Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation

Alexis M. Tourapis[1], Oscar C. Au[2], Ming L. Liou

Department of Electrical and Electronic Engineering,
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong

ABSTRACT

Motion Estimation (ME) is an important part of most video encoding systems, since it could significantly affect the output quality of an encoded sequence. Unfortunately this feature requires a significant part of the encoding time especially when using the straightforward Full Search (FS) algorithm. In this paper a new algorithm is presented named as the *Predictive Motion Vector Field Adaptive Search Technique* (PMVFAST), which significantly outperforms most if not all other previously proposed algorithms in terms of Speed Up performance. In addition, the output quality of the encoded sequence in terms of PSNR is similar to that of the Full Search algorithm. The proposed algorithm relies mainly upon very robust and reliable predictive techniques and early termination criteria, which make use of parameters adapted to the local characteristics of a frame. Our experiments verify the superiority of the proposed algorithm, not only versus several other well-known fast algorithms, but also in many cases versus even the Full Search algorithm.

Keywords: PMVFAST, Zonal Algorithms, Motion Estimation, Diamond Search, Prediction, Predictive Diamond Search, Block Matching, Video Coding

1. INTRODUCTION

Block Matching Motion Estimation (BMME) and compensation is an essential part of several video-coding standards, such as MPEG-1/2/4 and ITU-T H.261/263/263+, since it allows us to exploit the temporal correlation and reduce the redundancy that exists between frames of video sequences, which leads to high compression.

In BBME, the current frame is partitioned into square blocks of pixels, and the best match for these blocks is found inside a reference frame using a predefined distortion criterion. The best match is then used as a predictor for the block in the current frame, where as the displacement between the two blocks actually defines a motion vector (MV), which is associated with the current block. In the encoder. it is only necessary to send the motion vector and a residue block, defined as the difference between the current block and the predictor. This can require significantly fewer bits than the direct coding of the original.

The most common distortion measure used is typically the *mean absolute error* (MAE) or *mean absolute difference* (MAD), or the equivalent *sum of absolute difference* (SAD), which requires no multiplication and gives similar performance as the *mean square error* (MSE). The MAD or SAD of a block A of size N×N located at (x,y) inside the current frame compared to a block B located at a displacement of (v_x,v_y) relative to A in a previous frame is defined as:

$$MAD(v_x, v_y) = \frac{1}{N^2} \sum_{m,n=0}^{N-1} \left| I_t(x+m, y+n) - I_{t-i}(x+v_x+m, y+v_y+n) \right|, \quad (1)$$

$$SAD(v_x, v_y) = N^2 \cdot MAD(v_x, v_y), \quad (2)$$

where I_t is the current frame and I_{t-i} is a previously coded frame.

If a maximum displacement of W pixels in a frame is allowed, we will have $(2W+1)^2$ locations to search for the best match of the current block. Full Search (FS) is the brute force algorithm, which essentially examines all possible locations and is computational intensive. For a frame of size PxQ and a frame rate of T fps, the amount of computation is:

[1] Correspondence: Email: alexismt@ieee.org
[2] Correspondence: Email: eeau@ee.ust.hk; Telephone (+852) 2358-7053, Fax (+852) 2358-1485

$$T \cdot \left(\frac{P}{N} \cdot \frac{Q}{N} \right) \cdot (2W+1)^2 (2N^2 - 1) \cong 8TPQW^2 \cong 1.09 \times 10^{10}, \quad (3)$$

for a possible combination of T=30, P=288, Q=360 and W=21. In some cases this can consume approximately up to 80% of the computational power of the encoder, and is the most computational intensive part in the encoding process, especially when using the brute force Full Search (FS) algorithm.

Several techniques have been previously proposed in an effort to reduce the encoding complexity, such as for example the Three-Step Search (TSS) [1], 2-D log Search [2], New Three Step Search (NTSS) [3] and Diamond Search (DS) [5-6]. Unfortunately, even though these algorithms can make the search faster, they usually incur a significant loss in visual quality. Besides, a common drawback of these algorithms is that the motion vectors are found by minimizing a distortion measure such as the SAD, or integral projection. In other words, the motion vector is found independent of the subsequent DCT, quantization and variable length encoding. In particular, most of these algorithms do not take into consideration the bits required to encode the motion vectors, and thus are not exactly optimal.

Recently several new algorithms have been developed which try to take in consideration some of these factors in order to improve performance. In [4] we introduced the *zonal fast-search* which achieved a modest speed up gain versus FS and good visual quality by initially dividing the search area into rectangular shaped zones and by using different thresholds to enable the early termination of the search. The algorithm was further improved by using circular [7] and diamond zones [10] instead of square ones, where as also the concept of the *predictor motion vector candidate* was also included. These algorithms, even though were significantly faster than most traditional algorithms and could in some cases yield better visual quality than FS, were significantly affected in terms of performance by the type and amount of motion inside a sequence. Other additional criteria were added [8,9,11] which managed to enhance performance further, leading to the Advanced Diamond Zonal Search (ADZS) algorithm.

Another algorithm, rather similar to the zonal algorithms was introduced in [12-14] named as the *Motion Vector Field Adaptive Search Technique* (MVFAST). MVFAST also considered the motion vectors of spatially adjacent blocks in order to improve the preexisting DS algorithm, by selecting the best candidate and then performing a modified DS pattern using this motion vector as the center. Thresholding was also used, which could further assist in the early termination of the search.

In this paper we propose a new improved algorithm, named as the *Predictive Motion Vector Field Adaptive Search Technique* (PMVFAST). Similar to the MVFAST algorithm, PMVFAST initially considers a possible predictor candidate set from which the best candidate is chosen. In addition, adaptively calculated thresholding parameters are introduced which can assist in the early termination of the search. Even though PMVFAST is essentially an improvement of MVFAST, it can also be seen as a special case of the zonal algorithms.

In Section 2 we will first introduce the MVFAST algorithm where as in section 3 we will present the techniques used to improve MVFAST. PMVFAST will be further described in section 4. Finally, in section 5, we will present some simulation results and comparisons.

2. MOTION VECTOR FIELD ADAPTIVE SEARCH TECHNIQUE (MVFAST)

In [12-14] a new algorithm was introduced named as the *Motion Vector Field Adaptive Search Technique* (MVFAST). The algorithm was a significant improvement over the preexisting Diamond Search (DS) algorithm in both terms of visual quality and speed up by initially considering a small set of predictors, namely the (0,0) motion vector and the motion vectors of the three spatially adjacent blocks (left, top, top-right) as possible motion vector predictor candidates. A modified DS pattern (Figure 1) was then created, using the best motion vector predictor candidate as the center of the search.

Unlike DS where only a large moving diamond pattern was considered (Figure 1a), MVFAST also introduced a smaller moving diamond (Figure 1e). The diamond pattern keeps on moving towards the direction of the current minimum (Figure 1c-d-f), until the minimum SAD is found in the center. If the small diamond is used, the search would terminate at this point. If a large diamond is used, a final search is performed around the center (Figure 1b) using the small diamond.

In MVFAST, the selection between using the small diamond versus the large diamond was performed by an initial characterization of the motion type that most likely the current block will have. Motion was characterized as low when the magnitude of the largest motion vector from the three adjacent blocks used in the prediction was below a threshold L_1, medium if it was between L_1 and a second threshold L_2, and high if it was larger than L_2. If motion was considered as low, the small diamond was used with the (0,0) motion vector as the center. When motion was medium, the (0,0) position was again selected as the center but at this time the large diamond was used instead. Finally if motion was characterized as high,

the (0,0) motion vector and the additional spatial predictors were examined and the small diamond was used with the best predictor as the center. An additional thresholding step using a fixed threshold (T = 512) was also performed on the (0,0) MV, which was examined first. After examining (0,0) if the distortion was lower than T then the algorithm would terminate without examining any other locations.

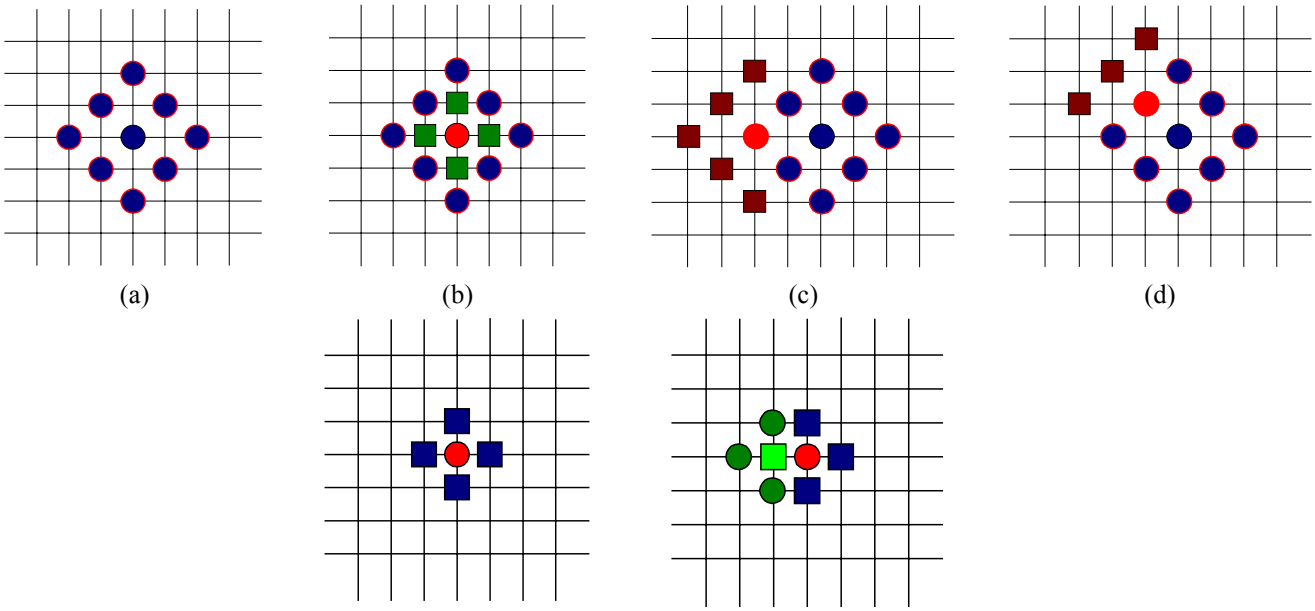


Figure 1: Definition of steps in the MVFAST algorithm. (a-d) large diamond pattern, (e-f) small diamond pattern.

3. GENERALIZED PREDICTOR SELECTION AND ADAPTIVE THRESHOLDING

The MVFAST algorithm managed a significantly improved performance versus DS in both terms of speed up and PSNR. After though making some additional observations related to the characteristics of an encoded sequence it is possible to further improve the algorithm. In particular, as is obvious from the description of the algorithm, MVFAST is a zero biased algorithm, and like DS always benefits the (0,0) motion vector. This is also the basis of several other algorithms including NTSS and the original Square based Zonal Algorithm and was based on the fact that for several sequences motion vectors were concentrated in a small area around the center of the search. This can also be seen in Figure 2a for the *Foreman* sequence. Unfortunately for some sequences this is not always true as can be seen in the *Bus* sequence Figure 2b, which will imply that these algorithms will have reduced performance in such cases.

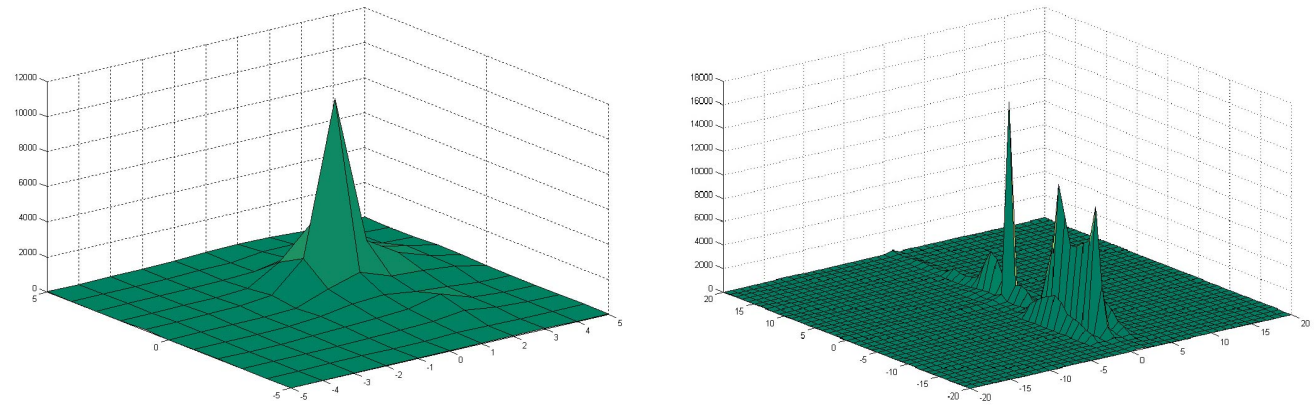
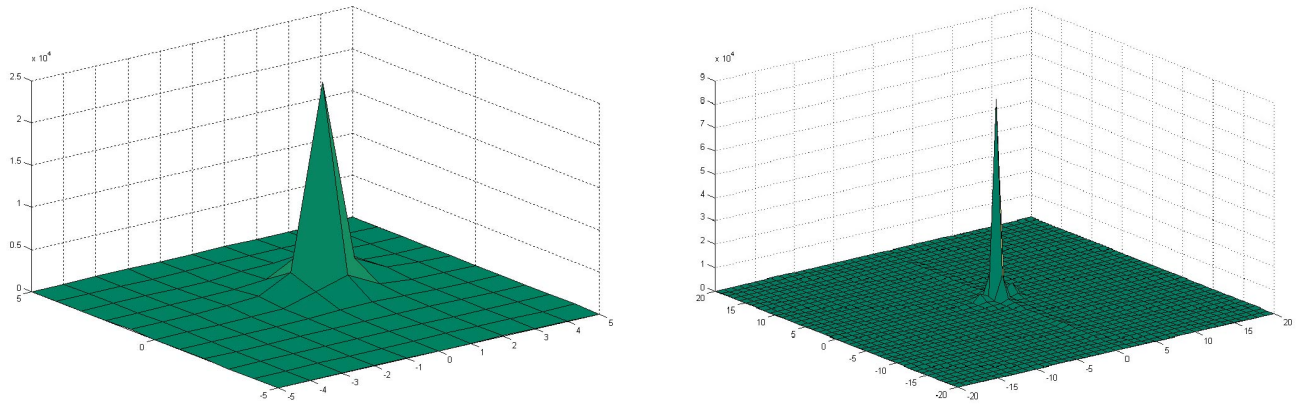


Figure 2: Motion vector distribution in (a) *Foreman* and (b) *Bus* sequences using the FS algorithm versus (0,0) MV

In other algorithms though, such as the zonal based algorithms, the median value of the motion vectors of the three spatially adjacent blocks, the one on the left, top, and top right, was used instead as a primary prediction, making these algorithms median biased. In addition, some of the most advanced video coding standards such as MPEG-4 and H.263 use this value as

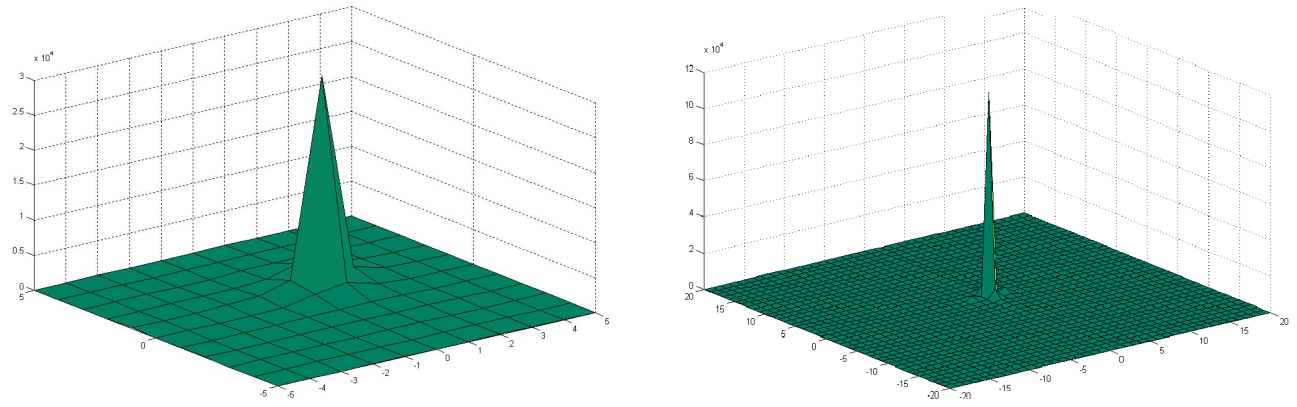
the prediction value in the differential coding of the motion vectors. This is rather justified from examining the distribution of the motion vectors compared to this predictor. From additional simulations we can observe that this median predictor seems to truly have a much higher correlation (Figure 3a) with the current motion vector than (0,0) even for non-center biased sequences such as the Bus sequence mentioned previously (Figure 3b). This suggests that, instead of initially examining the (0,0) position, we could achieve possibly better results if the median predictor was examined first and was given higher priority with the use of an early termination threshold.



(a) (b)

Figure 3: Motion vector distribution in (a) *Foreman* and (b) *Bus* sequences versus the median predictor

Furthermore, other possible predictors also have very high correlation with the motion vector of the current block. In MVFAST, apart from (0,0), the motion vectors of the three spatially adjacent blocks used for the median calculation (the left, top, and top-right blocks) were also used., finally comprising a set of 4 initial predictors. We propose using a larger set of such predictors, which apart from the previous four mentioned and the median predictor also includes several other highly probable candidates. Such candidates can include the motion vector of other spatially and temporally adjacent blocks, but we may also further consider linear or nonlinear combinations of these motion vectors, such as the mean or a weighted mean of all these candidates. We name the selection of such highly probable motion vector candidates as the Generalized Predictor Selection. In particular, we can define the motion vectors of the macroblocks on the left, top, and top-right, their median, the (0,0) motion vector, and the motion vector of the collocated block in the previous frame at time t-1 as Candidate Set A. These motion vectors are the easiest ones to select and do not increase computation significantly. It is obvious from Figure 4 that with the use of this set of motion vectors, correlation has increased further even compared to the median predictor (around 24% in the Foreman case). This might be more obvious when examining the entropy of the current motion vector (Table 1) calculated versus all these possible candidates. It is rather obvious that entropy versus the median motion vector is significantly lower compared to (0,0) for all sequences, where as it is further reduced when compared to the candidates in Set A.



(c) (d)

Figure 4: Motion vector distribution in (a) *Foreman* and (b) *Bus* sequences versus the median predictor and the additional spatial/temporal predictors.

Sequence	(0,0)	Median	Set A
Foreman	6.78	5.17	4.64
Stefan	7.07	4.96	4.41
Bus	6.86	5.31	4.37

Table 1: Motion Vector Entropy versus different Predictors

Apart from the predictor selection, we previously mentioned that MVFAST and the zonal algorithms also use a set of thresholding parameters. At certain intervals the current minimum distortion is compared to these thresholding values and depending on the outcome, we may select to terminate the search immediately, terminate after examining an additional number of checking points or zones, or simply continue the search. These thresholds were though fixed and were used without any consideration of any special characteristics of the encoded sequence. This unfortunately could affect significantly the complexity, and the quality of the estimation since the fixed thresholds might be too high or too low for some blocks. Also as was previously discussed, the selection of such thresholds can be a rather tedious process and could be highly dependent on special conditions of the encoding (for example noise, motion type, resolution etc). An adaptive calculation of these thresholds would be highly desirable since it could potentially improve the efficiency of these algorithms.

We propose to calculate thresholds adaptively in a per-block or even set of blocks basis, which in a sense enables us to adapt to the local characteristics of a frame, and thus improve the efficiency of the thresholding. Thresholds could be linear or non-linear functions of several available parameters, such as the distortion criteria of temporally or spatially adjacent blocks or even their motion vector values. For example, if the distortion criterion used is SAD and n predictors are used, then the calculation of thresholding parameter T_j can be seen as:

$$T_k = f_k\left(MSAD_1,\ldots,MSAD_i,\ldots,MSAD_n,\overrightarrow{MV}_1,\ldots,\overrightarrow{MV}_i,\ldots,\overrightarrow{MV}_n\right), \tag{4}$$

where $MSAD_i$ and $\overrightarrow{MV}_i$ are the minimum SAD and motion vector of thresholding predictor i respectively. Some possible combinations could be:

$$T_k = a_k \times \min\left(MSAD_1, MSAD_2,\ldots,MSAD_n\right) + b_k, \tag{5}$$

$$T_k = a_k \times \text{median}\left(MSAD_1, MSAD_2,\ldots,MSAD_n\right) + b_k, \tag{6}$$

$$T_k = \frac{a_k}{n} \times \sum_{i=1}^{n}(c_i \times MSAD_i) + b_k, \tag{7}$$

For example we can select as an early termination criterion, and basically a predictor of the SAD of the current block, the minimum (Equation5) or maximum $MSAD_i$ from the above set, the median (Equation 6), the mean or a weighted mean (Equation 7) (where the weights could be dependent on the motion vectors) etc. It is also possible to weigh these values or offset them by a fixed or adaptive parameter, or even allow these thresholds to adapt or change during the motion estimation process of the current block. From our experiments we have found that a good choice for the estimation of these thresholds is the selection of the minimum SAD value of the blocks used in the median predictor calculation, namely the blocks on the left, top, and top-right from the current block.

Since though in some cases the thresholds used can be excessively large or even too small, possibly causing either a premature early termination or increasing unnecessarily the complexity respectively, we may additionally restrict these thresholds according to some limiting parameters. Even though the use of an upper limit could reduce the speed-up, the visual quality for some blocks could be significantly improved. These upper and lower limits could be either fixed or even adaptive. For example, they can be functions of the mean distortion of the previous frame.

A more generalized formula for the threshold selection when using these limits could then be:

$$T_k = \min\left\{ \max\left[f_k(MSAD_1,\ldots,MSAD_n,\overrightarrow{MV_1},\ldots,\overrightarrow{MV_n}) \;,\; T_j \right] \;,\; T_m \right\}, \tag{8}$$

which implies that a threshold T_k is constrained by two other thresholds T_j and T_m.

A further analysis on the selection of the initial predictors as also on the adaptive calculation of the thresholding parameters can be found in [13].

4. PREDICTIVE MOTION VECTOR FIELD ADAPTIVE SEARCH TECHNIQUE (PMVFAST)

In this section, we propose a new algorithm, named *Predictive Motion Vector Field Adaptive Search Technique* (PMVFAST) that improves upon MVFAST by considering the proposed generalized predictor selection and adaptive thresholding techniques discussed in the previous section. PMVFAST uses basically the same architecture and patterns as MVFAST does. Initially a set of predictors is examined, which in contrast to MVFAST includes the median predictor and temporal candidates, and a set of thresholding parameters is adaptively determined. Unlike MVFAST where priority is given to the (0,0) motion vector, priority in PMVFAST is given to the median predictor by examining it first, making PMVFAST a median-biased algorithm. Thresholding versus a small threshold (e.g. T=256) is then applied allowing the algorithm to terminate the search after examining only one point. After the best one among the predictors is identified, thresholding is applied. This threshold T_1 is calculated adaptively as the minimum value of the three adjacent blocks (left, top, and top-right). If the current minimum distortion is small enough, the algorithm stops. Otherwise, the best predictor is used as the center of a diamond pattern search. The lower and upper limits for T_1 are set to 512 and 1024 respectively but could also be different or even disabled.

A significant difference of PMVFAST compared to MVFAST is the way the small versus the large diamond is selected. Unlike MVFAST where motion was characterized as low, medium or high by considering the largest motion vector candidate, in PMVFAST a different selection strategy is used. This strategy can improve the overall speed up of the algorithm by using the large diamond less often. As a first step, we calculate the $\|MedianMV\|_1$ where *MedianMV* is the median predictor of the three adjacent blocks to the current position. If this value is zero and a distortion predictor T_2 is relatively large, the large diamond is used. Otherwise we use the small diamond. Unlike MVFAST, all predictors are always examined, thus enhancing the accuracy of the prediction.

The distortion predictor T_2 is basically another thresholding parameter that is only used when the median predictor is found to be in the center of the search window, and can be seen as another prediction of the SAD of the current block. If T_2 is small enough, less than some value K, the SAD of the current block is also likely to be small. In this case the motion vector is likely to be found in a small neighborhood around the center, which suggests that the small diamond can be more efficient if used. Otherwise the large diamond should be used instead. In our case we used K = 1536. In our case T_2 was taken as T_1 + 256. We should also mention that an upper limit for T_2 was set equal to 1792.

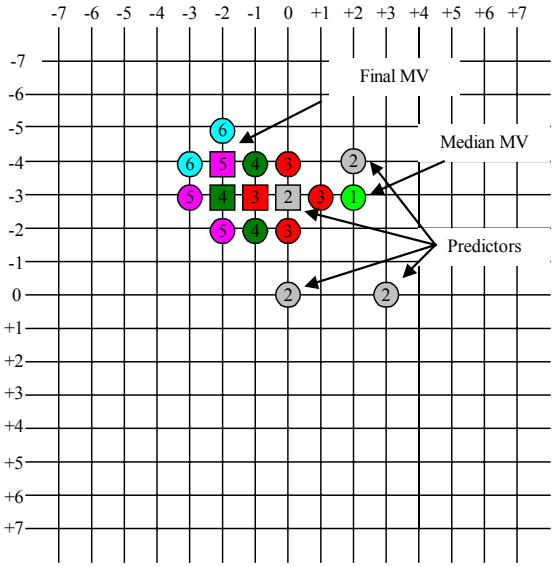


Figure 5: Example of the Implementation of the PMVFAST algorithm.
Square blocks represent the best candidate in each step.

Another characteristic that we have not yet exploited is the possibility that some or even all of the motion vector predictors used are the same or are very close to each other. In such a case it is also highly likely that the target motion vector is within a small neighborhood of the predictors. Thus we introduce some additional *confidence criteria*, which can improve the

speed up of the estimation even further. For example, if all spatially adjacent motion vectors are the same, or the median prediction is the same as the motion vector of the collocated block in the previous frame, there is a higher-than-usual confidence in the accuracy of the predictor and we can terminate the search after N iterations of the diamond pattern, for some small N. Even better, if both criteria are true at the same time, confidence becomes even stronger and we can use even less iterations of the diamond pattern, i.e. only one, or even terminate the search immediately after examining all these predictors without caring about the other early termination stopping criteria. We can also combine this method with an additional thresholding parameter T_3, which can be calculated by taking the *MSAD* value of the collocated block in the previous frame. If both criteria are true and the current minimum SAD is lower than T_3, our confidence is further enhanced and thus we may immediately terminate the search. The threshold T_3 could also be weighted or calculated in different ways as described previously. In all our simulations we have used an upper limit for this threshold, equal to 3072. An implementation of the PMVFAST algorithm can be seen in Figure 5.

It might be obvious that PMVFAST is a generalization of MVFAST. By adjusting different parameters and using different predictors, thresholding criteria, and functions, we may make several variations of the algorithm. PMVFAST is also a special case of the zonal based algorithms, as is also described in [16-17], where only zones 0, 1, and in some cases zone 2 (for the large diamond pattern) are used.

5. SIMULATION

The proposed PMVFAST algorithm was embedded in the MPEG-4 VM encoder software and was tested using the comprehensive test conditions proposed by the *MPEG Ad Hoc Group on Complexity evaluation of video tools* [18]. Further experiments including an implementation of PMVFAST for B frames and field motion estimation are presented in [16-17]. The thresholding parameters were calculated as described previously, with the minimum SAD value of the three spatially adjacent blocks being our primary threshold T_1.

In Table 2 PMVFAST is compared in terms of PSNR and checking points versus the FS, TSS, NTSS, DS, MVFAST, and the ADZS algorithms. It is obvious from the results that the proposed algorithm is significantly faster than all these algorithms while achieving similar and in most cases significantly better PSNR. On the average, for the sequences examined in this test, PMVFAST is roughly 4.5, 4.8, and 4 times faster whereas its PSNR is approximately 0.87dB, 0.81dB, and 0.73dB higher than TSS, NTSS, and DS, respectively. For the cases examined, PMVFAST is about 215 times faster than FS for Search Area (SA) 16, 654 times for SA 32, and 2735 times for SA 64 while having an average PSNR loss of only 0.06dB versus FS. We should note that in MPEG-4, SA 16 implies that the search area examined is (-16, +15) around the center.

On the other hand MVFAST and ADZS are only 2.32 and 1.59 times faster than DS with about 0.16dB and 0.35dB respectively lower PSNR than PMVFAST. In Figures 6 and 7, which show the per frame results, PMVFAST is obviously significantly better than all other fast motion estimation algorithms.

One possible drawback of PMVFAST is the additional memory that might be required for the storage of motion vectors or SAD values. In the case of motion vectors, there is no need for any additional memory since it is possible to reuse the current motion vector memory allocated in an encoder. It is rather important that this memory is not reinitialized at the end of every frame, thus allowing access to the collocated block's motion vector value. In the rather extreme case of a video resolution of 2048×2048 pixels, and if 4 bytes are needed to store one motion vector in the form (MV_x,MV_y), then a total of 64 KB is needed. For this resolution, additional 32 KB are needed for the frame's SAD values. However this number is relatively small especially when considering that 2 frames would require a total of 12MB in this extreme case. If a relatively small search range is needed (less than ±128) the memory required to store the motion vectors can be reduced by half.

In all our simulations an additional memory map was allocated for storing the checking points that have been previously examined, thus avoiding unnecessary repeated SAD computation for the same point and thus maximizing performance. This memory overhead is insignificant since only one bit is needed for each location. An efficient implementation was also presented [17], which ensured minimal overlapping and had only 2% worse case speed-up degradation.

It was also shown in [17] that PMVFAST, similar to other zonal based algorithms, has very smooth motion fields and can basically capture the true motion inside a sequence. This property enables the use of PMVFAST in several other applications such as for example transcoding, deinterlacing, and even video object segmentation.

6. CONCLUSION

In this paper we proposed a new highly efficient block-based motion estimation algorithm named as the *Predictive Motion Vector Field Adaptive Search Technique (PMVFAST).* The algorithm combines two techniques, the *Generalized Motion Vector Predictor* and the *Adaptive Thresholding* to significantly improve the existing MVFAST algorithm. Our simulation

results demonstrate the superiority of PMVFAST versus most if not all other fast motion estimation algorithms in both terms of speed up and visual quality. The algorithm can basically produce similar if not better visual quality than the Full Search algorithm at an insignificant fraction of the complexity. It should be mentioned that PMVFAST was accepted as part of the informative part of the MPEG-4 standard [19] after many rigorous and extensive experiments. PMVFAST is a relatively easy algorithm to implement, whereas it is capable of generating a smooth motion vector field. By further adjusting the different parameters, predictors, and adaptive thresholding techniques, different variants of PMVFAST are possible.

ACKNOWLEDGEMENT

This work was funded by the RGC CERG HKUST6057/99E grant from the Hong Kong Government.

REFERENCES

1. T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, "Motion compensated interframe coding for video conferencing," *Proc. Nat. Telecommun. Conf., New Orleans*, LA, pp. G5.3.1-G5.3.5, Dec'81.
2. J.R. Jain and A.K. Jain, "Displacement measurement and its application in interframe image coding," *IEEE Trans. on Communications*, vol. COM-29, pp.1799-808, Dec'81.
3. R. Li, B. Zeng, and M.L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 4, no. 4, pp. 438-42, Aug'94.
4. Z.L. He and M.L. Liou, "A high performance fast search algorithm for block matching motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.7, no.5, pp.826-8, Oct'97.
5. S. Zhu and K.K. Ma, "A new diamond search algorithm for fast block matching motion estimation," *Proc. of Int. Conf. Information, Communications and Signal Processing*, vol.1, pp.292-6, 1997.
6. 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. On Circuits & Systems for Video Technology*, vol.8, pp.369-77, Aug'98.
7. A.M. Tourapis, O.C. Au, and M.L. Liou, "Fast Motion Estimation using Circular Zonal Search", *Proc. of SPIE Sym. Of Visual Comm. & Image Processing*, VCIP'99, vol.2, pp.1496-1504, Jan. 25-27, 1999.
8. A.M. Tourapis, O.C. Au, and M.L. Liou, "A High Performance Algorithm for Fast Block Based Motion Estimation", *Proc. of Picture Coding Symposium, PCS'99*, pp.121-4, Apr 21-23, 1999.
9. A.M. Tourapis, O.C. Au, and M.L. Liou, "An Adaptive Center (Radar) Zonal based Algorithm for Motion Estimation," *Proc. of 6th IEEE Int. Conf. On Electronics, Circuits and Systems, ICECS'99,* Sept 5-8, 1999.
10. A.M. Tourapis, O.C. Au, M.L. Liou, and G. Shen, "An Advanced Zonal Block Based Algorithm for Motion Estimation", *Proc. of IEEE Int. Conf. On Image Processing (ICIP'99),* section 26PO3.1, Kobe, Japan, Oct'99.
11. A. M. Tourapis, O. C. Au, M.L. Liou, G. Shen, and I. Ahmad, "Optimizing the Mpeg-4 Encoder – Advanced Diamond Zonal Search", in Proc. of 2000 IEEE Inter. Symposium on Circuits and Systems (ISCAS-2000), Geneva, Switzerland, May, 2000.
12. P.I. Hosur and K.K. Ma, "Motion Vector Field Adaptive Fast Motion Estimation," *Second International Conference on Information, Communications and Signal Processing (ICICS '99),* Singapore, 7-10 Dec'99.
13. A. M. Tourapis, O. C. Au, and M. L. Liou, "Highly Efficient Predictive Zonal Algorithms for Fast Block-Matching Motion Estimation," *submitted to Circuits and Systems for Video Technology in Oct. 2000*.
14. K.K. Ma and P.I. Hosur, "Performance Report of Motion Vector Field Adaptive Search Technique (MVFAST)," in *ISO/IEC JTC1/SC29/WG11 MPEG99/m5851*, Noordwijkerhout, NL, Mar'00.
15. A.M. Tourapis, O.C. Au, and M.L. Liou, "Fast Block-Matching Motion Estimation using Predictive Motion Vector Field Adaptive Search Technique (PMVFAST)," *in ISO/IEC JTC1/SC29/WG11 MPEG2000/M5866*, Noordwijkerhout, NL, Mar'00.
16. A.M. Tourapis, O.C. Au, and M.L. Liou, "Core Experiment on Block Based Motion Estimation," *in ISO/IEC JTC1/SC29/WG11 MPEG2000/M5867*, Noordwijkerhout, NL, Mar'00.
17. A.M. Tourapis, O.C. Au, and M.L. Liou, "Implementation of the Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) algorithm in the Optimization Model 1.0," *in ISO/IEC JTC1/SC29/WG11 MPEG2000/M6194*, Beijing, China, Jul'00.
18. Implementation Study Group, "Experimental conditions for evaluating encoder motion estimation algorithms," in *ISO/IEC JTC1/SC29/WG11 MPEG99/n3141*, Hawaii, USA, Dec'99 (updated and modified January 2000).
19. "MPEG-4 Optimization Model Version 1.0", in *ISO/IEC JTC1/SC29/WG11 MPEG2000/N3324*, Noordwijkerhout, NL, Mar'00

Table 2: Simulation results of proposed algorithms at low/medium bitrates

Sequence	Format	BR	SA		Motion Estimation Algorithms						
					FS	*PMVFAST*	TSS	NTSS	DS	MVFAST	ADZS
Hall monitor	QCIF	10	16	PSNR	30.35	*30.34*	29.79	29.79	30.32	30.32	29.78
				Speed Up	1	*378*	41	55	77	239	175
Mother & Daughter	QCIF	24	16	PSNR	34.80	*34.78*	34.81	34.78	34.78	34.76	34.71
				Speed Up	1	*311*	41	54	74	206	215
Coastguard	QCIF	48	16	PSNR	28.88	*28.9*	28.77	28.75	28.73	28.83	28.83
				Speed Up	1	*158*	41	41	58	101	71
Coastguard	CIF	112	16	PSNR	27.03	*27.14*	26.71	26.66	26.44	27.05	27.07
				Speed Up	1	*114*	41	37	49	97	48
			32	PSNR	27.06	*27.19*	26.72	26.69	26.46	27.10	27.06
				Speed Up	1	*431*	156	142	187	370	181
Foreman	CIF	112	16	PSNR	30.04	*29.95*	29.46	29.54	29.58	29.91	29.69
				Speed Up	1	*106*	41	39	43	85	54
			32	PSNR	30.37	*30.24*	29.52	29.58	29.63	30.18	29.67
				Speed Up	1	*386*	156	147	158	307	204
Foreman	CIF	1024	16	PSNR	35.47	*35.54*	34.79	34.92	34.97	35.37	35.24
				Speed Up	1	*203*	41	44	55	126	121
			32	PSNR	35.53	*35.59*	34.79	34.91	34.97	35.41	35.24
				Speed Up	1	*763*	156	169	208	474	459
		512	16	PSNR	34.51	*34.56*	33.88	34.02	34.07	34.43	34.41
				Speed Up	1	*139*	41	40	47	98	78
			32	PSNR	34.84	*34.87*	33.90	34.02	34.09	34.70	34.40
				Speed Up	1	*526*	156	153	173	366	296
Table tennis	SIF	1024	16	PSNR	34.98	*34.98*	34.71	34.82	34.92	34.92	34.95
				Speed Up	1	*310*	41	50	68	182	175
			32	PSNR	35.00	*34.97*	34.72	34.83	34.90	34.92	34.91
				Speed Up	1	*1165*	155	191	258	686	658

Table 3: Simulation results of proposed algorithms at high bitrates (CCIR format)

Sequence	Format	BR	SA		Motion Estimation Algorithms						
					FS	*PMVFAST*	TSS	NTSS	DS	MVFAST	ADZS
Basket	CCIR 625	4000	64	PSNR	26.71	*26.49*	25.90	25.91	26.01	26.40	26.45
				Speed Up	1	*1913*	607	525	584	1326	748
		9000	64	PSNR	30.78	*30.57*	29.92	29.91	29.97	30.46	30.57
				Speed Up	1	*2037*	607	526	590	1366	832
Bus	CCIR 525	4000	64	PSNR	29.78	*29.53*	27.01	26.96	27.15	29.10	28.36
				Speed Up	1	*2655*	603	544	627	1524	809
		9000	64	PSNR	34.03	*33.91*	31.04	30.99	31.18	33.45	32.47
				Speed Up	1	*2923*	603	545	640	1570	863
Flower Garden	CCIR 525	4000	64	PSNR	28.33	*28.21*	27.54	27.83	27.88	28.13	28.15
				Speed Up	1	*3580*	603	553	729	1423	1272
		9000	64	PSNR	33.17	*33.07*	32.26	32.68	32.76	32.99	33.02
				Speed Up	1	*3704*	603	553	730	1426	1434
Stefan	CCIR 525	4000	64	PSNR	30.77	*30.51*	28.51	28.48	28.71	30.02	29.62
				Speed Up	1	*2473*	603	586	662	1751	923
		9000	64	PSNR	34.97	*34.81*	33.11	33.09	33.26	34.43	34.18
				Speed Up	1	*2594*	603	588	668	1813	996

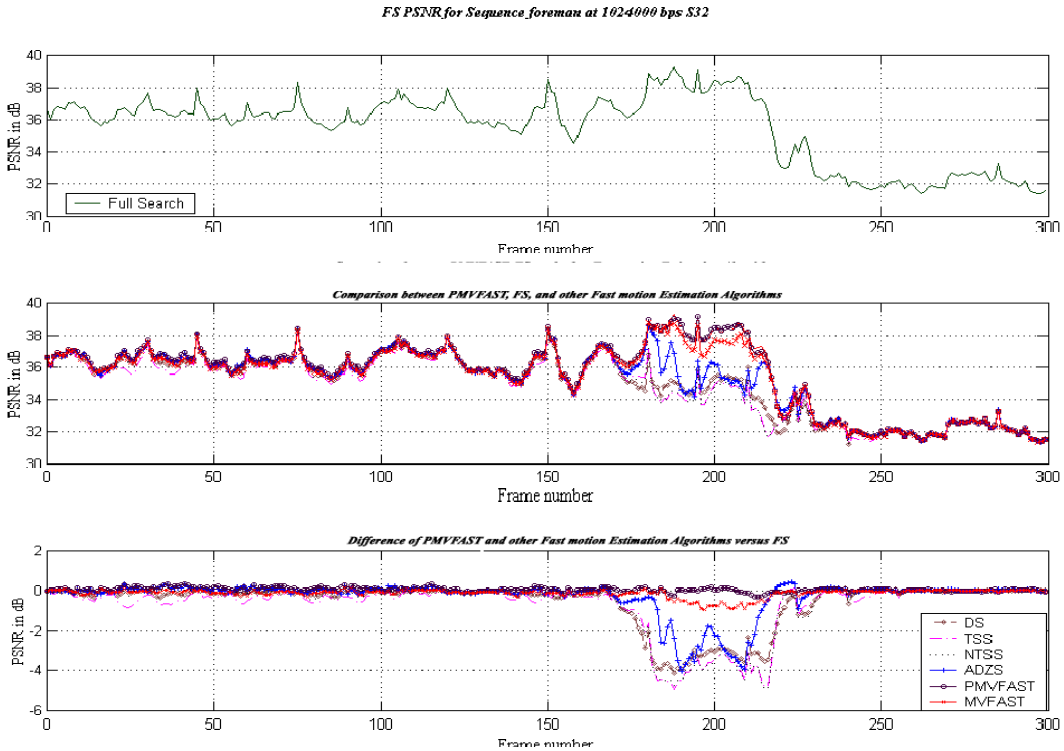


Figure 6: Per frame PSNR for *foreman* at 1024kbps (SA32)

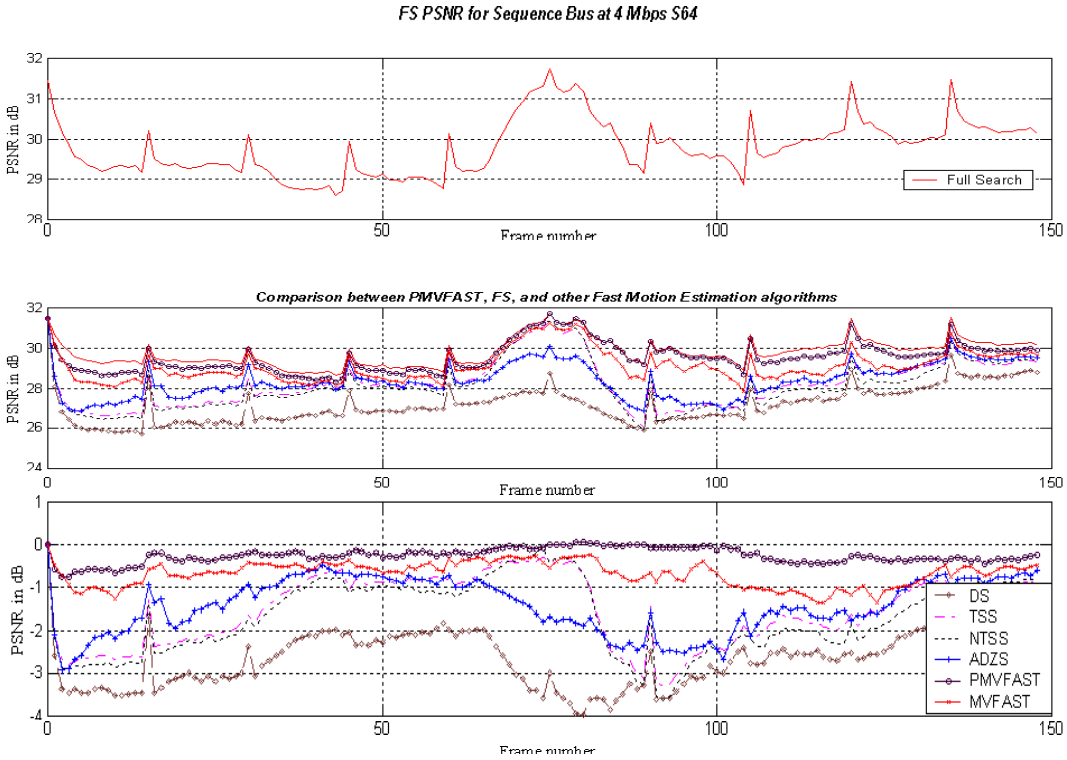


Figure 7: Per frame PSNR for *Bus* at 4000kbps (SA64)