A Study on Block Matching Algorithms for Motion Estimation

S. Immanuel Alex Pandian
Asst. Prof., Dept. of ECE,

Dr.G. Josemin Bala
Prof & Head , Dept.of EMT,

Becky Alma George
PG Student, Dept. of ECE,

Karunya University,
Coimbatore,India.

***Abstract*— Block matching motion estimation is the essence of video coding systems. This paper gives a review of the different block matching algorithms used for motion estimation in video compression. It compares 10 different types of block matching algorithms that range from the very basic Full Search to the recent fast adaptive algorithms like Pattern Based Search. The algorithms that are evaluated in this paper have been used in implementing various standards ranging from MPEG1 / H.261 to MPEG4 / H.263.**

Keywords- Block matching, motion estimation, video compression, full search, pattern based search

I. INTRODUCTION

The demand for communications with moving video picture is rapidly increasing. Video is required in many remote video conferencing systems, and it is expected that in near future cellular telephone systems will send and receive real-time video. A major problem in a video is the high requirement for bandwidth. A typical system needs to send dozens of individual frames per second to create an illusion of a moving picture. For this reason, several standards for compression of the video have been developed. Digital video coding has gradually increased in importance since the 90s when MPEG- 1 first emerged. Video coding achieves higher data compression rates without significant loss of subjective picture quality also eliminates the need of high bandwidth required. Generally speaking, video compression is a technology for transforming video signals that aims to retain original quality under a number of constraints, e.g. storage constraint, time delay constraint or computation power constraint. It takes advantage of data redundancy between successive frames to reduce the storage requirement by applying computational resources. The design of data compression systems normally involves a tradeoff between quality, speed, resource utilization and power consumption.

In a video scene, data redundancy arises from spatial, temporal and statistical correlation between frames. These correlations are processed separately because of differences in their characteristics. Hybrid video coding [1] architectures have been employed since the first generation of video coding standards, i.e. MPEG. MPEG consists of three main parts to reduce data redundancy from the three sources described above. Motion estimation and compensation are used to reduce temporal redundancy between successive frames in the time domain. Motion estimation examines the movement of objects in an image sequence to try to obtain vectors representing the estimated motion. Motion compensation uses the knowledge of object motion so obtained to achieve data compression. Transform coding, also commonly used in image compression, is employed to reduce spatial dependency within a frame in the spatial domain. Entropy coding is used to reduce statistical redundancy over the residue and compression data. This is a lossless compression technique commonly used in file compression. Each individual frame is coded so that redundancy is removed. Furthermore, between consecutive frames, a great deal of redundancy is removed with a motion compensation system.

Different search algorithms are used to estimate motion between frames. When motion estimation is performed by an MPEG-2 [2] encoder it groups pixels into 16×16 macro blocks. MPEG-4 AVC [3] encoders can divide these macro blocks into partitions as small as 4 × 4, and even of variable size within the same Macro block. Partitions allow for more accuracy in motion estimation because areas with high motion can be isolated from those with less movement.

II. MOTION ESTIMATION

A video sequence can be considered to be a discretized three-dimensional projection of the real four-dimensional continuous space-time. The objects in the real world may move, rotate, or deform. The movements cannot be observed directly. Changes between frames are mainly due to the movement of these objects. Using a model of the motion of objects between frames, the encoder estimates the motion that occurred between the reference frame and the current frame. This process is called motion estimation (ME). The encoder then uses this motion model and information to move the contents of the reference frame to provide a better prediction of

the current frame. This process is known as motion compensation (MC) [4]-[7], and the prediction so produced is called the motion-compensated prediction (MCP) or the displaced-frame (DF). In this case, the coded prediction error signal is called the displaced-frame difference (DFD). A block diagram of a motion-compensated coding system is illustrated in Figure 1. This is the most commonly used interframe coding method.

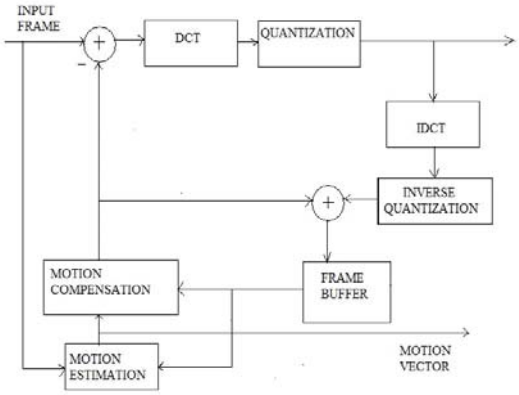

Figure 1. Motion compensated video coding

The reference frame employed for ME can occur temporally before or after the current frame. The two cases are known as forward prediction and backward prediction, respectively. In bidirectional prediction, however, two reference frames (one each for forward and backward prediction) are employed and the two predictions are interpolated (the resulting predicted frame is called B-frame). The most commonly used ME method is the block-matching motion estimation (BMME) algorithm, adopted in many compression standards.

ME is quite computationally intensive and can consume up to 80% of the computational power of the encoder if the full search (FS) is used by exhaustively evaluating all possible candidate blocks within the search window. Therefore, fast algorithms are highly desired to significantly speed up the process without sacrificing the distortion seriously. Many computationally variants [8]-[17] were developed, among which are typically the TSS, NTSS, 4SS, DS algorithm.

III. BLOCK MATCHING ALGORITHMS

The process of block-matching algorithm is illustrated in Figure 2. In a typical Block Matching Algorithm, each frame is divided into blocks, each of which consists of luminance and chrominance blocks. Usually, for coding efficiency, motion estimation is performed only on the luminance block. Each luminance block in the present frame is matched against candidate blocks in a search area on the reference frame. These candidate blocks are just the displaced versions of original block. The best candidate block is found and its displacement (motion vector) is recorded. In a typical interframe coder, the input frame is subtracted from the prediction of the reference frame. Consequently the motion vector and the resulting error can be transmitted instead of the original luminance block; thus interframe redundancy is removed and data compression is achieved. At receiver end, the decoder builds the frame difference signal from the received data and adds it to the reconstructed reference frames.

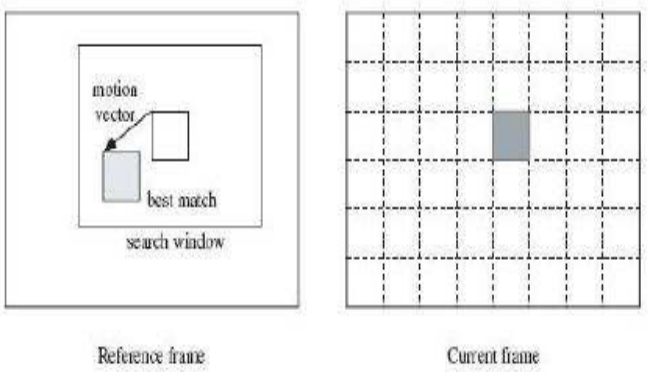

Figure 2. Block matching motion estimation

1. Full Search Motion Estimation:

In order to get the best match block in the reference frame, it is necessary to compare the current block with all the candidate blocks of the reference frames. Full search motion estimation calculates the sum absolute difference(SAD) value at each possible location in the search window. Full search computed the all candidate blocks intensive for the large search window.

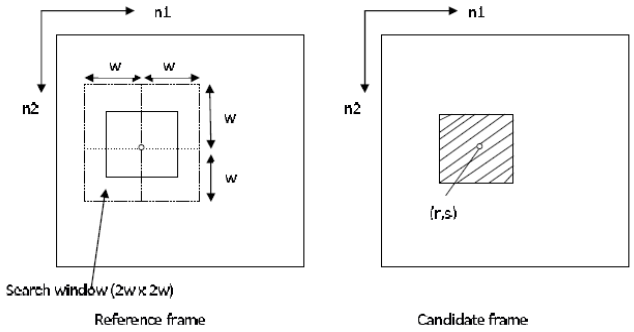

Figure 3. Full search motion estimation

Consider a block of N × N pixels from the candidates frame at the coordinate position (r, s) as shown and then consider a search window having a range ±w in both and directions in the references frame, as shown. For each of the (2w + 1)2 search position (including the current row and the current column of the reference frame), the candidate block is compared with a block of size N × N pixels, and the best matching block, along with the motion vector is determined only after all the (2w+1)2 search position are exhaustively explored.

By exhaustively testing all the candidate blocks within the search window, this algorithm gives the global minimum block distortion position which corresponds to the best matching block. However, a substantial computation load is demanded.

2. *Three Step Search Motion Estimation:*

TSS was introduced by Koga et al in 1981[4]. It became very popular for low bit-rate video applications because of its simplicity and also robust and near optimal performance. It searches for the best motion vectors in a course to fine search pattern. The algorithm may be described as:

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

Step 2: The step size is halved. The centre is moved to the point with the minimum distortion.

The point which gives the smallest criterion value among all tested points is selected as the final motion vector. TSS reduces radically the number of candidate vectors to test, but the amount of computation required for evaluating the matching criterion value for each vector stays the same. TSS may not find the global minimum (or maximum) of the matching criterion; instead it may find only a local minimum and this reduces the quality of the motion compensation system. On the other hand, most criteria can be easily used with TSS. It uses a uniformly allocated search pattern, which is not very efficient to catch small motions appearing in stationary or qusi-stationary blocks. To remedy this problem several adaptive techniques have been suggested to make the search more adaptable to motion scale and uncertainty. A dynamic search window scheme is presented in [18] which the search window is adjusted according to the uncertainty of motion scale. The uncertainty is estimated by the difference of block distortion measure among the checked points. A small difference indicates a large uncertainty and hence the search scope will be increased in next step.

3. *New Three Step Search Motion Estimation:*

In 1994, a new three step search algorithm [9] for fast ME is introduced. The search pattern in each step is fixed and no thresholding operations are involved in this algorithm. Nevertheless, it is made to better utilize the motion distribution of real world image sequences in low bit-rate video applications and is adaptive in the sense that the algorithm may stop at the second or third step. It will be shown that the NTSS algorithm retains the simplicity and regularity of the original TSS method, works better that TSS in terms of motion compensation error and robustness, and is quite compatible to TSS in terms of computational complexity.

NTSS differs from TSS by (1) assuming a center-biased checking point pattern in its first step and (2) incorporating a halfway-stop technique for stationary or qusi-stationary blocks. It constructed a center-biased search point pattern in the first step by adding 8 extra checking points which are neighbors of the window center. This has significantly reduced the average distance.

4. Four Step Search Motion Estimation:

The 4SS algorithm [10] utilizes a center-biased search pattern with nine checking points on a 5 x *5* window in the instead of a 9 x 9 window in the 3SS. This algorithm helps in reducing the number of search points when compared to the 3SS and hence is more robust. The 4SS algorithm is summarized as follows:

Step 1: A minimum block distortion method (BDM) point is found from a nine-checking point's pattern on a *5* x *5* window located at the center of the 15 x 15 searching area as shown in Figure. 4(a). If the minimum BDM point is found at the center of the search window, go to Step 4; otherwise go to Step 2.

Step 2: The search window size is maintained in 5 *x* 5. However, the search pattern will depend on the position of the previous minimum BDM point.

a) If the previous minimum BDM point is located at the corner of the previous search window, five additional checking points as shown in Figure. 4(b) are used.

b) If the previous minimum BDM point is located at the middle of horizontal or vertical axis of the previous search window, three additional checking points as shown in Figure. 4(c) are used. If the minimum BDM point is found at the center of the search window, go to Step 4; otherwise go to Step 3.

Step3: The searching pattern strategy is the same as Step 2, but finally it will go to Step 4.

Step4: The search window is reduced to 3 x 3 as shown in Figure. 4(d) and the direction of the overall motion vector is considered as the minimum BDM point among these nine searching points.

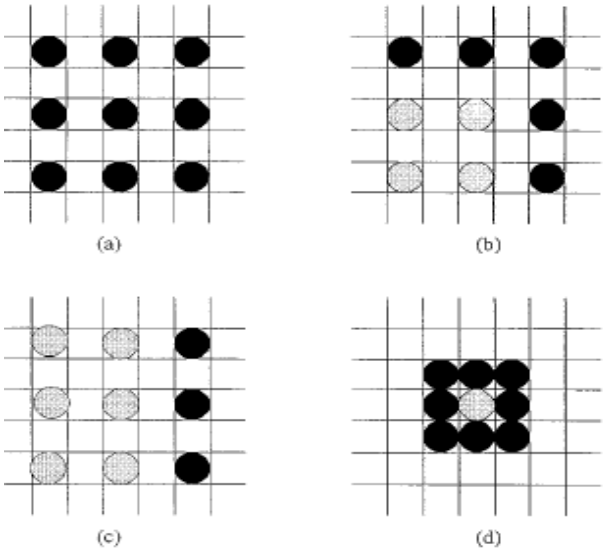

Figure 4. Search Patterns of 4SS. (a) First Step, (b) second/third step, (c) second/third step, (d) fourth step

From the algorithm, we can find that the intermediate steps of the 4SS may be skipped and then jumped to the final step with a 3 x 3 window if at any time the minimum BDM point is located at the center of the search window. Based on this four-step search pattern, we can cover the whole 15 x 15 displacement window even only small search windows, 5 x 5 and 3 x 3, are used.

This 4SS produce better performance than the 3SS and has similar performance as compared with the N3SS. While the worst case computational requirement of the 4SS is 27 block matches, which is only just 2 more block matches as compared with the 3SS. The 4SS also possesses the regularity and simplicity of hardware-oriented features.

5. Diamond Search Motion Estimation:

The DS algorithm [11] employs two search patterns. The first pattern, called large diamond search pattern (LDSP) shown in Figure. 5(a), comprises nine checking points from which eight points surround the center one to compose a diamond shape. The second pattern consisting of five checking points forms a small diamond shape, called small diamond search pattern (SDSP) shown in Figure. 5(b). In the searching procedure of the DS algorithm, LDSP is repeatedly used until the minimum block distortion (MBD) occurs at the center point. DS algorithm consistently performs well for the image sequence with wide range of motion content.

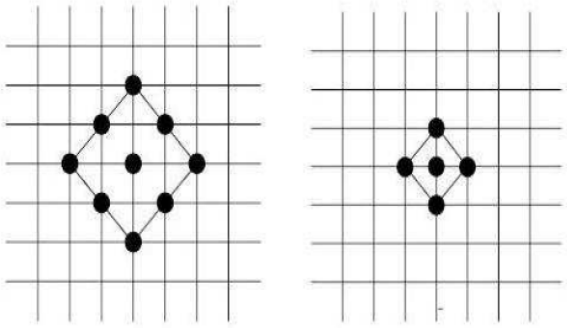

Figure 5. (a) Large Diamond Search Pattern (LDSP), (b) Small Diamond Search Pattern (SDSP)

The DS algorithm is summarized as follows.

Step1: The initial LDSP is centered at the origin of the search window, and the 9 checking points of LDSP are tested. If the MBD point calculated is located at the center position, go to Step 3; otherwise, go to Step 2.

Step2: The MBD point found in the previous search step is re-positioned as the center point to form a new LDSP. If the new MBD point obtained is located at the center position, go to Step 3; otherwise, recursively repeat this step.

Step3: Switch the search pattern from LDSP to SDSP. The MBD point found in this step is the final solution of the motion vector which points to the best matching block.

This algorithm consistently performs well for the image sequence with wide range of motion content. It also outperforms the well-known TSS algorithm and achieves close performance compared to NTSS while reducing computation by 20%-25% approximately.

6. Four Step Genetic Search Motion Estimation:

The four-step genetic search [12] for BMA combines GA and **4SS**. This method has similar performance to FS in terms of MSE. In addition, number of search points required by the proposed algorithm is approximately equal to 14% of FS and closed to 3SS. The speed up ratio between this and FS is 5.6 times. Figure. 6, shows the block diagram of the four step genetic search algorithm.GA can search individual points in any arbitrary search space with much less search steps. These individual points are represented by binary string. They are named **as** chromosome. In each generation, there will be a number of chromosomes**.** Each chromosome will be evaluated by a fitness function**.** A chromosome with higher fitness value means it is strong. The probability for that chromosome survival will be high. On the other hand, a chromosome with lower fitness value means the probability of that chromosome will survive for next generation is low**.** It will be replaced by a strong chromosome. Finally, chromosomes in each generation will become better. A set of genetic operator, namely crossover and mutation operators, will be used for generating new chromosomes. After several generations, the chromosome with the highest fitness value will be the best chromosome to represent the extreme within the search space. In addition, the search will stop if any chromosome's fitness value meet the stopping criteria.

Both 4GS and lightweight genetic search algorithm (LGSA) need more search points than 3SS and 4SS. They use 2 to 44 % more search points than 3%. The new method needs less number of search points than LGSA. It only takes ***2*** to 27 % more search points than the conventional 3SS. On the other hand, LGSA takes 28 to 44 % more search points than 3SS. In general, the computational cost of 4GS is lower than LGSA. In addition, 4GS takes 39% to 44% more search points than 4SS. However, the search number of this algorithm is much less than FS. The algorithm only takes approximate 14 % search points compared to FS. LGSA and 4GA take longer search time than 3% to compute motion vector. Amount of search time is increased by 25 % to 60%. On the other hand, the computation time of LGSA and 4GA are much less than FS. They save approximate 80% to 85% of computation time. Moreover, the search time of 4GS is shorter than LGSA by 6.7% to 13.4%. In general, 4GS and LGSA perform better than 3SS and 4SS. However, the performance of the genetic algorithm based BMA is depending on nature of image sequence. LGSA performs better with Football, Mobile, and Table Tennis sequence. This algorithm performs better with'Garden, Miss American and Saleman sequence. The average mean square error (MSE) of estimated sequence by LGSA and 4GS are very close to each other. Nevertheless, the 4GS and LGSA both perform comparatively to FS. As a result, this algorithm is suitable for H.261, MPEG-1 and MPEG-2. Since the new algorithm with regularity, it is also suitable for hardware implementation.

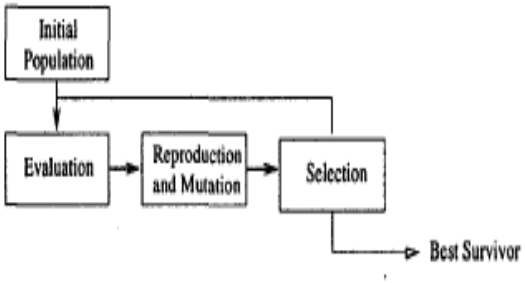

Figure 6. Four Step genetic search

7. Hexagonal Search Motion Estimation:

In block motion estimation, a search pattern with a different shape or size has a very important impact on search speed and distortion performance. A square-shaped search pattern is commonly adopted in many popular fast algorithms. Then, a diamond-shaped search pattern was introduced in fast block motion estimation, which has exhibited faster search speed. Based on an in-depth examination of the influence of search pattern on speed performance, a novel algorithm using a hexagon-based search pattern [13] to achieve further improvement. Analysis shows that a speed improvement rate of the hexagon-based search (HEXBS) algorithm over the diamond search (DS) algorithm can be as high as over 80% for locating some motion vectors in certain scenarios. In short, this HEXBS algorithm can find a same motion vector with fewer search points than the DS algorithm. Generally speaking, the larger the motion vector, the more search points the HEXBS algorithm can save.

The hexagonal search algorithm offers better improvement is speed and accuracy when compared to the previous methods. They are of two types as depicted in Figure. 7(a) and Figure.7(b).

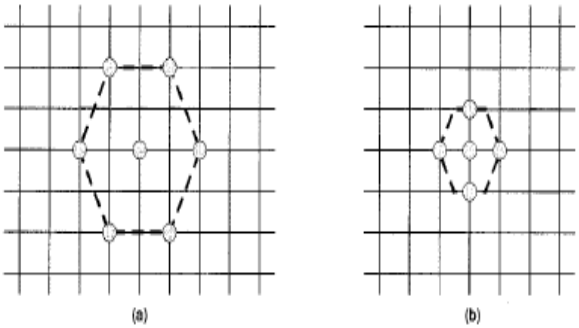

Figure 7. (a) Large Hexagonal Block Search(HEXBS), (b) Small Hexagonal Block Search(HEXBS)

The HEXBS algorithm is summarized as follows:

Step 1) The large hexagon with seven checking points is centered at, the center of a predefined search window in the motion field. If the MBD point is found to be at the center of the hexagon, proceed to Step (3) (Ending); otherwise, proceed to Step (2) (Searching).

Step 2) With the MBD point in the previous search step as the center, a new large hexagon is formed. Three new candidate points are checked, and the MBD point is again identified. If the MBD point is still the center point of the newly formed hexagon, then go to Step (3) (Ending); otherwise, repeat this step continuously.

Step 3) Switch the search pattern from the large to the small size of the hexagon. The four points covered by the small hexagon are evaluated to compare with the current MBD point. The new MBD point is the final solution of the motion vector.

From the procedure, it can be easily derived that the total number of search points per block will be-

$$N(mx,my)=7+3xn+4$$

where (mx,my) is the final motion vector found and n is the number of executions in step 2.

The superiority of this HEXBS to the other fast methods in terms of using the smallest number of search points with a very small penalty of marginal degradation in distortion. In terms of hardware implementation, this HEXBS approach adopts a regular hexagonal search pattern, which possesses the regularity and simplicity of hardware-oriented features. Compared with the hardware implementation for DS, the realization of HEXBS

algorithm may consume fewer MAD calculators due to its fewer search points each time, thus saving cost and hardware size.

8. Enhanced Hexagonal Search Motion Estimation:

Most motion estimation algorithms developed attempt to speed up the coarse search without considering accelerating the focused inner search. On top of the hexagonal search method recently developed, an enhanced hexagonal search algorithm is proposed to further improve the performance in terms of reducing number of search points and distortion, where a novel fast inner search is employed by exploiting the distortion information of the evaluated points. An enhanced HEXBS [14] is developed by making use of an elaborated 6-side-based fast inner search scheme. The reduction in number of search points for the enhanced HEXBS algorithm is expected to be from two aspects, i.e., one from the prediction for a good starting point using the predictive HEXBS, and the other from the proposed fast inner search.

The enhanced HEXBS has consistently achieved the fastest motion estimation in terms of number of search points with smaller MSE distortions than the original HEXBS and almost the same MSE distortions as the predictive HEXBS. The resulted enhanced HEXBS algorithm outperforms the original HEXBS remarkably in terms of search speed and distortion performance, up to 57% in both speed-improvement rate and distortion-decrease rate.

9. Genetic Rhombus Pattern Search Motion Estimation:

Pattern-based block motion estimation (PBME) is one of the most effective yet computational intensive tools in digital video coding standards. The weighting function is strongly related to the performance of a search algorithm, and, therefore, [15] design a fast search algorithm that has the smallest possible weighting function value at all locations. This is done by first constructing a target weighting function that is based on the prior knowledge and has the smallest possible values at all locations. Followed by a search pattern in Figure. 8, that achieves the desired weighting function.

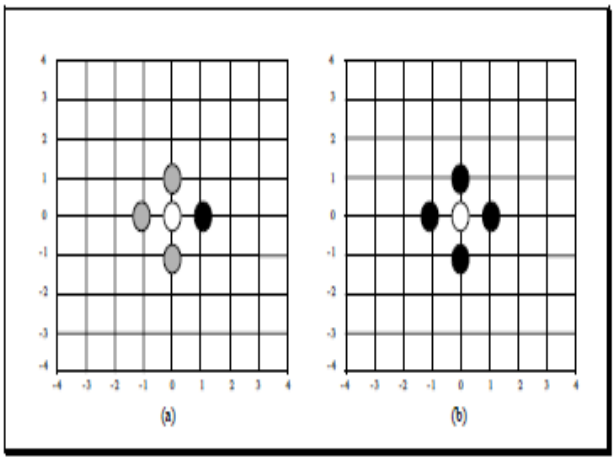

Figure 8. Search patterns for genetic rhombus search

Most pattern-based search algorithms typically consist of two stages: 1) coarse initial search stage and 2) fine ending search stage. Generally, the coarse search stage focuses on finding a rough location of the optimal motion vector, and the fine ending search stage locates its precise location. Thus one search pattern for each stage. In the coarse search stage, because the shortest path between two points on a plane is the strait line, the fastest search path for a search algorithm is the strait line from the starting point directly to the best motion vector. Flow chart of the search algorithm is shown in Figure 9. Mainly, it consists of 4 stages: initializing stage, mutation, competition and end stage.

Comparing to the other popular search algorithms, GRPS reduces the average search points for more than 20% while it maintains a similar level of coded image PSNR quality. The computing gain (CG) is defined as the ratio of the average number of search points (ASP) minus one, and the quality gain (QG) is defined as the PSNR difference. The ASP of GRPS on the average is 26% faster than that of ERPS, 51% faster than EHS, 125% faster than DS, 160% faster than FSS, and 136 times faster than the full search (FS). And the PSNR of GRPS on the average is the same or better than all the other search algorithms (0~+0.18 dB). It is clear that GRPS outperforms all the other search algorithms in terms of ASP for all sequences, while its quality on the average is comparable to all the other algorithms.

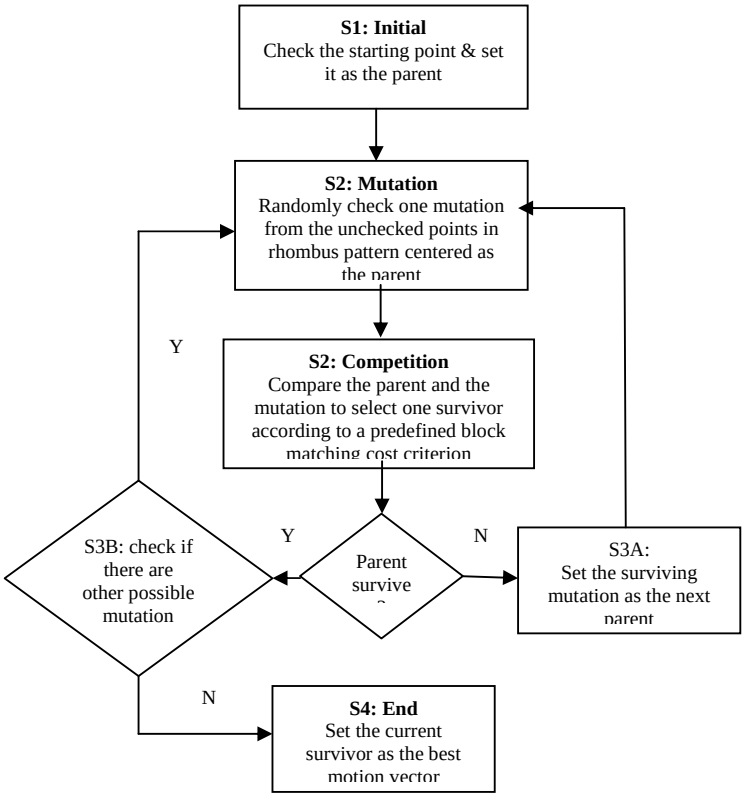

Figure 9. Flowchart of GRPS

10. *Adaptive Genetic Pattern Search Motion Estimation:*

Adaptive genetic pattern search algorithm [16] uses two search pattern sets, the genetic rhombus search patterns (GRPS) and the genetic enhanced hexagonal search patterns (GEHS).The flow chart of GRPS is shown in Figure. 9 and its search patterns are shown in Figure. 8. In step 2 (S2), it checks one of the search points in Figure. 8(a), and in step 3B (S3B), it examines if all the points in Figure. 8(b) has been checked. The flow chart of GEHS is shown in Figure. 10 and its associated search patterns are shown in Figure. 11. In step 2 (S2), it checks one of the search points in Figure. 11(a). In step 3B (S3B), it examines if all the points in Fig. 11(b) are checked. In step 4 (S4), six block matching costs are computed using the two-point patterns in the hexagonal pattern as indicated by Groups I to VI in Figure. 11. Then, determine which search pattern (direction) to be used next. When the smallest cost is in Groups II, III, V, and VI in Figure. 11(b), two extra points are checked, as exemplified in Figure. 11(c). Or, if the smallest cost is in Groups I and IV in Figure. 11(b), three extra points are checked, as exemplified in Figure. 11(d).

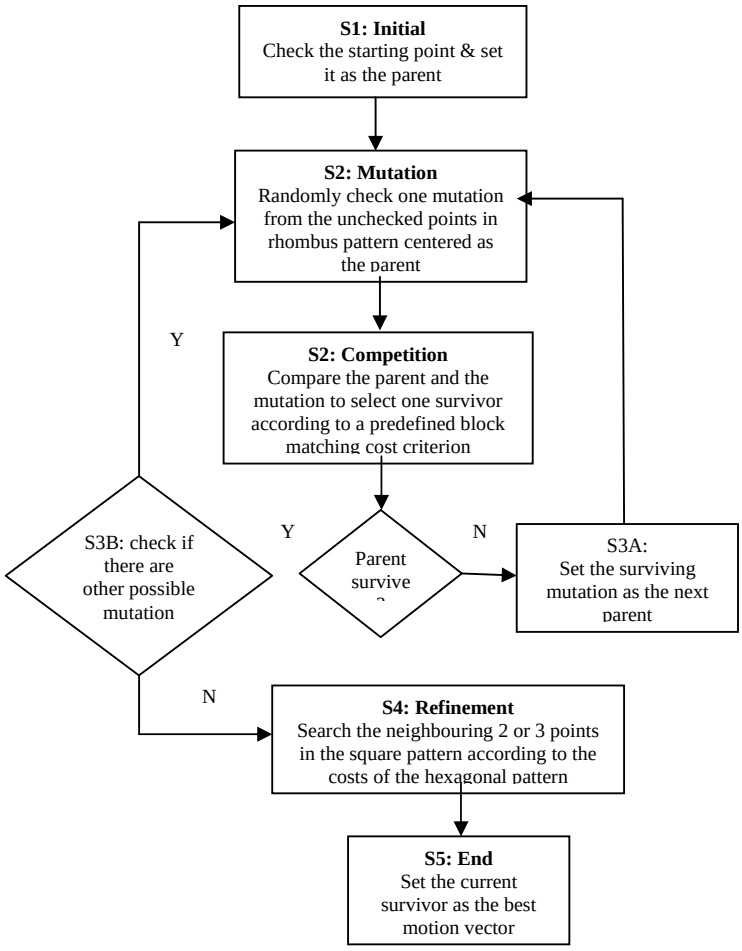

Figure 10. Flow chart of GEHS

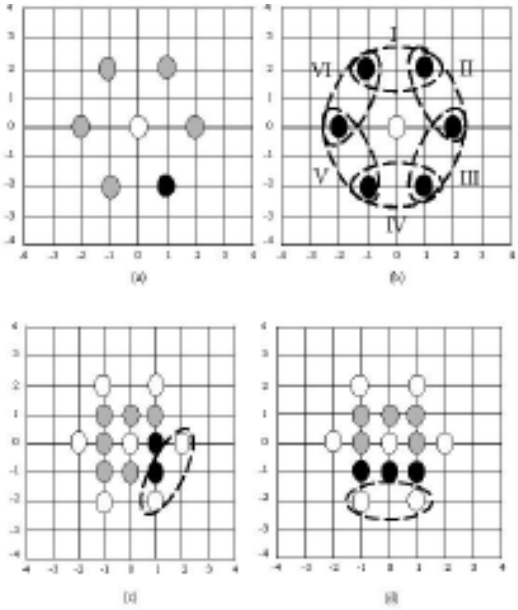

Figure 11. Search pattern of GEHS

An adaptive genetic search pattern search (AGPS) scheme which combines GRPS and GEHS. Using the standard deviation of MVs in the previous frame to decide which search algorithm to be used in the current frame, AGPS achieves more than 30% acceleration in terms of the average search points while it maintains a similar level of picture quality

Jang. *et. al* [17] look into every component of a typical PBME algorithm and fine tune the major components systematically to achieve the optimal or nearly optimal results. The methodology is developed based on proposed analytical model together with statistical tools. First, it uses the analytic model to analyze and design effective genetic-algorithm-based search patterns. Moreover, an adaptive switching strategy is proposed that dynamically switches between two search patterns. Because the contents of video sequences vary drastically, one single search pattern may not produce the best result in terms of speed and PSNR. Thus, the adaptive pattern-switching search algorithms were proposed [19]–[23]. These algorithms are empirically constructed and the switching criterion is often based on block (feature) classification. Second, this PBME model is extended to evaluate the efficiency of starting (initial search) points. A near optimal set of starting points is progressively identified. Last, the early termination threshold techniques are studied and suggest a metric in selecting an effective threshold. The *early termination* mechanism terminates the search process when the block-matching error produced by a MV (in the search area) is smaller than a pre-chosen threshold. And this MV is accepted as the best MV. Clearly, there is a trade-off between the MV quality (matching error) and the computational speed. Thus, the challenge is to find the termination threshold that maximizes the speed gain and minimizes the quality degradation. A systematic method is used to find the nearly optimal early termination threshold (ETT). Combining all these techniques, a PBME algorithm is developed that outperforms most popular algorithms.

This algorithm outperforms DS by 538% at average search points and FS by 405 times and the average PSNR quality is slightly better (0.10–0.19 dB) than all the other algorithms including FS. This may be due to the fact that our scheme often prefers a smaller value MV, which requires fewer bits in coding. And a few additional bits are available for texture (DCT coefficients) coding, which results in better overall PSNR.

IV. CONCLUSION

In this paper we presented an overview of Block matching motion estimation algorithms in video coding. In the overview of these techniques, we discussed various block matching algorithms that range from the very basic Full Search to the recent fast adaptive algorithms like Pattern Based Search. Normally ME is quite computationally intensive and can consume up to 80% of the computational power of the encoder if the full search (FS) is used by exhaustively evaluating all possible candidate blocks within the search window. As a consequence, the computation of video coding is greatly reduced with pattern based block motion estimation.

REFERENCES

[1] T. Wiegand,G. J. Sullivan, G. Bjontegaard, and A. Luthur, “Overview of the H.264/AVC video coding standard,” IEEE Trans. Circuits Syst. Video Technol., vol. 13, no. 7, pp. 560-576, Jul. 2003.

[2] Information Technology – Generic Coding of Moving Pictures and Associated Audio Information: Video, ISO/IEC 13 818-2-ITU-T Rec. H.262 (MPEG-2 Video), 1995.

[3] Information Technology – Generic Coding of Audio-Visual Objects, Part 2: Visual, ISO/IEC 14 496-2 (MPEG-4 Video), 1999.

[4] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion-compensated interframe coding for video conferencing,” Proc. NTC81, pp. C9.6.1-9.6.5, New Orleans, LA. Nov. 1981.

[5] J. R. Jain and A. K. Jain, :Displacement measurement and its application in interframe image coding,” IEEE Trans. Commun., vol. COM-29, pp. 1799-1808, Dec. 1981.

[6] R. Srinivasan and K. R. Rao, “Predictive coding based on efficient motion estimation,” IEEE Trans. Commun., vol. COM-33, pp. 888-896, Aug. 1985.

[7] S. Kappagantula and K. R. Rao, :Motion compensated interframe image prediction,” IEEE Trans. Commun., vol. COM-33, pp. 1011-1015, Sept. 1985.

[8] M. Ghanbari, “The cross-search algorithm for motion estimation,” IEEE Trans.Commun., vol. 38, pp. 950-953, July 1990.

[9] R. Li, B. Zeng, and M. L. Liou, “A new three-step search algorithm for block motion estimation,” IEEE Trans. Circuits Syst. Video Technol., vol. 4, pp. 438442, Aug.1994.

[10] L. M. Po and W. C. Ma, “A novel four-step search algorithm for fast block motion estimation,”IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 313317, June 1996.

[11] S. Zhu and K.-K. Ma, “A new diamond search algorithm for fast block-matching motion estimation,” in Proc. Int. Conf. Inf. Commun. Signal Process. (ICICS ’97), vol. 1, Sep. 9–12, 1997, pp.292–296.

[12] M. F. So and A. Wu, “Four-step genetic search for block motion estimation,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process.(ICASSP ’98), vol. 3. May 1998, pp. 1393–1396.

[13] C. Zhu, X. Lin, and L.-P. Chau, “Hexagon-based search pattern for fast block motion estimation,” IEEE Trans. Circuits Syst. Video Technol.,vol. 12, no. 5, pp. 349–355, May 2002.

[14] C. Zhu, X. Lin, L. Chau, and L.-M. Po, “Enhanced hexagonal search for fast block motion estimation,” IEEE Trans. Circuits Syst. Video Technol.,vol. 14, no. 10, pp. 1210–1214, Oct. 2004.

[15] J.-J. Tsai and H.-M. Hang, “A genetic rhombus pattern search for block motion estimation,” in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS ’07), New Orleans, LA, May 2007, pp. 3655–3658.

[16] J.-J. Tsai and H.-M. Hang, “On adaptive pattern selection for block motion estimation algorithms,” in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP ’07), Honolulu, HI, vol. 1. Apr. 2007, pp.I-1173–I-1176

[17] J.-J. Tsai and H.-M. Hang, “On the design of pattern-based block motion estimation algorithms,” IEEE Trans. Circuits Syst. Video Technol.,vol. 20, no. 1, pp. 136–143, Jan. 2010.

[18] L. W. Lee, J. F. Wang, J. Y. Lee, and J. D. Shie, "Dynamic search-window adjustment and interlaced search for block matching algorithm," IEEE Trans. Circuits Syst. Video Technol.,vol. 3, no. 1, Feb. 1993.
[19] S. F. Lin, M. T. Lu, H. Chen, and C. H. Pan, "Fast mult-frame motion estimation for H.264 and its applications to complexity-aware streaming," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'05), Kobe, Japan, vol.2, May 2005, pp. 1505-1508.
[20] S. Y. Huang, C. Y. Cho, and J. S. Wang, "Adaptive fast block-matching algorithm by switching search patterns for sequences with wide-range motion content," IEEE Trans. Circuits Syst. Video Technol.,vol. 15, no. 11, pp. 1373–1384, Nov. 2005.
[21] I. Ahmad, W. Zheng, J. Luo, and M. Liou, "A fast adaptive motion estimation algorithm," IEEE Trans. Circuits Syst. Video Technol.,vol. 16, no. 3, pp. 420-438, Mar. 2006.
[22] M. C. Chang, and J. S. Chein, "An adaptive search algorithm based on block classification for fast block motion estimation," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS'06), May 2006, pp. 3982-3985.
[23] Y. Liu and S. Oraintara, "Complexity comparision of fast block matching motion estimation algoritms," in Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP'04), vol.3, May 2004, pp. 341-344.