Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2015, Article ID 715651, 13 pages
http://dx.doi.org/10.1155/2015/715651

Research Article

Content-Aware Fast Motion Estimation Algorithm for IoT Based Multimedia Service

Ryong-Baek, Kyung-Soon Jang, Jae-Hyun Nam, and Byung-Gyu Kim

Department of Computer Engineering, SunMoon University, Asan-si, Republic of Korea

Correspondence should be addressed to Byung-Gyu Kim; bg.kim@mpcl.sunmoon.ac.kr

Received 1 May 2015; Accepted 15 June 2015

Academic Editor: Qun Jin

Copyright © 2015 Ryong-Baek et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Various applications based on IoT real-time multimedia are under the spotlight. To implement real-time multimedia service, motion estimation in video compression service has a high computational complexity. In this paper, an efficient motion search method based on content awareness is proposed consisting of three steps. The first step is motion classification using the center position cost distribution. The second step is calculation of a predictor based motion classification. The third step is setting the arm size of the search pattern based on adaptive use of the distance between the predictor and the center position. Experimental results show that the proposed algorithm achieves speed-up factors of up to 48.57% and 16.03%, on average, with good bitrate performance, compared with fast integer-pel and fractional-pel motion estimation for H.264/AVC (UMHexagonS), and an enhanced predictive zonal search for single and multiple frame motion estimation (EPZS) methods using JM 18.5, respectively. In addition, the proposed algorithm achieves a speed-up factor of up to 42.61%, on average, with negligible bitrate degradation, compared with the TZ search motion estimation algorithm for the multiview video coding (TZS) method on HM 10.0.

1. Introduction

The rapid development of Internet of Things (IoT) technology makes it possible for connecting various smart objects together through the Internet and providing more data interoperability methods for application purpose. Recent research shows more potential applications of IoT in information intensive industrial sectors such as healthcare services [1]. Also, with the vigorous development of the Internet of Things (IoT) technology, the wave of the Internet of Things applications followed by applications based on IoT real-time multimedia is under the spotlight [2].

Most viewers prefer digital television delivered via terrestrial, cable, satellite, or the Internet over analog service due to a greater choice of channels, electronic program guides, and high definition services. Analog TV has been switched off in many countries. Many TV programs can be provided via the Internet. However, due to the huge size of data for a video signal, in practical scenarios it is not possible to transmit raw data and video compression is required. To generalize the video data compression technique for use with different kinds of media, a global standard for video compression has been developed by the ISO/IEC motion picture experts group (MPEG) and the ITU-T video coding experts group (VCEG).

The ITU-T video coding expert group and the ISO/IEC moving picture experts group together proposed the H.264/AVC video coding standard in 2003 [3]. This standard is widely used for many applications, including broadcast of HD TV signals, video content acquisition, camcorders, and security applications. However, due to increasing demands for ultrahigh definition over and above HD video formats VCEG and MPEG established a Joint Collaborative Team on Video Coding (JCT-VC) January 2010 to implement the next-generation video compression standard, named high efficiency video coding (HEVC) [4]. The aim of this new standard is to further reduce the bitrate by 50% with the same reconstructed video quality. With these standards, Liu and Yan have introduced the multiview video technology, explained the principle and coding structure of multiview video, and discussed two kinds of video monitoring applications mode in Internet of Things, focusing on the application of the intelligent security systems in the IoT environment [5].

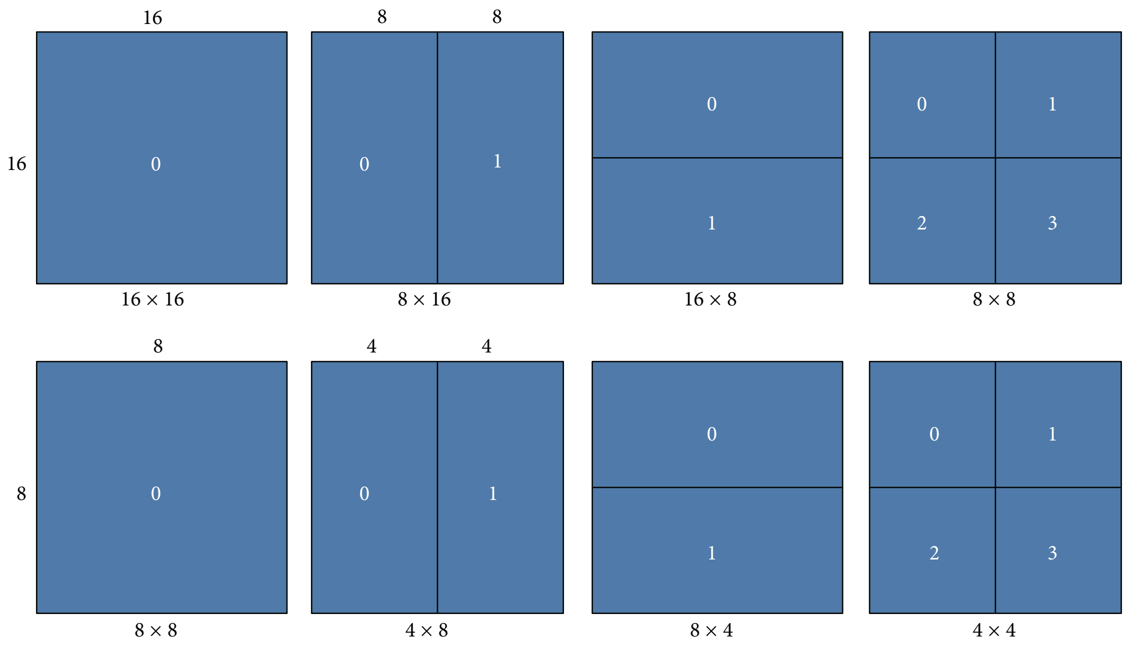


Figure 1: Different macroblock partitions.

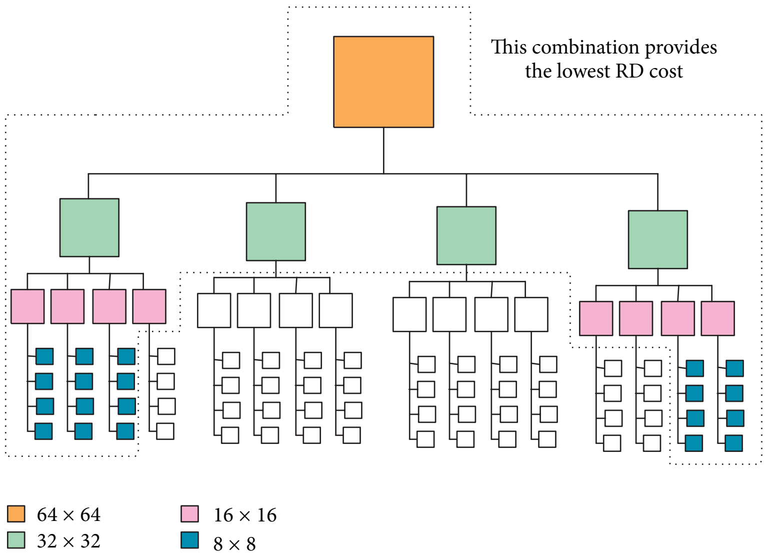
Both the H.264/AVC and HEVC video coding standards use both temporal and spatial redundancy to achieve compression. In the temporal domain, there is usually a high degree of correlation or similarity between video frames that were captured at nearly the same time. Temporally adjacent frames (successive frames in time order) are often highly correlated, especially if the temporal sampling rate or frame rate is high. In the spatial domain, there is usually a high degree of correlation between pixels that are close to each other as values of neighboring pixel are often similar.

In general, a video system comprises the two major modules of intraframe coding and interframe coding. The block based motion estimation (BME) algorithm plays a key role in interframe coding. In the H.264/AVC video coding standard, BME has 7 different block sizes that are used for interframe motion estimation and compensation [6]. These different block sizes actually form a two-level hierarchy inside a macroblock (MB). The first level comprises block sizes of 16×16, 16×8, and 8×16 that is called a large MB type. In the second level, the MB is specified as a P8×8 type block size where each 8×8 block can be one of the subtypes 8×8, 8×4, 4×8, or 4×4. These are referred to as the submacroblock type. The relationship between these different block sizes is shown in Figure 1.

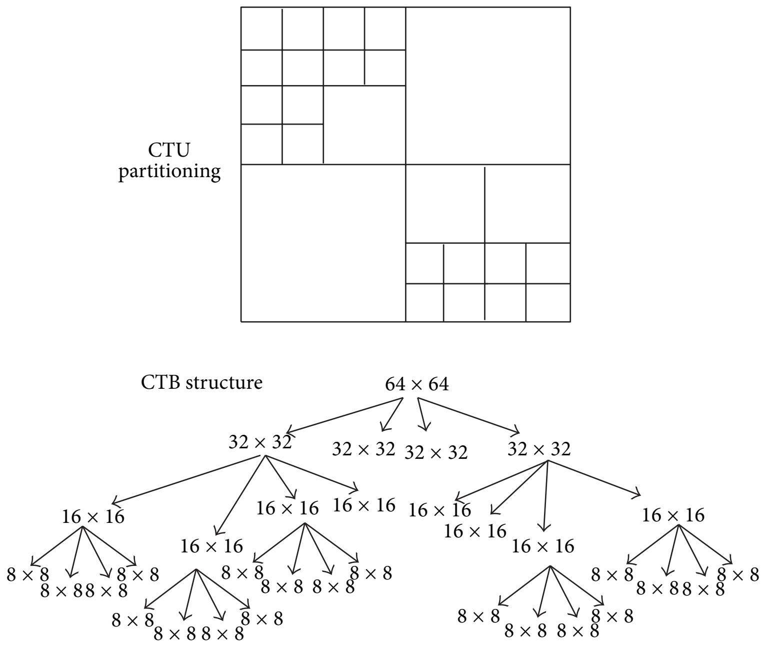
For HEVC, sizes of coding blocks are not fixed, as are macroblocks in H.264/AVC. A flexible coding block (CB) and a novel quadtree based coding tree unit (CTU) have been proposed for HEVC. Prediction of each block is performed in the prediction unit (PU). Illustration of the CTB structure for a CTU is shown in Figure 2(a) where 64×64 pixel CTU block is divided into smaller CU blocks. Upon calculation of the RD cost for every combination, the CUs that are under the red dotted area of Figure 2(a) give minimum RD values. The corresponding CTU partitioning and CTB structure for this particular (best) combination are shown in Figure 2(b). However, the interprediction technique is similar to H.264/AVC interprediction, for which the most computational time is needed for motion estimation, which is common for both standards.

Among traditional motion estimation (ME) algorithms, a full search (FS) algorithm requires much time due to evaluation of all candidate macroblocks to achieve optimal performance. However, the FS algorithm is not suitable in real-time video coding applications because high degree of computational complexity is involved.

To overcome this problem in use of the FS algorithm, variable fast block-matching algorithms (FBMA) have been proposed to increase the speed of the motion estimation process based on reduction of the number of search points using three different approaches. The first approach uses coarse-to-fine searching as a three-step search algorithm (TSS) [7], a 2D logarithmic search algorithm [8], and a four-step search algorithm (4SS) [9]. The second approach uses center-biased characteristics of the error distortion as a block based gradient descent search algorithm (BBGDS) [10] and a diamond search algorithm (DS) [11]. BBGDS is difficult to apply fast motion and simply falls to a local minimum position. DS is efficient for use in midlevel motion. However, many search points are required in slow and fast motion. The third approach uses temporal and/or spatial correlation for calculation of a predicted motion vector (MV). An adaptive rood pattern search (ARPS) and an enhanced predictive zonal search for single and multiple frame motion estimation (EPZS) are also used. These algorithms use a set pattern size



(a)



(b)

Figure 2: (a) CTB structure that provides the lowest RD cost for the CTU; (b) the corresponding CTU partitioning for the best CTB structure [9].

or start position from a previous frame and/or a neighboring current block MV. The EPZS and ARPS approaches also preserve the peak signal-to-noise ratio (PSNR), as does FS, while reducing the time required and maintaining the bitrate.

An adaptive FBMA has also been proposed based on pattern switching among TSS, DS, and BBGDS. However, some proposed algorithms only consider a fixed size MB [12]. In [13], a clustering-based approach has been proposed in which an algorithm periodically counts the motion vectors of past blocks to accumulate progressive clustering statistics; then it uses the clusters as MV predictors for following blocks.

To reduce the computational complexity of the motion estimation module, an efficient algorithm is herein proposed for classification of video motion content using an adaptive size for a search pattern. The proposed algorithm reduces computational complexity by use of a predictor.

The rest of the paper is organized as follows: Section 2 presents related motion estimation algorithms. The proposed algorithm is described in Section 3. Experimental results are presented in Section 4, and conclusions are drawn in Section 5.

2. Related Research

Block based encoding structure is used in the H.264/AVC and HEVC video standard. For interprediction, a motion estimation technique is at the core of video compression and for different video processing applications that extract motion information from a video sequence. Typically using motion estimation, a motion vector is generated for a block (MB or CU) in the video compression standard. The motion vector indicates displacement of a block of pixels from a current location due to motion of an object or the camera. This information is used to determine the best matching block in the reference frame that minimizes the rate-distortion cost. This technique is known as the block-matching algorithm (BMA). Based on study of different motion estimation algorithms used in the H.264/AVC and HEVC standards, existing BMAs can be classified into the three categories of (1) fixed search patterns, (2) search patterns based on block correlation, and (3) search patterns based on motion classification.

2.1. Fixed Search Patterns. Most methods are based on the assumption that the ME matching error decreases monotonically as search moves toward the position of the global minimum error. The motion vector of each block is searched independently using fixed search patterns. Examples are displacement measurement and applications in interframe image coding (2-LOGS), motion compensated interframe coding for video conferencing (TSS), a novel four-step search algorithm for fast block motion estimation (4SS), a block based gradient descent search algorithm for block motion estimation in video coding (BBGDS), a hexagon-based search pattern for fast block motion estimation (HEXBS) [14], a new diamond search algorithm for fast block-matching motion estimation (DS), and fast integer-pel and fractional-pel motion estimation for H.264/AVC (UMHexagonS) [15]. These algorithms reduce the number of search points with a tradeoff between complexity reduction and image quality.

Both 4SS and TSS are efficient for fast motion video sequences because MVs in fast motion sequences are far removed from the center point of a macroblock. However, in other cases, such as medium and slow motion sequences, the MV can be trapped in a local minimum. TSS also uses a constantly allocated checking point pattern in the first step, which becomes inefficient for estimation of slow motion. A new three-step search for block motion estimation (NTSS) [16], an efficient three-step search algorithm for block motion estimation (ETSS) [17], and a simple and efficient search algorithm for block-matching motion estimation (SES) have been proposed in order to improve the performance of simple fixed search pattern algorithms [18].

2.2. Search Patterns Based on Block Correlation. Instead of using predetermined search patterns, methods for exploitation of the correlation between the current block and an adjacent block in the spatial and/or temporal domains for prediction of candidate MVs are being considered. Predicted MVs are obtained based on calculation of statistical average (such as a mean, a median, or weighted mean/median) for neighboring MVs or selection of one of the neighboring MVs according to predetermined criteria [19]. One such candidate method, named accelerator MVs, is based on differentially increased or decreased MVs after consideration of the motion vector of the collocated frame in the previous frame and also of the frame before that.

The concept behind selection of such predictor is that a block may not follow a constant velocity but may be accelerated. This kind of approach uses spatial or/and temporal correlation to calculate the predictor as the ARPS and EPZS. These types of algorithms set pattern sizes or estimate positions from a previous frame or/and a neighboring current block MVs. Both EPZS and ARPS preserve the peak signal-to-noise ratio (PSNR) as does FS, and the time required is reduced with a similar bitrate. However, these procedures create much overhead in memory resources since they use Spatio-temproal information.

2.3. Search Patterns Based on Motion Classification. Apart from fixed or variable search patterns, the motion activity of a video sequence has been used for a block-matching algorithm. Video sequences can be broadly divided into the three categories of frame slow, medium, and fast sequences based on motion activity in successive frames. Algorithms use different schemes to classify video sequences.

A proposed search patterns switching algorithm for block motion estimation (SPS) [20] has combined two approaches of motion estimation. The first approach uses a coarse-to-fine technique for reduction of the number of search points, similar to 2-DLG and TSS. This approach is efficient for fast motion video sequences because, in these sequences, search points are evenly distributed over the search window and, thus, global minima far distant from window centers can be located more efficiently. The second approach uses the center-biased characteristic of MVs. Algorithms such as N3SS, 4SS, BBGDS, and DS use center-biased search patterns to use the center-biased global minima distribution. Compared with the first approach, a substantial reduction in the number of search points can be achieved for slow motion sequences. SPS algorithms combine the advantages of these two approaches by use of different search patterns according to the motion content of a block. The performance of an adaptive algorithm depends on the accuracy of motion content classification.

In real video sequences with slow, medium, and fast motion, different motions frequently occur together. The adaptive fast block-matching algorithm using switching search patterns for sequences with a wide-range of motion (A-TDB) can efficiently remove temporal redundancy of sequences containing a wide range of motion. Based on characteristics of a predicted profit list, A-TDB can adaptively switch search patterns among TSS, DS, and BBGDS according to the motion content [21].

An adaptive motion estimation scheme for video coding (NUMHexagonS) using statistics of the MV distribution has been developed. The algorithm uses method for prediction of the MV distribution and makes full use of MV characteristics. A combined MV distribution prediction with new search patterns also makes the search position more accurate [22].

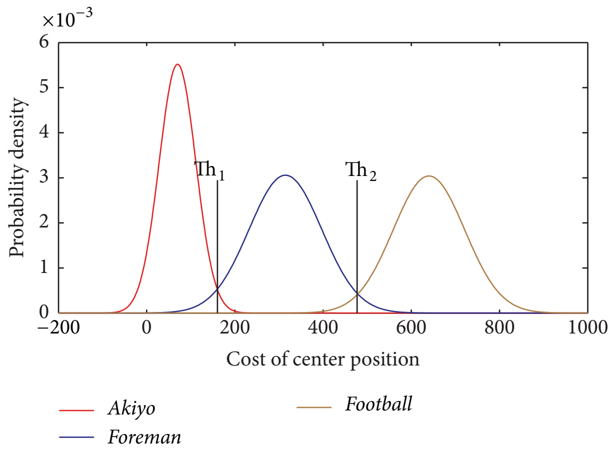


Figure 3: Intersection of *Akiyo*, *Foreman*, and *Soccer* sequence distributions.

3. Proposed Work

Most real world video sequences contain a substantial amount of background and many motionless objects that have a high temporal correlation between successive frames. In order to estimate movement accurately, different pattern sizes are used based on the type of motion content.

3.1. Motion Classification Based on Video Content. There are several measurements for classifying the motion degree of video content: RD cost, sum of absolute difference (SAD), and the mean of absolute difference (MAD). Usually, analysis of MAD value is robust to some noise components for separating the motion degree. Based on this, we also employ the MAD value to classify the motion degree.

We have analyzed the distribution of the mean absolute difference (MAD) of the center position for the *Akiyo*, *Foreman*, and *Football* sequences, typical representatives of slow, medium, and fast motion, respectively. Mean and standard deviation values were calculated and used to model Gaussian distribution as shown in Figure 3.

Only intermotion estimation was decomposed. Calculated MAD values are represented as

$$\mathrm{MAD} = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\left|\mathrm{Cur}_{i,j} - \mathrm{Ref}_{i,j}\right|, \quad (1)$$

where Cur_{ij} and Ref_{ij} are the macroblocks of the current and reference frames, respectively, with a size of N, and (i, j) are the spatial horizontal and vertical positions of the current and reference frames, respectively. Gaussian plots of slow, medium, and fast motion are shown in Figure 3. Video sequences are shown in which two intersection points of these distributions can be seen. These points of intersections are considered as thresholds Th_1 and Th_2. For motion classification, there should be two points of intersection as Threshold 1 = Th_1 and Threshold 2 = Th_2. The Th_1 value is a classification of the slow motion content between slow and medium motion, and Th_2 is classification of the fast motion content.

Table 1: Change in threshold points for different QP values.

QP	Th_1	Th_2
0	180	544.5
24	162	475.5
28	173	478
32	188	484.5
36	207	495

Analysis was performed for a particular QP value that varies from 20 to 36. Corresponding Th_1 and Th_2 values are calculated for each QP value. Different threshold points for changing QP values are shown in Table 1 for the combined *Akiyo*, *Foreman*, and *Football* distributions. The values of Th_1 and Th_2 change linearly with the QP value. Hence, it is natural to use a linear model for these thresholds for different QP values. The corresponding linear model is

$$\mathrm{Th}_1(\mathrm{QP}) = 0.125 * \mathrm{QP}^2 - 3.75 * \mathrm{QP} + 180,$$
$$\mathrm{Th}_2(\mathrm{QP}) = 0.125 * \mathrm{QP}^2 - 5.875 * \mathrm{QP} + 544.5. \quad (2)$$

To effectively locate the global minimum, predictor is designed to use the motion content. Predictor designs in many ME algorithms have been proposed and most use temporal and spatial information [23, 24]. While accurate predictors can be located, resource overhead and much memory are required. In the next section, calculation of the predictors to solve this problem is explained.

3.2. Calculation of Predictors. Based on a unimodal error surface assumption, block distortion monotonically decreases towards the global minimum. A predictor is calculated using three steps. First, distortion of the center point is compared to the threshold value. If the determined motion is slow, calculation of the predictor is not performed. Otherwise, distortions at the center and four adjacent center points are calculated. Among these five points, the one with the minimum distortion is recognized as the best point and not Min_1. Therefore, calculation of the distortion values of the five points is performed exclusively of Min_1 to locate the second minimum as Min_2. The Min_1 and Min_2 values have the direction of MVs.

From these Min_1 and Min_2 values a minimum of two positions is obtained. In the second step, if motion is classified as medium, the jumping distance is set to 2. If motion of the current block is classified as fast, then the jumping distance is set to 3. To justify the jumping distances of different sequences, analysis of the 3 representative video sequences *Akiyo*, *Foreman*, and *Football* was performed, and results are shown in Table 2 in which jumping distances of 2 and 3 were sufficient.

For the *Foreman* and *Football* sequences, two minima are calculated initially for a candidate point where the candidate point is shown as the center point and in black color in Figure 4, two separate loops are generated for each minima (Min_1 and Min_2) as shown in Figure 4.

From Min_1, in the vertical direction a new point_J is considered and the SAD value of this point is calculated. The

Table 2: Experimental results for determination of the jumping distance.

D	*Akiyo* PSNR (dB)	*Akiyo* Bit rate (Kbps)	*Akiyo* ME time (sec.)	*Foreman* PSNR (dB)	*Foreman* Bit Rte (Kbps)	*Foreman* ME time (sec.)	*Football* PSNR (dB)	*Football* Bit rate (Kbps)	*Football* ME time (sec.)
1	35.41	43.98	2.67	32.55	128.98	4.04	29.91	263.17	4.53
2	35.41	43.98	2.65	32.52	128.35	3.92	29.92	261.74	4.38
3	35.42	44.05	2.65	32.53	128.48	3.87	29.92	260.67	4.35
4	35.42	44.05	2.66	32.53	129.58	3.87	29.91	260.72	4.32
5	35.42	44.05	2.64	32.53	129.52	3.89	29.90	263.30	4.31
6	35.42	44.05	2.66	32.52	128.63	3.88	29.91	263.17	42.29
7	35.42	44.05	2.66	32.53	128.48	3.87	29.91	264.13	4.31
8	35.41	43.98	2.67	32.52	128.74	3.90	29.91	264.32	4.32
9	35.41	43.98	2.66	32.53	128.64	3.91	29.93	264.22	4.31
10	35.42	44.14	2.65	32.52	128.84	3.91	29.93	264.30	4.31

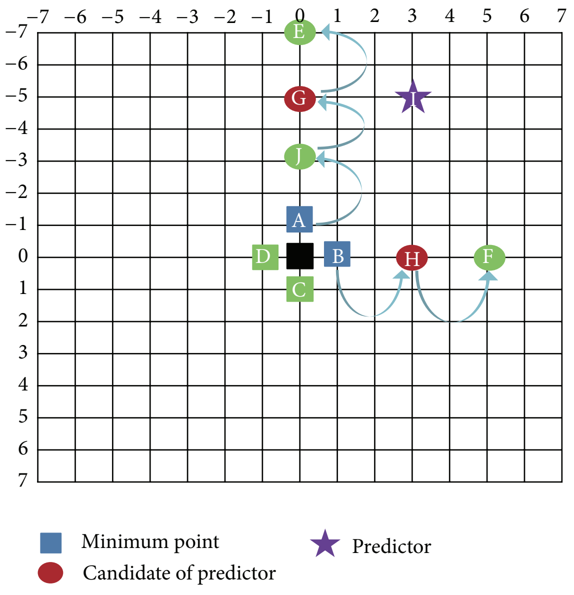


Figure 4: Candidates with consideration of directional prediction.

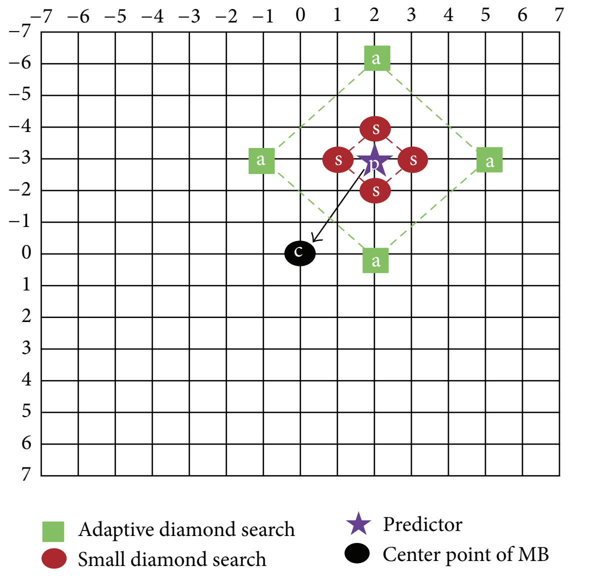


Figure 5: An adaptive diamond search pattern and a small diamond search pattern.

distance between point$_A$ and point$_J$ is the jumping distance of the sequence. If the SAD value of point$_J$ is smaller than for point$_A$, another point in the same direction is considered (point$_G$ in the example). This procedure will terminate when the SAD value of the new point is less than the SAD value of the previous point. For this example, point$_G$ has the minimum SAD value in this direction and is considered as the candidate point in the vertical direction. A similar procedure is also applied for point$_B$ (Figure 4). Considering point$_H$ as the candidate minimum point in the horizontal direction we identify two candidate points in two different directions. Using these two candidate points a third predictor point is calculated. The predictor point is shown as point$_I$ in Figure 4. The predictor point is calculated using (3). On the other hand, after calculation of the predictor, an adaptive diamond search is performed using the calculated predictor as the center point of the diamond. The size of the adaptive diamond search is given in (4) as

$$\text{predictor} = \{\text{point}_H(x) + \text{pint}_G(x), \text{point}_H(y) + \text{point}_G(y)\}, \quad (3)$$

$$\text{size} = \text{Max}(|\text{predictor}(x)|, |\text{predictor}(y)|). \quad (4)$$

3.3. Selection of a Search Pattern Size. A pattern size decision algorithm is introduced (Figure 5). Initially, the maximum distance from the predictor to the center is calculated, which is 3 in Figure 5. Using this maximum distance, a square region is considered with 4 points (Figure 5, marked as point$_a$).

SAD values of these 4 new points are calculated and centers are checked. If the minimum SAD value is the center of the diamond, then the large search pattern size decision

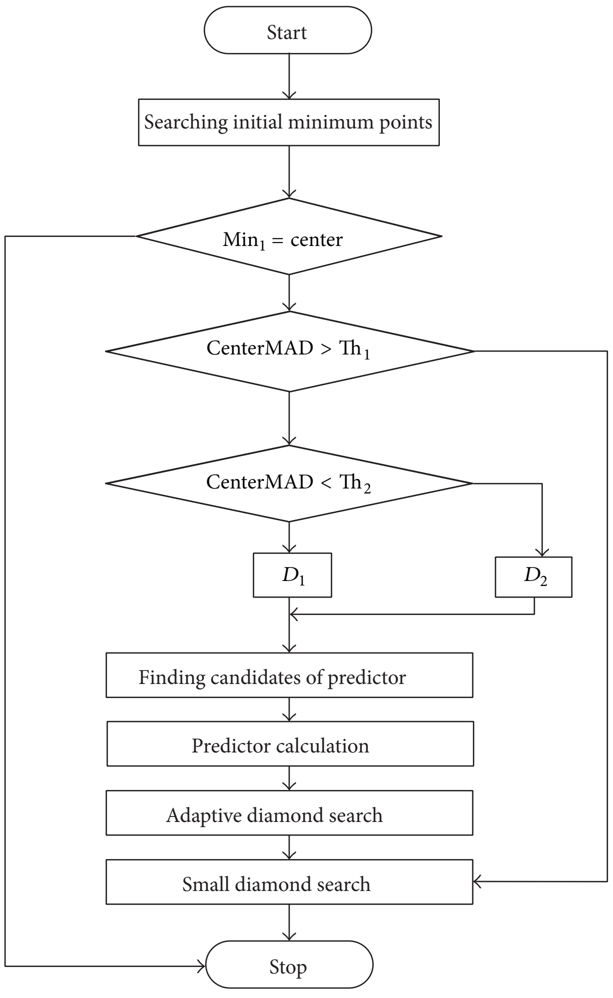


Figure 6: Overall procedure of the proposed algorithm.

algorithm terminates. For this case, a small diamond search region is considered again with 4 new points (shown in Figure 5 as point$_s$). In this second step, SAD values of the 5 points are checked. As in the previous decision, if the minimum value lies in the center, then the search terminates. Other conditions are shown in Figure 6 and discussed in detail in the next section.

3.4. Overall Procedure. The proposed algorithm can be divided into the steps shown in Figure 6:

(1) Initial decision for predictor candidates.

(2) Estimation of predictors.

(3) Setting a search pattern size.

(4) Performing search patterns.

Initially, values Min$_1$ and Min$_2$ are calculated. If Min$_1$ is the center, then the block has zero motion and no additional investigation is needed. Otherwise, the MAD value of the center is calculated as MAD$_c$, the value of which is compared with Th$_1$ and Th$_2$ for calculation of the jumping distance. Using the value of MAD$_c$ and the thresholds, block motion is determined, as shown in (5), as

$$\begin{aligned} &\text{slow motion: } \mathrm{MAD}_c < \mathrm{Th}_1(\mathrm{QP}), \\ &\text{medium motion: } \mathrm{Th}_1(\mathrm{QP}) < \mathrm{MAD}_c < \mathrm{Th}_2(\mathrm{QP}), \\ &\text{fast motion: } \mathrm{MAD}_c > \mathrm{Th}_2(\mathrm{QP}). \end{aligned} \tag{5}$$

For slow motion, only a small diamond search is considered in the proposed algorithm. On the other hand, medium and fast motion based blocks require calculation of the predictor. The overall procedure is shown in Figure 6.

4. Experimental Results

The proposed algorithm was implemented using JM 18.5 [25] and HM 10.0 [26] of HEVC. Measurement of ΔBit, ΔPSNR$_Y$, and ΔT was defined as

$$\begin{aligned} \Delta\mathrm{Bit} &= \frac{\left(\mathrm{Bitrate}_{\mathrm{proposed}} - \mathrm{Bitrate}_{\mathrm{original}}\right)}{\mathrm{Bitrate}_{\mathrm{original}}} \times 100\%, \\ \Delta\mathrm{PSNR}_Y &= \mathrm{PSNR}_Y^{\mathrm{proposed}} - \mathrm{PSNR}_Y^{\mathrm{original}}, \\ \Delta T &= \frac{\left(\mathrm{Time}_{\mathrm{proposed}} - \mathrm{Time}_{\mathrm{original}}\right)}{\mathrm{Time}_{\mathrm{original}}} \times 100\%, \end{aligned} \tag{6}$$

where ΔBit indicates the total bitrate change (percentage), ΔPSNR$_Y$ indicates ΔPSNR$_Y$ changes, and ΔT is the time saving factor (percentage). A positive value for the bitrate indicates an increase in the number of bits and a negative value indicates a decrease. An improvement in the ΔPSNR$_Y$ parameter gives a positive value and degradation gives a negative value. A negative value for the time saving factor indicates that the proposed algorithm consumes less time than the original reference software. We performed all simulation experiments on a PC with an Intel i5-3470 processor with a 3.2 GHz clock speed and 4 GB of RAM running on Windows 7. The proposed algorithm was tested using both H.264/AVC and HEVC.

4.1. Performance Evaluation Using H.264/AVC Standard. The experimental conditions were as follows:

(1) Frames To Be Encoded 100.

(2) Search Range 32.

(3) QP 24, 28, and 32.

(4) Framerate 20.

(5) Entropy coding method CABAC.

(6) The format of YUV 4:2:0.

(7) The coding environment IPPP frame structure.

(8) Five reference frames per sequence.

(9) Only integer motion estimation.

Video sequences considered were *News*, *Foreman*, *Soccer*, *Stefan*, and *Tractor*. *News* is representative of slow motion.

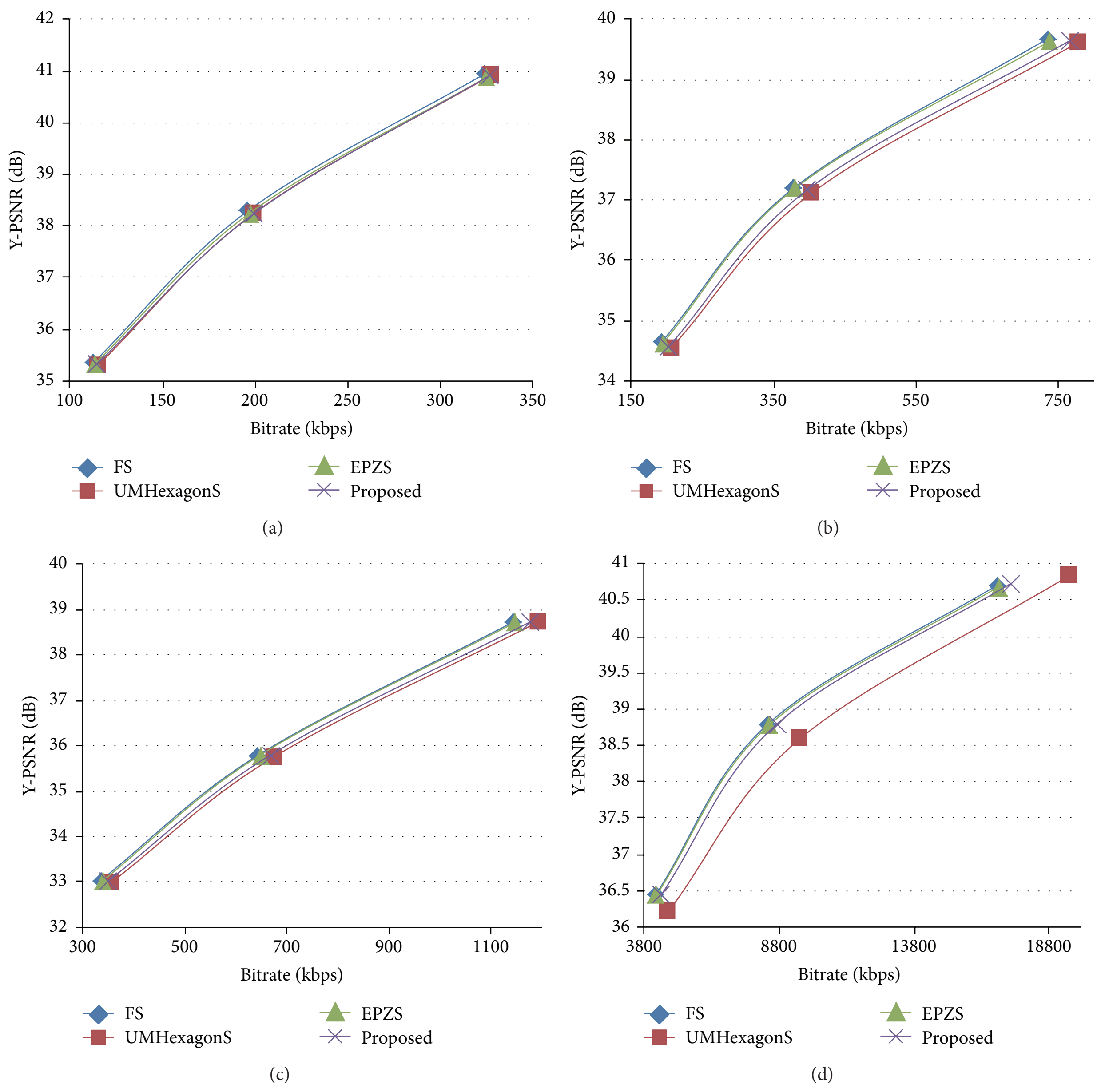


Figure 7: Rate-distortion (RD) curves for (a) *News*, (b) *Foreman*, (c) *Soccer,* and (d) *Tractor* sequences for slow motion, middle motion, and fast motion.

Foreman and *Tractor* are medium motion, and fast motion representative sequences are *Soccer* and *Stefan*. The frame size of *Tractor* was 1920 × 1080 (FHD). The other sequences used a frame size of 352 × 288 (CIF).

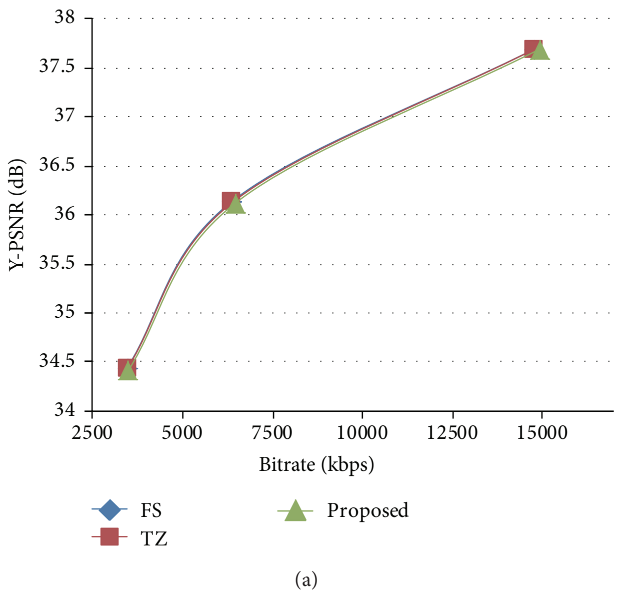
Experimental results are shown in Table 3 in which the proposed algorithm is compared with the full search (FS), UMHexagon search, and EPZ search algorithms. Deviations between the proposed algorithm and the comparison algorithms are shown. The proposed algorithm achieved reductions of 97.10% for total time, on average, compared with FS and achieved, on average, 29.43% and 8.28% more total time reductions than for UMHexagonS and EPZS, respectively. Considering the motion estimation time (ΔME time) the proposed algorithm provided 98.88%, 48.57%, and 16.03%, on average, time reductions compared with FS, UMHexagonS, and EPZS, respectively.

Quality degradation of the proposed algorithm is also shown based on the ΔPSNR and ΔBit metrics. The proposed algorithm provided a negligible loss of quality, compared with the FS, UMHexagonS, and EPZS algorithms, all with larger time reductions.

To show PSNR deviation from a low to a high bitrate, RD curves of the proposed and compared algorithms are plotted in Figure 7 for all sequences. All of the algorithms provided almost the same RD curve. For the *Forman* sequence, the proposed and UMHexagonS algorithms produced similar

Table 3: The performance comparison of the proposed algorithm with the FS, UMHexagonS, and the EPZS on the JM 18.5.

Sequences	QP	FS				UMHexagonS				EPZS			
		ΔPSNR (dB)	ΔBit (%)	ΔME time (%)	Δ*T* time (%)	ΔPSNR (dB)	ΔBit (%)	ΔME time (%)	Δ*T* time (%)	ΔPSNR (dB)	ΔBit (%)	ΔME time (%)	Δ*T* time (%)
News	24	−0.02	0.93	−98.91	−96.68	0	−0.22	−25.4	−9.7	0.03	0.54	9.68	2.85
	28	−0.03	0.93	−98.91	−96.36	0	−0.22	−25.4	−13.77	0.03	0.54	9.68	1.09
	32	−0.05	1.13	−98.61	−96.07	0.01	−0.69	−40.26	−18.83	−0.03	0.51	−3.77	−1.67
Foreman	24	−0.03	4.25	−98.92	−97.43	0.03	−1.58	−39.62	−21.35	0	3.6	−17.18	−7.76
	28	−0.02	5.4	−98.76	−97.12	0.05	−1.47	−45.36	−26.16	−0.02	4.76	−21.42	−10.35
	32	−0.07	5.35	−98.57	−96.75	0.02	−1.74	−51.94	−31.88	−0.05	4.6	−25.64	−12.85
Soccer	24	0.01	2.94	−99.13	−97.88	0	−1.47	−56.78	−34.99	0.01	2.5	−18.52	−8.29
	28	0.01	2.94	−99.13	−97.84	−0.01	−1.47	−56.78	−40.65	0.01	2.5	−18.52	−13.66
	32	0	4.83	−99.02	−97.74	0.04	−1.4	−66.03	−45.52	0.01	3.65	−34.62	−18.29
Stefan	24	0	2.99	−99.13	−97.64	0.01	−0.4	−62.98	−38.36	0.01	2.93	−18.34	−7.44
	28	−0.02	4.19	−99.07	−97.61	−0.01	−0.44	−65.59	−42.29	−0.01	3.99	−22.03	−9.7
	32	−0.02	5.61	−99.01	−97.56	−0.01	−0.44	−67.74	−45.71	0	5.31	−26	−12.25
Tractor	24	0.02	3	−98.76	−97.18	−0.1	−11.21	−37.06	−20.99	0.04	2.53	−9.68	−4.43
	28	0.01	4.17	−98.67	−96.97	0.2	−8.89	−41.13	−23.89	0.01	3.56	−17.33	−8.24
	32	−0.01	4.61	−98.61	−96.77	0.24	−5.42	−46.43	−27.38	0	4.08	−26.57	−13.19
Average		**−0.01**	**3.55**	**−98.88**	**−97.17**	**0.03**	**−2.47**	**−48.57**	**−27.38**	**0**	**3.04**	**−16.02**	**−8.28**



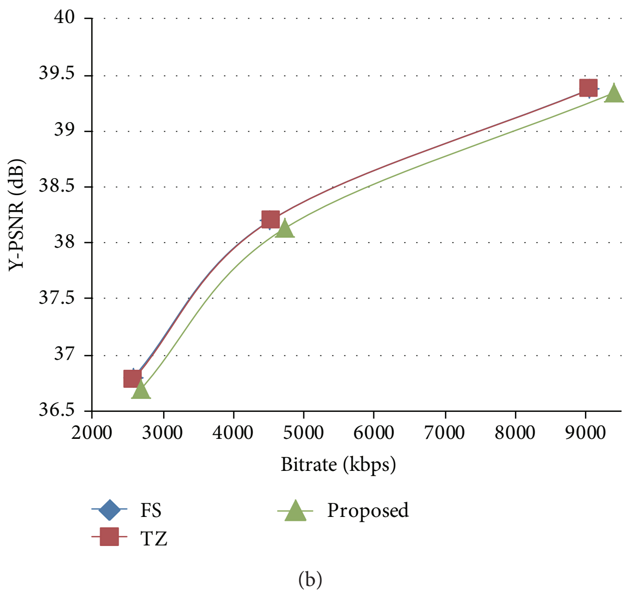


Figure 8: Rate-distortion (RD) curves for (a) *Cactus* and (b) *BasketballDrive* sequences for Class B in low-delay-p, main condition.

RD performance. The proposed algorithm achieved a better result than UMHexagonS for the *Tractor* sequence and almost the same performance as FS for the same sequence.

4.2. Performance Evaluation Using HEVC Standard. The proposed algorithm was tested in HM 10.0 reference software using the HEVC encoder. The test environment using the same hardware as for the H.264/AVC comparison was based on the following configuration:

(1) QP values are 24, 28, and 32.

(2) The first 30 frames of each sequences were considered.

(3) All analyses were performed in low-delay main mode.

(4) Fast modes were set in the order of FEN: 1, ECU: 0, FDM: 1, CFM: 0, ESD: 0.

(5) Sequences were tested, including *Cactus* (Class B), *BasketballDrive* (Class B), *BQMall* (Class C), *PartyScene* (Class C), and *BasketballPass* (Class D).

Results are shown in Table 4 in which the proposed algorithm was compared with the FS and the TZS algorithms [27]. The proposed algorithm reduced the overall encoding time, on average, 95.57% and 14.27%, compared with FS and TZS, respectively. Considering only the motion estimation time, the proposed algorithm achieves 99% and 42.61% time reductions, compared with the FS and TZS algorithms.

To evaluate quality degradation, ΔPSNR and ΔBit values are also shown in Table 4. The proposed algorithm provided a similar PSNR loss with a marginal bit reduction.

RD curves of all sequences are shown in Figures 8, 9, and 10 for Class B, Class C, and Class D. The proposed algorithm achieved almost the same result as the FS and TZS algorithms. For the *BasketballDrive* sequence, the proposed method achieved a degradation of the bitrate, compared with FS, due to the complexity of the sequence.

In IoT environment, multimedia contents can be also generated locally or downloaded from the Internet through any WiFi, DSRC, WiMAX, 4G LTE connections available to the mobile/smart devices and enable the end-user or end-system to take appropriate actions and be aware of the environmental conditions based on rich visual information. In view of this, real-time content delivery is very important, especially based on HEVC video standard. The proposed algorithm can provide a scheme for supporting the real-time UHD video content distribution system.

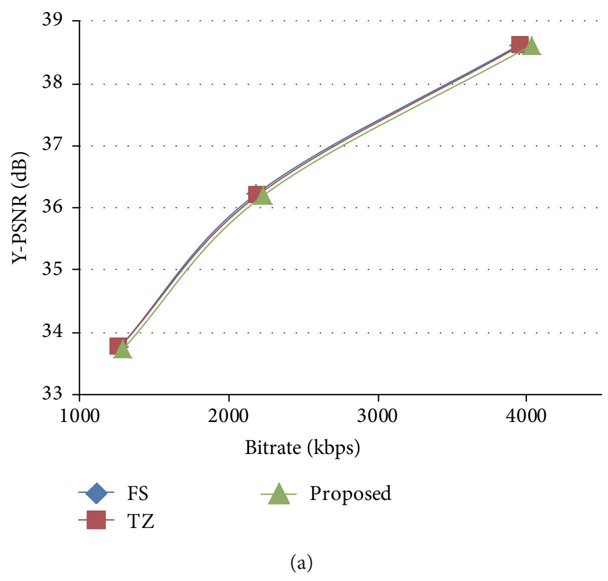
5. Conclusions

A video content-based fast motion estimation algorithm is herein proposed. Based on motion classification, a predictor is used and an adaptive search pattern size is set. Using H.264/AVC, experimental results showed that the proposed algorithm achieved speed-up factors of up to 48.57% and 16.03%, on average, compared with UMHexagonS and EPZS, respectively, while maintaining good bitrate performance.

For HEVC, the proposed algorithm achieved a speed-up factor of up to 42.61%, on average, compared with TZS. Also, for medium and fast motion sequences, the proposed algorithm substantially reduced the consumed time for motion estimation. The proposed scheme will be helpful for implementation of a real-time video encoding system because of no need for large memory comparing with EPZS and UMHexagonS algorithms. It means that the proposed algorithm can give easy hardware implementation of HEVC encoder on portable (embedded) devices.

Table 4: The performance comparison of the proposed algorithm with the FS and the TZS algorithm on the HM 10.0.

Class	Sequences	QP	FS				TZS			
			ΔPSNR (dB)	ΔBit (%)	ΔME time (%)	ΔT time (%)	ΔPSNR (dB)	ΔBit (%)	ΔME time (%)	ΔT time (%)
B	*BasketballDrive*	24	−0.03	3.88	−98.87	−94.24	−0.03	3.80	−60.11	−20.23
		28	−0.07	4.55	−98.88	−95.21	−0.07	4.43	−55.83	−20.36
		32	−0.11	3.78	−98.89	−95.93	−0.02	3.81	−51.90	−21.54
	Cactus	24	0.00	1.03	−99.01	−95.02	0.00	0.98	−44.21	−12.71
		28	−0.02	1.30	−99.02	−96.02	−0.01	1.09	−41.55	−14.07
		32	−0.03	0.70	−99.03	−96.60	−0.02	0.81	−39.61	−15.79
C	*PartyScene*	24	0.00	0.77	−99.04	−93.71	0.00	0.84	−46.69	−11.16
		28	0.00	1.28	−99.04	−95.01	−0.01	1.23	−44.54	−12.53
		32	−0.02	1.40	−99.05	−95.88	−0.03	0.47	−43.01	−15.67
	BQMall	24	−0.01	2.01	−99.00	−94.94	−0.01	1.58	−40.26	−11.13
		28	−0.03	2.04	−99.01	−95.82	−0.02	1.36	−38.09	−12.09
		32	−0.03	1.79	−99.02	−96.41	0.00	1.70	−37.31	−14.23
D	*BasketballPass*	24	−0.02	0.61	−99.05	−95.78	−0.02	0.65	−33.57	−9.80
		28	0.00	0.48	−99.05	−96.30	−0.03	0.39	−32.20	−10.40
		32	−0.02	0.14	−99.04	−96.73	0.00	1.08	−30.22	−12.35
	Average		**−0.03**	**1.72**	**−99.00**	**−95.57**	**−0.02**	**1.61**	**−42.61**	**−14.27**



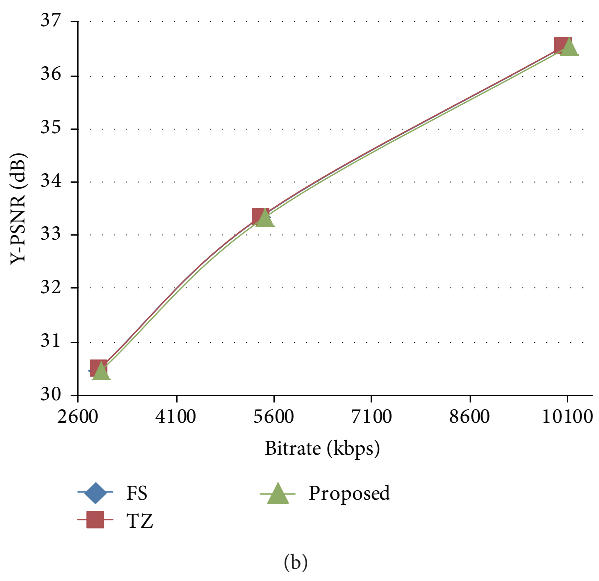


FIGURE 9: Rate-distortion (RD) curves for (a) *BQMall* and (b) *PartyScene* sequences for Class C in low-delay-P, main condition.

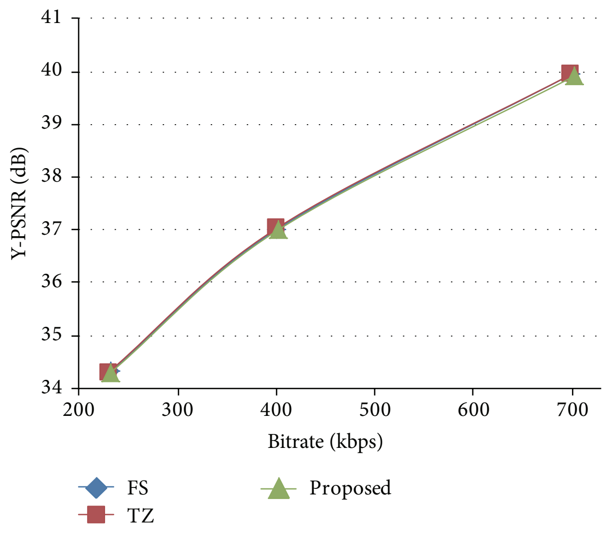


FIGURE 10: Rate-distortion (RD) curves for *BasketballPass* sequences for Class D in low-delay-P, main condition.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] B. Xu, L. D. Xu, H. Cai, C. Xie, J. Hu, and F. Bu, "Ubiquitous data accessing method in iot-based information system for emergency medical services," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1578–1586, 2014.

[2] W. Jiang and L. Meng, "Design of real time multimedia platform and protocol to the internet of things," in *Proceedings of the 11th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom '12)*, pp. 1805–1810, IEEE, Liverpool, UK, June 2012.

[3] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no. 7, pp. 560–576, 2003.

[4] G. J. Sullivan, J.-R. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649–1668, 2012.

[5] Z. Liu and T. Yan, "Study on multi-view video based on IOT and its application in intelligent security system," in *Proceedings of the International Conference on Mechatronic Sciences, Electric Engineering and Computer (MEC '13)*, pp. 1437–1440, IEEE, Shenyang, China, December 2013.

[6] I. E. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*, John Wiley & Sons, 2004.

[7] T. Koga, K. Ilinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion-compensated interframe coding for video conferencing," in *Proceedings of the NTC '81*, New Orleans, La, USA, November 1981.

[8] J. R. Jain and A. K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Transactions on Communications*, vol. 29, no. 12, pp. 1799–1808, 1981.

[9] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 3, pp. 313–317, 1996.

[10] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, no. 4, pp. 419–422, 1996.

[11] S. Zhu and K.-K. Ma, "A new diamond search algorithm for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 9, no. 2, pp. 287–290, 2000.

[12] K. Goswami, B. G. Kim, D. S. Jun, S. H. Jung, and J. S. Choi, "Early coding unit-splitting termination Algorithm for high

efficiency video coding (HEVC)," *ETRI Journal*, vol. 36, no. 3, pp. 407–417, 2014.

[13] K. Chen, Z. Zhou, and W. Wu, "Clustering based search algorithm for motion estimation," in *Proceedings of the 13th IEEE International Conference on Multimedia and Expo (ICME '12)*, pp. 622–627, July 2012.

[14] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 5, pp. 349–355, 2002.

[15] Z. Chen, J. Xu, Y. He, and J. Zheng, "Fast integer-pel and fractional-pel motion estimation for H.264/AVC," *Journal of Visual Communication and Image Representation*, vol. 17, no. 2, pp. 264–290, 2006.

[16] R. Li, B. Zeng, and M. L. Liou, "New three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438–442, 1994.

[17] X. Jing and L.-P. Chau, "An efficient three-step search algorithm for block motion estimation," *IEEE Transactions on Multimedia*, vol. 6, no. 3, pp. 435–438, 2004.

[18] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, pp. 429–433, 1997.

[19] L. Luo, C. Zou, X. Gao, and Z. He, "A new prediction search algorithm for block motion estimation in video coding," *IEEE Transactions on Consumer Electronics*, vol. 43, no. 1, pp. 56–61, 1997.

[20] K.-H. Ng, L.-M. Po, K.-M. Wong, C.-W. Ting, and K.-W. Cheung, "A search patterns switching algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 5, pp. 753–759, 2009.

[21] 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 Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 11, pp. 1373–1384, 2005.

[22] P. Liu, Y. Gao, and K. Jia, "An adaptive motion estimation scheme for video coding," *The Scientific World Journal*, vol. 2014, Article ID 381056, 14 pages, 2014.

[23] Tourapis and M. Alexis, "Enhanced predictive zonal search for single and multiple frame motion estimation," *Electronic Imaging 2002*, vol. 7, no. 2, pp. 1069–1079, 2002.

[24] Y. Nie and K.-K. Ma, "Adaptive rood pattern search for fast block-matching motion estimation," *IEEE Transactions on Image Processing*, vol. 11, no. 12, pp. 1442–1449, 2002.

[25] Joint Video Team (JVT) reference software version 18.5, http://iphome.hhi.de/suehring/tml/download/old_jm/.

[26] HM Reference Software 10.0, HEVC Software, http://hevc.hhi.fraunhofer.de/svn/svn_HEVCSoftware/tags/HM-10.0/.

[27] N. Purnachand, L. N. Alves, and A. Navarro, "Improvements to TZ search motion estimation algorithm for multiview video coding," in *Proceedings of the 19th International Conference on Systems, Signals and Image Processing (IWSSIP '12)*, pp. 388–391, IEEE, Vienna, Austria, April 2012.