the current frame. Despite the simplicity of the block matching technique it is very efficient yet. It has been by far the most popularly utilized motion estimation technique in video coding. In fact, it has been adopted by most international video coding standards: ISO MPEG-1, MPEG-2 and MPEG-4, and ITU H.261, H.263 and H.264. The purpose of this work is to provide a comparative study of fast full block matching search algorithms, used in motion estimation.

2. Block Matching Algorithms

Block matching techniques based on segmentation of the current frame into blocks and the determination of all pixels inside the block have the same displacement vector. Motion vectors are estimated by finding the best-matched counterpart in the previous frame(s). The block size needs to be chosen properly. In general, the smaller the block size, the more accurate, but leading to more motion vectors to be estimated and encoded, which means an increase in both computation and overhead information. A block of size 16×16 pixels is considered to be a good choice – this has been specified in the international video coding standards such as H.261, H.263 and MPEG-1, MPEG-2 [2, 3]. Figure 2 illustrates the principle idea of block matching techniques, where the segmentation of an image frame at the moment t_n into non-overlapped $p \times q$ rectangular blocks is performed. Considering one of the blocks centered at (x, y), it is assumed that the block is translated as a whole. Consequently, only one displacement vector needs to be estimated for this block.

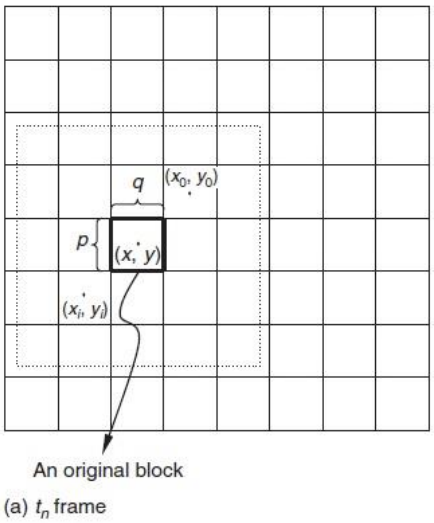

Figure 2. Principle Idea of Block Matching

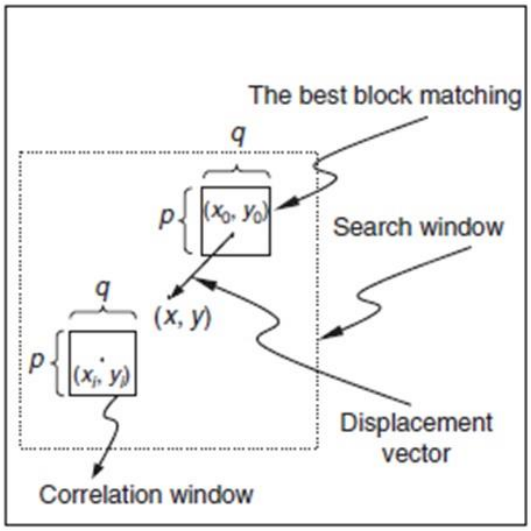

Figure 3. Matching Process

In order to estimate the displacement vector, a rectangular search window is opened in the frame t_n-1 and centered at pixel (x, y), as in Figure 3. A rectangular correlation window of the same size $p \times q$ is opened with the pixel located in its center, and a certain type of similarity measure (correlation) is calculated. After the matching process is being completed for all

candidate pixels in the search window, the correlation window corresponding to the largest similarity becomes the best match for the block under consideration in frame t_n. There are various cost functions to calculate the best matching, of which the most popular and less computationally expensive are the Mean Absolute Difference (MAD) [9], given in equation (i), the Mean Squared Error (MSE) [2], given in equation (ii) and the Sum of Absolute Difference (SAD) [4], given in equation (iii).

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

$$MSE = \frac{1}{N^2}\sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\left(C_{ij} - R_{ij}\right)^2 \quad \text{(ii)}$$

$$SAD = \sum_{j=0}^{N-1}\sum_{i=0}^{N-1}\left|C_{ij} - R_{ij}\right| \quad \text{(iii)}$$

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. The relative position between these two blocks (the block and its best match) gives the displacement vector. In the past, many techniques were developed to accelerate the block matching process. We classify these techniques into two basic categories:

- Category 1: reduce the number of positions searched.
- Category 2: reduce the computational cost of the matching error for each search position.

The work in this paper provides a revision of seven different types of block matching algorithms, which vary in the number of positions searched and computational cost of the matching error. A comparison among these algorithms regarding the PSNR quality, which is given in equation (iv), and computational time are considered.

$$PSNR = 10\ \log_{10}\left[\frac{(Peak\ to\ Peak\ value\ of\ original\ data)^2}{MSE}\right] \quad \text{(iv)}$$

2.1. Full Search (FS)-

Full Search [10] algorithm is the first and a simple method used for block matching. In an exhaust searching for the best matching, the correlation window is moved to each candidate position within the search, where the maximum number of checked points is 225. The minimum dissimilarity will give the best matching. Full search algorithm provides the highest PSNR, but at the same time suffers from a long computational time. For that an improvement is needed to maintain the PSNR and at the same time reduce the computational complexity. There are a number of block matching algorithms developed from the full search algorithm in order to reduce the searching time – this poses great challenges on the real-time codec implementation, either by having different search patterns or less number of searching points.

2.2. Fast Computational Full Search (FCFS)

Fast Computational Full Search [4] keeps the same quality of decompressed video as in FS algorithm, but reduces the computational time required to determine the matching macro-block from the reference frame to the macro-block in the current frame. In principle the algorithm first makes a simple check to detect whether a candidate block is possible to be the best matching one and stops the calculation of cost function between the pixels when the current uncompleted sum absolute value is greater than the previous calculated one. In this case, only the potential candidate blocks are further processed on detailed distortion calculation. By doing so, a large part of unnecessary computation for impossible candidate block can be avoided. The proposed algorithm works as follows:

- Step 1: Compute the sum of absolute difference (SAD_{min}) between the current macro-block MB and the macro-block at the same location in the reference frame. Consider this value to be the minimum SAD.
- Step 2: Compute the sum of absolute difference between pixels of next candidate block and the current block. If the new value exceeds SAD_{min} then stop computing the SAD for the rest of the pixels and move to step 3. Otherwise, continue the process to compute SAD for the rest of the pixels then move to step 4.
- Step 3: Move to the next macro-block in the search area and go back to step 2.
- Step 4: Assign the new SAD from step 2 to the SAD_{min} and move to the next candidate MB then go back to step 2.
- Step 5: The last SAD_{min} will give the matching macro-block.

Fast Computational Full Search algorithm improves the time to determine the matching block without compromising the quality of the Full Search which has the same number of searched points.

2.3. Three Step Search (TSS)

This algorithm was introduced by Koga *et al.*, [5]. It became very popular because of its simplicity and near optimal performance. It searches for the best motion vectors in a course to fine search pattern. The algorithm (as shown in Figure 4) is explained as:

- Step 1: An initial step size is chosen. Eight blocks at a distance of step size from the center (around the center block) are considered for the comparison.
- Step 2: The step size is halved. The center is moved to the point with the minimum distortion.
- Steps 1 and 2 are repeated until the step size becomes smaller than 1.

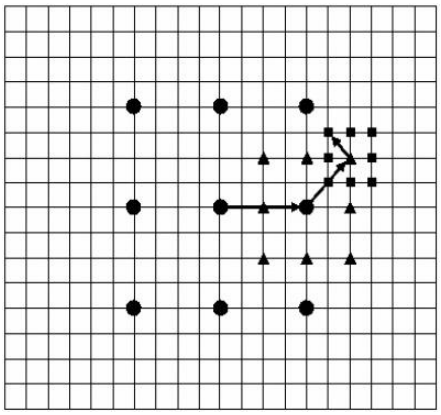

Figure 4. Three Step Search Algorithm

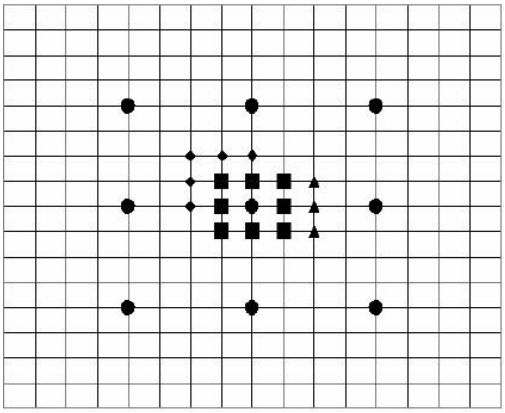

Figure 5. New Three Step Search Block Matching

The three step search algorithm has less number of checked points than FS. This means that is has less computational time and the number of checked points is reduced to 25. One problem the Three Step Search suffers from is that it uses a uniformly allocated checking point pattern in the first step, which becomes inefficient for small motion estimation. This algorithm is used in the MPEG video standard.

2.4. New Three Search Step Search (NTSS)

The improvement introduced by the New Three Step Search (NTSS) [6] is to achieve better estimation for the motion. It has the same steps as is TSS with adding eight points neighboring the center point of block to be checked in the first step, as shown in Figure 5. If the minimum is matched in one of these points the search will stop. Otherwise, the search continues as in the TSS. Although this process might need a minimum of 17 points to check every macro-block, it has the worst-case scenario of 33 locations to check.

2.5. Winner-Update Three Step Search (WinUpTSS)

The basic idea of the winner-update strategy is to avoid, at each search position, the costly computation of matching error when a lower bound larger than the global minimum matching error exists. But the winner-update strategy [11] explained below can help to solve this problem. At the beginning, we lay all the cards face down except the first one of each hand. The lower bound of the penalty score for each player is initialized as the value of the first card. Among all the players, only the temporary winner who has the minimum lower bound is allowed to turn up the face of the next card of his/her hand and update (increase) his/her intermediate lower bound of the penalty score. A new temporary winner is then selected and this process is repeated until the temporary winner has no card laid facedown and becomes the final winner. Table 1 lists the operations step-by-step to demonstrate the process for the example given in Figure 6. This comparison strategy, called the winner-update strategy, is a special case of the branch-and-bound strategy [12]. In block matching for motion vector estimation, the matching errors between the template block and most of the candidate matching blocks are usually very large compared to the minimum matching error. That is, most of the players hold cards with relatively large values except the winner and a few tough competitors. Therefore, by using this winner-update strategy, the number of the cards laid facedown, which implies the saved computations, can be enormous.

Table 1. Step-by-Step Calculation of the Penalty Scores of each Player. The Score Numbers in Boldface are the Temporary Minima after each Step

Operation	P_1	P_2	P_3	P_4	P_5
Initialize	**3**	12	4	8	6
Turn up #2 of P_1	12	12	**4**	8	6
Turn up #2 of P_3	12	12	**6**	8	6
Turn up #3 of P_3	12	12	9	8	**6**
Turn up #2 of P_5	12	12	9	**8**	10
Turn up #2 of P_4	12	12	**9**	21	10
Turn up #4 of P_3	12	12	11	21	**10**
Turn up #3 of P_5	12	12	**11**	21	20
P_3 is winner	12	12	11	21	20

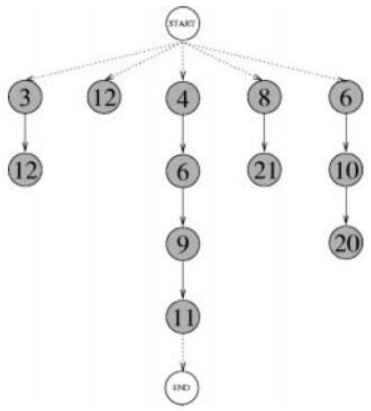

Figure 6. Winner-Update Strategy the Search Process Conducted in Table I

In each iteration, TSS algorithm needs to calculate and compare the matching errors of nine or eight search positions. Meanwhile the winner-update algorithm can be applied to efficiently find the one with the minimum matching error among these search positions. Because the winner-update algorithm can find the best matching position in each iteration, the accuracy of this combined algorithm is the same as in TSS algorithm, while the computational efficiency will be much better.

2.6. Four Step Search (FSS)

Four Step Search [7] also depends on center biased searching. FSS sets a fixed pattern size of step = 2 for the first step. Thus, it looks at 9 locations in a 5×5 window. If the least weight is found at the center of search window the search jumps directly to fourth. 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 and the search window is still maintained as 5×5 pixels wide. Depending on where the least weight location is, FSS might terminate checking weights at 3 locations or 5 locations. The patterns are shown in Figure 7.

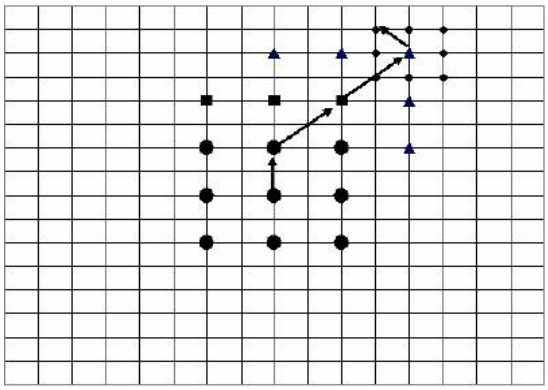

Figure 7. Four Step Search Algorithm

Once again if the least weight location is at the center of the 5×5 search window, the algorithm jumps to the fourth step, otherwise, it moves on to third step. The third is exactly the same as the second step. In the fourth step the window size is dropped to 3×3 with size of step = 1. The location with the least weight is the best matching macro-block and the motion vector is set to point to that location. This search algorithm has the best performance for 17 checking points and worst performance for 27 checking points.

2.6. Diamond Search (DS)

DS [8] algorithm is exactly the same as FSS, 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. Diamond Search 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 Figure. 8. Just like in FSS, the first step uses LDSP, and if the least weight is at the center location, DS jumps to the 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, as illustrated in the second and third steps of procedure shown in Figure 8. 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 in accurate and efficient way. The end result of PSNR should be close to that of FS, while computational expense is significantly less.

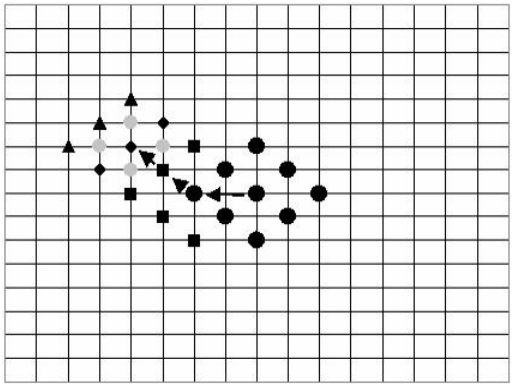

Figure 8. Diamond Search Algorithm

2. Simulation Results

All of the above seven algorithms under consideration have been implemented using luminance popular video sequences of 100 frames CIF format (352×288) for large motion activity ”Football” video sequence. The distance between the current frame and reference frame is considered to be 2 to generate the frame-by-frame results. Table 2 shows the maximum value of PSNRs for these search algorithms using the “Football” video for the first 100 frames, the average number of searches required per macro block and the speed up of the fast full algorithms in relation to the full search algorithm.

Table 2. CIF (352×288) “Football” Video with distance of 2

CIF (352×288) “Football” Video with distance of 2							
	FS	FCFS	TSS	NTSS	WinUpTSS	FSS	DS
Maximum of PSNR	23.36	23.36	23.10	22.99	23.10	22.79	22.76
Number of searches per macro-block	204.3	204.3	21.06	20.93	21.06	18.40	17.81
Speed up	1.00	1.60	9.70	9.77	12.13	11.10	11.82

3. Conclusion

This paper analyzed and compared seven different block matching algorithms on large motion activity video sequence “Football“. Full Search (FS) and fast computational full search algorithms give the best results in terms of image quality. Diamond search algorithm gives the best results regarding the number of searches per macro-block and Winner-update. Three Step Search has the best computational time. FCFS can speed up the computational process by 1.6 comparing to FS despite having the same number of searched point and PSNR. Combining the same principle used in FCFS with any other algorithm, the computational time can be reduced by approximately 60%. This means that we can have a new algorithm with less computational time without changing the quality. The Winner-Update Strategy can speed up the computational time by 12.13 comparing to FS with same quality with TSS. Again, to combine the same strategy used in Winner-Update with any algorithm the computational time will be reduced by approximately 82% without affecting the quality value. These results have an impact to video compression and transfer in many projects related to video processing.

Acknowledgements

This research has been realized under the support of the following grants: “Tools and Methods for Video and Image Processing for the Fight against Terrorism” – MV VG20102015006 (CZ), “Security-Oriented Research in Information Technology” – MSM0021630528 (CZ) and “The IT4Innovations Centre of Excellence” – IT4I-CZ 1.05/1.1.00/02.0070 (CZ).

References

[1] I. E. G. Richardson, "Video Coding Concepts", H.264 and MPEG-4 Video Compression, Wiley, Essex, England, **(2003)**, pp. 27-42.
[2] A. Barjatya, "Block Matching Algorithms for Motion Estimation", Dept. of Electrical Engineering, Utah State University, Utah, DIP 6620, **(2004)**.
[3] D. V. Manjunatha, "Comparison And Implementation Of Fast Block Matching Motion Estimation Algorithms for Video Compression", International Journal of Engineering Science and Technology, vol. 3, no. 10, **(2011)**.
[4] Z. Ahmed, A. Hussain and D. Al-Jumeily, "Fast Computations of Full Search Block Matching Motion Estmation (FCFS), PGNeT Conference, **(2011)**.
[5] T. Koga, "Motion compensated interframe image coding for video conference", Proceedings of NTC81, **(1981)**.
[6] R. Li, B. Zeng and M. L. Liou, "A new three-step search algorithm for block motion estimation", IEEE Transactions of Circuits and Systems for Video Technology, vol. 4, no. 4, **(1994)**, pp. 438-442.
[7] L. M. Po and W. C. Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation", IEEE Transactions of Circuits and Systems for Video Technology, vol. 6, no. 3, **(1996)**, pp. 313-317.
[8] S. Zhu and K. K. Ma, "A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation", IEEE Transactions of Image Processing, vol. 9, no. 2, **(2000)**, pp. 287-290.
[9] C. M. Lin and S. C. Kwatra, "Motion compensated interframe color image coding", International Conference on Communications, Part 1, **(1988)**, pp. 516-520.
[10] M. Ahmadi and M. Azadfar, "Implementation of fast motion estimation algorithms & comparison with full search method in H.264", IJCSNS International Journal of Computer Science & Network Security, vol. 8, no. 3, **(2008)**, pp. 139-143.
[11] Y. -S. Chenm, Y. -P. Hung and C. -S. Fuh, "Fast Block Matching Algorithm Based on the Winner-Update Strategy", IEEE Transactions on Image Processing, vol. 10, no. 8, **(2001)**, pp. 1212-1221.
[12] P. H. Winston, "Artificial Intelligence", 3rd Ed. Reading, MA: Addison- Wesley, (**1992**).

Authors

Ibrahim Nahhas graduated in 2008 at the University of Aleppo, Faculty of Electrical & Electronic Engineering – Communication Systems Department, in Syria. He had his diploma thesis of communication engineering. He is now a Ph.D. Student at the Brno University of Technology, Faculty of Information Technology in the Czech Republic. His research topics include image and video processing, video compression standards and motion estimation and block matching. For more information – see please http://www.fit.vutbr.cz/~inahhas/.

Martin Drahanský graduated in 2001 at the Brno University of Technology, Faculty of Electrotechnics and Computer Science in the Czech Republic and simultaneously at the FernUniversität in Hagen, Faculty of Electrotechnics, Germany. He achieved his Ph.D. grade in 2005 at the Brno University of Technology, Faculty of Information Technology in the Czech Republic and his habilitation grade at the same faculty in 2009. Now he works as associate professor at the BUT FIT, DITS. His research topics include biometrics, security and cryptography, artificial intelligence and sensoric systems. For more information – see please http://www.fit.vutbr.cz/~drahan/.