
Cheong Seong Chee, Asral Bahari Jambek, Razaidi Hussin
Universiti Malaysia Perlis
calvin_cheong2003@yahoo.com, asral@unimap.edu.my, shidee@unimap.edu.my

Abstract—This paper reviews the energy efficiency of several popular block-matching motion estimation algorithms that can be used in wireless video sensor network applications. Full search motion estimation provides the best image quality but requires high computing power. Therefore, only fast search algorithms are considered for deployment in wireless video sensor networks, which generally operate in remote, battery-constrained areas. However, the image quality of fast search algorithms also needs to be considered in the comparison. In this paper, those block-matching algorithms are compared using two criteria: 1) computing cost/energy consumption, and 2) image quality. The purpose of this review is to select the most energy efficient algorithm that retains image quality and can be applied to wireless video sensor network video compression applications.

Keywords—block matching, motion estimation, wireless video sensor networks.

I. INTRODUCTION

Wireless video sensor networks are used to monitor geographical areas in battery-constrained conditions. In order to reduce the power consumption in bit transmissions/receptions over wireless video sensor networks, the video captured must be encoded or compressed before it is transmitted to the base station [1]. Motion estimation is an important part of a standard video coder [2] and is the process of determining motion vectors that describe the transformation of one 2D image to another. Motion estimation uses the most power and is the most computationally intensive component of video compression systems [3]. It consumes 70% of the computational capability [6]. Since a motion estimation engine is the most costly computing block in video compression design, selecting an energy efficient block-matching motion estimation algorithm is a highly important part of designing a low power, video compression algorithm that can be used in wireless video sensor networks.

In this paper, we review several popular block-matching motion estimation algorithms in terms of energy efficiency and image quality. Peak signal-to-noise ratio (PSNR) is used to measure the image quality. Although PSNR may not be the best measure of image quality for different types of motion estimation algorithms, it is still acceptable as an indicator of quality [2]. The number of searching points that are used for calculating the best match motion vector of the current block are used as a benchmark to compare operational power consumption.

II. BLOCK-MATCHING MOTION ESTIMATION ALGORITHMS

Until recently, block-matching motion estimation has been an effective method in reducing temporal redundancy in video coding and, therefore, it is currently included in many video coding standards [4-5]. Numerous motion estimation algorithms have been developed for various usages. In this paper, we focus on a few select algorithms. These algorithms are selected based on their market popularity. Additionally, a few new proposed power saving algorithms were selected for comparison. The remainder of this section will provide detailed explanations of these algorithms.

The Sum of Absolute (SAD) is widely used because it is an extremely simple algorithm that is used to measure the similarity between image blocks that only involve summation and subtraction. Using the absolute difference, a comparison of each pixel’s luminance is made between the current frame and the reference frame. Assuming that the size of a macroblock is N x N, the SAD for the candidate macroblock at position (u, v) is defined as follows:

$$SAD(u, v) = \sum_{i=1}^{N} \sum_{j=1}^{N} r(i, j) - r(i+u, j+v)$$ (1)

where $r(i, j)$ and $s(i+u, j+v)$ are luminance values at position $(i, j)$ of the reference frame macroblock and at position $(i+u, j+v)$ in the current frame candidate macroblock $(u, v)$ in the search area, respectively.

The best match motion vector of $u$ and $v$ with minimum SAD $(u, v)$ is obtained by using the following equation:

$$SAD(u_{\text{mp}}, v_{\text{mp}}) = SAD(u, v)_{\text{min}} \leq SAD(u, v)$$ (2)

A. Full Search (FS) [2,8]

The full search algorithm is the direct brute-force motion estimation algorithm that evaluates all of the possible displacements inside a search area in order to obtain the best...
match [2]. For one macroblock, there will be 16x16 = 256 pels that need to undergo SAD computation. This means that FS can be accurately compared to other block-matching motion estimation algorithms. The algorithm is illustrated in Figure 1. Even though this particular algorithm requires a significant amount of computing power, it is useful as a benchmark to perform comparison between other motion estimation algorithms. The search area range is (2p + 1)² pels, where p is the search range defined by the user. The SADs for (2p + 1) candidate macroblocks in each horizontal and vertical direction, for a total of (2p + 1)² candidate macroblocks for each reference macroblock, are compared to determine the match that has the minimum SAD value.

B. Diamond Search (DS) [8]

Diamond search involves the usage of two diamond shapes for motion vector searching. This method includes a large diamond with nine search points and a small diamond with five search points. This algorithm initially searches for the minimum SAD location by using the large diamond shape. If the minimum SAD location is not located at the center of the diamond, the center point is then replaced by the new minimum SAD location that is found. The searching continues by using a large diamond until the minimum SAD location is located at the center of the diamond. After that, it switches to a small diamond shape to perform a refinement process. If the minimum SAD point is located at the four searching points at the surrounding of the small diamond center point, then the minimum SAD point will be the new center point. This process is repeated until the minimum SAD is located at the center point. Figure 2 shows the DS algorithm. The total number of search points is as follows:

\[ 9 + k \cdot (5 \times N_1 + 3 \times N_2) + 4 \]  

(3)

Where k is 1 if involved in the second step, N₁ is the number of steps with 5 new search points, and N₂ is the number of steps with 3 new search points.

C. Three-Step Search (TSS) [2,8]

Three-Step Search assumes that the residue values increase radically from the absolute minimum point within the search area. This algorithm searches for the direction of the minimum SAD value and, from there, continues to find the best match minimum SAD point. Initially, TSS compares nine candidate macroblocks surrounding a center point, with step size, p, equal to or larger than half of the maximum search range, r. When the nine search point SAD is evaluated, a minimum SAD is selected and becomes the new center point of the next search. In the next step, the step size is halved and 8 new search points surrounding the new center point are evaluated and a minimum SAD is selected. In the third step, the step size is halved again and becomes equal to one. The minimum SAD best match point is found at this stage. Figure 3 shows the TSS algorithm. The total number of search points is as follows: 

\[ 9 + k \cdot (8 \times N_1) + 8 \]  

(4)

Where k is 1 if involved in the second step, N₁ is the number of steps with 8 new search points.

Figure 1. Current macroblock and the respective search range on the video frame.

Figure 2. Diamond Search algorithm.

Figure 3. Three Step Search algorithm.
D. Four-Steps Search (FSS) [8,11]

In the first step, the four-step search algorithm searches by using a square search pattern with nine checking points on a 5 x 5 window, instead of a 9 x 9 window that is used TSS. This window is located at the center point (0, 0). The center point of the search window will switch to the minimum SAD point found at the searching points around the center point until the minimum SAD point is located at the center point. Then, it will perform the refinement step by using a 3 x 3 search window. The minimum SAD point obtained in this step is the best match that provides a motion vector value. The algorithm of FSS is explained in detail in Figure 4. The total number of search points for FSS is as follows:

$$9 + k.( 3 \times N_1 + 5 \times N_2 ) + 8$$  \hspace{1cm} (5)

Where $k$ is 1 if involved in the second step, $N_1$ is the number of steps with 3 new search points, and $N_2$ is the number of steps with 5 new search points.

E. Hexagon-Based Search (HEXBS) & Hexagon Diamond Search (HDS) [7,16]

HEXBS/HDS is accomplished by using a hexagonal search-point configuration. Initially, the minimum SAD point searching begins by using a large hexagonal pattern with seven checking points. It will continue to use the minimum SAD point as the new center until the minimum SAD is located at the center of the hexagon. Once the minimum SAD is found at the center, HEXBS will switch to using the shrunken hexagonal pattern, which includes four checking points for the refinement search. Figure 5 explains the algorithm in detail. The total number of search points for HEXBS is as follows:

$$7 + k_.( 3 \times N_1 ) + 4$$  \hspace{1cm} (6)

Where $k$ is 1 if involved in the second step, $N_1$ is the number of steps with 3 new search points.

F. Improved Four-Step Search (IFSS) [14]

IFSS is the modified version of FSS. In this version, the refinement step of FSS is changed to the diamond search pattern. The conventional FSS with four step searching is modified to five step searching. Using this method can effectively reduce the search points and achieve fast searches in both large and small motion content. Step 1 to step 3 of this algorithm is exactly the same as FSS. However, at step 4, the minimum SAD point obtained from step 3 is used as the center point of a five search point small diamond search pattern to continue searching for the minimum SAD. Next, the minimum SAD from step 4 is used as the center point and the upper and lower point is checked for best match of the minimum SAD. Figure 6 shows the algorithm of IFSS. The whole process of this algorithm involves the total number of search points as follows:

$$9 + k_1.( 5 \times N_1 + 3 \times N_2 ) + 4 + 2.k_2$$  \hspace{1cm} (7)

Where $k_1$ is 1 if involved in the second step, $k_2$ is 1 if involved last refinement step, $N_1$ is the number of steps with 3 new search points, $N_2$ is the number of steps with 5 new search points.
G. Modified Diamond-Square Search (MDSS) [13]

The MDSS is the modified version of the Diamond Search (DS). This method reduces the large diamond shape search points from nine to five and modifies the small diamond shape, which is used during the refinement step, into a square shape. The authors in [13] claim that this will help reduce the searching points during the recursive search process until the best match is obtained. Basically, the searching algorithm is exactly the same as the conventional Diamond Search algorithm. The first step uses a five search point diamond search pattern to search for the minimum BDM. Next, the minimum BDM point becomes the new center point for the diamond search pattern. Then the refinement step is performed by using a square search pattern where the center point is using the minimum BDM point that was obtained in the previous step. The MDSS algorithm is explained in Figure 7. The total search point number of MDSS is as follows:

\[ 5 + k(3 \times N_1) + 4 \quad (8) \]

Where \( k \) is 1 if involved in the second step, \( N_1 \) is the number of steps with 3 new search points.

![Figure 7. MDSS algorithm.](image)

H. Hardware-Oriented, Modified diamond Search (HMDS) [12]

HMDS is designed to compensate for the decreased adaptability and search efficiency. The purpose of HMDS is to track large motion problems that occur as a fixed set of search patterns algorithms, such as TSS, NTSS, FSS, DS and HEXBS. These algorithms assume the error surface increases monotonically as the search position moves away from the global minimum. After studying various basic shape search patterns, the authors in [12] claim that the diamond shape is extremely efficient at finding true motion vectors. HMDS modified the diamond search pattern by enlarging the search range to the outermost of the search range. HMDS involves a square shape with nine search points at the center and a diamond shape with eight search points located at the outer portion of the square. This algorithm uses the fixed 3 step search to determine the minimum distortion block. Initially, the minimum block distortion measure is found from the search points by using the modified diamond search pattern. Then, the minimum distortion block will be used as the center of a large diamond search pattern (LDSP) to continue to identify the new minimum BDM point. Similar to the second step, the last step uses the minimum BDM, which was obtained from the second step as the center point, to search for the minimum BDM. The searching process terminates after the third step and the minimum BDM obtained is the best match. This is unlikely compared to the conventional diamond search algorithm, which searches recursively until it determines the minimum distortion block located at the center of the diamond before performing the refinement step. This helps to save on computation costs. In the worst case scenario, the total search point number of HMDS is thirty-seven. Figure 8 illustrates the algorithm.

![Figure 8. HMDS algorithm.](image)

I. Enhanced modified orthogonal search (EMOS) [15]

Orthogonal Search (OS) was first introduced in 2005 by [17] and it is a hybrid of the TSS and the Two Dimensional Logarithmic Search. It recursively employs the vertical stage and the horizontal stage to obtain the minimum SAD point. In 2010, a Modified Orthogonal Search Algorithm (MOS) was introduced by [18]. The only change in this algorithm was the addition of a nine search point square search pattern at the first searching step. This overcomes the weakness of the novel Orthogonal Search, which was inefficiently used to estimate small motions. Later, the Enhanced Modified Orthogonal Search (EMOS) algorithm was introduced and claimed to be better than MOS. This new algorithm modified the first search step nine search point square search pattern of the existing MOS to a five search point diamond search pattern. The EMOS algorithm involved six search steps. The first step uses a five search point diamond pattern at the center of the search window and an extra two search points at a distance of step...
size in the horizontal direction from the center of the search window. The minimum SAD point is searched among these points. During the second step, the minimum SAD point from the first step is used as the new center point. Two search points at a distance of step size from the new center point in the vertical direction are examined to determine the minimum SAD point. Then, again the new minimum SAD point is used as a new center point to search for two search points at a distance of half a step size in the horizontal direction. This searching process is repeated alternatively in the horizontal and vertical direction until the step size is equal to one. EMOS algorithm is illustrated in Figure 9. The total number of search points for this algorithm is as follows:

$$5 + k(2 \times N_1)$$ \hspace{1cm} (9)

Where k is 1 if involved in the second step, N_1 is the number of steps with 2 new search points.

![Figure 9. EMOS algorithm.](image)

### III. COMPUTATIONAL COMPLEXITY AND PERFORMANCE ANALYSIS

In this section, we will analyze the motion estimation efficiency and power consumption as reported in the following. The analysis is done by comparing the number of searching points needed to go through each algorithm before they are able to determine the best match with a minimum SAD. This is because the number of computational operations required to find the best match is directly proportional to the power consumption. The data for all algorithms was obtained from reference papers. From Table I, we can see that FS requires the largest computation bandwidth in order to get the best match motion vector. The computation bandwidth consumption decreases as follows: HMDS, TSS, FSS, IFSS, DS, HEXBS/HDS, MDSS, and finally, the minimum computation bandwidth consumption is the EMOS algorithm. By comparing this data, we see that the typical EMOS algorithm is the most cost-saving algorithm, in terms of computational resources.

However, we cannot solely judge the power consumption based on the theoretical calculated total number of search points for each algorithm. This is because some of the algorithms have a larger step size, p, and are able to obtain the best match motion vector in a fewer number of search points. Thus, data from Table II, which is taken from [15, 16], is very important. The data is collected by simulating all the algorithms by using several types of video sequences with CIF 30 fps. The experimental data shows that the HEXBS/HDS and EMOS are able to get the best match motion vector by using the least number of search points. From the data comparison, we can see that EMOS and HEXBS/HDS both have better power saving potential compared to others. The experimental data proved that the HEXBS/HDS and the EMOS are better than the DS algorithm in term of intelligent searching algorithm, which lowered the number of search points required. From Table IV [13], we can see that the average search point needed by MDSS is better than DS, while the PSNR is slightly less than the DS. This means that the MDSS saves on computational costs without sacrificing the quality of video. Both the IFSS and the HMDS enhanced the search range to minimize the global minimum error [12, 14]. However, through this enhancement, the number of search points required is also reduced. This helps to save on computation costs as well.

Although the HEXBS/HDS algorithms are better in terms of power saving, the image quality is not as good when compared to other algorithms. Peak signal-to-noise ratio (PSNR) is used to measure the quality of reconstruction of lossy compression codecs. The higher PSNR value indicates that the reconstruction quality is higher. From Table III, we can see that the PSNR value of the HEXBS/HDS algorithm is lower than other algorithms. This means that the HEXBS/HDS algorithm is close to other algorithms in terms of image quality. The data in Table III is taken from [16]. The FS algorithm has the ideal image quality because it considers every possibility by searching all the pixels that are available in the search area. By comparing the ideal case, TSS and FSS have an image quality that is similar to the FS algorithm. This is because TSS and FSS have smaller searching steps size when compared to the HEXBS/HDS. However, for wireless video sensor network applications, the main concern is lowering power consumption while maintaining acceptable image quality. In this case, the HEXBS/HDS, MDSS and EMOS are still the best choice to use in wireless video sensor networks.

### TABLE I

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>No. of search points</th>
<th>Typical</th>
</tr>
</thead>
<tbody>
<tr>
<td>FS</td>
<td>(2N + 1)^2</td>
<td>255</td>
</tr>
<tr>
<td>TSS</td>
<td>9 + k(8 x N_1) + 8</td>
<td>25</td>
</tr>
<tr>
<td>DS</td>
<td>9 + k(5 x N_1 + 3 x N_2) + 4</td>
<td>21</td>
</tr>
<tr>
<td>FSS</td>
<td>9 + k(5 x N_1 + 3 x N_2) + 8</td>
<td>25</td>
</tr>
<tr>
<td>HEXBS</td>
<td>7 + k(3 x N_1) + 4</td>
<td>14</td>
</tr>
<tr>
<td>MDSS</td>
<td>Maximum 37</td>
<td>37</td>
</tr>
<tr>
<td>IFSS</td>
<td>9 + k(5 x N_1 + 3 x N_2) + 4 + 2k_2</td>
<td>23</td>
</tr>
<tr>
<td>EMOS</td>
<td>5 + k(2 x N_1)</td>
<td>7</td>
</tr>
</tbody>
</table>
IV. CONCLUSION

From the data comparison, it is easy to see that the HEXBS/HDS, MDSS and EMOS algorithms are able to perform low computation cost motion estimation. This is due to the significant reduction in the number of searching points required in order to obtain the best match motion vector. However, when compared to other algorithms, these algorithms are unable to provide a high quality image. Although the image quality is not that good, it is still within an acceptable range. For battery-constrained wireless video sensor network applications, power saving is the first priority. Thus, the HEXBS/HDS, MDSS and EMOS are still the best choices. According to this review, there are a few trade-offs that need to be considered before building up a motion estimation algorithm. By using a larger number of search points, the coverage is larger and the image quality is better but the computation costs are higher. So, future work needs to develop a new motion estimation algorithm for wireless video sensor networks. This key point must be balanced in order to achieve the desired low power consumption and acceptable image quality. Hopefully, future experiments and simulations will address this need.

REFERENCES
