An Adaptive Pattern Search Algorithm for Motion Estimation

Ching-Huang Wei and Yeng-Hwa Chiang
Department of Computer and Communication Engineering
National Kaohsiung First University of Science and Technology, Kaohsiung, Taiwan
chwei@ccms.nkfust.edu.tw, u9211826@ccms.nkfust.edu.tw

Abstract—In this paper we propose an efficient fast block-matching algorithm for motion estimation. Two different patterns, 3x3 and 5x5 cross patterns alternative in inner ring, and three search strategies, including prediction, threshold, and hierarchy, are utilized to form the procedure of the proposed algorithm. The simulation results show that number of search points required in the proposed algorithm is about 2-3% of full-search algorithm, 33-40% of diamond search algorithm, and approximated to that of dual-cross search algorithm while providing the competitive coding quality in comparison with both fast algorithms.

I. INTRODUCTION

Due to the great development and demand on video compression systems like H.261, H.263, MEG-1, MPEG-2, MPEG-4, and H.264, block-matching algorithm (BMA) for motion estimation and compensation plays an important role in video coding standards because the high computational complexity is consumed on it. Full-search (FS) block-matching algorithm provides an optimal result of motion vectors and residue blocks by exhaustively searching all area within the search window, but it yields very high computational complexity and makes it difficult to be implemented in real-time applications. Hence, there were many fast BMAs developed for optimizing search speed while preserving the distortion as small as possible, such as three-step search (TSS) [1], new three-step search (NTSS) [2], four-step search (4SS) [3], block-based gradient descent search (BBGDS) [4], diamond search (DS) [5], hexagon-based search (HEXBS) [6], enhanced HEXBS (E_HEXBS) [7], and adaptive dual-cross search algorithm (ADCS/DCS) [8] etc.

However, these fast BMAs still have some drawbacks [5]. For example, TSS may mislead the search path to a wrong direction in the first step and become inefficient for the estimation of small motions. NTSS can achieve good solution, but it still costs higher complexity and loses the regularity and simplicity of TSS. 4SS obtains better performance than TSS and similar performance compared to NTSS, but it even needs at least 17 search points in each block. BBGDS works very well in small motion sequences, but it’s easy to be trapped into local minimum in large motion sequences. DS exists some redundancy among the search points because it uses compact pattern in terms of distance between neighboring points. HEXBS is likely to lose optimal solutions because it does not check the four diagonal points around the hexagon’s center at its last search step. ADCS needs additional memory to save the minimal sums of absolute differences (SAD) of two previous adjacent blocks to set the threshold for the current block.

In this paper, we propose a new fast block-matching algorithm, called adaptive pattern search algorithm (APS), which could have faster search speed than the existing fast algorithms and provide the competitive coding quality.

II. OBSERVATION

For fast BMAs, adopting suitable shape and size to form the search patterns always plays a very important role and influences the performance very obviously. Like BBGDS algorithm, putting more search points around the center is helpful for stationary blocks. However, if we want to get better results in non-stationary blocks, we should distribute more search points faraway from the center as exploited in the first step of TSS algorithm. NTSS algorithm may conform these two requests and provide good solutions, but it is still required to test too many search points for each block. Hexagon shape yields fewer search points on average, but it usually loses optimal solutions. Hence, we develop a fast BMA to provide a better solution while dealing with both stationary and non-stationary blocks well and requiring search points as few as possible simultaneously.
First, we put more search points around the center. However, if we directly adopt a $3 \times 3$ checking block as exploited in BBGDS algorithm, it may need too many search points. Hence, we use a $3 \times 3$ cross pattern, like the small diamond search pattern (SDSP) of DS algorithm. Second, in order to search motion vectors in non-stationary blocks more quickly and accurately, it’s necessary to put some search points far away from the center as exploited in the first step of TSS. By combining those two requests, we form the initial pattern as shown in Fig. 1(a). Hence, the initial pattern is coupled by two parts, inner ring, which is denoted as five black points, and outer ring, which is labeled from $\circ_1$ to $\circ_8$.

![Fig. 1. Initial pattern and two different patterns used in the proposed algorithm. (a) Initial pattern, (b) $3 \times 3$ cross pattern, and (c) $5 \times 5$ cross pattern.](image)

III. THREE ADDITIONAL STRATEGIES

Three strategies, including prediction, threshold, and hierarchy, are utilized in the proposed algorithm to further reduce unnecessary search points.

A. Prediction

Predicting the motion vector of the current block with the information of previous adjacent blocks is a useful pre-process which has been used in many fast BMAs like [7]-[12], etc. We use two previous motion vectors of the left and above adjacent blocks for prediction, denoted as $(mv_x^L,mv_y^L)$ and $(mv_x^A,mv_y^A)$. Each current block should check the SAD values at $(mv_x^L,mv_y^L), (mv_x^A,mv_y^A),$ and $(0,0)$ first and decide a start point, denoted as $(mv_x^0,mv_y^0)$, the one with the lowest SAD value among these three positions.

B. Threshold

A threshold value could be compared with SAD values to decide whether it is likely to be good enough as the optimal solution for early search termination. Selecting a suitable threshold value can save lots of search points without degrading too much coding quality. We set the threshold value fixed as $ThD = 512$ in our experiments, a trade off between the coding quality and the search speed. ADCS [8] adopts a variable threshold, but it needs additional memory.

C. Hierarchy

The third strategy used in the proposed algorithm, called hierarchy, works on the outer ring of initial pattern, where the search points are denoted from $\circ$ to $\circ$ as shown in Fig. 1(a), and decides which points on the outer ring should be tested or not. If the initial start point after prediction is correspondent with

$$| mv_x^0 | \leq 5 AND | mv_y^0 | \leq 5$$

then we don’t test any points of outer ring any more; otherwise, we change inner ring from $3 \times 3$ cross pattern as shown in Fig. 1(b) to $5 \times 5$ cross pattern as shown in Fig.1(c), test all five search points of inner ring, and only test an appropriate part of search points of outer ring. This additional step can save at least five search points from the whole outer ring. The decision rule of hierarchy is explained as follows

If $| mv_x^0 | \leq 5 AND | mv_y^0 | \leq 5$ then

- test $3 \times 3$ cross pattern of inner ring

else

- change inner ring to $5 \times 5$ cross pattern
- test $5 \times 5$ cross pattern of inner ring and
- If $mv_x^0 < 0$ and $mv_y^0 < 0$
  - test points $\circ$, $\circ$, and $\circ$
If \(mv_x^0 \geq 0\) and \(mv_y^0 \leq 0\)

- test points \(\circ, \circ, \text{ and } \circ\)

If \(mv_x^0 \leq 0\) and \(mv_y^0 \geq 0\)

- test points \(\circ, \circ, \text{ and } \circ\)

If \(mv_x^0 > 0\) and \(mv_y^0 > 0\)

- test points \(\circ, \circ, \text{ and } \circ\)

IV. THE PROPOSED ALGORITHM

The proposed algorithm making use of three strategies mentioned above is described as follows:

Step 1) Compute SAD values at three search points: \((0,0), (mv^x_L, mv^y_L), \text{ and } (mv^x_A, mv^y_A)\). Select the point with the lowest SAD value as the start point, \((mv_x^0, mv_y^0)\).

Step 2) If the SAD value of the start point is less than or equal to the threshold, \(ThD=512\), then stop the search and set start point as the motion vector; otherwise, go to next step.

Step 3) If \(|mv_x^0| \leq 5 \text{ AND } |mv_y^0| \leq 5\), then only test five search points of 3×3 cross pattern in inner ring as initial pattern; otherwise, change inner ring pattern from 3×3 to 5×5 cross pattern and choose an appropriate part of search points in outer ring by the decision rule of hierarchy mentioned above as initial pattern.

Step 4) Locate the center of initial pattern as the new start point and test all search points on initial pattern. If the minimum SAD value appears at the start point, then stop the search; otherwise, relocate the new center and go to next step.

Step 5) Use 3×3 cross pattern for refinement until the minimum SAD value appears at the center of 3×3 cross pattern.

Fig. 2 illustrates two examples for the proposed algorithm. In Fig. 2(a), assume that the start point is \((-3,-1)\) after prediction and its SAD value is larger than threshold \(ThD\). There is a search path to lead to the motion vector \((-4,-2)\). Fig. 2(b) shows another example where the start point is located at \((-6,-2)\) and the found motion vector is \((-7,-6)\). It is noted that because the start point exceeds the range of \((\pm 5, \pm 5)\), the search points \(\circ, \circ, \circ\), and \(\circ\) of outer ring are added to be tested.

V. SIMULATION RESULTS

In our simulation experiments, the proposed algorithm is compared with several well-known block-matching algorithms, including FS, TSS [1], NTSS [2], 4SS [3], BBGDS [4], DS [5], HEXBS [6], E_HEXBS [7], and DCS [8], in terms of average number of search points per block and the average
mean squared error (MSE) per frame. Because ADCS [8] requires additional memory, we would not consider it in comparison. Three different video sequences are tested including “salesman”, “flower”, and “football” sequences all in CIF 352 × 288 format, 50 frames. The block size is chosen as 16×16 pixels and the search widow size is set to 15×15 pixels. In the simulations, we treat the sum of absolute differences (SAD) as the block distortion measurement (BDM).

Table I shows the performance comparison in terms of average MSE values per frame and average number of search points per block. FS algorithm has the lowest MSE values in each tested sequences and stands for the optimal solutions, but it yields the largest number of search points, too. The proposed algorithm is superior to other fast BMA in terms of number of search points, and almost has competitive MSE distortion with both DS and DCS algorithms. Table I could be illustrated in Fig. 3 where the more the performance of algorithm is close to origin, the more the algorithm performs well. Both the proposed algorithm and DCS algorithm outperform the other fast BMAs. Number of search points required in the proposed algorithm is about 2-3% of FS algorithm, 33-40% of DS algorithm, and approximated to that of DCS algorithm while providing the competitive coding quality compared with both fast algorithms.

Fig. 4 shows the MSE distribution frame by frame. The proposed algorithm could achieve the coding quality approximate to both DS and DCS algorithms while requiring much fewer search points than DS algorithm.

VI. CONCLUSION

We have presented a fast block-matching algorithm for motion estimation based on an adaptive search pattern. By the analysis of motion vectors for both stationary and non-stationary video sequences, we could adopt two different patterns, 3×3 and 5×5 cross pattern in inner ring, and three additional strategies, including prediction, threshold, and hierarchy, to form the proposed APS algorithm. The simulation results demonstrate that the proposed

<table>
<thead>
<tr>
<th>Sequence</th>
<th>FS</th>
<th>TSS</th>
<th>NTSS</th>
<th>4SS</th>
<th>BBGDS</th>
<th>DS</th>
<th>HEXBS</th>
<th>E_HEXBS</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search points</td>
<td>293.627</td>
<td>342.787</td>
<td>304.421</td>
<td>322.587</td>
<td>314.057</td>
<td>317.029</td>
<td>332.223</td>
<td>303.726</td>
<td>299.619</td>
</tr>
<tr>
<td>MSE</td>
<td>299.619</td>
<td>5.32</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Search points</td>
<td>328.022</td>
<td>299.541</td>
<td>314.057</td>
<td>322.587</td>
<td>314.057</td>
<td>317.029</td>
<td>332.223</td>
<td>303.726</td>
<td>299.619</td>
</tr>
<tr>
<td>Search points</td>
<td>328.022</td>
<td>299.541</td>
<td>314.057</td>
<td>322.587</td>
<td>314.057</td>
<td>317.029</td>
<td>332.223</td>
<td>303.726</td>
<td>299.619</td>
</tr>
</tbody>
</table>

Fig. 3. Performance comparison on MSE (in y-coordinate) versus average search points costs (in x-coordinate) for (a) salesman, (b) flower, and (c) football sequences.
algorithm could provide a better performance than other fast block-matching algorithms and have the competitive coding quality compared with both DS and DCS algorithms.

**REFERENCES**


