Multimedia Tools and Applications manuscript No.
(will be inserted by the editor)

Motion estimation with chessboard pattern prediction strategy

Hadi Amirpour[1] · **Mohammad Ghanbari** [2,3] · **Antonio Pinheiro**[1] · **Manuela Pereira**[1]

Received: date / Accepted: date

Abstract Due to high correlations among the adjacent blocks, several algorithms utilize movement information of spatially and temporally correlated neighbouring blocks to adapt their search patterns to that information. In this paper, this information is used to define a dynamic search pattern. Each frame is divided into two sets, black and white blocks, like a chessboard pattern and a different search pattern is defined for each set. The advantage of this definition is that the number of spatially neighbouring blocks is increased for each current block and it leads to a better prediction for each block. Simulation results show that the proposed algorithm is closer to the Full-Search algorithm in terms of quality metrics such as PSNR than the other state-of-the-art algorithms while at the same time the average number of search points is less.

Keywords Video compression · Motion estimation · Dynamic search pattern · Prediction · PSNR

The authors are very grateful to the Portuguese FCT-Fundação para a Ciência e Tecnologia and co-funded by FEDER-PT2020, Portugal partnership agreement under the project PTDC/EEI-PRO/2849/ 2014 - POCI-01-0145-FEDER-016693, and under the project UID/EEA/50008/ 2013.

[1]Instituto de Telecomunicações
Universidade da Beira Interior, Covilhã, Portugal
Tel.: +351-918 246 733
E-mail: hadi.amirpour@gmail.com

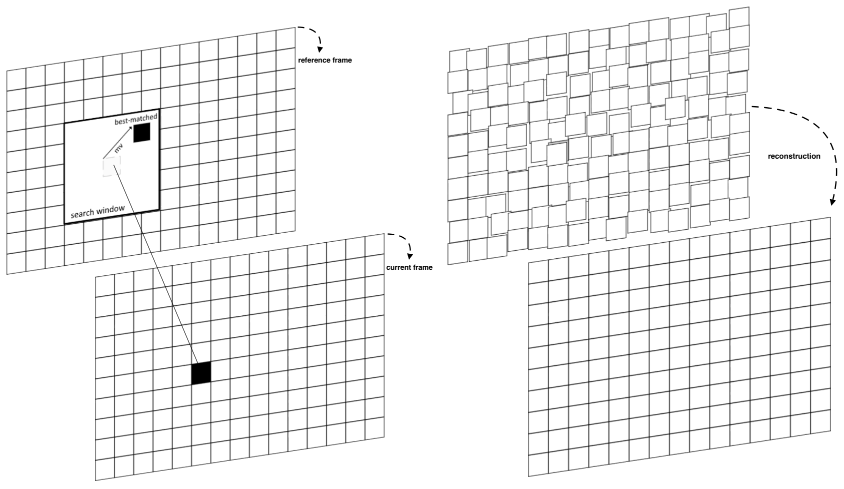
[2] School of Electrical and Computer Engineering, College of Engineering
University of Tehran, Tehran, Iran

[3]School of Computer Science and Electronic Engineering
University of Essex, Colchester, UK

1 Introduction

With the increasing usage of video, namely in social media, as well as the increase in video quality and resolution, the need to reduce the volume of compressed video for storage and transmission over the network is felt more than before. Video coding standards try to reduce redundancy between successive frames and motion estimation is the critical part of this process. Block-matching motion estimation algorithms due to their efficiency and simplicity are broadly used in the state-of-the-art video coding standards like HEVC [24]. As shown in Fig. 1a, these algorithms divide the current frame into non-overlapping blocks and search for the best-matched block in a corresponding search window in the previous frame(s), called reference frame(s). A motion vector is defined as the difference in positions between a candidate block and its best-matched block in the corresponding search window. When motion vectors for all the blocks are determined, the blocks in the reference frame are used to reconstruct the current frame (Fig. 1b) [9].

The most straightforward method to find the best-matched block in the defined search window is the Full-Search (FS) algorithm, which searches for all the blocks inside the search window and results in the best possible matched block. However, searching all the blocks demands a high computational cost. Several algorithms have been proposed to reduce this computational cost while they try to keep the quality close to FS. Traditional fast motion estimation algorithms such as three step search (TSS) [10], four step search (FSS) [18], diamond search (DS) [27], hexagonal search (HS) [26], cross search (CS) [5],



(a) (b)

Fig. 1: Motion estimation and compensation. (a) shows motion estimation process. (b) indicates how corresponding best-matched blocks found in the reference frame make current frame.

orthogonal search (OS) [19] and etc. use fixed search patterns to find best-matched block in the search window based on a unimodal error surface assumption [2]. Examples of search paths of these algorithms are shown in Fig.2 [6].

Although these algorithms reduce computational complexity compared to FS, they are not flexible to the various types of movements. They also have some fundamental problems, such as being trapped into a local minimum and over-searching. To have an early estimate of the current block's motion and avoid using a fixed search pattern for all types of movements, some algorithms utilize movement information of neighbouring blocks to get an early estimate of the current block's movement. This is due to the high correlation among the movements of adjacent blocks. Each block has four spatially neighbouring coded blocks which are shown in Fig. 3, as well as one temporally neighbouring block, where their motion vectors can be used as predictors.

According to the algorithms such as ARPS [15], APSP [14], long rood [13], PMVFAST [25], PTSS [4] and so on, motion vectors of adjacent blocks can be used to determine size, type or center of the search pattern, region of the search or other kinds of predictions. Moreover, DPS [23], Enhanced DPS [22], DISP [16] and, Prioritized-DSP [3] define a dynamic search pattern based on motion vectors of neighbouring blocks which can eliminate unnecessary search points and improve the performance of the algorithms. However, these algorithms

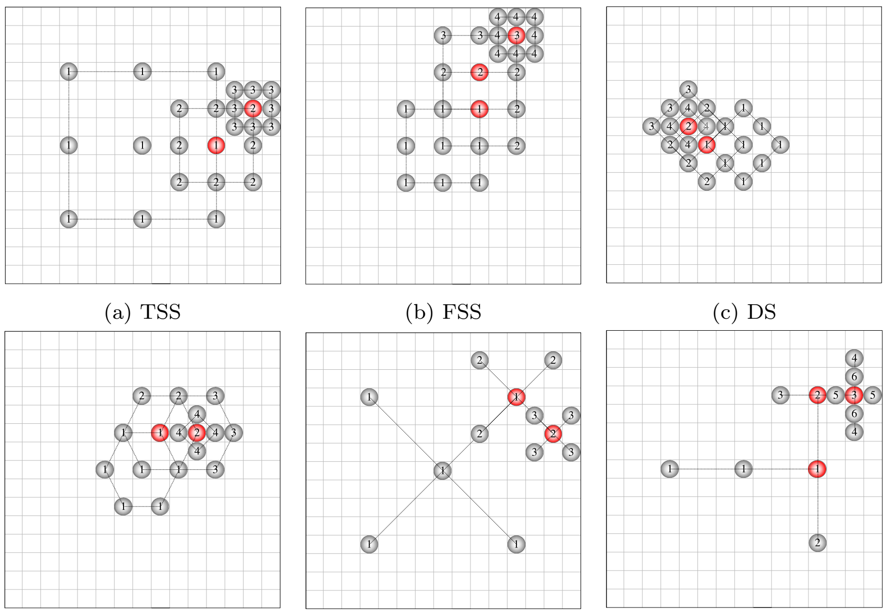

(a) TSS (b) FSS (c) DS (d) HS (e) CS (f) OS

Fig. 2: Examples of search paths of some traditional motion estimation algorithms. Numbers in each circle indicate the step and red points show best-matched block in each step.

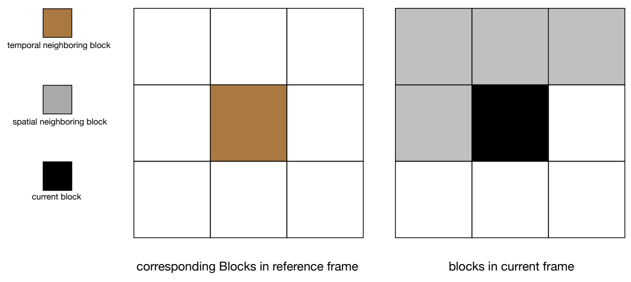

temporal neighboring block
spatial neighboring block
current block
corresponding Blocks in reference frame
blocks in current frame

Fig. 3: Spatial and temporal neighbouring blocks.

are limited to the four spatially neighbouring blocks because of using the conventional raster scan. Hence, they cannot utilize movement information of eight immediate spatially neighbouring blocks. In this paper, a new search method that allows the algorithm to use more motion vectors of neighbouring blocks and increase the performance of the search is proposed. The reminder of this paper is organized as follows: In Section 2, a systematic and detailed review of the state-of-the-art algorithms especially those which use dynamic search patterns is presented. The proposed algorithm which uses chessboard instead of conventional raster scan method is introduced in Section 3. Experimental results and discussions are covered in Section 4 and finally conclusion is drawn in Section 5.

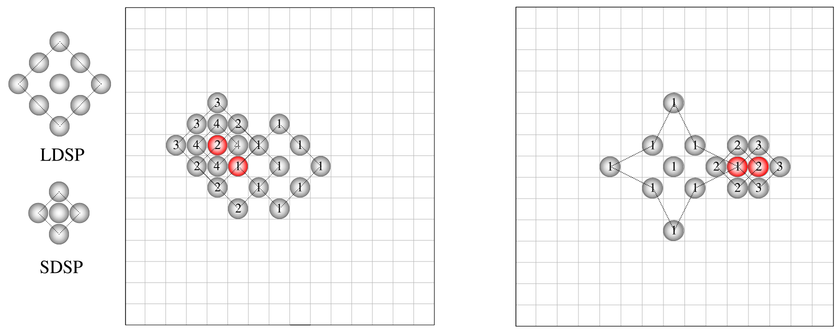
2 Related work

Motion estimation algorithms can be categorized into two types. First group of algorithms use fixed search patterns to find an optimal motion vector for any kind of video. DS is the most-known method in fixed search pattern algorithms that has a good performance in terms of quality/complexity. In this algorithm, a Large Diamond Search Pattern (LDSP) is repeatedly searched to find the best-matching block until the central point is optimal. Then the LDSP is changed to a Small Diamond Search Point (SDPS) and search is continued until the central point becomes optimal resulting in the final motion vector. Based on analysis of motion vectors' distribution, LDSP has changed to a star search model in star diamond (SD) [7] algorithm. Examples of DS and SD are shown in Fig.4.

State-of-the-art video encoders such as HEVC use TZSearch algorithm to find the motion vectors. TZSearch (based on default setting of HM software 16.19 [1]) is formed by the following steps [21,20,8]:

Best predictor definition: First, median of left, top and top-right predictors is computed. Then the predictor with minimum cost among the median and zero predictors is selected as the center for next step.

Initial grid search: Diamond pattern with different stride lengths that are multiples of two, are employed. This pattern is shown in Fig. 5a. Distance of



(a) Diamond Search (b) Star Diamond

Fig. 4: Examples of DS and SD algorithms. Numbers in each circle indicate the step and red points show best-matched block in each step.

best point is saved in variable uBestDistance. Decision on the next step of algorithm is made based on uBestDistance:

- if $uBestDistance = 0$, algorithm stops.
- if $uBestDistance = 1$, Two-point search is performed.
- if $uBestDistance > iRaster$, Raster search is performed.
- if $1 < uBestDistance < iRaster$, Star refinement is performed.

Two-point search: In the cases that uBestDistance is 1, there are two out of eight unsearched points around the best-matched block of the last step. When these two points are searched, the algorithm stops. Fig. 5b shows an example of two-point search.
Raster search: When best-matched block is far from the center in the initial grid search - uBestDistance is more than predefined *iRaster* - raster search is performed. Raster search is a down-sampled version of the search window with a sampling factor *iRaster*. A raster search with $iRaster = 4$ has been illustrated in Fig. 5c
Star refinement: like the initial grid search with the difference that its center point has been changed to the best-matched block defined in the previous steps. A flowchart of the TZSearch algorithm is shown in Fig. 6.

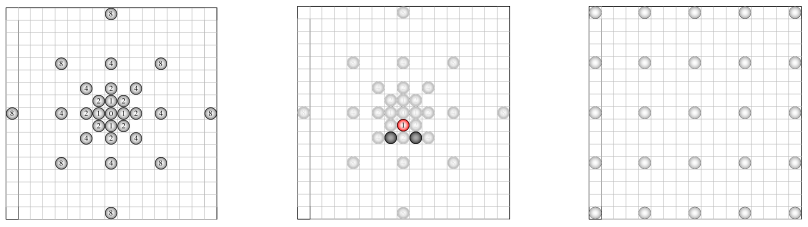

(a) Initial grid search (b) Two-point search (c) Raster search

Fig. 5: Search patterns used in TZSearch algorithm.

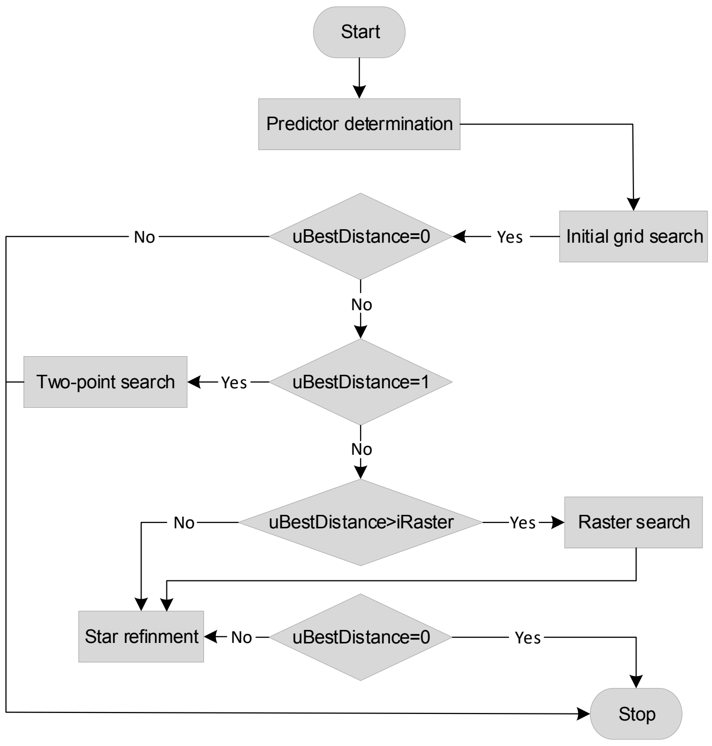


Fig. 6: TZSearch Flowchart.

Since these types of algorithms use fixed search patterns, they are not suitable for all type of video, especially those video frames which have a mixture of motions varying from very slow to fast movements. They also may do over-searching for stationary and semi-stationary video scenes.

The high similarity among motions of adjacent blocks is utilized in the second group of motion estimation algorithms to define a dynamic search pattern. In this case, the search is adapted to the motions of neighbouring blocks instead of using only one fixed search pattern for all kind of motions. One of the first attempts in this direction is the Adaptive Rood Pattern Search (ARPS) [15] that decomposes motion vector of the left block to horizontal and vertical components and the maximum of these components is used to determine the size of search pattern. This motion vector is also used as an initial search point. DPS [23] is a modified version of ARPS which adds the motion vector of the corresponding block in the reference frame to the initial search points set. Moreover, Enhanced DPS [22] gives to the initial search points a priority and stops the search algorithm if any of them meets a threshold criterion. As a result, in these types of algorithms videos with fast moving objects benefit from using large search pattern, and stationary and semi-stationary videos benefit from small-size search patterns. While in most of motion estimation algorithms, the center of search pattern for each current block is placed at the corresponding location of the current block in the reference frame, some algorithms such as [13] update locations of the search pattern centers based on motion information of adjacent blocks. In order to eliminate searching unnecessary points to reduce complexity, algorithms like PTSS [4] divide a search window into different regions and search only the points that are placed in the same region of neighbouring blocks' optimal matched block. Moreover, some other algorithms including [12] use various type of search patterns and based on the obtained motion information make a decision to select an appropriate pattern. Adaptive pattern selection (APS) [17] adaptively uses SDSP, LDSP and half-way stop techniques to modify the DS algorithm.

Above mentioned algorithms although try to adapt their search patterns to information of the obtained motions, however, they use a fixed search pattern and modify it. The state-of-the-art algorithms use spatially and temporally adjacent motion vectors to form an initial search pattern and for each candidate block the initial search pattern can vary. Dynamic initial search pattern (DISP) [16] is an algorithm that its initial search points are completely dependent on the motion vectors of neighbouring blocks. To construct an initial search point set, motion vectors of top (T), left (L) and top-left (TL) neighbouring blocks are decomposed into their horizontal and vertical components. Horizontal components consist of $S_x : \{x_L, x_T, x_{TL}\}$ and vertical components form $S_y : \{y_L, y_T, y_{TL}\}$. Now, an initial search points set is obtained by Cartesian product of S_x and S_y as:

$$S = S_x \times S_y \tag{1}$$

Fig. 7 details the definition of these initial search points. After searching the points in set S , a cross search pattern is centered at the optimal point and

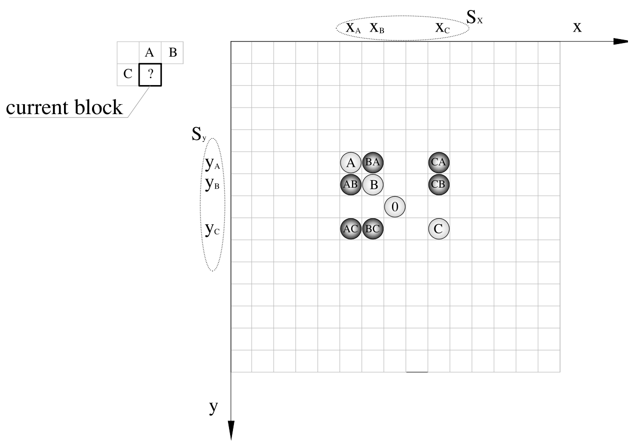


Fig. 7: Initial search pattern in the DISP algorithm. Darker blocks are added using Cartesian product of S_x and S_y.

searching is continued until the center is the best matching block. Prioritized dynamic search pattern (PDSP) [3] uses all the available spatially and temporally adjacent neighbouring blocks' (left, top-left, top, right and temporal) motion vectors along with (0,0) to form a dynamic search pattern and prioritizes them based on their matching criterion that has been calculated between these blocks and their best-matched blocks. Then these points are searched one by one and respectively each point that meets the threshold criterion is selected as optimum and the algorithm stops; otherwise, refinement step is started and continues until the center is the best point.

3 Chessboard search pattern (CSP)

The above survey of the literature in motion estimation indicates that for a better estimate of the motion vector of the current block, its surrounding motion vectors are investigated. In this regard, whether the search range is kept constant or varied dynamically, an increase in the estimation accuracy is highly desired. In the following section, we introduce a new way of exploiting the neighbouring motion vectors, named as chessboard search pattern (CSP).

3.1 Initial dynamic search pattern.

As shown in Fig. 8, conventional motion estimation algorithms use raster scan order to find the motion vector of the current block. Fig. 8 shows how this scan order leads to the definition of four spatially neighbouring blocks for each current block. In this figure, gray blocks have been searched previously, the current block is shown with a question mark and its four spatially neighbouring blocks are marked with a star. As discussed in the previous section, movement information of neighbouring blocks is useful and can give a prediction for motion of the current block. To address this, motion vectors for two sequences, namely *bus* and *coastguard*, are stored using FS method. Then the similarity of

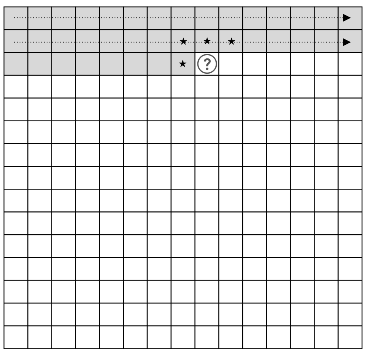
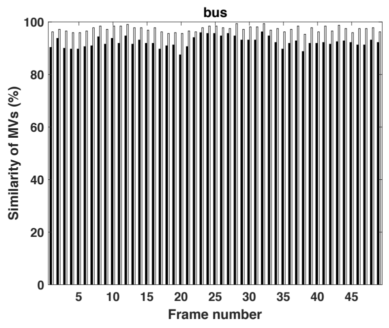


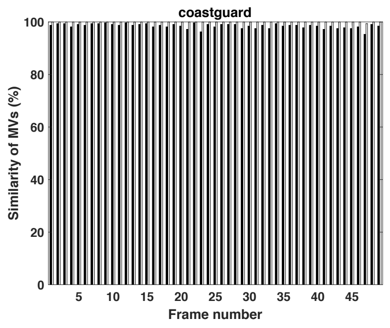
Fig. 8: Raster Scan.

motion vectors with motion vectors of all eight spatially neighbouring blocks, temporal block, and zero motion vector is computed. To do this, the number of blocks that have at least one similar motion vector with the above-mentioned predictors is divided by the total number of the blocks. Secondly, the similarity of motion vectors with first groups along with their four immediate blocks is also computed. In Fig. 9, black and white bars show similarity with first and second groups, respectively.

However, because of the raster scan order, each current block has access to motion vectors of only four spatially neighbouring coded blocks: left, top-left, top and top-right blocks. To utilize motion vectors of other neighbouring blocks a chessboard scan can be used. In [11] partitioning of the current frame into three groups of macroblocks with different number of available predictors is proposed. In our paper, blocks in each current frame are divided into two different sets - white and black blocks - like a chessboard pattern. Firstly, the algorithm starts to find the best-matched block for black blocks (Fig. 10a) and assigns them an early motion vector. When black blocks are searched, each block has three neighbouring blocks: top-left, top-right and temporal. The number of motion vectors that is used as predictors is the same as EPZS but the predictors were changed. In the proposed algorithm, the left predictor has been changed to top-left and top temporal predictors. These predictors along with (0,0) are used as initial search points for black blocks. When initial motion vectors are found for all the black blocks, the algorithm starts to search for white blocks. An example of a white block is shown in Fig. 10b. As can be seen from this figure, when the algorithm starts to find motion vectors for a white block all the black blocks had already been searched and their motion vectors are already available. Consequently, motion vectors of four neighbouring black blocks including left, top, right and bottom are available as predictors. Also, top-left and top-right predictors whom blocks have been searched as white



(a) bus



(b) coastguard

Fig. 9: Similarity of black and white blocks motion vectors with their neighbouring blocks.

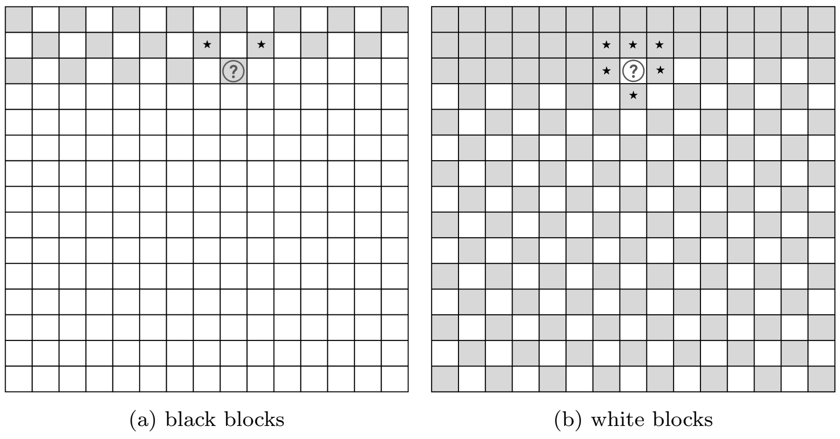

(a) black blocks (b) white blocks

Fig. 10: Available neighbouring blocks for black and white blocks.

blocks are also available for the current white block. Moreover, temporal and (0,0) motion vectors will be used as predictors, too.
Therefore, motion vectors of seven neighbouring blocks will be available to be used as a predictor for white blocks: left, top-left, top, top-right, right, bottom and temporally co-located block along with (0,0) there will be eight points to be searched. Predictor motion vectors used in this paper are named in Table. 1. For both black and white blocks, the algorithm uses motion vectors of neighbouring blocks to define a dynamic search pattern.

Table 1: Terminology table.

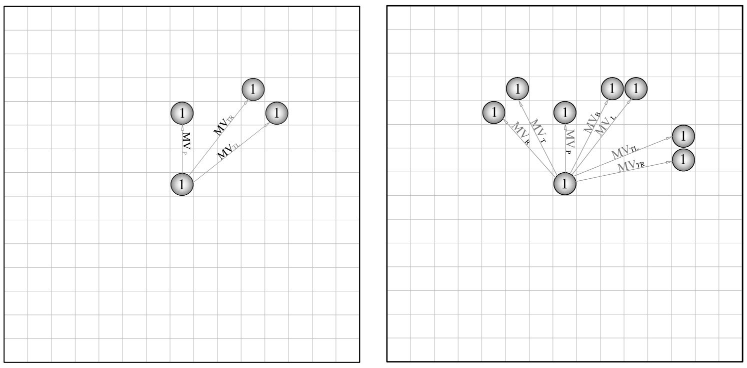
Term	Definition
MV_L	motion vector of left neighboring block
MV_{TL}	motion vector of top-left neighboring block
MV_T	motion vector of top neighboring block
MV_{TR}	motion vector of top-right neighboring block
MV_R	motion vector of right neighboring block
MV_{BR}	motion vector of bottom-right neighboring block
MV_B	motion vector of bottom neighboring block
MV_{BL}	motion vector of bottom-left neighboring block
MV_P	motion vector of temporal neighboring block
MV_0	zero motion vector
T	threshold to stop the algorithm
n_0	number of the black blocks to start parallel processing
n_r	number of the blocks that can be processed in parallel way

As an example, for a current black block, if top-left (MV_{TL}), top-right (MV_{TR}) and temporally neighbouring (MV_P) motion vectors have following values:
MV_{TL}=(3,2),
MV_{TL}=(2,3),

MV_{TL}=(0,2),
along with MV_0=(0,0) make a dynamic search pattern which is called dynamic black search pattern and can be seen in Fig. 11a.
Similarly, to the current black block, a dynamic white search pattern is defined for white current blocks. For instance, if the motion vectors of adjacent blocks for a white block are:
MV_R=(-2,2),
MV_T=(-1,3) ,
MV_P=(0,2),
MV_B=(1,3),
MV_L=(2,3),
MV_{TL}=(3,2),
MV_{TR}=(3,1)
along with MV_0=(0,0) define a dynamic white search pattern that is shown in Fig. 11b.

3.2 Proposed algorithm

Predictors can give an early estimation for the motion of the current block. As in most of the cases, the motion of adjacent blocks is similar, these predictors can avoid the algorithm from over-searching and being trapped into a local minimum. Moreover, defining a dynamic search pattern can prevent the algorithm from searching unnecessary points in the fixed search patterns that are not in the similar direction of the adjacent blocks' motions. Considering the right direction for movement of adjacent blocks, searching blocks that are in the left direction not only will increase the number of search points, but it also may increase the risk of being trapped into a local minimum. In the developed algorithm, as discussed in the previous section, black and white search patterns will be used as initial search patterns. However, the number of



(a) dynamic black search pattern (b) dynamic white search pattern

Fig. 11: Examples of initial search patterns for a black and white block.

initial search points for black blocks is not as enough as white points, therefore, the algorithm will find an early motion vector for the black blocks that will be useful in stationary blocks and blocks that have large motions and can be named as early best motion vector (EBMV). Then after searching white blocks, the algorithm will return to black blocks again. Now, for each black block motion vectors four white blocks are added as initial search points and will be compared to EBMV and the best point will be used as a center for the next step. The algorithm of the chessboard search method introduced in the previous section is as follows:

■ Black blocks:

- a dynamic black search pattern based on available motion vectors of adjacent blocks (MV_{TL}, MV_{TR} and MV_P) and $MV(0,0)$ is made, the matching criterion for all of them is computed and the best point is selected.
- an early search termination for best point is checked and if it is less than a threshold (T), the algorithm stops; Otherwise, it goes to the next step.
- unity size diamond search pattern is centered at the best-matched block, the matching criterion is calculated for corresponding points of the search pattern and the point with minimum matching error is selected as EBMV.

❏ White blocks:

- a dynamic white search pattern based on available motion vectors of adjacent blocks (MV_L, MV_{TL}, MV_T, MV_{TR}, MV_R, MV_B, MV_P) and MV(0,0) is made.
- early search termination for the best point is checked and if it is less than threshold (T), the algorithm stops; Otherwise, it goes to the next step
- unity size diamond search pattern is centered at the best matched block, the matching criterion is calculated for each point of the search pattern and the point with minimum matching error is selected as the best point.
- if the best point is at the center, the algorithm stops searching for this block and goes to other white blocks, else it returns to the previous step.

■ Black blocks refinement:

- MV_L, MV_T, MV_R and MV_B are compared with EBMV and the best point is selected.
- early search termination for the best point is checked and if it is less than a threshold (T), the algorithm stops; Otherwise, it goes to the next step.
- unity size diamond search pattern is centered at the best-matched block, the matching criterion is calculated for each point of the search pattern and the point with minimum matching error is selected as the best point.
- if the best point is at the center, the algorithm stops searching for this block and goes to other white blocks, else it goes to the previous step.

A flowchart of the proposed algorithm is depicted in Fig. 12. An example of the proposed algorithm for a black and a white block is shown in Fig.13a and

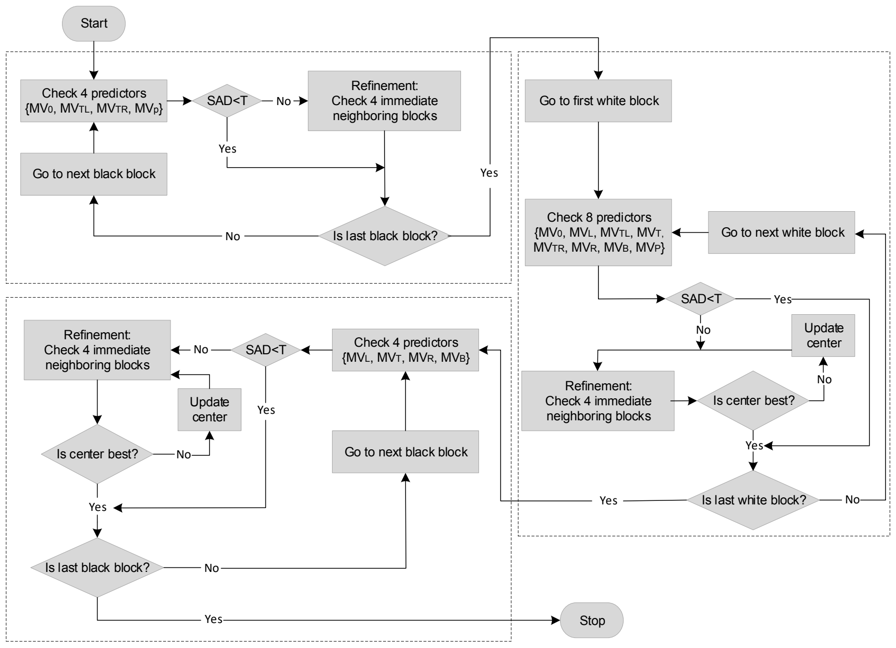
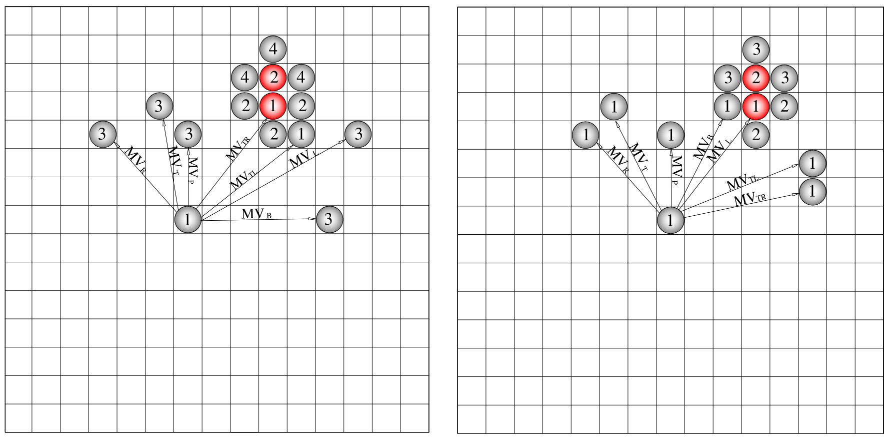

Start
Check 4 predictors {MV0, MVTL, MVTR, MVP}
SAD<T
No
Yes
Refinement: Check 4 immediate neighboring blocks
Go to next black block
Is last black block?
Go to first white block
Check 8 predictors {MV0, MVL, MVTL, MVT, MVTR, MVR, MVB, MVP}
Go to next white block
Update center
Is center best?
Is last white block?
Check 4 predictors {MVL, MVT, MVR, MVB}
Stop

Fig. 12: Flowchart of proposed algorithm.

Fig.13b, respectively. In these figures, the step number of the search has been written inside the circle representative of each point. Red points represent the best-matched block in each step. In Fig.13a, the algorithm starts with 4



(a) example for a black block (b) example for a white block

Fig. 13: Examples of the proposed algorithm for a black and a white block.

initial points which are shown with circles and the red circle is the best point. Then this best point is compared with the threshold and as it does not satisfy the threshold a diamond search pattern is centered at the best point and the corresponding points of the search pattern (points with number 2) are checked and the best one is represented with a red circle. When the algorithm returns to this block again, four motion vectors of white neighbouring blocks (points with number 3) are added to the initial search points and it updates the best-matched point (in this example best point does not change). If matching errors of these points are not less than the threshold, the refinement step is started, unity size diamond search is centered at the best point and the algorithm continues until the center is the best match. In the white block example, each initial search point is compared with the threshold and if none of them meets the threshold criterion, the best of them is selected as the best point which is represented with a red circle in the figure. A diamond search pattern is centered at the best point and the corresponding points are searched and the algorithm goes further until the center becomes the best point.

3.3 Parallel processing

The proposed method can be used in both off-line and on-line codecs of H.264/H.265 using parallel processing. Motion vectors of neighbouring blocks are available for the white blocks when black blocks in two rows are processed. Hence, searching white blocks can be started after processing two lines of black blocks which enables parallel processing. In this way, remaining white and black blocks can be run simultaneously. Fig. 14, depicts an example of how black and white blocks can be searched in parallel. In this figure, numbers inside each block indicate the step when corresponding blocks are searched. Black and white blocks with the same numbers can be used concurrently in different processors. Considering Fig. 14, the number of black blocks needed to be searched till parallel search can be started is $n_0 = 6$. After that, remaining blocks ($n_r = 94$) can be processed in parallel. Therefore, the maximum saving time can be calculated as:

$$Saving\ time(\%) = 1 - \frac{n_0 + n_r/2}{n_0 + n_r} = 1 - \frac{6 + 47}{100} * 100 = 47\%. \tag{2}$$

As it is clear, nearly 50% of time complexity can be saved through parallel processing.

4 Experimental results

Several video sequences listed in Table 2 have been used to compare the performance of the proposed algorithm with other block matching motion estimation techniques. For a comprehensive comparison, the video sequences are selected from test sequences of various motions and resolutions varying from QCIF to

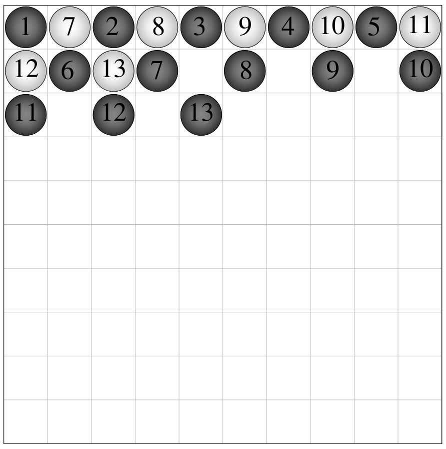


Fig. 14: Black and white blocks processed in parallel.

1080p. Simulations have been carried out in Matlab environment and previous frames have been chosen as reference frames for the current frame. The window search area is limited to 61×61 (± 30) pixels and the similarity measure of a current block with the blocks in the search window is based on the SAD matching criterion. The SAD for two $N \times N$ pixel blocks A and B is computed as below:

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |(A(i,j) - B(i,j))| \tag{3}$$

The quality of the reconstructed frames is measured in terms of PSNR which is defined as:

$$PSNR = 10\ log10\ \frac{2^n - 1}{MSE} \tag{4}$$

where n is number of bits per pixel. MSE is the Mean Squared Error between the current and reconstructed frames, defined as:

$$MSE = \frac{1}{m \times n} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (f_{rec}(i,j) - f_{ref}(i,j))^2 \tag{5}$$

where (m,n) is the size of each frame, f_{rec} is the reconstructed frame and f_{ref} is the reference frame. The threshold (T) is used as early termination criterion for best predictors and is set to $T = 512$.

Various known block matching algorithms are compared with each other in terms of quality and speed. Table 3 compares performance of the proposed

Table 2: Video Sequences used in simulations

Sequence	Format	Resolution	No. of frames
coastguard	QCIF	176x144	300
mother	QCIF	176x144	300
silent	QCIF	176x144	300
carphone	QCIF	176x144	300
foreman	QCIF	176x144	300
akiyo	QCIF	176x144	300
mobile	QCIF	176x144	300
setfan	SIF	352x240	300
mobile	CIF	352x288	300
paris	CIF	352x288	300
stefan	CIF	352x288	90
bus	CIF	352x288	150
waterfall	CIF	352x288	260
highway	CIF	352x288	300
beauty	1080p	1920x1080	100
readySetGo	1080p	1920x1080	100
yachtRide	1080p	1920x1080	100
jockey	1080p	1920x1080	100
shakeNDry	1080p	1920x1080	100
pedestrian	1080p	1920x1080	300
riverbed	1080p	1920x1080	250

Table 3: Performance comparison in terms of average PSNR(dB) for 16×16 pixel blocks.

	TSS	DS	ARPS	APS	DISP	TZS	Proposed	FS(±15)	FS(±30)
coastguard	32.42	32.44	32.46	32.45	32.46	32.47	32.46	32.48	32.49
mother	40.86	40.88	40.86	40.84	40.85	40.91	40.88	40.92	40.94
silent	34.83	34.75	34.65	34.62	34.65	34.9	34.79	34.99	35.04
carphone	31.73	31.78	31.7	31.69	31.73	31.92	31.82	32.06	32.12
foreman	31.8	31.78	31.83	31.83	31.83	32.11	31.95	32.2	32.25
akiyo	44.08	44.08	44.08	44.08	44.08	44.08	44.08	44.08	44.08
mobile	26.06	26.06	26.06	26.05	26.06	26.06	26.06	26.06	26.06
stefan	21.37	21.22	22.24	22.36	22.81	23.18	23.35	22.82	23.68
mobile	24.32	24.54	24.55	24.52	24.54	24.6	24.56	24.64	24.67
paris	31.65	31.71	31.65	31.64	31.67	31.82	31.75	31.88	31.9
stefan	25.08	24.56	25.44	25.43	25.59	25.81	25.75	25.91	26.02
bus	23.07	22.24	23.17	22.76	23.66	24.09	24.81	25	25.17
waterfall	34.57	34.57	34.57	34.57	34.57	34.57	34.57	34.57	34.57
highway	34.44	34.5	34.25	34.46	34.47	35.09	34.99	35.76	35.78
beauty	34.56	35.4	35.56	35.41	35.62	35.77	35.69	35.72	36.28
readySetGo	32.89	31.58	33.56	33.85	33.88	34.2	34.12	34.42	34.68
yachtRide	32.56	32.84	33.32	33.27	33.48	33.56	33.61	33.8	33.91
jockey	25.31	29.18	36.08	36.23	36.97	36.98	37.16	30.71	37.91
pedestrian	26.98	31.39	32.57	32.58	33.5	33.1	33.55	30.77	33.69
riverbed	21.79	23.29	24.12	23.62	24.22	24.39	24.33	24.17	25.75
average	30.518	30.939	31.636	31.613	31.832	31.98	32.014	31.648	32.349

algorithm with the other block matching methods in terms of quality (PSNR) for block size of 16 × 16 pixels and in Table 4 in terms of speed (average search points per block). In order to better study the detailed behaviour of the proposed algorithm over other algorithms, the PSNR per frame and average search points per block for the first 100 frames of the bus sequence, are plotted in Fig. 15 and Fig. 16, respectively.

Table 4: Performance comparison in terms of average search points for 16×16 pixel blocks.

	TSS	DS	ARPS	APS	DISP	TZS	Proposed	FS(±15)	FS(±30)
coastguard	21.61	13.08	7.95	4.57	4.81	25.73	3.91	782.21	2714.8
mother	21.52	11.89	6.42	2.09	5.06	27.04	1.82	782.21	2714.8
silent	21.49	11.98	6.56	2.92	5.2	27.17	2.47	782.21	2714.8
carphone	21.63	13.26	7.96	5.6	6.06	29.11	4.54	782.21	2714.8
foreman	21.65	14.95	8.44	5.59	6.11	28.49	4.76	782.21	2714.8
akiyo	21.48	11.43	5.87	1.32	4.61	26.08	1.25	782.21	2714.8
mobile	21.48	11.44	6.10	4.35	4.64	26.04	4.56	782.21	2714.8
stefan	23.15	19.14	10.77	8.01	7.40	34.06	6.25	859.45	3142.6
mobile	23.23	13.32	8.21	5.02	5.15	27.86	4.81	869.33	3198.3
paris	23.21	12.65	6.51	2.64	5.19	28.59	2.32	869.33	3198.3
stefan	23.30	17.2	9.24	5.97	6.47	31.78	5.3	869.33	3198.3
bus	23.57	20.36	10.71	8.49	7.13	33.89	6.45	869.33	3198.3
waterfall	23.21	12.27	6.26	4.35	4.91	28.01	3.94	869.33	3198.3
highway	23.44	15.34	8.92	7.24	7.08	33.98	6.26	869.33	3198.3
beauty	24.71	19.95	11.77	13.21	10.39	49.41	10.88	939.48	3597.1
readySetGo	24.66	23.23	11.79	4.74	8.49	37.02	4.71	939.48	3597.1
yachtRide	24.65	26.37	12.26	5.88	8.63	37.76	5.15	939.48	3597.1
jockey	24.72	36.49	13.46	5.00	8.68	36.35	4.74	939.48	3597.1
pedestrian	24.61	29.15	14.32	11.54	10.45	44.44	9.33	939.48	3597.1
riverbed	24.65	36.26	24.77	29.27	24.49	71.95	24.61	939.48	3597.1
average	23.10	18.49	9.91	6.90	7.55	34.24	5.90	859.39	3145.90

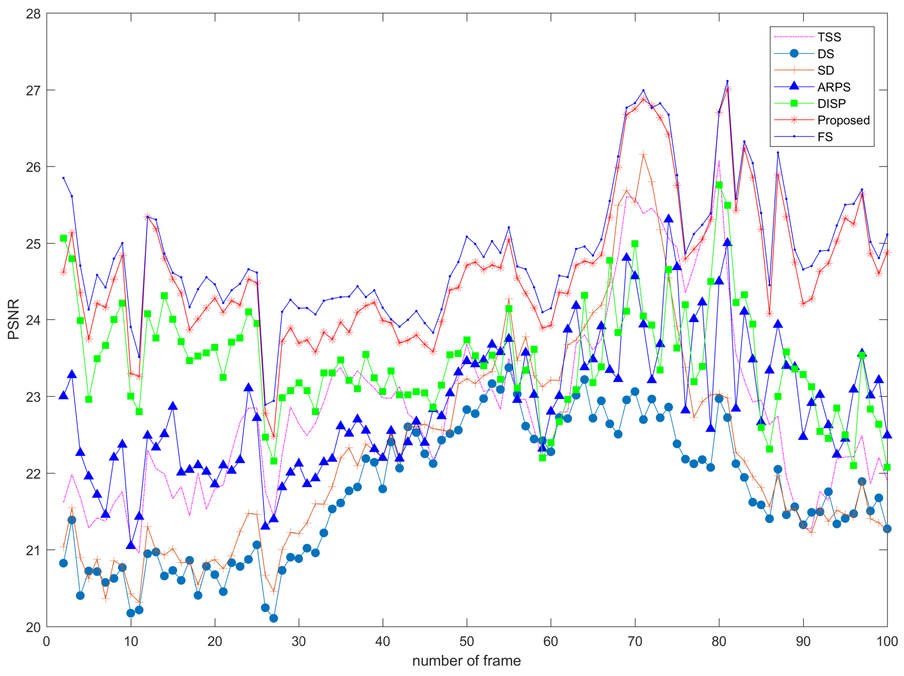


Fig. 15: PSNR of different methods for 100 frames of the bus sequence.

As these Tables indicate while the motion compensation efficiency of the proposed algorithm is as good as TZSearch, its estimation time is much faster.

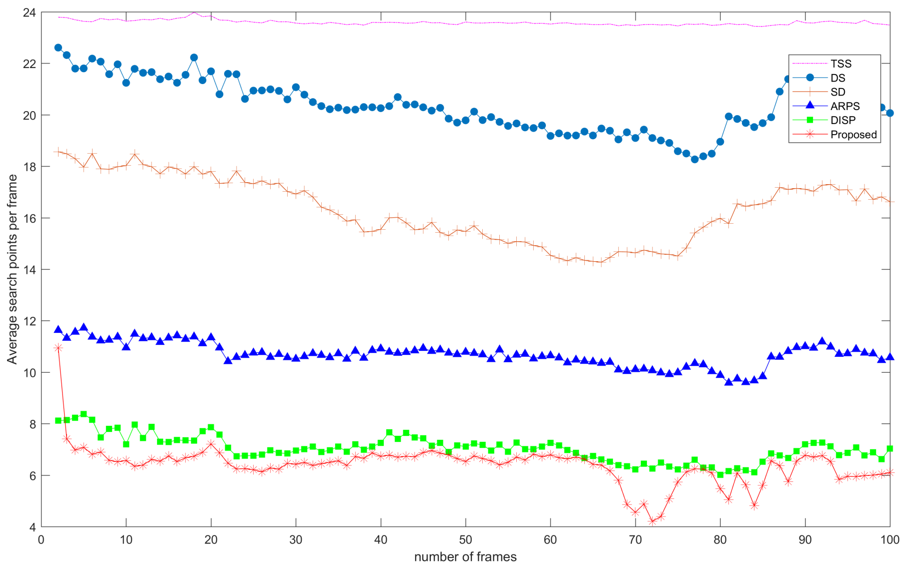


Fig. 16: Average search points of different methods for 100 frames of the bus sequence.

The reason for such property is as follows: On the search speed, since the proposed algorithm uses all the surrounding motion vectors, and the final motion vectors is closer to one of them, then it needs fewer points to find it. The number of the points is variable from one to up 8. Whereas, the TZSearch, searches at least 29 points in the initial search grid. Also, when the best-matched point in the initial search grid is far from the center, TZSearch does raster search which adds to the complexity. In addition, refinement can be stopped with 4 points for the proposed method while TZSearch requires more refinement steps and hence its search time is increased.

5 Conclusion

In this paper, a fast motion estimation algorithm has been proposed which divides the current frames into black and white blocks like a chessboard pattern. Using this definition, a larger number of predictors are available to define an initial dynamic search pattern. First, early motion vectors are obtained for the black blocks and then the white blocks are searched, and final motion vectors are obtained for them. Then the motion vectors of white neighbouring blocks are added to the initial search pattern of the current black block as predictors increasing the number of predictors for the black blocks. The performance of the proposed method is compared against the other known methods in terms of PSNR. The experimental results show that the proposed algorithm is closer to FS than the other methods. Moreover, the proposed algorithm has the best compression efficiency, and has the fastest response for calculating the best motion vector.

References

1. Hm software 16.19. URL https://hevc.hhi.fraunhofer.de/trac/hevc/browser/tags/HM-16.19
2. Al-Mualla, M., Canagarajah, N.C., Bull, D.R., Canagarajah, C.N.: Video Coding for Mobile Communications: Efficiency, Complexity, and Resillience. Academic Press, Inc., Orlando, FL, USA (2002)
3. Amirpour, H., Mousavinia, A.: A dynamic search pattern motion estimation algorithm using prioritized motion vectors. Signal, Image and Video Processing **10**(8), 1393–1400 (2016). DOI 10.1007/s11760-016-0906-5. URL https://doi.org/10.1007/s11760-016-0906-5
4. Amirpour, H., Mousavinia, A., shamsi, N.: Predictive three step search (PTSS) algorithm for motion estimation. In: Machine Vision and Image Processing (MVIP), 2013 8th Iranian (2013)
5. Ghanbari, M.: The cross-search algorithm for motion estimation (image coding). IEEE Transactions on Communications **38**(7), 950–953 (1990). DOI 10.1109/26.57512
6. Jakubowski, M., Pastuszak, G.: Block-based motion estimation algorithms — a survey. Opto-Electronics Review **21**(1), 86–102 (2013). DOI 10.2478/s11772-013-0071-0. URL https://doi.org/10.2478/s11772-013-0071-0
7. Kerfa, D., Belbachir, M.F.: Star diamond: an efficient algorithm for fast block matching motion estimation in H264/AVC video codec. Multimedia Tools and Applications **75**(6), 3161–3175 (2016)
8. Khemiri, R., Bahri, N., Belghith, F., Sayadi, F.E., Atri, M., Masmoudi, N.: Fast motion estimation for HEVC video coding. In: 2016 International Image Processing, Applications and Systems (IPAS), pp. 1–4 (2016). DOI 10.1109/IPAS.2016.7880120
9. Kim, B.G., Goswami, K.: Basic Prediction Techniques in Modern Video Coding Standards, 1st edn. Springer Publishing Company, Incorporated (2016)
10. Koga T. Iinuma K., H.A., Y., I.: Motion compensated interframe coding for video conferencing. In: Proceedings of NTC81, p. G5.3.1G5.3.5. New Orleans (1981)
11. Kuo, T.Y., Kuo, C..J.: Fast overlapped block motion compensation with checkerboard block partitioning. IEEE Transactions on Circuits and Systems for Video Technology **8**(6), 705–712 (1998). DOI 10.1109/76.728412
12. Lin, L., Wey, I.C., Ding, J.H.: Fast predictive motion estimation algorithm with adaptive search mode based on motion type classification. Signal, Image and Video Processing **10**(1), 171–180 (2016). DOI 10.1007/s11760-014-0723-7. URL https://doi.org/10.1007/s11760-014-0723-7
13. Lin, Z., Zou, Y.: Long-rood motion estimation algorithm based on starting search point prediction. In: Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on, pp. 1327–1331 (2009)
14. Luo, J., Yang, X., Liu, L.: A fast motion estimation algorithm based on adaptive pattern and search priority. Multimedia Tools and Applications **74**(24), 11821–11836 (2015). DOI 10.1007/s11042-014-2280-z. URL https://doi.org/10.1007/s11042-014-2280-z
15. Nie, Y., Ma, K.K.: Adaptive rood pattern search for fast block-matching motion estimation. Image Processing, IEEE Transactions on **11**(12), 1442–1449 (2002)
16. Pan, Z., Ku, W., Wang, Y.: Dynamic initial search pattern defined on cartesian product of neighboring motion vectors for fast block-based motion estimation. Multimedia Tools and Applications (2017)
17. Pan, Z., Zhang, R., Ku, W., Wang, Y.: Adaptive pattern selection strategy for diamond search algorithm in fast motion estimation. Multimedia Tools and Applications (2018). DOI 10.1007/s11042-018-6353-2. URL https://doi.org/10.1007/s11042-018-6353-2
18. Po, L.M., Ma, W.C.: A novel four-step search algorithm for fast block motion estimation. Circuits and Systems for Video Technology, IEEE Transactions on **6**(3), 313–317 (1996)

19. Puri, A., Hang, H.., Schilling, D.: An efficient block-matching algorithm for motion-compensated coding. In: ICASSP '87. IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. 12, pp. 1063–1066 (1987). DOI 10.1109/ICASSP.1987.1169777
20. Purnachand, N., Alves, L.N., Navarro, A.: Fast motion estimation algorithm for HEVC. In: 2012 IEEE Second International Conference on Consumer Electronics - Berlin (ICCE-Berlin), pp. 34–37 (2012). DOI 10.1109/ICCE-Berlin.2012.6336494
21. Purnachand, N., Alves, L.N., Navarro, A.: Improvements to TZ search motion estimation algorithm for multiview video coding. In: 2012 19th International Conference on Systems, Signals and Image Processing (IWSSIP), pp. 388–391 (2012)
22. Purwar, R.K.: Enhanced dynamic pattern search algorithm with weighted search points for fast motion estimation. Signal, Image and Video Processing **11**(6), 1001–1007 (2017)
23. Purwar, R.K., Rajpal, N.: A fast block motion estimation algorithm using dynamic pattern search. Signal, Image and Video Processing **7**(1), 151–161 (2013)
24. Sullivan, G.J., Ohm, J.R., Han, W.J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Cir. and Sys. for Video Technol. **22**(12), 1649–1668 (2012)
25. Tourapis, A.M., Au, O.C., Liou, M.L.: Predictive motion vector field adaptive search technique (PMVFAST) – enhancing block based motion estimation. In: IN THE OPTIMIZATION MODEL 1.0, IN ISO/IEC JTC1/SC29/WG11 MPEG2000/M6194, pp. 883–892. Noordwijkerhout, NL (2001)
26. Zhu, C., Lin, X., Chau, L.P.: Hexagon-based search pattern for fast block motion estimation. Circuits and Systems for Video Technology, IEEE Transactions on **12**(5), 349–355 (2002)
27. Zhu, S., Ma, K.K.: A new diamond search algorithm for fast block-matching motion estimation. Trans. Img. Proc. **9**(2), 287–290 (2000)