Journal of Balkan Libraries Union

ISSN 2148-077X

http://www.balkanlibraries.org/journal
http://dergipark.gov.tr/jblu

Block-Matching Motion Estimation Algorithms for Video Processing and Compression: a Brief Overview

Samsun Mustafa Başarıcı

Adnan Menderes University, Aydın, Turkey
Tel.: +90-256-213-75-03 (Ext 3618); fax: +90-256-213-66-86; e-mail: sbasarici@adu.edu.tr

Research Article

ARTICLE INFORMATION

Article history:
Received 11 January 2017
Received in revised form 18 March 2017
Accepted 2 April 2017
Available online 30 May 2017

Journal of Balkan Libraries Union
Vol. 5, No. 1, pp. 28-34, 2017.

ABSTRACT

Humanity created different methods for sharing information. One of the first forms of sharing information and knowledge were images. In the beginning, the process of sharing was relying on static appearances. With the invention of moving pictures by Eadweard Muybridge in the first part of 1870s, this exchange and sharing gained a new quality. Now it was possible to show and preserve motion too. Since that time, technology has changed rapidly. The latest discoveries and improvements from the point of view of technology use computer and IT technologies extensively. Today it is possible for everybody to create and record movies by themselves using affordable and convenient technological devices.

Also the process of sharing evolved rapidly and become cheaper and cheaper. We are now able to record some movies and share them through the Internet or other carriers in real time or near real time. However, this also creates serious problems due to the huge volume of data to be sent through the data lines. Therefore, research has concentrated on methods to decrease the data volume without losing the quality. One way to do that is to create effective CODECs. A major drawback of moving pictures is the motion itself. CODECs have to minimize the size of videos without paying the price of quality losses but have also to reduce the computational complexity. Both of these requirements can be achieved with a solid knowledge of motion estimation among others. This paper gives a general overview and survey of some existing and important approaches without the claim of having a complete overview of the field.

Keywords: Video processing, Video compression, Motion estimation, Block-matching.

I. Introduction

One of the main problems in video processing and compression is motion estimation. This problem affects the main process of designing video CODECs, and is not only relevant in the design process but has also huge influence on computational complexity. Without a good knowledge of motion estimation, it is nearly impossible to create an effective video CODEC. Here we want to give a general overview and survey of existing approaches. This paper should therefore be considered only as an outline and does not have the claim to develop and improve new methods.

The area of video processing is a very dynamic area, which shows an emerging field of research. Because of this here, we focus only to the last few years. The historical evolution and basic concepts of video and image processing can be read in (Bovik, 2000; Richardson, 2002; Richardson 2003).

The rest of this paper gives an overview on block-matching motion estimation in general and concluded with suggestions and cautious projection.

II. Block-Matching Motion Estimation (BMME)

One of the essential parts of every video coding standard is block-matching motion estimation (BMME). In general, full-search block-matching is used as a benchmark in the reference software. The idea of BMME is to partition the image into several blocks in the reference frame, perform a search inside a previously coded frame, and select the best match by using a predetermined criterion. The best match is used to predict the current block, whereas the displacement between the two blocks defines a motion vector (MV), which is associated with the current block.

Full-search block-matching motion estimation (FS-BMME) algorithm estimates motion vector by testing all

possible positions in the search area exhaustively. As it can been seen this approach brings a high computational complexity both in time and space requirements. Many algorithms are introduced who sacrifice reducing the computational complexity compared to FS.

Based on the centre-biased characteristics of MVs, several fast BMAs have been developed, including the three-step search (TSS), the new TSS (NTSS), the block-based gradient-descent search (GDS), the diamond search (DS), and PMVFAST.

The block-matching algorithm (BMA) is extensively employed to extract motion vectors. Typically, the algorithm consumes 60–80% of the total computation in a video encoder, and it strongly affects the visual quality at a given bit-rate. The regular data flow of full-search BMA makes it especially amenable to hardware implementation. Many efficient hardware designs have been proposed for FS in recent decades, with many focusing on data reuse. However, FS—a brute-force algorithm—does not utilize information on motion activity in video sequences, and hence it is possible to improve the performance of FS hardware designs by considering this issue. Although these fast BMAs greatly reduce the computation required for motion estimation, their irregular search patterns result in complicated hardware design.

In general all fast search algorithms use only a subset of the search area in order to reduce the total number of searches. Most of the existing fast algorithms focus on macro-block full-pel (MBFP) ME.

There are four rules, which should be considered in the design of BMAs:

1. Searching points should be chosen in the direction of the current best improvement for faster convergence to an optimum solution.
2. The spatial and temporal correlation of MVs should be exploited to determine the initial searching point.
3. Searching points should be examined in a pattern around the initial position so as to exploit the centre-biased distribution of MVs.
4. The search should stop as soon as possible once the matching is good enough.

The baseline MPEG-4 introduces a new 8x8 block full-pel (BFP) ME. The H.264 standard defines seven different modes for variable block sizes (Fig. 1).

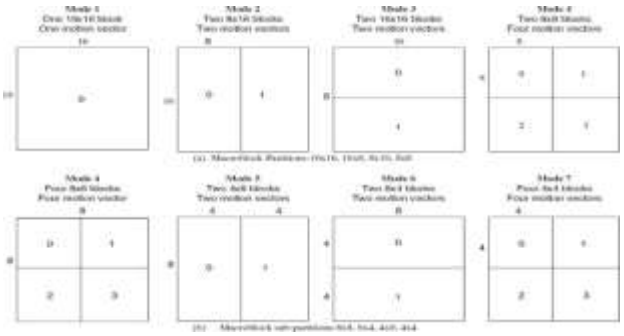


Fig. 1. Variable block sizes in H. 264 (Khan, Masud & Ahmad, 2006)

The baseline MPEG-4 ME algorithm consists of four main tasks. These are a full-pel search for 16x16 MB (MBFP), a full-pel search for 8x8 block (BFP), a half-pel search for 16x16 MB (MBHP) and a half-pel search for 8x8 block (BHP) as follows:

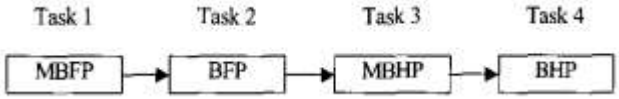


Fig. 2. Tasks for MPEG-4 baseline ME algorithms (Yang, 2003)

Considering these tasks an example for the ME estimation process can be given.

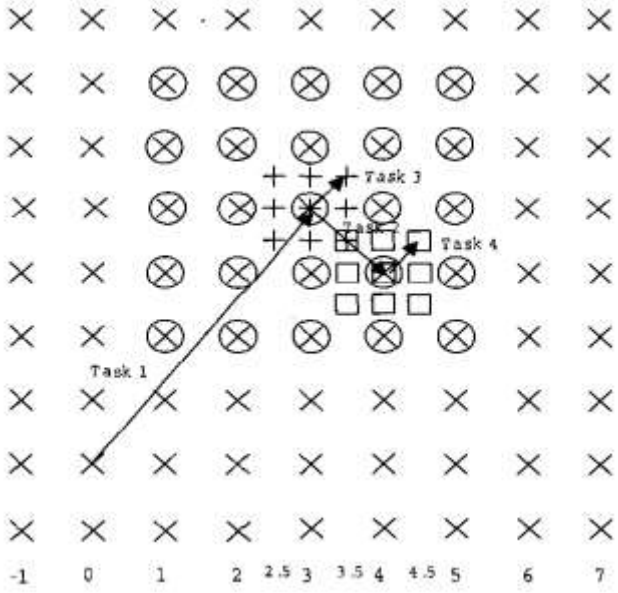


Fig. 3. Example of a baseline MPEG-4 ME estimation process (Yang, 2003)

X denotes search points in MBFP process+ denotes search points in MBHP process

0 denotes search points in BFP process denotes search points in BHP process

In the example above, the first task (MBFP) matches the current MB with every candidate at the full-pel position in the search window in the reference frame. The search window is centred at the same coordinate as the current MB and is extended in each direction by an amount determined by the search range. The second task (BFP) matches each 8x8 block (blocks inside current MB) with every candidate at full-pel positions in the search window. Each window center is at the corresponding blocks of the best matched MB from the first task and each side is extended by W (default value is 2 in reference software) full pixels for a total number of (2*W+1) candidate search points. The third task (MBHP) then matches the current MB with every candidate at the half-pel position in the search window. The window center is at the position of the best-matched MB from the first task and each side is extended by one half-pel for a total of nine candidate MBs. Finally, the last task (BHP) matches the four blocks of the current MB with every candidate at the half-pel position in their respective windows. The window centers are at the positions of the corresponding best matched blocks from the second task and each side is extended by half-pel for a total of nine candidate blocks (Yang, 2003).

In this model detecting of optimal motion vector is based on the mean absolute differences (MAD) error

calculation at all possible positions. The MAD matching criterion is a widely used method because of its low complexity and relatively good matching results respect to other methods such as mean square error (MAE). The definition of MAD is:

$$MAD(m,n) = \frac{\sum_{i=0}^{I-1}\sum_{j=0}^{J-1} I_S(i,j,k) - I_S(i+m, j+n, l)}{IJ} \quad (1)$$

where *m* an *n* denotes the relative displacement in search area, which is defined by (M, N), *I* and *J* are the sizes of the predicted block and $I_S(i, j, k)$ is the *k*th frame image.

In (Yang, 2003) the author proposes a new ME implementation which is shown below.

Start of task 1: MBFP

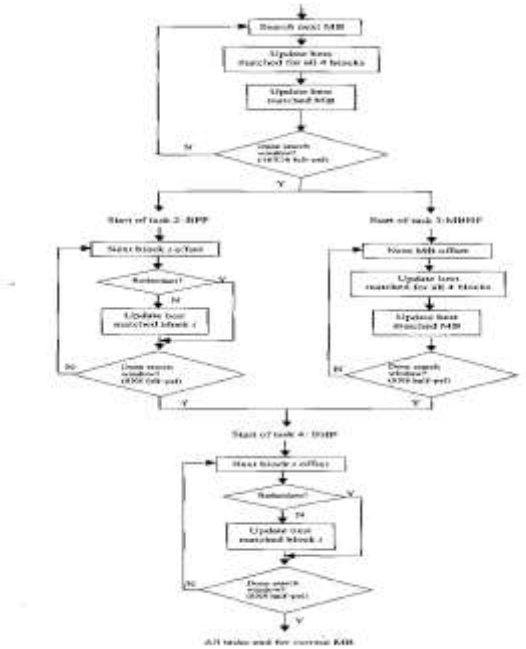


Fig. 4. Flowchart of the proposed MPEG-4 ME algorithm (Yang, 2003)

In task l (MBFP), this implementation breaks down the 16x16 full-pel search cost into four 8x8 full-pel block costs. When the 16x16 full-pel search is performed in first task, the vectors and costs for the four best matched 8x8 full-pel block are stored. Stored values can be passed to task 2. Task 2 will check the redundancy when doing iteration of full-pel block estimation. The redundant points are discarded, i.e. do not need to compute again. When processing task 2, video encoder can simultaneously process task 3, because there are no data dependency between these two tasks. This implies that an effective multi-core video encoder can do the procession in task 2 and 3 in distributed parallel way to improve its real-time performance.

Similarity with the flow progress of the relation between MBFP and BFP tasks, this implementation breaks down the 16x16 half-pel search cost into four 8x8 half-pel block costs. When the 16x16 half-pel search is performed in MBHP task, the vectors and costs for the four best matched 8x8 half-pel blocks are stored and passed to BHP task. BHP task will check the redundancy when doing iteration of half-pel block estimation in 9 candidate positions. The redundant points are discarded, i.e. do not need to compute again. The author also gives the required formulae to calculate the number of discarded and non-discarded points in the 8x8 blocks (Yang, 2003). Hence it is not our purpose to go further into the details of each introduced algorithm we don't give the formulae and the test results.

Based on the four principles mentioned before, a directional squared search (DSS) algorithm and a pipelined parallel architecture are presented in (Huang & Tsai, 2004). The authors of the introduced DSS follow the first rule, as shown in Fig.5. A square 3x3 search window of nine points is initially applied to the search area with the center recommended by the second rule. The algorithm stops if the center of the 3x3 window is the position of the best matching point; otherwise, the search center is moved to the best matching point. Only neighbouring points of the center position are investigated in the next search step. If the best matching point is in the corner, then five additional points should be checked, whereas three points must be checked when the best matching point is an edge point. The above process is repeated until the center of the window is the position of the best matching point (Huang & Tsai, 2004).

The main purpose of (Huang & Tsai, 2004) is to implement the algorithm as hardware. Typically, the initial search step of GDS can be considered to be a special case of FS with a ±1 search range, and it can be efficiently implemented in many hardware designs. However, the data flow when investigating neighbouring points during subsequent searches is not as regular as during the initial step, and hence special hardware is required. Actually, FS is the best choice for motion estimation from the viewpoint of hardware implementation. The basic idea of (Huang & Tsai, 2004), as shown in Fig. 5, is to modify the searching points employed by the subsequent searches of GDS such that they can also be performed by ±1 FS without redundant computation. Irrespective of whether the best matching point is located at a corner or edge, the modified GDS (MGDS) successively applies a square 3x3 search window of new nine points.

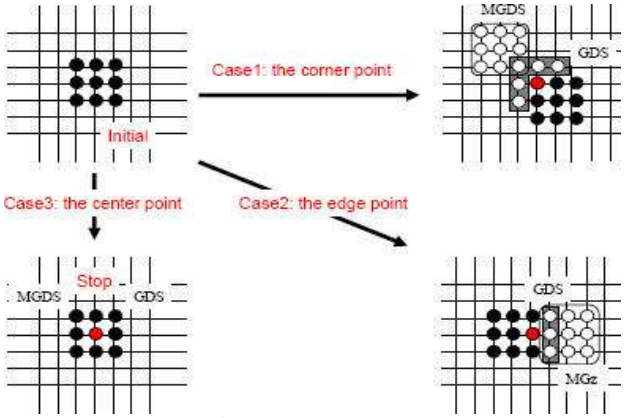


Fig. 5. Structure of GDS and MGDS (Huang & Tsai, 2004)

The major advantage of the proposed MGDS scheme is that all operations can be performed by ±1 FS. Thus, we can employ an array of ±1 FSs as the search engine of motion estimation. This feature is especially useful for

personal visual communication because types of handheld devices that are now being used are various, such as PDAs and handsets. The large variation in the computation power amongst these heterogeneous devices makes conventional BMAs impractical, since their parameters cannot be tuned automatically according to the available computational power. In contrast, the computational power of the search engine can be easily updated by changing the number of elements in the FS array. To efficiently utilize the available computation of the search engine, an adaptive computation distribution mechanism is further presented in (Huang & Tsai, 2004). Experimental results indicate that MGDS can uniformly achieve a quality improvement over original FS under the same computation. That is, the computation distortion (CD) performance of FS hardware designs can be improved when they are assembled into the proposed array structure and cooperate in the manner of MGDS.

Current video codec predicts MV of a given macroblock (MB) based on the MVs of neighbouring MBs for efficient entropy coding. However, the search center, (x_c, y_c) of the current MB can only be set as (0, 0) for parallel extracting MV in MGDS. A square 3x3 search window of nine points is then applied to the initial center, i.e. (0, 0). In MGDS, distortion is measured by the sum of absolute differences (SAD) due to its lower computation cost. Similar to GDS, MGDS immediately stops if the center is the position of the best matching point; otherwise a series of subsequent searches will be performed toward the best matching point. Unlike GDS, a square 3x3 search window of nine points is applied successively irrespective of whether the best matching point is located at a corner or edge. The center of the next search (x_n, y_n) is generated by

$$(x_n, y_n) = (x_c, y_c) + (3*i_c, 3*j_c) \quad (2)$$

where (i_c, j_c) is the placement of the best matching point.

In MGDS, three additional stopping conditions are employed to reduce the computation. Let the immediately preceding search center and the SAD of the best matching point be (x_p, y_p) and SAD_p, respectively. It is pointless examining the searched area if the next search center (x_n, y_n) is equal to one of the previous search centers. For simplicity, only the immediately preceding search center (x_p, y_p) is checked in MGDS. Therefore, MGDS stops if (x_n, y_n) = (x_p, y_p). The second stopping condition is $SAD_p \leq SAD_c$, where SAD_c is the best SAD of the current MB. This condition implies that the optimum occurs at the best matching point of the previous nine points. The third stopping condition is $SAD_c \leq TH$, where TH is the given threshold. This condition indicates that the current SAD is below an acceptable threshold.

Suppose the value of TH is 350. MGDS first examines nine searching points labeled as *1* in the figure, resulting in (1, 0) and 500 as the displacement and SAD of the best matching point, respectively. Since the best matching point is not at the center and the best SAD is larger than TH (i.e. 500>350), the second search center (3, 0) is derived using the equation above and nine searching points depicted as *2* in the figure are then checked, resulting in (1, 1) as the displacement and 400 as the SAD. Similarly, nine searching points labeled as *3* in the figure with the search center (6, 3) are examined because the displacement of the best matching point is not (0, 0), the current SAD is smaller than that of the previous (i.e. 400<500), the next search center (6, 3) is not examined and the best SAD is still larger than TH (i.e. 400>350) in this situation. After the third ±1 FS, (i_c, j_c) = (-1, 0) and SAD_c = 300. The algorithm then stops because the current best SAD is below TH (i.e., 300<350). Therefore, the MV of this MB is (5, 3) since the current SAD is smaller than the previous value. Another example for explaining MGDS is shown in Fig. 2(b). The processing of this example is initially the same as for the above example, except the resulting displacement of the second ±1 FS is (-1, 0). MGDS stops because the next search center is (0, 0) = (3 x -1, 3 x 0) which has already been examined by the first ±1 FS. Thus, (2, 0) is the final MV (Huang & Tsai, 2004).

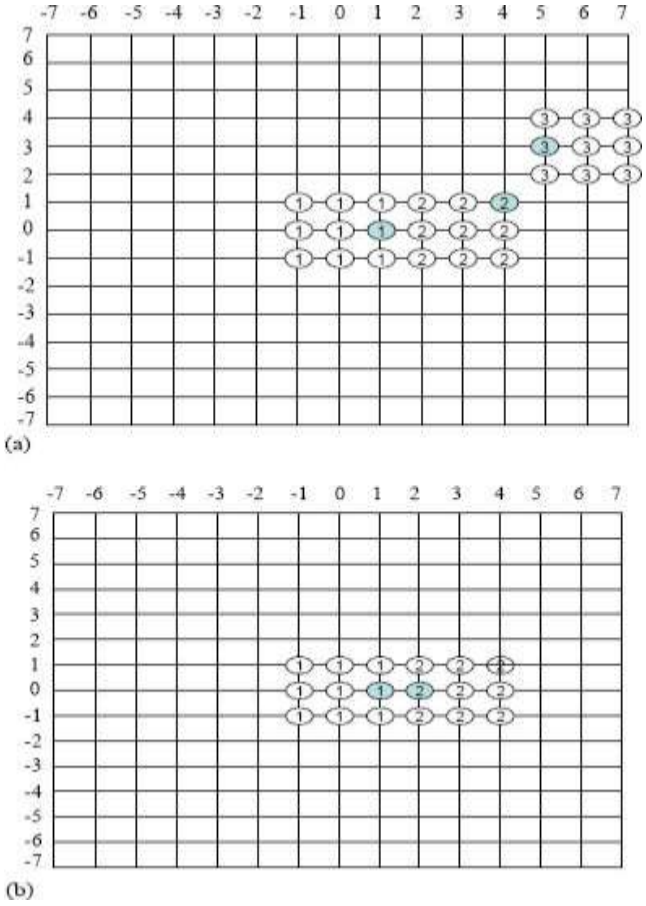


Fig. 6. (a), (b) Examples to illustrate the MGDS algorithm (Huang & Tsai, 2004)

Another interesting approach is introduced in (Tsai & Pan; 2006). In this approach a 3-D predict hexagon search algorithm for fast block motion estimation on H.264 is used. The main motivation of the authors can be coarsely given as the growing of the internet. Because the internet is more and more universal and the technology of multimedia has been progressed largely, the communication of the video data is an essential part in our life.

Just for retrospection the basic structure of multiframe motion estimation in MPEG-4/AVC/H.264 is given in Fig. 7. For each block of the encoding mode, the motion vector is searched in a frame by frame manner. Adopting

the full search scheme to search the motion vector for each encoding mode in each reference frame consumes considerable search time. The computational load of motion estimation increases markedly in H.264 owing to the new features. As we mentioned before, according to statistics, it consumes approximately 60%-80% of the entire encoding time (Chen, Li, Chiang &Hsu, 2006).

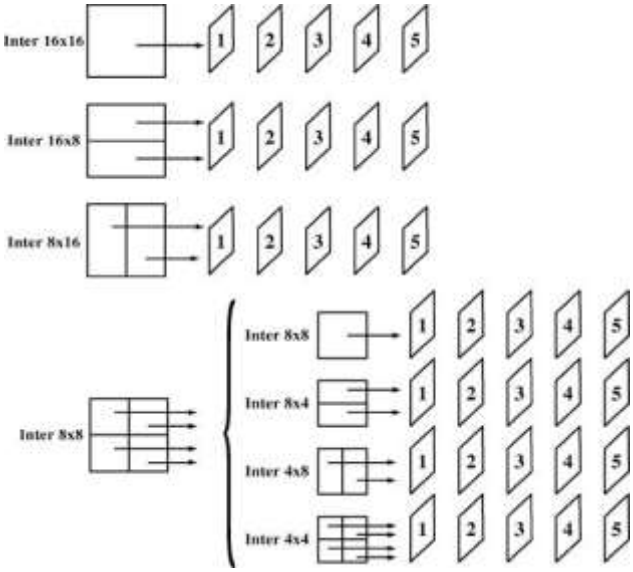


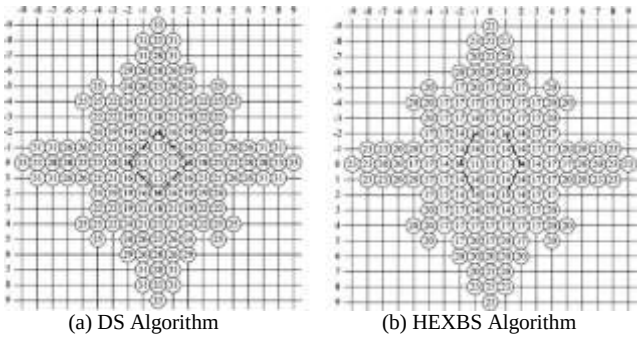
Fig. 7. Multiframe motion estimation in MPEG-4/AVC/H.264 (Chen, Li, Chiang &Hsu, 2006)

As mentioned before the main critical issue in H.264 motion estimation is to reduce the complexity of the motion estimation. According to Tsai & Pan (2006), there are three methods to achieve it.

- Reducing the complexity of mode decision when doing motion estimation.
- Reducing the complexity of reference frames when doing motion estimation.
- Reducing the number of search points: the well-known Full Search (FS) algorithm exhaustively evaluates all possible candidate motion vectors over a predetermined neighbourhood search window to find the global minimum block distortion position. Although FS can get the best matching blocks but it expenses a high computational complexity.

Tsai & Pan (2006) is mainly focused on the effect of reducing search points. The 3-D consideration indicates the three critical predictions; it includes the object movement in vertical and horizontal directions, the search center with variable block sizes, and the search center with multiple reference frames. In addition, because the analysis of motion vector distribution is used to make a local search range, two different search patterns are used to reduce the search points effectively.

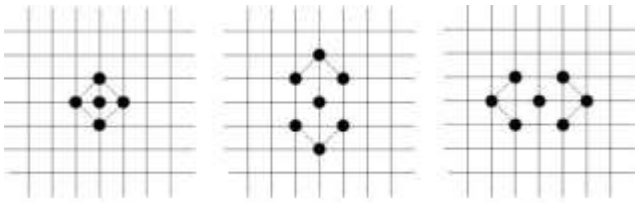
In most of the previous algorithms, such as DS and HEXBS (Hexagon Based Search), the searching process often uses the large search pattern first and then uses the small search pattern. The difference between them can be seen in Fig. 8.



(a) DS Algorithm (b) HEXBS Algorithm

Fig. 8. Minimum possible search points for each motion vector (Tsai & Pan; 2006)

In the proposed Predict Hexagon Search (PHS) algorithm the search pattern is constructed as shown in Fig. 9.



(a) (b) (c)

Fig. 9. (a) Small PHS Pattern (SPHSP), (b) Vertical Large PHS Pattern (vertical LPHSP), (c) Horizontal Large PHS Pattern (horizontal LPHSP) (Tsai & Pan; 2006)

The algorithm proposes the search process as follows.

Step 1) The SPHSP with five search points is used. If the minimum RD-Cost point is located in the center point of SPHSP, the center point is the final point of the motion vector; otherwise, the point which is the minimum RD-Cost (Rate Distortion) point will be the center point and the flow proceeds to step 2. This case is shown in Fig. 10(a). If the minimum RD-Cost point is located on up or down dots, we identify the object is moving in the vertical direction and vertical LPHSP will be used in step 3. On the other hand, if the minimum RD-Cost point is located on left or right dots, horizontal LPHSP will be used in step 3.

Step 2) With the minimum RD-Cost point in the previous searching step as the center, the SPHSP is formed and still used in this step. Three new candidate points are checked and the minimum RD-Cost point is identified again. If the minimum RD-Cost point is located on the center point of SPHSP, the center point is the final point of the motion vector; otherwise, the point which is the minimum RD-Cost point will be the center point and the flow proceeds to step 3. When finishing the step 1 and step 2, we complete the rood side in 2 searching first.

Step 3) With the minimum RD-Cost point in the previous searching step as the center, switch the search pattern from SPHSP to suitable LPHSP. For case 2 as shown in Fig. 10(b), three new candidate points are checked and the minimum RD-Cost point is identified again. For case 3 as shown in Fig. 10(c), four points are added as the new candidate points. If the minimum RD-Cost point is located on the center point of LPHSP, then the flow goes to step 5; otherwise, the flow proceeds to step 4.

Step 4) With the minimum RD-Cost point in previous searching step checked as the center point, a new large

hexagon is generated. Three new candidate points are checked and the minimum RD-Cost point is identified again. If the minimum RD-Cost point is the center point of the LPHSP, then the flow goes to step 5; otherwise, the flow repeats this step continuously.

Step 5) Switch the search pattern form LPHSP to SPHSP. In Fig. 10(d), four new candidate points are evaluated to compare with the current minimum RD-Cost point. The new minimum RD-Cost point is the final point of the motion vector. Fig. 11 illustrates the overall scheme of proposed PHS algorithm for H.264.

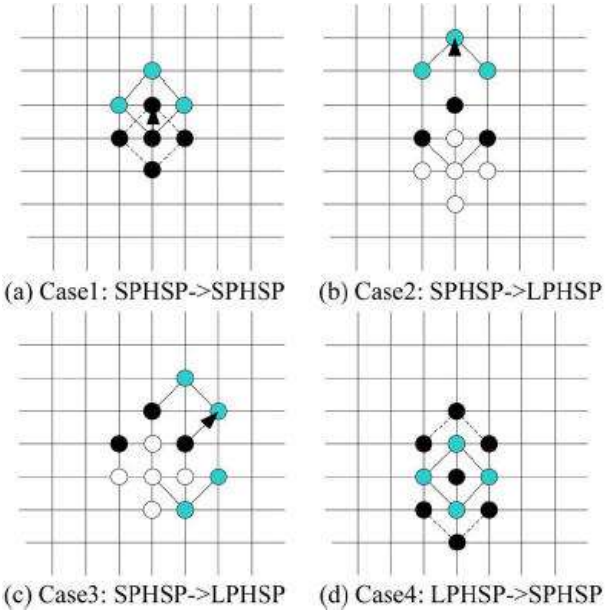


Fig. 10. Four special cases of checking points overlapping when the minimum RD-Cost point found in the previous search step (Tsai & Pan, 2006)

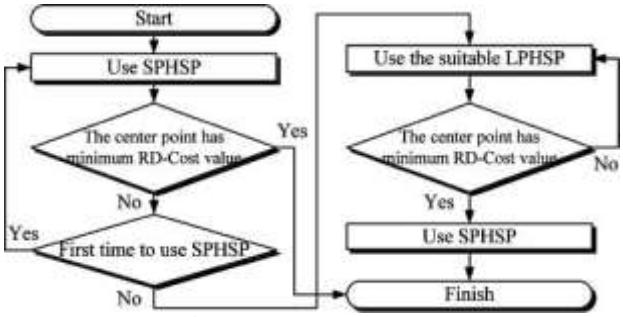


Fig. 11. Flowchart for proposed PHS algorithm (Tsai & Pan, 2006)

In Fig. 12 an example is given to show the search path strategy. In this example, the motion vector is (5, -1) and six searching steps are needed. Totally, there are 21 search points with 5, 3, 3, 3, 3, and 4 search points in each sequential step.

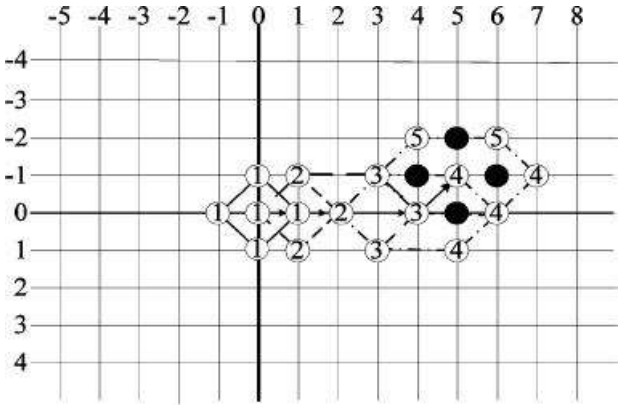


Fig. 12. Search path example leading to the MV (5, -1) in six searching steps (Tsai & Pan, 2006)

Just for comparison Fig. 13 shows the minimum possible number of search points for each MV location by PHS algorithm.

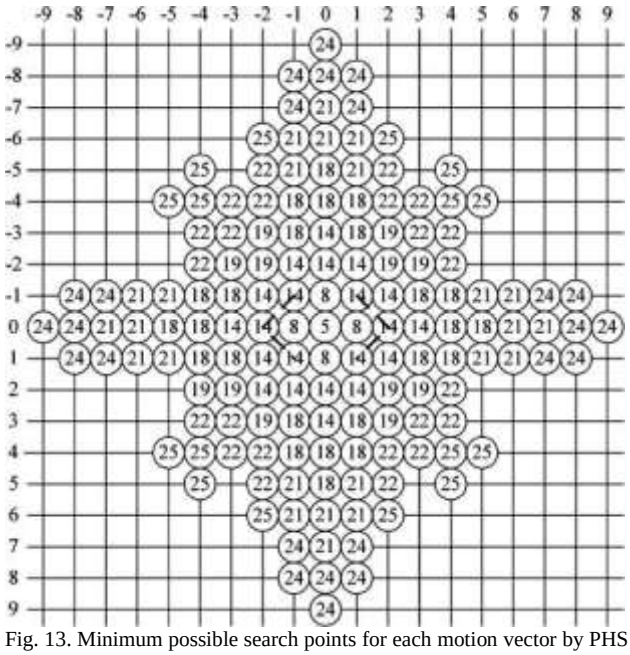


Fig. 13. Minimum possible search points for each motion vector by PHS (Tsai & Pan, 2006)

III. Conclusion and Future Work

Here we introduced briefly three different methods for motion estimation. There are many different approaches just like using likelihood and correlation of motion field (Kuo & Chan, 2006), subpixel accuracy (Hill, Chiew, Bull & Canagarajah, 2006), contextual knowledge (Namuduri, 2004), or even well known mathematical models like Markov (Chen, Chen, Hung, Fang, Shie & Lai, 2006). It would blast the frame of this work to examine all of them so we just give a list of papers as references. This list does not have the claim of being complete. It should be seen as an impulse to be concerned with this area. It is expected that the next years will give us more sophisticated works.

References

Bovik, Al (ed.). (2000) *"Video compression", handbook of image and Video processing.* 555-629, San Diego: Academic Press.

Chen, M., Li, G., Chiang, Y. & Hsu, C. (2006). Fast multiframe motion estimation algorithms by motion vector composition for the MPEG-4/AVC/H.264 standard. *IEEE Trans. on Multimedia*, *8*(3), 478-487.

Chen, P.H., Chen, H.M, Hung, K.J., Fang, W.H., Shie, M. C. & Lai, F. (2006). Markov model fuzzy-reasoning based algorithm for fast block motion estimation. *Journal of Visual Communication and Image Representation*, *17* (1), 131-142.

Hill, P. R., Chiew, T. K., Bull, D. R & Canagarajah, C. N. (2006). Interpolation free subpixel accuracy motion estimation. *IEEE Trans. on Circuits and Systems for Video Technology*, *16*(12), 1519-1526.

Huang, S. & Tsai, W. (2004).A simple and efficient block motion estimation algorithm based on full-search array architecture. *Signal Processing: Image Communication*, *19*(10), 975-992.

Khan, N.A., Masud, S. & Ahmad, A. (2006). A variable block size motion estimation algorithm for real-time H.264 video encoding. *Signal Processing: Image Communication*, *21*(4), 306-315.

Kuo, T.Y. & Chan, C.H. (2006). Fast variable block size motion estimation for H.264 using likelihood and correlation of motion field. *IEEE Trans. on Circuits and Systems for Video Technology*, *16*(10),1185-1195.

Lee, Y.G. & Ra, J.B. (2006). Fast motion estimation robust to random motions based on a distance prediction. *IEEE Trans. on Circuits and Systems for Video Technology*, *16*(7), 869-875.

Liang, Y., Ahmad, I., Luo, J., Sun, Y. & Swaminathan, W. (2005). On using hierarchical motion history for motion estimation in H.264/AVC. *IEEE Trans. on Circuits and Systems for Video Technology*, *15*(12), 1594-1603.

Montrucchio, B. & Quaglia, D.(2005). New sorting-based lossless motion estimation algorithms and a partial distortion elimination performance analysis. *IEEE Trans. on Circuits and Systems for Video Technology, 15*(2), 210-220.

Namuduri, K. R. (2004). Motion estimation using spatio-temporal contextual information. *IEEE Trans. on Circuits and Systems for Video Technology*, *14*(8), 1111-1115.

Po, L.M., Ting, C. W., Wong, K.M. & Ng, K. H. (2007). Novel point-oriented inner searches for fast block motion estimation. *IEEE Trans. on Multimedia*, *9*(1), 9-15.

Richardson, I.E. (2002). *Video CODEC design.* New York: Wiley.

Richardson, I.E. (2003). *H.264 and MPEG-4 video compression*. NewYork: Wiley.

Tsai, T. & Pan, Y. (2006). A novel 3-D predict hexagon search algorithm for fast block motion estimation on H.264 video coding. *IEEE Trans. on Circuits and Systems for Video Technology, 16*(12), 1542-1549.

Tu, Y.K., Yang, J.F, Sun, M.T. & Tsai, Y.T. (2005). Fast variable-size block motion estimation for efficient H.264/AVC encoding. *Signal Processing: Image Communication*, *20*(7), 595-623.

Yang, W. (2003). An efficient motion estimation method for MPEG-4 video encoder. *IEEE Trans. on Consumer Electronics*, *49*(2), 441-446.

Zhou, Z., Xin, J. & Sun, M.T. (2006). Fast motion estimation and Inter-mode decision for H.264/MPEG-4 AVC encoding. *Journal of Visual Communication and Image Representation*, *17*(2), 243-263.

Samsun Mustafa Başarıcı is an Asst. Prof. at the Department of Computer Engineering in Adnan Menderes University, Turkey. He has authored papers in refereed journals and international conference proceedings and has been actively serving as a reviewer for international journals and conferences. He has also authored a book and edited a conference proceeding.