Microsyst Technol (2018) 24:2529–2537
https://doi.org/10.1007/s00542-017-3625-0

TECHNICAL PAPER

Performance analysis of diamond search algorithm over full search algorithm

Rahul Priyadarshi[1] · Surender Kumar Soni[1] · Rampal Bhadu[1] · Vijay Nath[2]

Received: 3 September 2017 / Accepted: 6 November 2017 / Published online: 10 November 2017

Abstract Motion estimation is a progression used to estimate motion vectors between two or more images with a high degree of temporal redundancy. It is commonly used in video compression to attain high compression ratios as well as used in several applications for object tracking. In this paper a novel approach for diamond search algorithm has been recommended to overcome the problem encountered by several existing block matching algorithms especially with full search algorithm in reference of peak signal-to-noise ratio, required number of examine or search points as well as computational complexity. Simulation results reflect that recommended algorithm acting well compared to all existing algorithms. Experimentally 88–99% of the motion vectors are found inside the circle which has radius of 3-pixel unit and fixed on the place of zero motion. The proposed algorithm is used to implement various standards examples such as MPEG1 and MPEG4.

✉ Rahul Priyadarshi
rahul.glorious91@gmail.com

Surender Kumar Soni
surender.soni@gmail.com

Rampal Bhadu
errampal.bhadu@gmail.com

Vijay Nath
vijaynath@bitmesra.ac.in

[1] Department of Electronics and Communication Engineering, National Institute of Technology, Hamirpur, HP, India

[2] Department of Electronics and Communication Engineering, Birla Institute of Technology Mesra, Ranchi, JH, India

1 Introduction

Motion estimation is most vital difficulties for development of video coding applications. Between all motion estimation methodologies, block matching (BM) algorithm is the utmost standard techniques because of efficiency and effortlessness in case of both software as well as hardware executions (Barjatya 2004; Feghali 2005). BMA methodologies take responsibility that measure of pixels inside a definite area of current frame can be demonstrated as a paraphrase of pixels confined in the earlier frame. So in this process, motion vector is found by lessening sum of absolute differences (SAD) or mean absolute difference (MAD) created by the micro-block of the existing frame over a determined examine window from earlier frame. The SAD estimation is computationally costly and exemplifies the overwhelming task in BMA progression. The most popular BMA technique is full search (FS) algorithm (Lin and Tai 1997) that finds most precise motion vector, computing comprehensively SAD values for most of the element of examine window. In recent years several fast BMAs have been recommended to diminish the number of SAD tasks by computing simply a stable subsection of search positions at the expense of poor accurateness. A general mechanism of block-matching algorithm is shown in Fig. 1. There are several video compression criterions have been suggested for different applications such as H.261, MPEG-1 and MPEG-2 (Sullivan and Wiegand 2005; Sullivan et al. 2012). Generally it has been seen that most of the multimedia data are constituted by video data (Wiegand 2003). Coding of video is also very important for the usage of limited amount of bandwidth as well as storage medium. High quantity of compression is enabled by temporal correlation which generally exists between successive image frames (Pal 2015). To exploit temporal

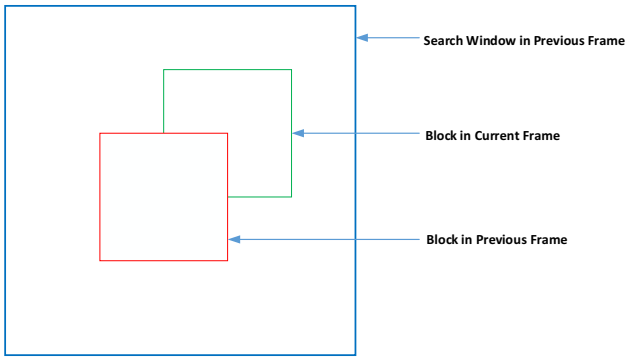

Fig. 1 Mechanism of block-matching algorithm

correlation, motion estimation is an important tool (Luo and Konofagou 2010; Dikbas and Altunbasak 2013).

Block based motion estimation with non-overlapping rectangular blocks are widely used in numerous video coding standards (Bergen et al. 1992; Karunakar and Pai 2009). In this case, frames of image are divided into several non-overlapping blocks and the match who has been found to be best, searched within a pre-defined search range using all possible positions for each block (Jung and Ye 2010). Although full FS algorithm which is also called as exhaustive search (ES) algorithm provides optimal quality but pointedly it suffers from computational load (Brünig and Niehsen 2001). Various motion estimation methods exist but block matching is used for all most all type of international standards video compression such as MPEG1, MPEG2, MPEG4 and H.264 (Ertürk 2007; Luo et al. 2008). A general video compression flow process has been shown in Fig. 2. Motion Estimation for Block matching is the procedure for so many compensated video coding standards where temporal redundancies between consecutive frames are efficiently removed (Luo et al. 2008). BMA begins by dividing a reference image up into equal size

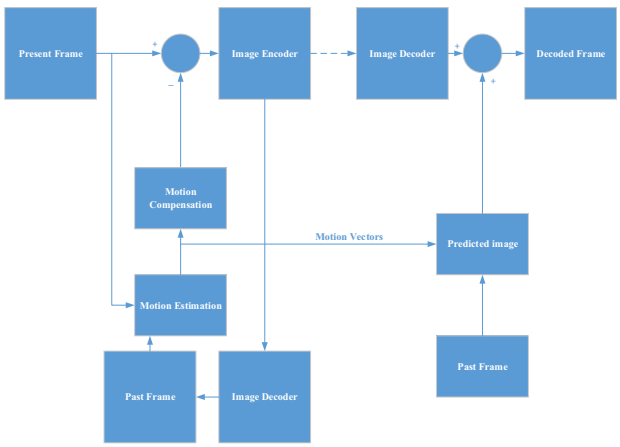

Fig. 2 Video compression flow progression

squares blocks. Then each block in the reference image is compared to a block in current image. Blocks in current image which are centered at different locations called search points inside a designated searching area of the current image called searching window (Farsiu et al. 2004). The distortion is calculated between two blocks centered at each designated search point and the point which has the lowest distortion is the center for the reference block in current image. Distortion can be calculated by means of number of different ways including mean square error (MSE), MAD as well as SAD technique. Selecting number of search points and pixels which become search points is a main difference for many of the block matching algorithms (Gao et al. 2000).

As we know most accurate and most expensive algorithm is FS algorithm which checks every point inside the search window. Search patterns in block matching motion estimation with altered size or shape and also the individualities of motion vector which are centrally biased have a huge influence on examining quality and speed of performance. Many other algorithms have tried to enhance computational complexity level of FS algorithm despite the fact maintaining a reasonable level of accuracy. Some such algorithms are three step search (TSS), four step search (FSS), diamond search (DS) and logarithmic search (LS) (Jing and Chau 2004; Po and Ma 1996a) but most of the algorithms have poor accuracy as well as computational complexity is too large.

2 Block matching algorithms

2.1 Three step search algorithm

Three step search algorithm (TSSA), suggested in (Li et al. 1994), is a satisfactory search mechanism. It turns out to be very standard algorithm because of simple and performance is robust and optimum. It examines for the finest motion vector to make search pattern precise. The TSSA is process which confines total number of inspection points in examines area. Also TSSA uses a homogeneously circulated search pattern in every single stage and for that reason displays straightforwardness and uniformity. One major difficult that happens with the TSSA, uses a homogeneously distributed inspection point shape in 1st phase which further turn out to be ineffective for insignificant motion estimation. Figure 3 shows the TSSA algorithm with selection of block in different stages.

There are also various step search algorithm related to TSSA have been proposed by authors in Sun et al. (2009) and Kim and Choi (1998), but most of the BMAs have disadvantages in terms of accuracy, complexity and as well as in performances. Authors in Jing and Chau (2004) and

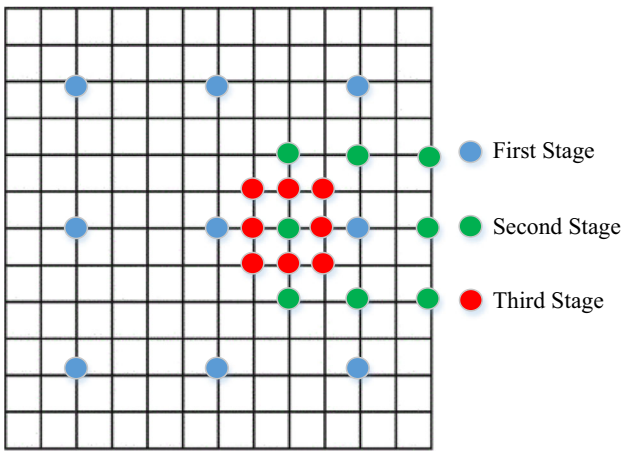

Fig. 3 Block search pattern in TSSA

Kim and Choi (1998) proposed new three step search algorithm by changing some search pattern which reflects better results in comparison of exiting TSAA but it is also not up to the mark for practical implementation because of its complexity.

2.2 Four step search algorithm

Authors in Po and Ma (1996a 1996b) first proposed four step search (4SS) algorithm which is basically based on actual world image agreement's distinguishing of mid-point-influenced motion and half way stop methods. This algorithm also activities the midpoint influenced appearances of the actual world video orders by means of using a lesser preliminary step size related with TSSA. The preliminary step size is 4th of determined motion dislocation. By reason of lesser preliminary step size, 4SS algorithm wants four examining steps to touch the periphery of a examine window. A general 4SS process has been shown in Fig. 4 where all the four stages are shown to find the pattern.

There are also various step search algorithm related to 4SS have been projected by authors in Nisar and Choi (2000), but most of the BMAs have disadvantages in terms of accuracy, complexity and as well as in performances.

2.3 Full step search algorithm

FS algorithm method counterparts all probable displace contender block inside the search area in reference frame with the purpose of discover the block with minimum distortion (Brünig and Niehsen 2001). So FS algorithm has large motion and more searching point to do the blocks matching and thus the computations may be too complete. A general FS algorithm search pattern is shown in Fig. 5. FS algorithm is frequently used in hardware execution of motion estimation for the reason that of its consistency and its simplicity ideal result. The commonly used metric is to find the finest match for FS algorithm in hardware is the MAD technique (Cai and David Pan 2012). To find the minimum MAD from all candidate blocks, performing search iteration for each candidate block. The performance of existing FPGA technology permits to new designs to resolve the motion estimation problem.

This work defines a study about recent motion estimation designs (Karunakar and Pai 2009). FS algorithm is a computationally expensive BMA of all existing algorithm. Cost function in this FS algorithm is calculated at each and every possible place in search or examine window. As an outcome, it finds finest likely match and gives the maximum PSNR amongst any BMA. Fast block matching algorithm attempts to attain similar PSNR doing as slight calculation as probable. The noticeable shortcoming of FS

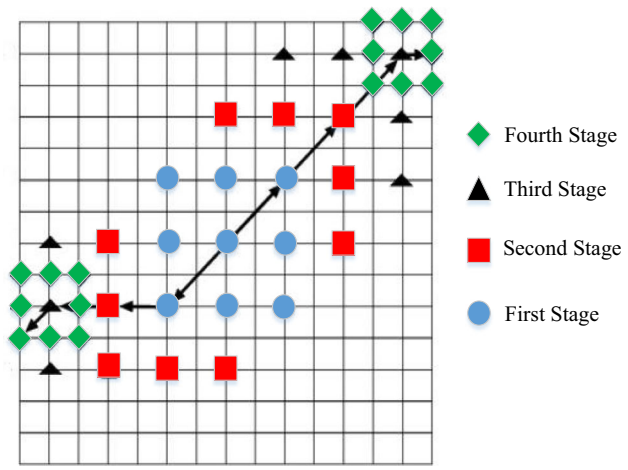

Fig. 4 Block search pattern in 4SS algorithm

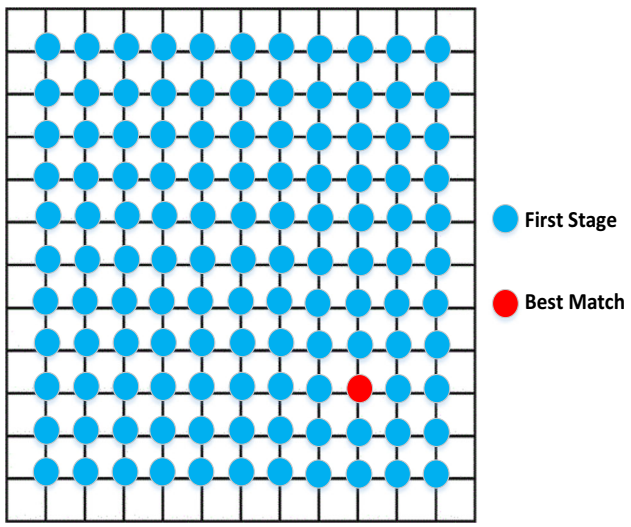

Fig. 5 Block Search pattern in FS algorithm

is that number of computations required is very high since a contender block is matched with all reference blocks in search window thus making, processing slower (Ouyang et al. 2012; Kim 2005).

A unique diamond search method which is planned to be realized in this paper are totally different from exiting all BM algorithm and we compare the performance with FS algorithm as well as with some other existing BM algorithm in reference of total number of essential search points, PSNR and computational complication (Yaakob et al. 2013; Ouyang et al. 2012).

3 Evaluation metrics

An evaluation criterion for matching the macro-block in frames with a different block is on the basis of cost function. Most standard in reference of computational expense is MAD as well as MSE. MAD can be calculated by Eq. (1):

$$MAD = \frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left|C_{ij} - R_{ij}\right|. \tag{1}$$

MSE can be calculated by using Eq. (2):

$$MSE = \frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left(\left|C_{ij} - R_{ij}\right|\right)^2, \tag{2}$$

where N is size of the macro-block, C_{ij} is pixels compared in the Present macro-block and R_{ij} is pixels compared in reference macro-block.

Goodness of a match is also found by SAD criteria (Vassiliadis et al. 1998; Niitsuma and Maruyama 2010). It is similar to MSE criteria but as an alternative of totaling the value of square differences, absolute differences are added which is given by Eq. (3):

$$SAD = \frac{1}{N^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left|C_{ij} - R_{ij}\right|. \tag{3}$$

Motion compensated image which is generated using motion vectors as well as macro-blocks from reference frame is categorized as PSNR which is calculated by Eq. (4):

$$PSNR = 10\log_{10}\left(\frac{Peak\ to\ Peak\ Value}{MSE}\right)^2. \tag{4}$$

4 Proposed algorithm

A novel diamond search (DS) algorithm has been implemented. There is pre-assumption in proposed algorithm is that motion vector should must be centrally biased. There are 25 crosses which are showing all different–different checking points inside the circle which has been shown in Fig. 6. Proposed DS algorithm basically using three search patterns and these are,

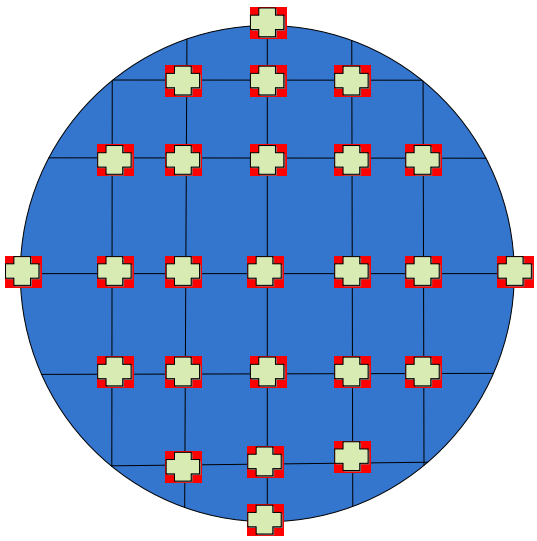

Fig. 6 Search pattern of proposed algorithm

- Small diamond search pattern.
- Medium diamond search pattern.
- Large diamond search pattern.

Small diamond search pattern defined with 5 cross points with one search point is at center and another 4 point is around center cross point with length value 1 which is shown in Fig. 7. Small diamond search pattern is last stage in proposed DS algorithm to find the best matches in the current frames. Medium diamond search pattern defined with 9 cross points with one search point is at center, 4 point is around center cross point with length value $\sqrt{2}$ and another 4 points is around center with length value 2 which is shown in Fig. 8. Large diamond search pattern defined with 13 cross points with one search point is at center, 8 point is around center cross point with length value 5 with different–different arrangement and another 4 points is located at extreme end of diameter of circle with length value 3 which is displayed in Fig. 9.

The proposed algorithm runs as follows:

Step 1: Begin with search position at midpoint. Search 24 positions about origin using diamond search pattern. Select between 25 positions that are searched, one which has smallest cost function. If smallest weightiness is achieved at midpoint for search window then follow the Step 2 and if smallest weightiness is achieved at one of the 24 positions other than midpoint then fix new origin to this position and start repeating Step 1.

Step 2: Fixe new search origin and start search 8 positions. Select between 9 positions that are searched, one with smallest cost function. If smallest weightiness is

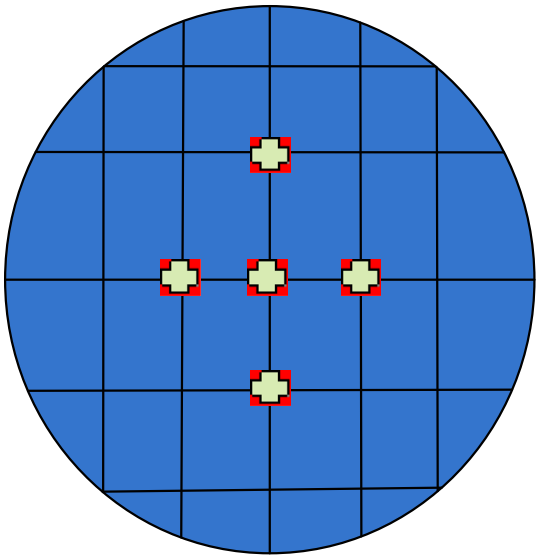

Fig. 7 Small diamond search pattern

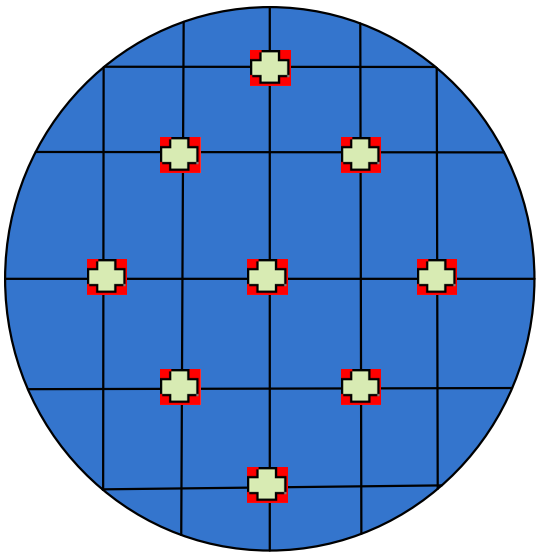

Fig. 8 Medium diamond search pattern

achieved at midpoint for search window then follow the Step 3 and if smallest weightiness is achieved at one of the 8 positions other than midpoint and fix new origin to this position and starting repeating Step 2.

Step 3: Fix new search origin and repeat search process to evaluate position with smallest weightiness. Choose position with smallest weightiness as motion vector.

5 Simulation results and discussion

Simulation of proposed algorithm is done using MATLAB tool and also results have been compared to the some existing algorithms. The below plots reflects the average

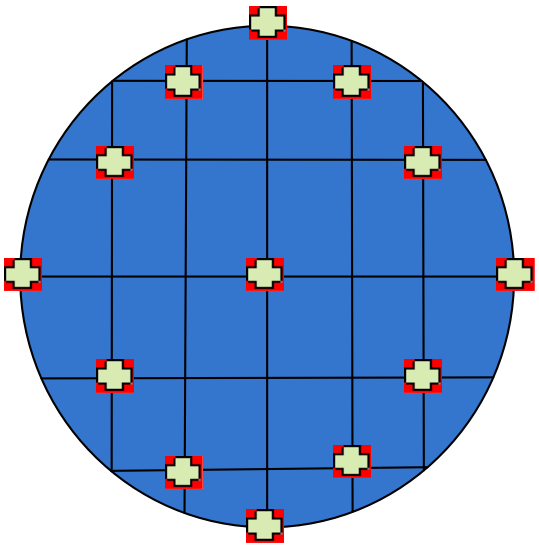

Fig. 9 Large diamond search pattern

Table 1 Simulation parameters

Parameters	Values
Number of frames	33
Search parameters	7
Size of macro-block	16 × 16
Error function	MAD criterion

number of searches which is essential per macro block for video sequence for different–different algorithms. Simulation parameters with its value are listed in Table 1.

5.1 Three step search algorithm

Figure 10 represents the plot for computational level (search per macro block) for TSS algorithm for frame number 0–33. For better understanding of this plot Table 2 shows the number of computations needed for TSS algorithm.

Figure 11 represents plot for PSNR values for TSS algorithm for frame number 0–33. For better understanding of this plot Table 3 shows the PSNR values for different frame numbers.

5.2 Four step search algorithm

Figure 12 represents the plot for computational level (search per macro block) for 4SS algorithm for frame number 0–33. For better understanding of this plot Table 4 shows the number of computations needed for 4SS algorithm.

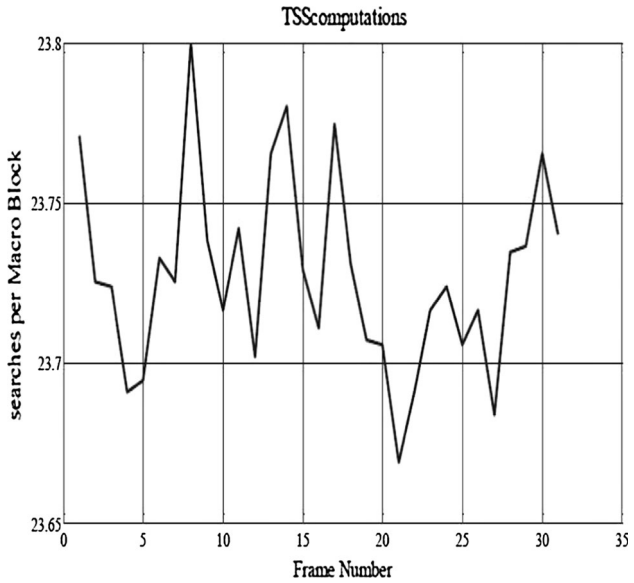

Fig. 10 Computation levels for TSS algorithm

Table 2 Computations needed for TSS algorithm

Frame no.	No. of computations needed (alternate value for frame no. is provided)				
1–10	23.77	23.72	23.69	23.73	23.74
11–20	23.74	23.77	23.73	23.77	23.71
21–30	23.67	23.72	23.71	23.68	23.74

Figure 13 represents plot for PSNR values for 4SS algorithm for frame number 0–33. For better understanding of this plot Table 5 shows the PSNR values for different frame numbers.

The PSNR result of 4SS algorithm is improved compared to the TSS algorithm with the significant amount (see in Figs. 11, 13). TSS takes average value around 23 searches for every macro-block while 4SS takes around 19 searches per macro block and hence drops the search per macro block (see in Figs. 10, 12) which reflects that performance is also improved and computational complexity has been decreased up to significant level.

5.3 Full search algorithm

Figure 14 represents the plot for computational level (search per macro block) for FS algorithm for frame number 0–33. For better understanding of this plot Table 6 shows the number of computations needed for FS algorithm.

Figure 15 represents plot for PSNR values for FS algorithm for frame number 0–33. For better understanding of this plot Table 7 shows the PSNR values for different frame numbers.

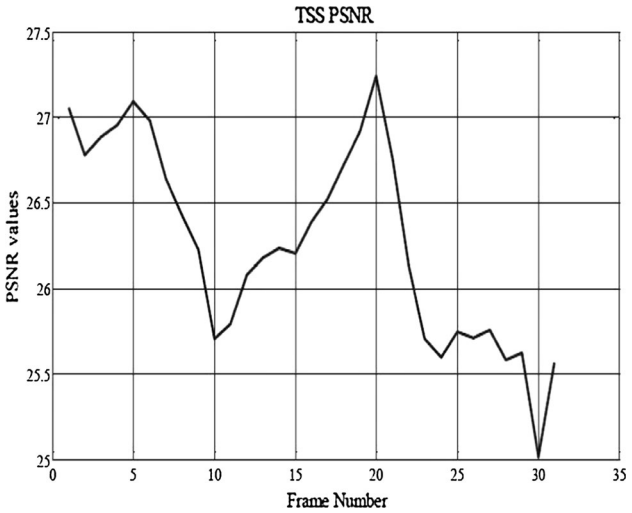

Fig. 11 PSNR levels for TSS algorithm

Table 3 PSNR values for TSS algorithm

Frame no.	PSNR values (alternate value for frame no. is provided)				
1–10	27.06	26.89	26.98	26.43	25.70
11–20	25.80	26.18	26.39	26.73	27.24
21–30	26.75	25.70	25.71	25.58	25.01

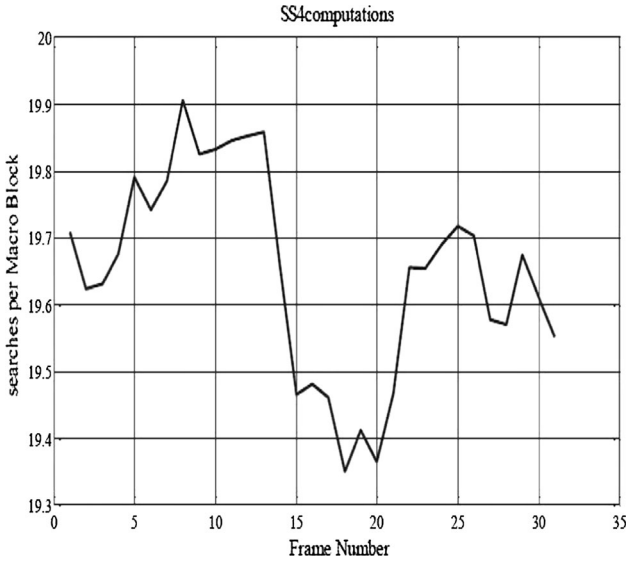

Fig. 12 Computation levels for 4SS algorithm

5.4 Proposed diamond search algorithm

Figure 16 represents the plot for computational level (search per macro block) for proposed DS algorithm for frame number 0–33. For better understanding of this plot

Table 4 Computations needed for 4SS algorithm

Frame no.	No. of computations needed (alternate value for frame no. is provided)				
1–10	19.71	19.63	19.79	19.79	19.83
11–20	19.85	19.86	19.47	19.46	19.41
21–30	19.46	19.65	19.72	19.59	19.67

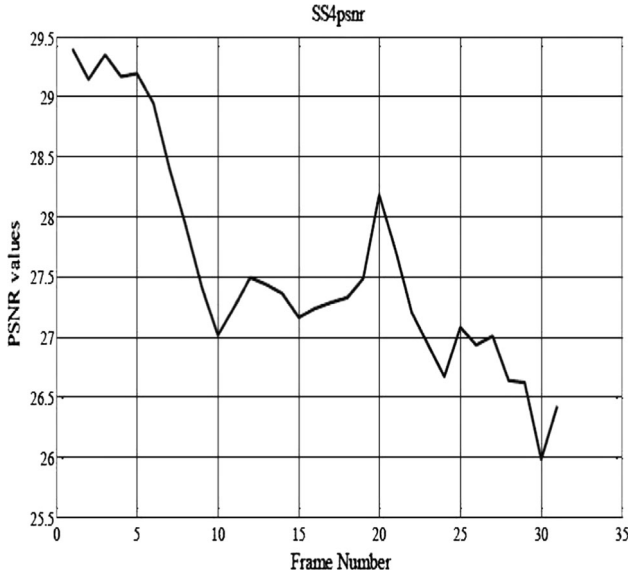

Fig. 13 PSNR levels for 4SS algorithm

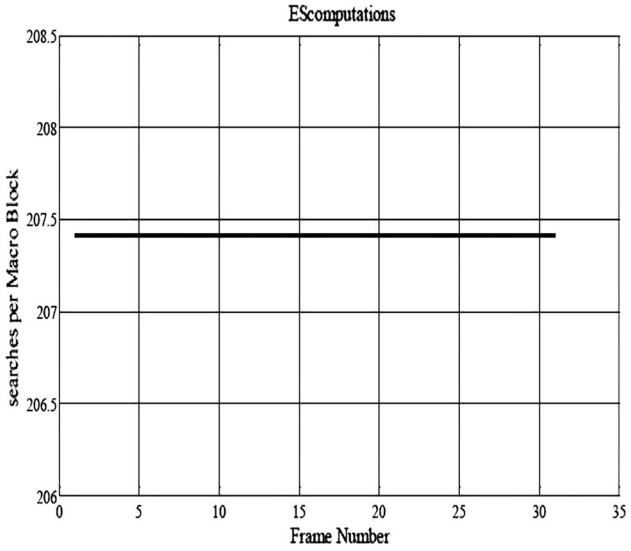

Fig. 14 Computation levels for FS algorithm

Table 5 PSNR values for 4SS algorithm

Frame no.	PSNR values (alternate value for frame no. is provided)				
1–10	29.40	29.14	29.17	28.94	27.94
11–20	27.02	27.49	27.36	27.24	27.33
21–30	28.18	27.20	26.67	26.93	26.64

Table 6 Computations needed for FS algorithm

Frame no.	No. of computations needed (alternate value for frame no. is provided)				
1–10	207.41	207.41	207.41	207.41	207.41
11–20	207.41	207.41	207.41	207.41	207.41
21–30	207.41	207.41	207.41	207.41	207.41

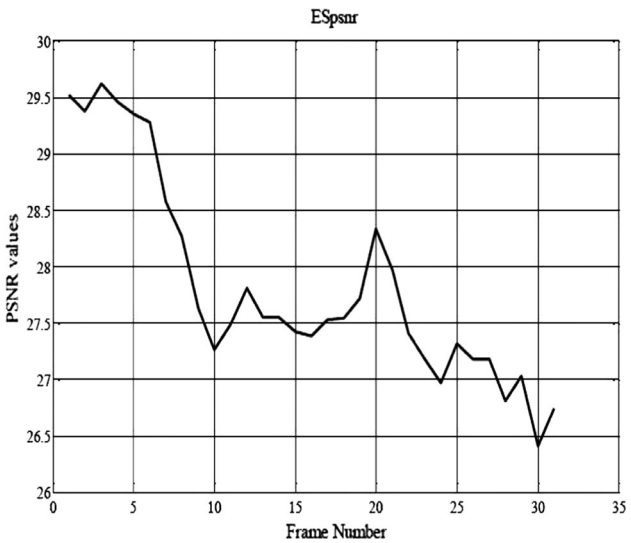

Fig. 15 PSNR levels for FS algorithm

Table 8 shows the number of computations needed for TSS algorithm

Figure 17 represents plot for PSNR values for DS algorithm for frame number 0–33. For better understanding of this plot Table 9 shows the PSNR values for different frame numbers.

The PSNR result of proposed DS algorithm is almost same (average 27) as the PSNR results of FS algorithm (see in Figs. 15, 17) and far better than TSS and similar to 4SS algorithm but main problem with 4SS algorithm is that it not searches all the macro-blocks in specified area which leads to sometimes inaccuracy in performances while DS algorithm follow one of the best hierarchy to search the macro-blocks to find the best matches in the specified area

Table 7 PSNR values for FS algorithm

Frame no.	PSNR values (alternate value for frame no. is provided)				
1–10	29.52	29.62	29.35	28.57	27.63
11–20	27.48	27.55	27.42	27.53	27.71
21–30	27.97	27.18	27.31	27.18	27.03

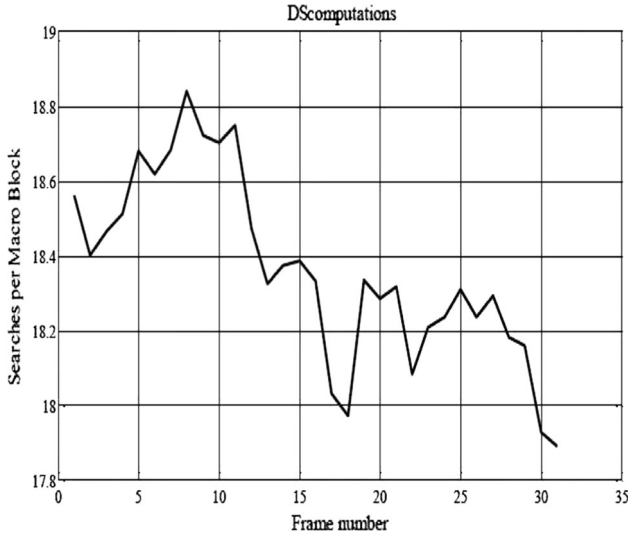

Fig. 16 Computation levels for proposed DS algorithm

Table 8 Computations needed for DS algorithm

Frame no.	No. of computations needed (alternate value for frame no. is provided)				
1–10	18.56	18.47	18.68	18.68	18.72
11–20	18.75	18.33	18.39	18.03	18.34
21–30	18.32	18.21	18.31	18.29	18.16

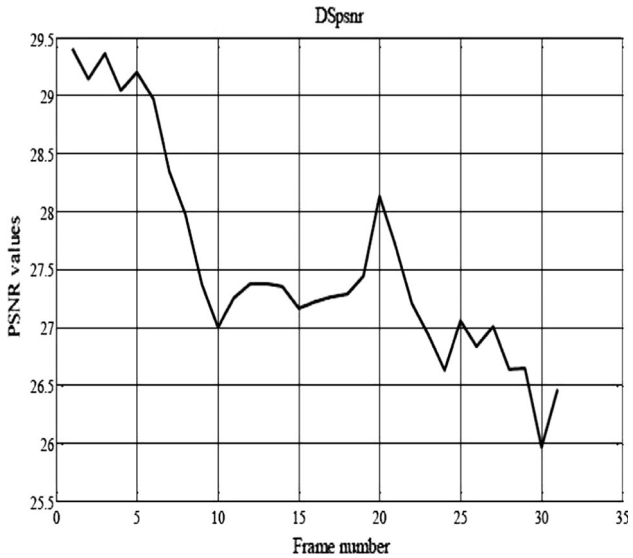

Fig. 17 PSNR levels for DS algorithm

Table 9 PSNR values for proposed DS algorithm

Frame no.	PSNR values (alternate value for frame no. is provided)				
1–10	29.40	239.37	29.12	28.35	27.37
11–20	27.25	27.38	27.16	27.26	27.44
21–30	27.69	27.71	27.06	27.01	26.64

and leads to the almost same performance compared to the FS algorithm. FS takes average value around 207 searches for every macro -block while DS takes around 18 searches for every macro- block and hence drops number by the huge margin (see in Figs. 14, 16) which reflects that performance is almost similar to FS algorithm and computational complexity has been decreased up to significant level. Experimentally it has been also found that 88–99% of the motion vectors are found inside the circle which has radius of 3-pixel unit and centered on location of zero motion.

6 Conclusion

The proposed DS algorithm outperforming well compared to the existing algorithms in reference of PSNR, required number of search points and computational complexity. Experimentally 88–99% of the motion vectors are found inside the circle which has radius of 3-pixel unit and pinpointed on place of zero motion. The proposed process is used to implement various standards examples such as MPEG1 and MPEG4.In future results can be also improved by considering various parameters such as matching criteria, outsized search region for large multifaceted motions, small search region for less multifaceted motions as well as bidirectional motion approximation. There is a further scope to develop new methods which will decrease complication of MPEG video coding.

References

Barjatya A (2004) Block matching algorithms for motion estimation. IEEE Trans Evol Comput 8(3):225–229

Bergen JR, Anandan P, Hanna KJ, Hingorani R (1992) Hierarchical model-based motion estimation. In: Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics), LNCS 588:237–252

Brünig M, Niehsen W (2001) Fast full-search block matching. IEEE Trans Circuits Syst Video Technol 11(2):241–247

Cai J, David Pan W (2012) On fast and accurate block-based motion estimation algorithms using particle swarm optimization. Inf Sci (Ny) 197:53–64

Dikbas S, Altunbasak Y (2013) Novel true-motion estimation algorithm and its application to motion-compensated temporal frame interpolation. IEEE Trans Image Process 22(8):2931–2945

Ertürk S (2007) A new perspective to block motion estimation for video compression: high-frequency component matching. IEEE Signal Process Lett 14(2):113–116

Farsiu S, Robinson MD, Elad M, Milanfar P (2004) Fast and robust multiframe super resolution. IEEE Trans Image Process 13(10):1327–1344

Feghali R (2005) Multi-frame simultaneous motion estimation and segmentation. IEEE Trans Consum Electron 51(1):245–248

Gao XQ, Duanmu CJ, Zou CR (2000) A multilevel successive elimination algorithm for block matching motion estimation. IEEE Trans Image Process 9(3):501–504

Jing X, Chau LP (2004) An efficient three-step search algorithm for block motion estimation. IEEE Trans. Multimed 6(3):435–438

Jung H, Ye JC (2010) Motion estimated and compensated compressed sensing dynamic magnetic resonance imaging: what we can learn from video compression techniques. Int J Imaging Syst Technol 20(2):81–98

Karunakar AK, Pai MMM (2009) Motion-compensated temporal filtering with optimized motion estimation. J Real-Time Image Process 4(4):329–338

Kim M (2005) A fast VLSI architecture for full-search variable block size motion estimation in MPEG-4 AVC/H.264. Proc. ASP-DAC 2005. Asia South Pacific Des Autom Conf 1:631–634

Kim JN, Choi TS (1998) A fast three-step search algorithm with minimum checking points using unimodal error surface assumption. IEEE Trans Consum Electron 44(3):638–648

Li R, Zeng B, Liou ML (1994) A new three-step search algorithm for block motion estimation. IEEE Trans Circuits Syst Video Technol 4(4):438–442

Lin YC, Tai SC (1997) Fast full-search block-matching algorithm for motion-compensated video compression. IEEE Trans Commun 45(5):527–531

Luo J, Konofagou E (2010) A fast normalized cross-correlation calculation method for motion estimation. IEEE Trans Ultrason Ferroelectr Freq Control 57(6):1347–1357

Luo J, Ahmad I, Liang Y, Swaminathan V (2008) Motion estimation for content adaptive video compression. IEEE Trans Circuits Syst Video Technol 18(7):900–909

Niitsuma H, Maruyama T (2010) Sum of absolute difference implementations for image processing on FPGAs. In: Proceedings—2010 international conference on field programmable logic and applications, FPL 2010, pp 167–170

Nisar H, Choi T-S (2000) Fast four step search algorithm using UESA and quadrant selection approach for motion estimation. In: Proceedings of SPIE—the international society for optical engineering, vol. 3974

Ouyang W, Tombari F, Mattoccia S, Di Stefano L, Cham WK (2012) Performance evaluation of full search equivalent Pattern matching algorithms. IEEE Trans Pattern Anal Mach Intell 34(1):127–143

Pal M (2015) An optimized block matching algorithm for motion estimation using logical image. Int Conf Comput Commun Autom ICCCA 2015:1138–1142

Po Lai-man, Ma Wing-chung (1996a) A novel four-step search algorithm for fast block motion estimation. IEEE Trans Circuits Syst Video Technol 6(3):313–317

Po L, Ma W (1996b) A novel four-step search algorithm for fast block motion estimation. Circuits Syst Video Technol 6(3):313–317

Sullivan GJ, Wiegand T (2005) Video compression-from concepts to the H.264/AVC standard. Proc IEEE 93(1):18–31

Sullivan GJ, Ohm JR, Han WJ, Wiegand T (2012) Overview of the high efficiency video coding (HEVC) standard. IEEE Trans Circuits Syst Video Technol 22(12):1649–1668

Sun NN, Fan C, Xia X (2009) An effective three-step search algorithm for motion estimation. In: ITME2009—proceedings 2009 IEEE international symposium on IT in medicine and education, pp 400–403

Vassiliadis S, Hakkennes EA, Wong JSSM, Pechanek GG (1998) The sum-absolute-difference motion estimation accelerator. In: Proceedings. 24th EUROMICRO Conf. (Cat. No.98EX204), vol. 2

Wiegand T (2003) Overview of the H.264/AVC video coding standard. Syst Video 13(7):560–576

Yaakob R, Aryanfar A, Halin AA, Sulaiman N (2013) A comparison of different block matching algorithms for motion estimation. Procedia Technol 11(Iceei):199–205