A Fast Octagon Based Search Algorithm for Motion Estimation

Lap-Pui Chau, Ce Zhu

School of Electrical & Electronic Engineering, Nanyang Technological University, Singapore

Abstract
Fast motion estimation technique plays a very important part in video compression standard. In this paper, we propose a novel algorithm using octagon-based search (OCTBS) pattern for fast block motion estimation. The proposed OCTBS algorithm can find motion vector with fewer search points than the diamond search (DS) algorithm. The speedup gain of the OCTBS method over the DS algorithm is more striking for finding large motion vectors. Experimental results justify the performance of the OCTBS algorithm compared with several other popular search algorithms.

Keywords
Motion Estimation, Video Compression.

1. Introduction
In video coding standards, block motion estimation plays a very important part. The quests for finding the best block motion estimation become a challenge. Therefore, developing fast algorithms are highly desired to significantly speed up the process without sacrificing the distortion seriously. Many computationally efficient motion estimation algorithms [1-6] were developed, typically among which are three-step search (TSS) [1], new three-step search (NTSS) [2], four-step search (4SS) [3], block-based gradient descent search (BBGDS) [4] and diamond search (DS) [5-6] algorithms.

In TSS, NTSS, 4SS and BBGDS algorithms, square-shaped search patterns of different sizes are employed. On the other hand, the DS algorithm adopts a diamond-shaped search pattern, which has demonstrated faster processing with similar distortion in comparison with NTSS and 4SS. This causes us to investigate why diamond search pattern can get speed improvement over some square-shaped search patterns and what is the reason behind. As a result, one may wonder whether there is any other pattern shape better than diamond for faster block motion estimation. In the following, we propose a octagon-based search algorithm that can achieve substantial speed improvement over the diamond search algorithm with just slightly worse distortion performance.
2. Octagon Based Search Algorithm

2.1 Octagonal Search Pattern

Ideally, a circle shaped search pattern in which a uniform distribution of a minimum number of search points is desirable to achieve the lowest error and the fastest search speed. Each search point can be equally utilized with maximum efficiency, where the redundancy among search points should be removed maximally. Referring to the diamond search pattern [5-6], we can see the diamond shape is not approximate enough to a circle, which is just a 90 degree rotation of a square. Therefore, a more circle-approximated search pattern is expected with a uniform distribution of a minimum number of search points. A octagon based search pattern is devised which is depicted in Fig. 1.

![Fig. 1 Octagon-based search (OCTBS): large and small Octagon patterns.](image)

There are two different sizes of octagonal search patterns. The larger one consists of 9 checking points marked as 1 with the centre surrounded by 8 endpoints. A small octagon based pattern, has 4 checking points. In the search process, the large octagon based search pattern keeps advancing with the centre moving to any of the 8 endpoints. Whichever endpoints the centre of the large octagon based search pattern moves to, there are always three new endpoints emerging with the other three endpoints being overlapped as shown in Fig. 2. In any direction the octagon advances 5 points, but the two shaded search points are ignored because it is highly co-related to check point 3 and 9 respectively. Therefore, each octagon step advances 3 new checking points.

![Fig. 2 Each octagon step advances 3 new checking points.](image)
2.2 Algorithm Development

With the development of the Octagon Based Search (OCTBS) algorithm, the search procedures are as follows. In the first step, the large octagonal search pattern with 9 checking points is used for search. If the optimum is found at the centre, then switch to use shrunk octagon pattern including 4 checking points for the focused inner search. Otherwise, the search continues around the point with mean square error (MSE) using the same large hexagon pattern. Note that while the large octagonal search pattern moves along the direction of decreasing distortion, only three new checking points non-overlapping will be evaluated as candidates each time. Fig. 3 shows an example of the search path strategy leading to motion vector (+5, -4). Note that a small OCTBS pattern is applied in the final step after the best candidate search point at step 3 remains the best at step 4. Totally 22(9+3+3+3+4) search points are evaluated in 5 steps.

![Fig. 3 Search path example locating the motion vector (+5, -4) by OCTBS.](image)

The proposed Octagon Based Search (OCTBS) algorithm can be summarised in the following steps.

Step i) Starting: The large octagon with 9 checking points is centred at (0,0), the centre of a predefined search window in the motion field. If the MSE point is found to be at the centre of the octagon, proceed to Step 3) (Ending); otherwise, proceed to Step 2) (Searching).

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

Step iii) Ending: Switch the search pattern from the large size of octagon to the small size of octagon. The four points covered by the small octagon are evaluated to compare with the current MSE point. The new MSE point is the final solution of motion vector.

From the above procedure, it can be easily derived that the total number of search points per block will be
\[ N_{\text{OCTBS}}(m_x, m_y) = 9 + 3 \times n + 4 \]  
(1)

where \((m_x, m_y)\) is the final motion vector found, and \(n\) is the number of execution of Step ii).

2.3 Analysis of the Algorithm

In this subsection, discussion will be on the proposed OCTBS algorithm as compared with the diamond search (DS) algorithm for number of points checked to find same motion vectors. For block motion estimation, computational complexity can be measured by number of search points required for each motion vector estimation. The diamond search method requires the following number of search points per block

\[ N_{\text{DS}}(m_x, m_y) = 9 + M \times n_d + 4 \]  
(2)

where \(M\) is either 5 or 3 depending on the search direction and \(n_d\) depends on the search distance. The \(n_d\) in equation (2) is always greater than or equal to the \(n\) in equation (1) for finding a same motion vector \((m_x, m_y)\). Especially in the direction of \(\tan^{-1}(\pm 2) \pm k \pi\), \(k=0, 1\), the speed improvement rate (SIR) of OCTBS over DS can be even up to 30% where the SIR can be obtained by

\[
\text{SIR} \left( \% \right) = \left( \frac{9 + M \times n_d + 4 - (9 + 3 \times n + 4)}{9 + 3 \times n + 4} \right) \times 100\%
\]

The number of search points saved does not sacrifice the distortion. The percentage increase in MSE can be obtained by

\[
\text{MSE} \left( \% \right) = \left( \frac{\text{MSE}_{\text{OCTBS}} - \text{MSE}_{\text{DS}}}{\text{MSE}_{\text{DS}}} \right) \times 100\%
\]

In short, the new OCTBS scheme can find motion vector in motion field with fewer search points than the DS algorithm. Generally speaking, the larger the motion vector, the more search points the OCTBS algorithm saves, which is justified by the following experimental results.

3. Experimental Results

The experimental set-up is as follows: the distortion measurement of mean square error (MSE) is used. Block size is of 16\( \times \)16, and search window size of \(\pm 15\). Six standard video sequences are used, which varies in motion content as well as frame size. The six video sequences are “Claire”, “Dancewolf”, “Football”, “Salesman”, “Suzie” & “Tennis”.

Average MSE values and the search point numbers are summarized in Table 1 and Table 2 for different algorithms including FS, NTSS, 4SS, DS and our proposed OCTBS, respectively. It can see that the proposed OCTBS algorithm consumes the smallest number of search points with slightly increase in MSE compared with other fast algorithms. The number of search points used by the OCTBS method is substantially smaller than that by NTSS, 4SS or DS, nearly half the number of NTSS. Here we mainly...
compare the DS with the proposed OCTBS algorithm in terms of number of search point as well as MSE. Table 3 tabulates the speed improvement rate (SIR) and average MSE increase in percentage of the proposed OCTBS over DS. For “Claire” sequences with motion vectors limited within a small region around (0, 0), the proposed OCTBS algorithm achieves close to 2.5% speed improvement over DS. For "Football" and "Dancewolf" which contain large motion, the OCTBS algorithm has obtained higher speed improvement over DS, of nearly 30%. The larger the motion in a video sequence, the larger the speed improvement rate of OCTBS over DS or the other fast algorithms can be achieved. Fig. 4 plot a frame-by-frame comparison numbers of search point per block for the different algorithms applied to "Dancewolf" sequence.

4. Conclusion
The development of a novel fast algorithm using octagon-based search pattern in block motion estimation, help to demonstrates significant speedup gain over the diamond-based search and other fast search methods. We would like to stress that our proposed method would not imply the optimization in those video sequences with small motion vector, as you can see the MSE and SIR for different sequences from the Table 3. Generally speaking, the large the motion, the more efficient of our proposed method can be achieved. Our contribution is verified by the experimental result.

Reference
Table 1: Average MSE per pixel for different methods and different video sequence.

<table>
<thead>
<tr>
<th></th>
<th>Claire</th>
<th>Dancewolf</th>
<th>Football</th>
<th>Salesman</th>
<th>Suzie</th>
<th>Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>4.744</td>
<td>51.005</td>
<td>139.264</td>
<td>17.529</td>
<td>18.125</td>
<td>117.117</td>
</tr>
<tr>
<td>N3SS</td>
<td>4.889</td>
<td>55.271</td>
<td>192.822</td>
<td>18.407</td>
<td>21.687</td>
<td>152.673</td>
</tr>
<tr>
<td>4SS</td>
<td>4.938</td>
<td>56.223</td>
<td>193.535</td>
<td>18.969</td>
<td>19.888</td>
<td>146.155</td>
</tr>
<tr>
<td>DS</td>
<td>4.769</td>
<td>57.527</td>
<td>210.618</td>
<td>17.814</td>
<td>18.489</td>
<td>143.795</td>
</tr>
<tr>
<td>OCTBS</td>
<td>5.182</td>
<td>59.819</td>
<td>216.732</td>
<td>18.201</td>
<td>20.680</td>
<td>146.344</td>
</tr>
</tbody>
</table>

Table 2: Average number of search points per block with respect to different methods and different video sequence.

<table>
<thead>
<tr>
<th></th>
<th>Claire</th>
<th>Dancewolf</th>
<th>Football</th>
<th>Salesman</th>
<th>Suzie</th>
<th>Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>860.444</td>
<td>910</td>
<td>916.067</td>
<td>860.444</td>
<td>782.212</td>
<td>916.067</td>
</tr>
<tr>
<td>4SS</td>
<td>24.607</td>
<td>30.009</td>
<td>29.797</td>
<td>24.873</td>
<td>25.737</td>
<td>28.516</td>
</tr>
<tr>
<td>OCTBS</td>
<td>12.031</td>
<td>17.159</td>
<td>17.357</td>
<td>12.204</td>
<td>11.673</td>
<td>14.968</td>
</tr>
</tbody>
</table>

Table 3: Speed improvement rate and average MSE increase in percentage of OCTBS over DS.

<table>
<thead>
<tr>
<th></th>
<th>Claire</th>
<th>Dancewolf</th>
<th>Football</th>
<th>Salesman</th>
<th>Suzie</th>
<th>Tennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIR(%)</td>
<td>2.49</td>
<td>30.82</td>
<td>28.05</td>
<td>3.15</td>
<td>8.16</td>
<td>15.29</td>
</tr>
<tr>
<td>MSE increase(%)</td>
<td>8.66</td>
<td>4.0</td>
<td>2.90</td>
<td>2.17</td>
<td>11.85</td>
<td>1.77</td>
</tr>
</tbody>
</table>
Fig. 4 Search points in Dance Wolf sequence