International Journal of Electronics Communication and Computer Engineering
Volume 6, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Efficient Search Points Reduction and Ranking of Motion Estimation Algorithms in Video Coding

Md. Salah Uddin Yusuf*, Monira Islam, Mohiuddin Ahmad
Department of Electrical and Electronic Engineering
Khulna University of Engineering & Technology, Bangladesh
*Email: ymdsalahu2@gmail.com

Abstract – **In this paper, the performance of commonly used Motion Estimation (ME) algorithms are analyzed in terms of search point reduction and ranking for hardware implementation. Since number of searching point is not a direct measure of hardware performance, we first define a cost function of accessing data in memory. Then, we make adjustments on the existing ME algorithms to minimize the problem of memory access by reordering the sequence of searching points, where the number of searching points and MAE/PSNR remains the same as the original algorithms. Finally, we calculate the memory cost improvement of each algorithm. Using the present analysis we see that the cost function of accessing data in memory can be saved from 68% to 80% and the number of search point can be saved from 87% to 94%. The experimental results showed that the ranking based on memory access cost for small motion video is $\boldsymbol{SCDS} > BBGDS > CDS > DS > 4\boldsymbol{SS} > 3\boldsymbol{SS} > FS$ and for high motion video the ranking is memory access cost is $\boldsymbol{BBGDS} > CDS > SCDS > 4\boldsymbol{SS} > DS > 3\boldsymbol{SS} > FS$ with lower computational complexity and similar video quality is maintained.**

Keywords – **H.264/AVC, Memory Access Cost, Motion Compensation, Motion Estimation, Search Algorithm, Search Point.**

I. INTRODUCTION

Motion Estimation has proven to be effective to exploit the temporal redundancy of video sequences and is therefore a central part of the ISO/IEC MPEG-1, MPEG-2, MPEG-4, CCITT H.261/ITU-TH.263 and H.264 video compression encoder algorithms [1]. Motion estimation [2] is used also for other applications than video encoding, like image stabilization, motion segmentation, and image analysis. However, the requirements for these applications differ significantly for video encoding algorithms: Motion vectors [3] have to reflect the real motion within the image sequence; otherwise the algorithms will not give the desired results. For video encoding the situation is different. Motion vectors are used to compensate the motion within the video sequence and only the remaining signal (prediction error) has to be encoded and transmitted. Therefore, the motion vectors have to be selected in order to minimize the prediction error and the number of bits required to code the prediction error. Motion estimation is in most cases based on a search scheme which tries to find the best matching position of a 16×16 macro block (MB) of the current frame with a 16×16 blocks within a predetermined or adaptive search range in the previous frame. The matching position relative to the original position is described as the motion vector which is (after subtraction of the predictor and variable length coding) transmitted in the bit stream to the video decoder.

Fast motion estimation (ME) algorithms [4] shown that they can save a lot of searching points, but they all based on an assumption that all data are accessed randomly, that means they assume the whole memory block is refreshed for each searching points and also the sequence of searching points does not accounted under this assumption. It is very different from the actual hardware environments that direct implementation of these algorithms may not be efficient.

There are two major factors for a ME algorithm affects the hardware to be efficient. One is memory access efficiency and another is pipeline efficiency [5]. For memory access, while using a simple architecture that the video block data is accessed row-by-row (or column-by-column); if the block distortions are measured on neighboring locations, it is not necessary to refresh the whole block thus save the bandwidth of the data bus. For instant, one step up, down, left, right movements are very efficient since it only requires replacing single row or column of memory instead of reloading the whole block. We can see that the memory access will be efficient if we keep the searching steps small. On the contrary, we should prevent diagonal move or random jump since these moves requires refreshing more data or almost the whole memory block.

For pipeline efficiency, if a fixed searching sequence can be defined, all video data can be queued up so the block distortion measure can keep working thus the hardware can be utilized. But for fast ME algorithms the further searching path always depends on distortions of the previous searched points [6]. For handling uncertain future searching point, prediction can be used to pre-fetch the data for next searching point. But when the prediction is wrong the penalty might be serious since the pipeline need to be cleared and data should be reloaded. If we want a fast ME algorithm to be pipeline friendly we should keep the number of branches to be as small as possible or make the branches to be predictable. In this paper the standard ME algorithm are analyzed in details and efficient techniques are introduced for ordering them in different kind of high and low motion video. The search point reduction and calculation of memory access cost will be helpful to minimized computational complexity and motion estimation time for further hardware implementation.

The rest of the paper organized with section II describes the memory access cost function. Section III gives the adjustment of memory access on different well-known

International Journal of Electronics Communication and Computer Engineering
Volume 6, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

motion estimation ME, section IV describes experimental set up, and conclusion is given in section V.

II. Memonry Access Cost Function

The definition of memory cost function is based on the following assumptions for simple hardware architecture [7]:

(i) Time of accessing row or column memory is the same.
(ii) No carry-in from previous motion estimation.
(iii) Single port memory is used.
(iv) The video is on a 2D array in the memory.

The memory cost function is defined as:

$$C_{mem}(x,y) = \min(M,N) + \sum_{i=2}^{n} c_{mem}(sp_i, sp_{i-1})$$

$$c_{mem}(sp_i, sp_j) = \begin{cases} d_{L1}(sp_i, sp_j) & \text{while } d_{L1}(sp_i, sp_j) < \min(M,N); \\ \min(M,N) & \text{else} \end{cases}$$

$$d_{L1}(sp_i, sp_j) = |x_i - x_j| + |y_i - y_j|$$

where, C_{mem} is the memory cost function for motion estimation at point (x,y), M, N is the macro block size, usually 16×16, c_{mem} is the memory cost for moving from searching point $sp_j$to sp_i, dL_1 is the City Block distance between the searching point sp_j and sp_i, City Block distance is used since it is the same as the number of row/columns need to be replaced in the movement, n is the number of searching points.

III. Adjustment for Memory Access Friendly on Motion Estimation Algorithm

A. Full Search

First, it is the exhaustive search [8, 9] that scans all the points in the searching area and it is trivial to form a pattern for efficient memory access. For a $(\pm 7, \pm 7)$ area, there are 225 searching points and the memory cost will be $16 + 224 = 240$. An example of full searching is shown in Fig.1 below.

B. Three Step Search (3SS)

3SS uses square search patterns three times with the searching pattern size reduced by half for each step[10]. For a searching area $(\pm 7, \pm 7)$, searching patterns are 7×7, 5×5 *and* 3×3. The number of searching point is fixed at 25. The memory access friendly algorithm is straight forward and it is adjusted as follows:

Step 1: Do the ordinary first step start corner and stop and center and a temporary minimum point is found.

Step 2: Set the search pattern center at the minimum point found in step 1 and set the starting point that closest to the end point in step 1 (center). Then go through the searching pattern.

Step 3: Set the search pattern center at the minimum point found in step 2 and set the starting point that closest to the end point in step 2. Then go through the searching pattern.

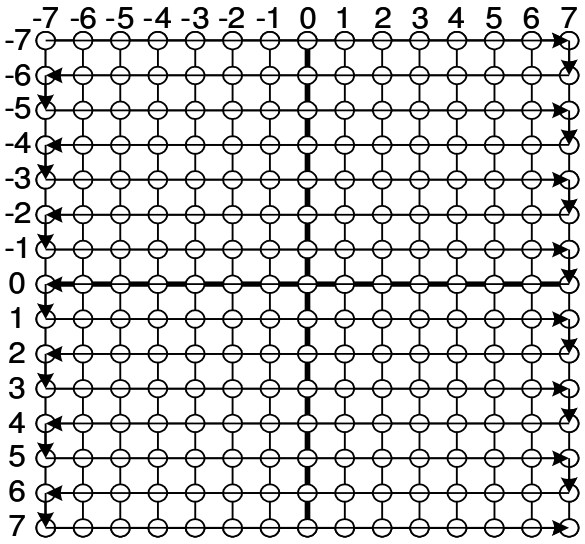
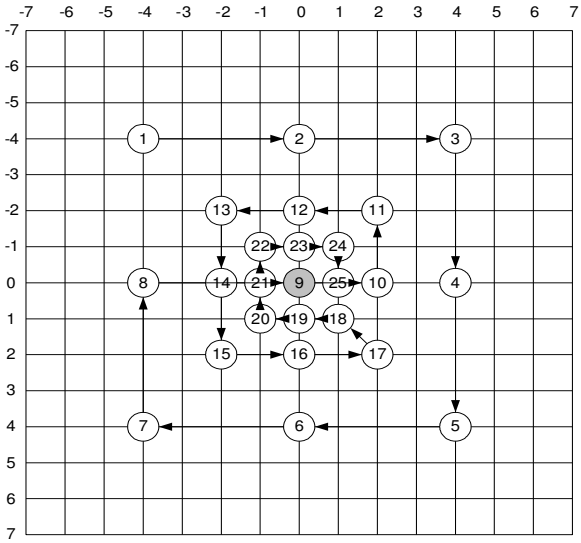


Fig.1. Full search



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	4	4	4	4	4	4	4	4	2	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
c_{mem}	2	2	2	2	2	2	2	2	1	1	73
Spt	**21**	**22**	**23**	**24**	**25**						
c_{mem}	1	1	1	1	1						

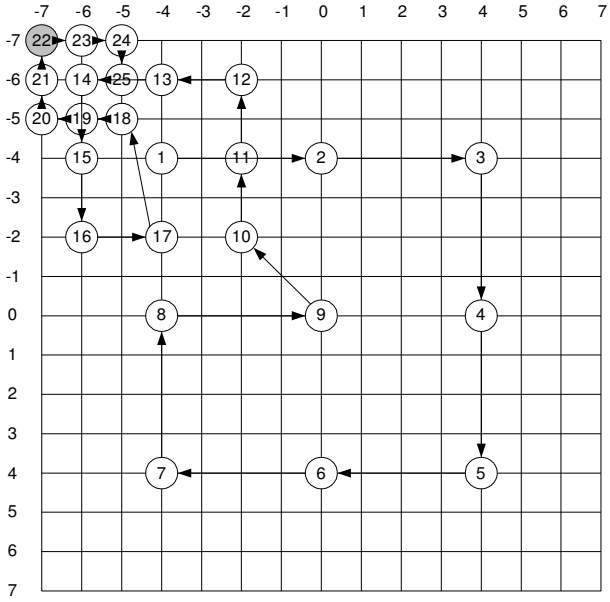
Fig. 2 3SS searching with global minimum at $(0,0)$

Examples of 3SS searching with global minimum point at $(0,0)$ *and* $(-7,-7)$ are shown in Fig.2 and Fig.3.

C. Four Step Search (4SS)

4SS also uses square search patterns, but with four times and different search pattern size [11]. For a searching area $(\pm 7, \pm 7)$,searching patterns for first three steps is 5×5 and the final step is 3×3. The number of searching point is varying between 17 and 27. The memory friendly algorithm is adjusted number of searching point is varying between 17 and 27.The memory friendly algorithm is adjusted as the following:

International Journal of Electronics Communication and Computer Engineering
Volume 6, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	4	4	4	4	4	4	4	4	4	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
c_{mem}	2	2	2	2	2	2	2	4	1	1	73
Spt	**21**	**22**	**23**	**24**	**25**						
c_{mem}	1	1	1	1	1						

Fig.3. 3SS searching with global minimum point at $(-7,-7)$

Step 1: Do the ordinary first step start corner and stop and center *clockwise* and a temporary minimum point is found.

Step 2: Set the search pattern center at the minimum point found in step 1. Then find the starting point and go through the unsearched points *anti-clockwise*. To find the starting point, find an unsearched point that the previous point is searched.

Step 3: Set the search pattern center at the minimum point found in step 2. Then find the starting point and go through the unsearched points clockwise. To find the starting point, find an unsearched point that the previous point is searched.

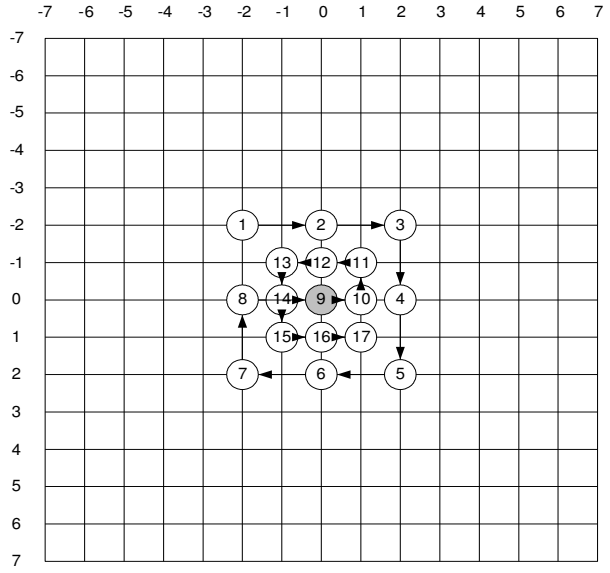
Step 4: Set the search pattern center at the minimum point found in step 3 and set the starting point that closest to the end point in step 3. Then go through the searching pattern.

Examples of 4SS searching with global minimum point at $(0,0)$ *and* $(7,-7)$ are shown in the Fig. 4 and Fig. 5.

D. Block Based Gradient Decent Search(BBGDS)

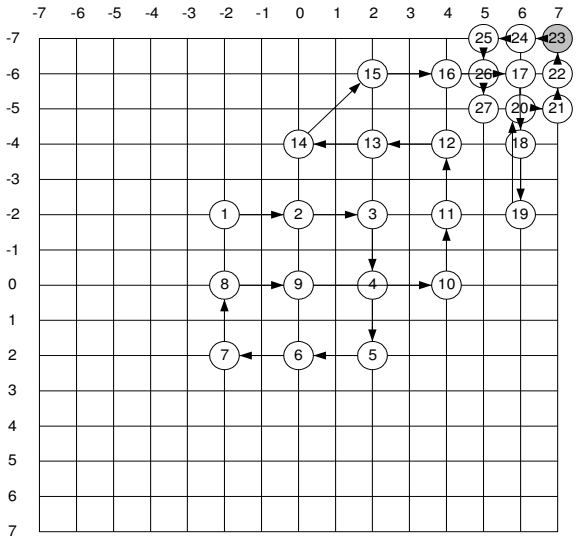
BBGDS is a very simple algorithm that only uses a $\mathbf{3\times 3}$ square as the search pattern [12]. A MV will be decided if a searched point is found as minimum distortion among eight surrounding pints. The memory friendly algorithm is adjusted as the followings:

Step 1: Do the first step $(3\times 3\ square)$ with the sequence shown in points 1-9 from Fig.6 and a temporary minimum point is found. If minimum point is $(0,0)$, end the process.



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	2	2	2	2	2	2	2	2	1	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**				40
c_{mem}	1	1	1	1	1	1	1				

Fig.4 4SS searching with global minimum point at $(0,0)$



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	2	2	2	2	2	2	2	2	4	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
c_{mem}	2	2	2	2	4	2	2	2	3	1	65
Spt	**21**	**22**	**23**	**24**	**25**	**26**	**27**				
c_{mem}	1	1	1	1	1	1	1				

Fig.5. 4SS searching with global minimum point at $(7,-7)$

Step 2: Search the same search pattern centered at the minimum point found in step 1 (e.g. point 10, 11, 12 in Fig.7), if the minimum point is the minimumamong all surrounding searched points, end the process. Otherwise, repeat step 2.

International Journal of Electronics Communication and Computer Engineering
Volume 6, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

Examples of BBGDS searching with global minimum point at $(0,0), (7,0)\ and\ (7,-7)$ are shown in Fig. 6, Fig.7 and Fig.8.

E. Small Cross Diamond Search(SCDS)

SCDS gives a very impressive solution for small motion video by using a Small Cross Search Pattern (SCSP) for the first step and the following steps remain the same as CDS [13,14].Since there is no one-way-trip for the SCSP, some memory access cost is wasted. The memory friendly algorithm is adjusted as the followings:

Step 1: Do the first step (SCSP) with the sequence shown in points 1-5 from Fig.9 and a temporary minimum point is found. If minimum point is $(0,0)$, end the process.

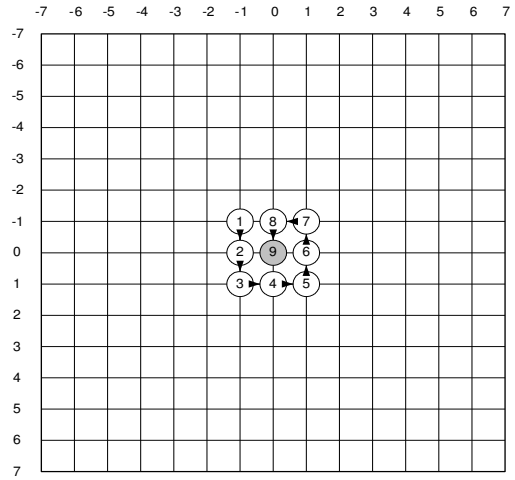
Step 2: Search LCSP points and half diamond points at the direction of the minimum point found in step 1 (e.g. point 10, 11 in Fig. 10), if the minimum point is surrounded by 4 searched points, end the process.

Step 3: Set the search pattern (LDSP) center at the minimum point found in step 2 (last step). Then find the starting point and go through the unsearched point reverse direction from last step (clockwise for the first time). To find the starting point, find an unsearched point that the previous point is searched. If minimum point is center go to Step 3; if not, move the center to the minimum point and repeat.

Step 4: Set the search pattern (SDSP) center at the minimum point found in step 3 and set the starting point that closest to the end point in step 3. Then go through the searching pattern. Examples of SCDS searching with global minimum point at $(0,0), at\ (-2,6)$ are shown in Fig.9 and Fig.10.

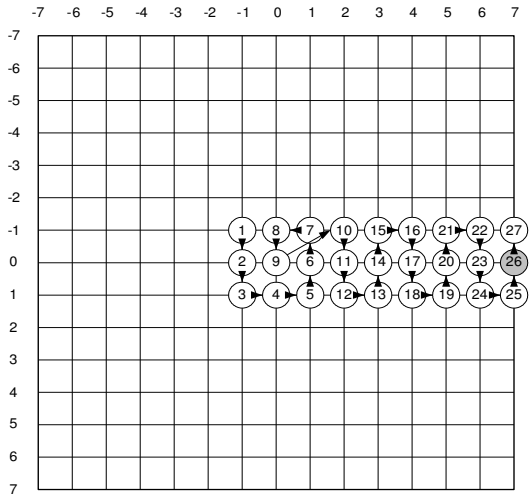
Simulation has also been carried out on other standard search algorithms. But due to page limitation, all the algorithms and simulation results are not tabulated here. For example Diamond Search (DS) / Unrestricted Center Biased [15] use a diamond shape pattern instead of square shape pattern that 3SS and 4SS does and with a converging mechanism like 4SS.

The major different from 3SS and 4SS is that it does not limit the total searching step. Although the theoretical maximum number of searching points is really high, the average number of searching point is even smaller than 4SS. On the other hand CDS further improve the DS by changing the first step searching pattern. It uses a Large Cross Searching Pattern (LCSP) in first step and introduced a new technique called first step stop that can substantially reduce the minimum number of searching points for small motion videos. Based on center biased property, most motion vectors will be within $(\pm1, \pm1)$, and especially for small motion videos, most of motion vectors will be zero motion. So, the number of searching points and memory access cost for minimum point at $(0,0)$ is a critical factor, from the Table I we found that CDS and SCDS can use smallest number of searching point and the memory access cost.



Spt	1	2	3	4	5	6	7	8	9	Total
c_{mem}	16	1	1	1	1	1	1	1	1	24

Fig.6. BBGDS searching with global minimum point at $(0,0)$



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	1	1	1	1	1	1	1	1	3	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
c_{mem}	1	1	1	1	1	1	1	1	1	1	44
Spt	**21**	**22**	**23**	**24**	**25**	**26**	**27**				
c_{mem}	1	1	1	1	1	1	1				

Fig.7. BBGDS searching with global minimum point at $(7,0)$

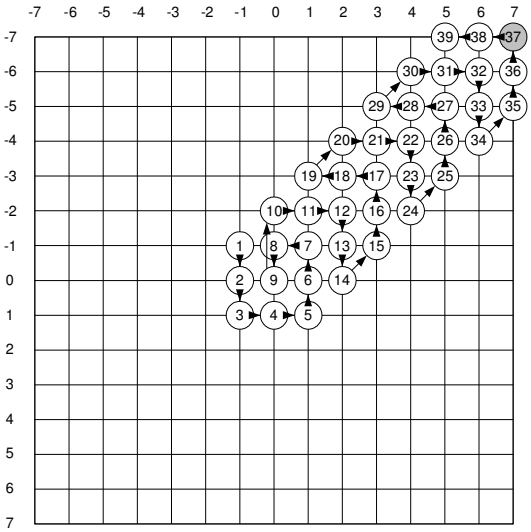
IV. EXPERIMENTAL SETUP AND SIMULATION

The simulation was performed under the following condition:

- Advanced prediction: on (i.e. $[-2,+2]$ pel search for 8×8 blocks around the best 16×16 of a fast ME algorithm
- half-pel motion estimation: on
- disabled: rate control, error resilience, SADCT
- enabled: DC/AC-prediction
- deblocking filter: rectangular VO: off, arbitrarily shaped VO: on

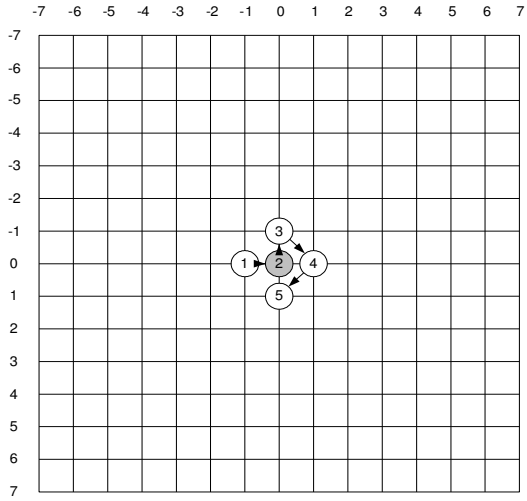
International Journal of Electronics Communication and Computer Engineering
Volume 6, Issue 6, ISSN (Online): 2249–071X, ISSN (Print): 2278–4209

- combined motion/shape/texture coding,
- h.264-quantization



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	4	4	4	4	4	4	4	4	2	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
c_{mem}	1	1	1	1	2	1	1	1	1	2	
Spt	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**	**29**	**30**	60
c_{mem}	1	1	1	1	2	1	1	1	1	2	
Spt	**31**	**32**	**33**	**34**	**35**	**36**	**37**	**38**	**39**		
c_{mem}	1	1	1	1	2	1	1	1	1		

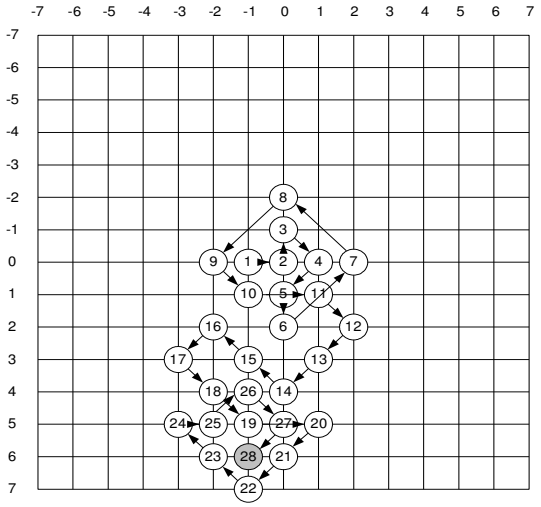
Fig.8. BBGDS searching with global minimum point at $(7, -7)$



Spt	1	2	3	4	5	Total
c_{mem}	16	1	1	2	2	22

Fig.9. SCDS searching with global minimum point at $(0, 0)$

The simulation were carried out using low, medium and high motion details sequence video as Akiyo (CIF), Football (SIF) and Stefan (CIF) for observing the best performance of different motion estimation. The performances are tabulated in Table II.



Spt	1	2	3	4	5	6	7	8	9	10	Total
c_{mem}	16	1	1	2	2	1	4	4	4	2	
Spt	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**	
c_{mem}	2	2	4	2	2	2	2	2	2	2	72
Spt	**21**	**22**	**23**	**24**	**25**	**26**	**27**	**28**			
c_{mem}	2	2	2	2	1	2	2	2			

Fig.10. SCDS searching with global minimum point at $(-2, 6)$

From the analysis it is found that the cost function of accessing data in memory can be saved from 68% to 80% and the number of search point can be saved from 87% to 94%. So this approach can perform better where the memory savings is greatly needed. The experimental results also showed that the ranking of memory access cost for small motion video is $SCDS > BBGDS > CDS > DS > 4SS > 3SS > FS$and for high motion video the ranking of memory access cost is $BBGDS > CDS > SCDS > 4SS > DS > 3SS > FS$ with lower computational complexity and similar quality is maintained.

V. Conclusion

The performance of commonly used ME algorithms are analyzed in details on the basis of memory saving and number of search point reduction. From the analysis it is found that it can significantly reduce the search point and memory access cost. We can also get ranking between the ME algorithms for different types of video content. From this ranking it can be concluded that if we can search using this order then we can obtained more better result for motion estimation and motion compensation. The design of hardware architecture will be studied for further enhancements in near future.

Acknowledgement

This research work is partially funded from the Research Project' 2015-2016 approved by CASR of Khulna University of Engineering & Technology (KUET), Khulna, Khulna-9203, Bangladesh.

Table I: Comparison of Searching Point and Memory Acess Cost for Zero motion vector

Search Algorithm	Search Point (Spt)	Cmem
FS	225	240
3SS	25	73
4SS	17	40
DS	14	41
CDS	9	28
BBGDS	9	24
SCDS	5	22

Table II; Performance Comparison of Different Motion Estimation Algorithms

(A) For Akiyo (CIF)

Akiyo (CIF)	C_{mem}/frame	MAE	MSE	PSNR	MAE / pixel	Spt	C_{mem} saved	Spt saved
FS	86836.0	61392.87	3.94052	42.7795	0.60560	204.28	-	-
3SS	27475.7	61923.43	4.07492	42.6569	0.61083	23.21	68.359%	88.64%
4SS	15300.4	61701.28	4.00621	42.7084	0.60864	15.85	82.380%	92.24%
BBGDS	9391.5	61405.69	3.93750	42.7818	0.60572	8.55	89.185%	95.82%
SCDS	8911.7	61497.18	3.98259	42.7417	0.60662	5.07	89.737%	97.52%

(B) For Football (SIF)

Football (SIF)	C_{mem}/frame	MAE	MSE	PSNR	MAE / pixel	Spt	C_{mem} saved	Spt saved
FS	71626.0	635238.94	208.086	25.6195	7.51940	202.05	-	-
3SS	22914.1	659199.96	230.139	25.1866	7.80303	23.10	68.01%	88.57%
4SS	14454.9	657698.57	227.648	25.2993	7.78526	17.63	79.82%	91.28%
BBGDS	9609.7	664136.70	235.656	25.2057	7.86147	12.87	86.58%	93.63%
SCDS	12423.0	662556.65	234.986	25.1908	7.84276	11.04	82.66%	94.53%

(C) For Stefan (CIF)

Stefan (CIF)	C_{mem}/frame	MAE	MSE	PSNR	MAE / pixel	Spt	C_{mem} saved	Spt saved
FS	86836.0	843357.3	303.407	24.6120	8.31910	204.28	-	-
3SS	27690.7	894472.87	341.119	24.1907	8.82332	23.30	68.11%	89.57%
4SS	18431.3	951493.35	385.155	23.6722	9.38579	18.65	78.77%	91.88%
BBGDS	12385.0	1003221.97	428.156	23.2255	9.89605	14.75	85.74%	93.81%
SCDS	18023.8	973488.53	404.538	23.4690	9.60275	14.33	79.24%	94.01%

REFERENCES

[1] I.Richardson, H.264 and MPEG-4 Video Comprsion- Video Coding for Next Generation Muledia. John Wiley&Sons, Chichester, 2003.

[2] R. Srinivasan and K. R. Rao, "Predictive coding based on efficient motion estimation," *IEEE Trans. Commun.*, vol. 33, no. 8, pp. 888–896, Aug. 1985.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," in *Proc. Nat.Telecommun. Conf.*, New Orleans, LA, Dec. 1981, pp. G5.3.1-G5.3.5.

[4] I. Ahmad, W. Zheng, J. Luo, and M. Liou, "A fast adaptive motion estimation algorithm," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 3, pp. 420–438, Mar. 2006.

[5] J. Kim and T. Park. A novel VLSI architecture for fullSearch variable block-Size motion estimation. IEEE Transactions on Consumer Electronics, 55(2): 728-733, 2009.

[6] J.C. Tuan, T.S. Chang, and C.W.Jen, "On the data reuse and memory bandwidth analysis for full-search block-matching VLSI architecture," IEEE Trans. Circuits Syst. Video Technol., vol. 12, pp. 61–72, Jan. 2002.

[7] S.B. Lopes, I. S. Silva and L.V Agostini, "An efficient memory hierarchy for full search motion estimation on high definition digital videos," Proceedings of the 24th symposium on Integrated circuits and systems design,pp.131-136, August 2011.

[8] S.-S. Lin, P.-C. Tseng, and L.-G. Chen, "Low- power parallel tree architecture for full search block-matching motion estimation," in Proc.IEEE International Symposium on Circuits and Systems, pp. 313–316, May 2004.

[9] M. Mohammadzadeh, M. Eshghi, M. M. Azadfar, "Parameterizable implementation of full search block matching algorithm using FPGA for real-time applications," ICCDCS 2004: Fifth International Caracas Conference on Devices, Circuits and Systems, pp. 200–203, 2004.

[10] R. Li, B. Zeng, and M. L. Lio, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*vol. 4, no. 4, pp. 438–443, Aug. 1994.

[11] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[12] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.

[13] L. K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst.Video Technol.*, vol. 6, no. 4, pp. 419–422, Aug. 1996.

[14] C. H. Cheung and L. M. Po, "A novel cross-diamond search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 12, pp. 1168–177, Dec. 2002.
[15] C. H. Cheung and L. M. Po, "Novel cross-diamond-hexagonal search algorithms for fast block motion estimation," *IEEE Trans. Multimedia*, vol. 7, no. 1, pp. 16–22, Feb. 2005.
[16] J. Y. Tham, S. Ranganath, M. Ranganath, and A. A. Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, no. 4, pp. 369–377, Aug. 1998.
[17] Chun-Ho Cheung, Lai-Man PO, "A novel cross-diamond search algorithm for fast block motion estimation," Circuits and Systems for Video Technology, IEEE Transactions, Vol.12, No.12, pp.1168 – 1177, Dec 2002.
[18] H.264/AVC SOFTWARE: http://iphome.hhi.de/suehring/~.

AUTHOR'S PROFILE

Md. Salah Uddin Yusuf

received his B.Sc. and M. Sc. Engineering degree in Electrical and Electronic Engineering from Khulna University of Engineering & Technology (KUET), Bangladesh in 1999 and 2005, respectively. He is currently pursuing his PhD in the same department also serving as Associate Professor. His research interest includes signal and image processing, video compression and multimedia communication.

Monira Islam

received her B.Sc. Engineering degree in Electrical and Electronic Engineering from Khulna University of Engineering & Technology (KUET), Khulna-9203, Bangladesh in 2013. She is currently pursuing her M.Sc. engineering in the same department and also serving as Lecturer in the same department. Her research interest includes signal processing and multimedia communication.

Mohiuddin Ahmad

received his BS degree with Honors Grade in Electrical and Electronic Engineering (EEE) from Chittagong University of Engineering and Technology (CUET), Bangladesh and his MS degree in Electronics and Information Science (EIS) from Kyoto Institute of Technology of Japan in 1994 and 2001, respectively. He received his PhD degree in Computer Science and Engineering (CSE) from Korea University, Republic of Korea, in 2008. From November 1994 to August 1995, he served as a part-time Lecturer in the Department of Electrical and Electronic Engineering at CUET, Bangladesh. From August 1995 to October 1998, he served as a Lecturer in the Department of Electrical and Electronic Engineering at Khulna University of Engineering & Technology (KUET), Bangladesh. In June 2001, he joined the same Department as an Assistant Professor. In May 2009, he joined the same Department as an Associate Professor and now he is a full Professor. Moreover, Dr. Ahmad had been serving as the Head of the Department of Biomedical Engineering from October 2009 to September 2012. Prof. Ahmad served as the Head of the Department of Electrical and Electronic Engineering from September 2012 to August 2014. His research interests include Biomedical Signal and Image Processing, Computer Vision and Pattern Recognition, Human Motion Analysis, and Energy Conversion. Technology (KUET), Bangladesh. In June 2001, he joined the same Department as an Assistant Professor. In May 2009, he joined the same Department as an Associate Professor and now he is a full Professor. Moreover, Dr. Ahmad had been serving as the Head of the Department of Biomedical Engineering from October 2009 to September 2012. Prof. Ahmad served as the Head of the Department of Electrical and Electronic Engineering from September 2012 to August 2014. His research interests include Biomedical Signal and Image Processing, Computer Vision and Pattern Recognition, Human Motion Analysis, and Energy Conversion.