Enhanced Predictive Zonal Search for Single and Multiple Frame Motion Estimation

Alexis M. Tourapis[1]

XiWave plc
Bath, BA1 2PH, United Kingdom

ABSTRACT

Motion Estimation (ME) for video coding has taken an entirely different direction with the emergence of zonal algorithms. These algorithms made it possible to significantly reduce, if not almost eliminate, the tremendous computational overhead of motion estimation that was incurred in a video encoding system, while having little if any loss in quality. In this paper we further improve on these algorithms with the introduction of the Enhanced Predictive Zonal Search Algorithm (EPZS). By considering an additional set of predictors, improving the thresholding process, and simplifying the search pattern employed by these algorithms, we not only manage in achieving better output quality, but also reduce complexity of the motion estimation process even further. Our algorithm was compared with the algorithms accepted in the Optimization Model 1.0 of MPEG-4, and our simulations prove its outright superiority versus the existing algorithms. Furthermore we present a 3-Dimensional implementation of EPZS, which can be easily applied in the case of multiple frame motion estimation (ITU H.263++ and H.26L).

Keywords: EPZS, Zonal Algorithms, Motion Estimation, Diamond, Circular, Prediction, Block Matching, N-dimensional Zonal Search, 3D-EPZS, Multi-frame Prediction

1. INTRODUCTION

Motion estimation 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 can effectively exploit temporal correlation and reduce the redundancy that exists between frames of video sequences, which leads to high compression. Block Based Motion Estimation (BBME) techniques are the ones most widely used due to their relative implementation simplicity and efficacy. The idea is to essentially partition the current frame into square blocks of pixels, and by using a predefined distortion criterion, locate the best match for these blocks inside a reference frame. This best match is then used as a predictor for the block in the current frame, where as the displacement between the two blocks defines a motion vector (MV), which is associated with the current block. Thus, it is only necessary in the encoder to send the motion vector and an additional residue block, defined as the quantized difference between the current block and the predictor. This method, in general, requires significantly fewer bits than the direct coding of the original, thus improving coding efficiency.

The distortion measure most commonly used is the *sum of absolute difference* (SAD) due to its simplicity, since it requires no multiplication, where as it has similar performance with other measures such as the *mean square error* (MSE). The 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:

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

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

If a maximum displacement of p pixels/frame is allowed, then the maximum possible locations to search for the best match of the current block will be equal to $(2\times p+1)^2$. The algorithm that examines all these locations is called the brute force exhaustive search or Full Search (FS). Obviously this algorithm requires a significant part of the computational power of an encoder, which could reach up to 80% or even higher. Unfortunately this can be undesirable and very expensive for several applications especially where real time encoding is essential.

Several algorithms have been proposed in an effort to reduce complexity by reducing the number of checking points examined. Well-known techniques are for example the *3-Step Search* [1], the *New 3-Step Search* [2], and the *Diamond*

[1] Correspondence: Email: alexismt@ieee.org

Search [3-4] algorithms. These techniques, even though could make the search faster, unfortunately at the same time were also affecting considerably the quality of the encoded video.

Recently several new high efficiency algorithms were presented [6-8] that not only significantly reduce the number of checking points examined, but can also retain, and even in many cases improve, video quality. These techniques can accomplish this by initially considering several highly likely predictors, and by introducing very reliable early-stopping criteria to terminate the search at any checking point. In addition, very efficient checking patterns are used for optimizing and improving the search even further. Improved video quality could be achieved since these algorithms in many cases reduce the bits required for motion vector encoding, thus also increasing the number of bits that could be used for encoding the error difference. In particular, the algorithm named as the *Motion Vector Field Adaptive Search Technique* (MVFAST) [6] considered the motion vectors of spatially adjacent blocks and the (0,0) motion vector as initial predictors, combined with a two stage diamond search type algorithm. The algorithm also introduced a fixed early-stopping criterion after examing the (0,0) predictor. The algorithm named as *Predictive Motion Vector Field Adaptive Search Technique* (PMVFAST) [7] further enhanced the performance of MVFAST, in both terms of speed up and output quality, by introducing an additional set of predictors, such as the median predictor and the motion vector of the collocated block in the previous frame. Unlike the fixed early stopping criterion used in MVFAST, PMVFAST used adaptively calculated early stopping criteria, which were more reliable, sequence independent, and were based on correlations between adjacent blocks. Both of these algorithms, due to their high efficiency, were accepted by the MPEG-4 Optimization Model (MPEG-4 Part 7) as recommendations for motion estimation [9]. On the other hand, the *Advanced Predictive Diamond Zonal Search* (APDZS), while also using the same concepts and predictors as the PMVFAST algorithm, achieved better quality at an insignificant cost in speed up, even though better than MVFAST, by using multiple stage diamonds. APDZS could also be seen as a more general algorithm that contains PMVFAST and even MVFAST if some of the parameters or the levels of the diamond shape are preset.

In this paper we propose a new algorithm, which could be considered an improvement of PMVFAST and APDZS, named as the *Enhanced Predictive Zonal Search* (*EPZS*). EPZS improves upon these algorithms by introducing an additional set of predictors, where as the early stopping criteria used are more efficiently selected. Furthermore, due to the enhanced reliability of the predictors, only one checking pattern is used thus considerably reducing any associated overhead of the algorithm. The checking pattern, depending on the implementation requirements, could be either a diamond or square. The algorithm, similar to other zonal type algorithms, could also be implemented in an N-Dimensional framework, such as for example in multi-frame motion estimation.

In Section 2 we will first present the PMVFAST algorithm, where as in section 3 we will introduce and analyze the EPZS algorithm and the main difference from PMVFAST. A 3-Dimensional version of EPZS, named as 3D-EPZS, will be introduced in section 4. Finally, in section 5, we will present some simulation results and comparisons.

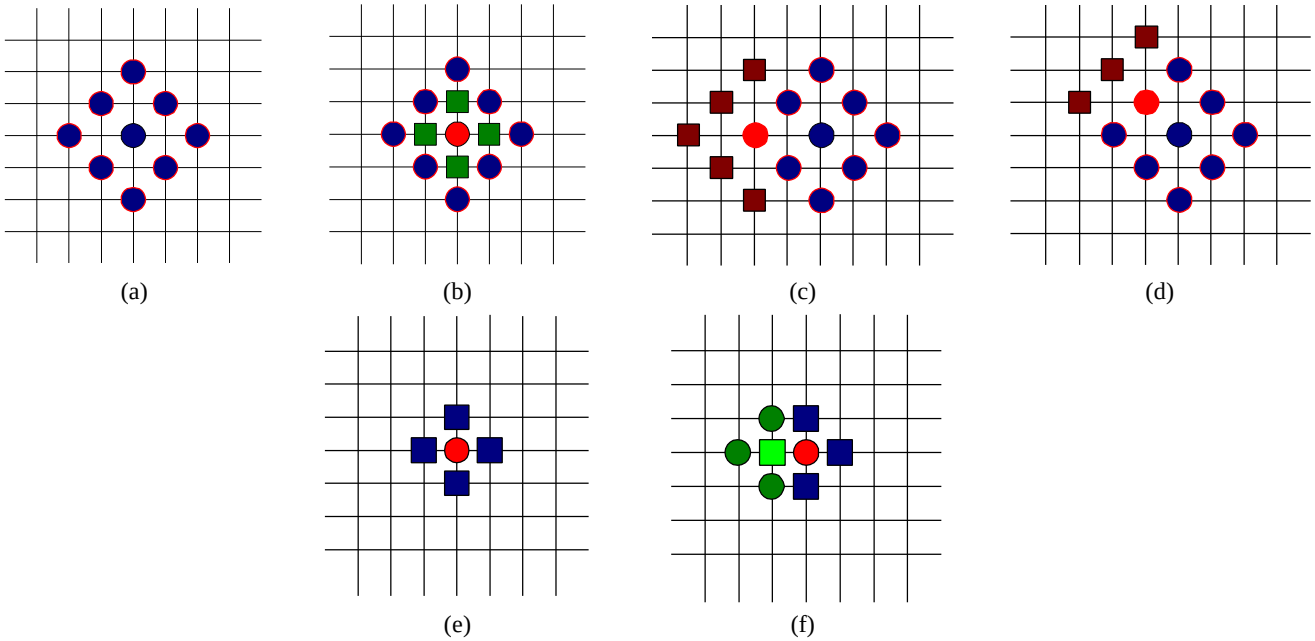
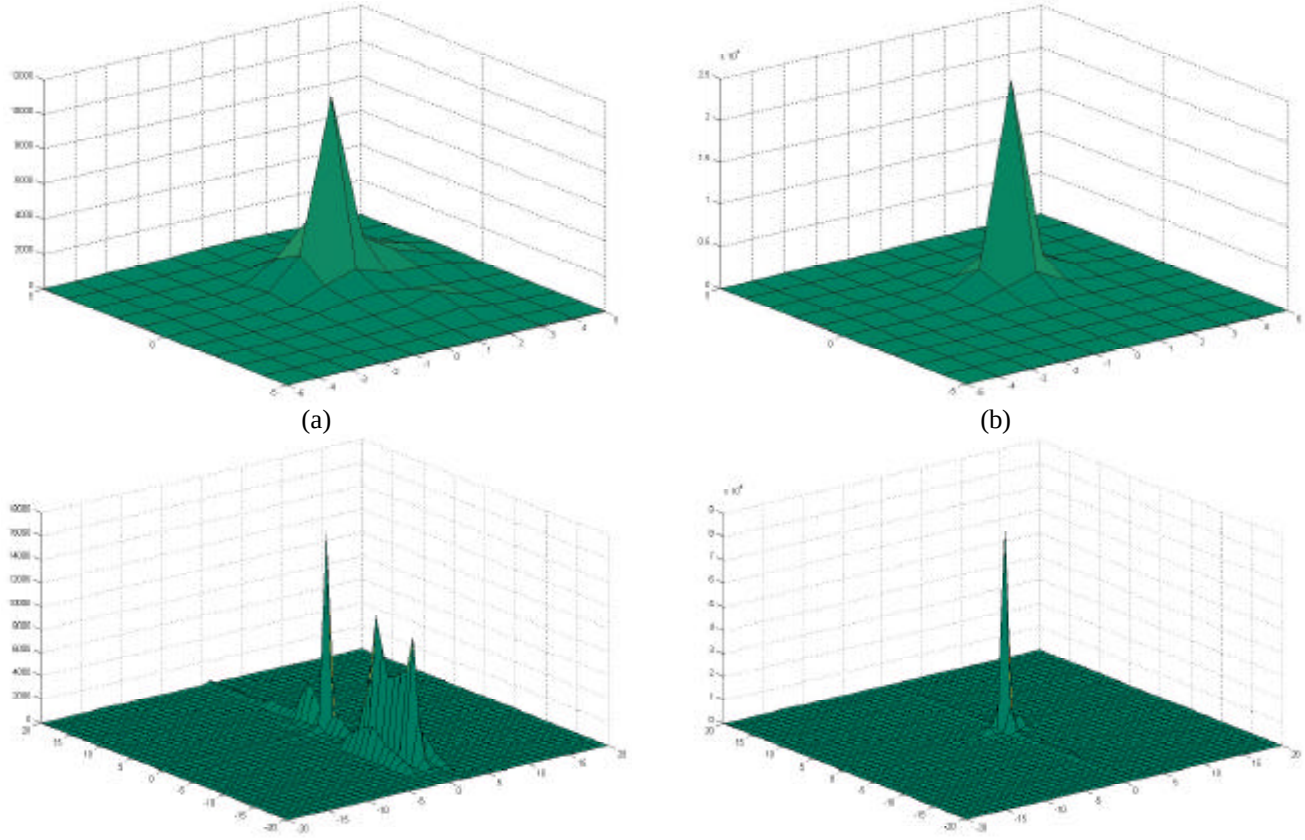


Figure 1: Definition of steps in the PMVFAST algorithm. (a-d) large diamond pattern, (e-f) small diamond pattern.

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

As we have previously discussed, the PMVFAST algorithm managed to improve significantly upon the performance of MVFAST in both terms of speed up and PSNR by enhancing several aspects of the algorithm. Even though both algorithms make use of two different diamond patterns as can also be seen in Figure 1, they differ significantly in several other aspects. More specifically, in PMVFAST, instead of initially examining the (0,0) motion vector as is done in MVFAST, the median predictor, used also for motion vector encoding, is examined instead. This is done since the Median predictor is more reliable and has higher probability to be the true predictor, especially for nonzero biased sequences. This is quite obvious in Figure 2, where even though the sequence *Bus* is not zero biased, it does appear to be median biased. Thus, PMVFAST could obviously perform better in such cases. Furthermore, in the MVFAST algorithm a set of only three additional predictors were examined, namely the motion vectors of the three adjacent blocks (left, top, top-right) in the current frame. In PMVFAST though, apart from these three predictors, we also consider the motion vector of the collocated block in the previous frame, since high correlation exists not only in the spatial but also in the temporal domain. This essentially results in PMVFAST initially examining a set of six predictors, namely the median (seen also as the primary predictor), the (0,0) MV, the MVs from the three adjacent blocks, and the MV of the collocated block in the previous frame.



(a) (b)

(c) (d)

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

A key difference between these two algorithms was the efficient use of early stopping criteria and thresholding. In the MVFAST algorithm after examining the (0,0) motion vector, a single fixed thresholding value was compared to the current position SAD, and if it was larger the search immediately stopped. Even though such a technique enhanced speed up considerably, considering also that the threshold used was not minor, for cases where variations in the sequence were small, or motion was not zero biased, the algorithm could potentially have problems and in the end reduce the overall quality. Instead, in the PMVFAST algorithm an entirely different approach was used. Even though a fixed threshold was also used after examining the primary, median predictor, this was significantly smaller, thus reducing potentially harmful early terminations. Furthermore, additional early stopping criteria were introduced after all predictors were examined. These though were adaptively selected and calculated by considering correlations that existed between blocks and frames. In particular, if the current minimum SAD value is below a threshold, selected as the minimum of the SAD values of the three

adjacent blocks, the search stops. The search also stops if the current best motion vector is the same with the motion vector, and has a smaller SAD value than that, of the collocated block in the previous frame. These thresholding values were though restricted with fixed limits in order to avoid and reduce possible errors resulting from the early termination.

Another major difference between MVFAST and PMVFAST was the selection of the large (Figure 1a-d) versus the small (Figure 1e-f) diamond. In MVFAST the selection was performed by classifying motion as small, medium, or large and depending on this classification the diamond pattern was selected. In PMVFAST though, the large diamond was used only in the case that the median vector was equal to (0,0) and if the adaptive threshold predictor was of a significant value. This was because in such case, it might be quite likely that the predictors have all failed since for such block the error is quite significant, and it is possible that a larger diamond pattern could improve the prediction. The different architecture of PMVFAST versus MVFAST, allowed the algorithm to be around 70% faster than MVFAST, where as it could achieve significantly better visual quality (around 0.2dB on the average).

3. ENHANCED PREDICTIVE ZONAL SEARCH (EPZS) FOR MOTION ESTIMATION

In this section, we propose a new algorithm, named as the *Enhanced Predictive Zonal Search* (*EPZS*) algorithm that improves upon PMVFAST, but also upon APDZS, by considering several other additional predictors in the generalized predictor selection phase of these algorithms. The algorithm also selects a more robust and efficient adaptive thresholding calculation where as, due to the high efficiency of the prediction stage, the pattern of the search can be considerably simplified.

(a) Predictor Selection

The predictor selection of zonal algorithms such as PMVFAST and APDZS appears to be the most important feature and the key to their performance. As we have previously discussed, the predictors are initially defined and examined within a set, where as afterwards it is only necessary to select the best possible candidate among them and perform a local search with a predefined pattern for refining the prediction. In particular, we have observed that motion vectors are highly correlated with the motion vectors of temporally and spatially adjacent blocks that have been previously calculated and could be considered as initial predictors. The same goes for the median value of the motion vectors of the three adjacent blocks, the blocks on the left, top, and top-right from the current position. The median predictor, as we have mentioned previously, appears to be the most likely candidate to be the *optimal* predictor in our set and can be considered on its own as a predictor candidate Subset A. All other candidates that we have previously mentioned, including the (0,0) motion vector, constitute a Subset B and can be examined together. Subsets A and B are already used in PMVFAST and APDZS. We, though, further notice that there exist several other possible candidates that may be examined and which could improve performance even further. In particular, one such candidate, that could be named as the *accelerator motion vector* (Figure 3), is the differentially increased/decreased motion vector taken after considering not only the motion vector of the collocated frame in the previous frame, but also of the frame before that. The concept behind the selection of such predictor is that a block may not be following a constant velocity but may instead be accelerating.

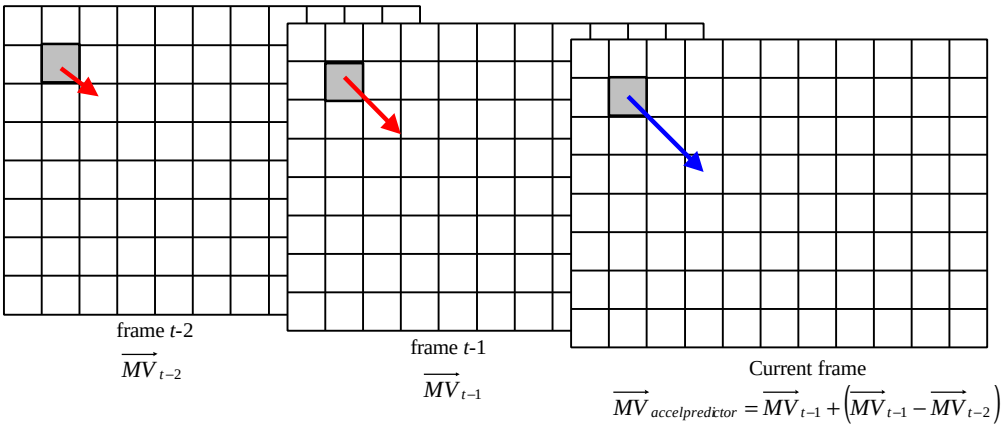
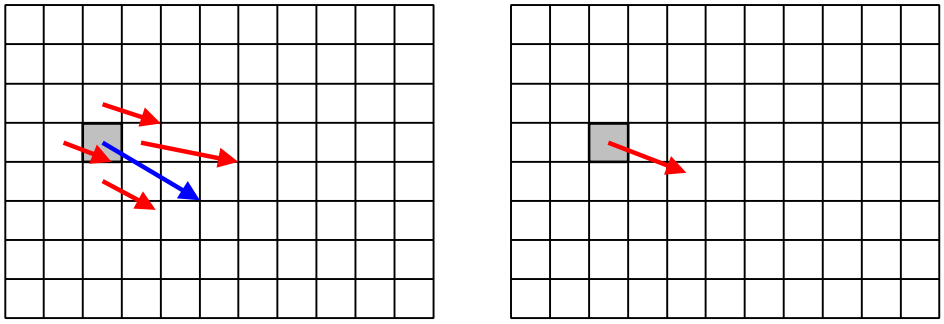


Figure 3: Use of acceleration information as a Motion Vector predictor.

We further notice that the current block might not only be correlated with the adjacent blocks in the current frame or the collocated block in the previous one, but could also be highly correlated with adjacent blocks to the collocated block in the previous frame (Figure 4). In particular, it is quite possible that the collocated block has a very high velocity, which might have caused it to become totally uncorrelated with the current block. Instead, it might be possible that its adjacent blocks,

especially if having a motion vector direction towards the location of the current block, have higher correlation and possibly be better candidates for our prediction phase. A very simple example could be of an object moving vertically or horizontally, from left to right or up to down, with a constant velocity of 16 pixels per frame. If we assume that this object takes up an entire 16×16 block, it is quite evident that the block will probably have little if any relation with the collocated block in the previous frame, but might be related with the motion vector of the left or upper block respectively in the previous frame. Even though we may define 8 such adjacent predictors around the collocated block, for simplicity and since no real benefit was seen from our simulations, we remove the diagonal blocks from our set and only select the four left.



frame *t*-1 Current frame

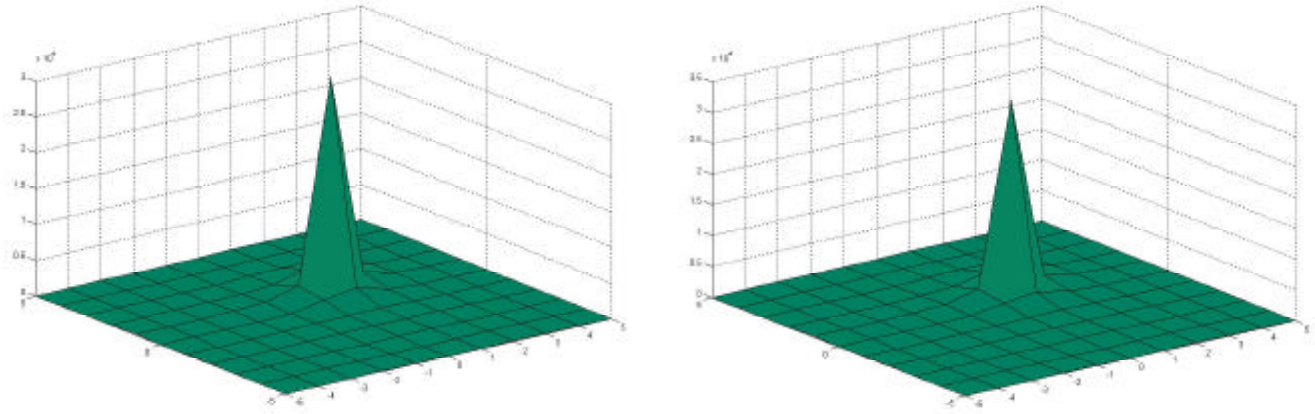
Figure 4: The MV of the current block might have more relationship with the motion vectors of the blocks around the collocated block in the previous frame.

These additional five predictors constitute a final third subset, which we will call as Subset C, of highly likely predictors. To further support our selection of these predictors we present the entropy of the difference between the predictor motion vector values versus the motion vectors generated by Full Search (Table 1). It is quite obvious that our 5 predictors decrease the entropy even further, as would have been expected from our previous arguments.

Sequence	(0,0)	Subset A	Subset B	Subset C
Foreman	6.78	5.17	4.64	4.49
Stefan	7.07	4.96	4.41	4.21
Bus	6.86	5.31	4.37	4.03

Table 1: Entropy of difference between predictors and the motion vectors generated using Full Search

This could also be observed from the motion vector distribution of the *Foreman* sequence (Figure 5) when considering up to the predictor *Subset B* currently used by PMVFAST and APDZS, and *Subset C* proposed for EPZS.



(a) (b)

Figure 5: Motion vector distribution of *Foreman* sequence using the FS algorithm compared to (a) predictor Subset B and (b) predictor Subset C.

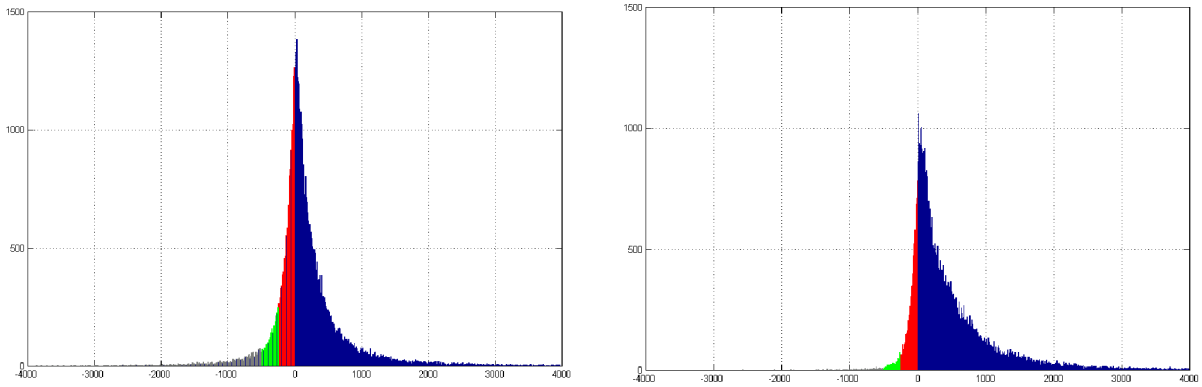
Considering though that these additional predictors might incur additional complexity overhead in the estimation process, we further introduce a three-stage thresholding process in our algorithm:

- After examining all predictors in *Subset A* (median predictor) examine whether the current minimum SAD satisfies a threshold T_1 (in our case T_1 =256). If so terminate algorithm, otherwise continue,
- examine all predictors in *Subset B*. If current minimum SAD satisfies an adaptively calculated threshold T_2 then terminate. Otherwise continue,
- finally examine all predictors in *Subset C*. If current minimum SAD satisfies a threshold T_3 *(in our case* $T_3 = T_2$) then terminate. Otherwise continue by using refinement pattern.

The above process obviously can reduce the computation overhead introduced by the additional predictors since they may only be examined only and only if all other predictors do not satisfy any of the early termination criteria.

(b) Improved Adaptive Thresholding

In both PMVFAST and APDZS thresholding parameters were adaptively calculated by considering the minimum SAD value of the three adjacent blocks. It is though quite possible for the current block to have little if any relationship with these blocks in terms of SAD value, while it is quite likely to be highly related to the collocated block in the previous frame. On the other hand it might also be possible even for the collocated block itself to be in some cases uncorrelated with the current block. Such could, for example, be the case that was also explained previously (large motion vector), or for the case of scene change. Thus, instead of choosing either the minimum of the three spatially adjacent blocks or the collocated block, we make a compromise and select the minimum of all these candidates for the calculation of our thresholding parameter. This can significantly reduce the possibility of error and of erroneous and inadequate early termination. The relationship of the Minimum SAD value compared to the minimum value of the adjacent blocks' SAD, and for the case where the SAD of the collocated block is also considered can be seen in Figure 6. Even though there are more values above zero, meaning that we will have fewer stopping cases by considering the collocated block in the threshold calculation, it is obvious that the number of stopping cases with significant error is also considerably reduced.



(a) (b)

Figure 6: Correlation of the current Minimum SAD value in *Foreman* Sequence using the FS algorithm versus (a) Min of adjacent blocks' SAD and (b) Min of (a) and collocated block in previous frame SAD

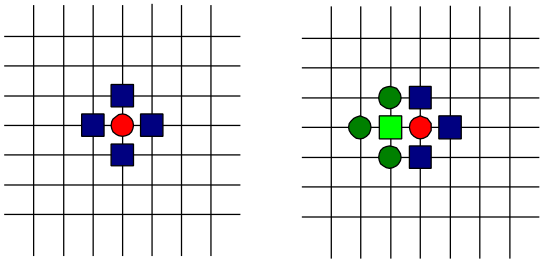
We may instead increase the stopping cases, while not at the same time considerably increasing cases with significant error by scaling and shifting our thresholding parameter. The thresholding parameter could in general be seen as:

$$T_k = a_k \times \min\left(MSAD_1, MSAD_2, \ldots, MSAD_n\right) + b_k \qquad (2)$$

where a_k and b_k are fixed parameters.

If we examine Figure 6 we observe that there exist several thresholding parameters that have really no effect in the early termination of our algorithm. These are basically all thresholding parameters above 0, while this implies that thresholds are too low compared to the actual minimum SAD value of the current block. We also note that these are mostly consentrated close to 0. By shifting these parameters towards the left of this graph we may allow the thresholding to be satisfied more times, thus speeding the search even further, with little cost if any in quality. The weighting could also significantly help, when considering that some thresholds maybe much smaller than others. In our case we have selected a_k =1.2 and b_k =128. The above thresholding technique allows us to completely remove the limits used in the calculation, thus again simplifying implementation.

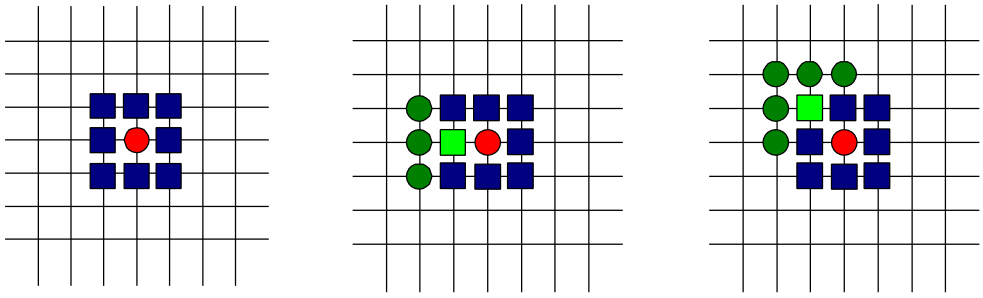
(c) Simplified Search Pattern



(a) (b)

Figure 7: Small diamond pattern used in EPZS

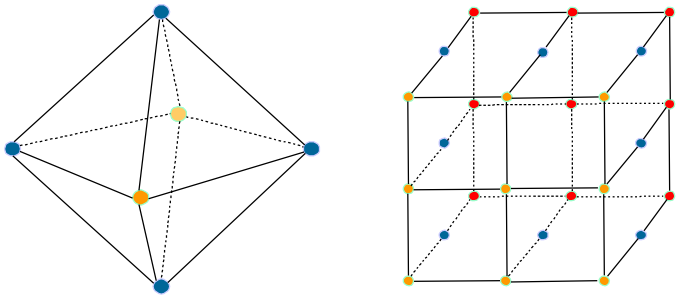
The above methods can significantly improve the accuracy of the prediction thus reducing the effect that the pattern itself will have in the search. Especially considering that in PMVFAST and APDZS we may use relatively large diamond zone shapes (of order 2 and above), it might be desirable to simplify our search pattern further and thus reducing not only the computational complexity, but also the algorithmic design and implementation complexity. This could be quite beneficial for certain applications and especially hardware designs. In particular, it is rather obvious that we may completely discard the second order diamond pattern used in PMVFAST and only use the rather simple first order diamond pattern in our search (Figure 7). Another possible pattern we may select in using would be the 8-point square pattern around the current minimum (Figure 8). Both patterns are very simple and easy to implement not only in a software implementation but also in hardware. As a consequence, considering that in PMVFAST a second threshold was used in deciding whether the large diamond would be used or not, this threshold is no more necessary thus reducing the computation overhead due to thresholding calculation even further. We name the EPZS implementation using the square pattern is named as EPZS^2 (EPZS square).



(a) (b) (c)

Figure 8: EPZS using the square/circular pattern (EPZS^2)

4. ENHANCED PREDICTIVE ZONAL SEARCH (EPZS) FOR MULTIPLE FRAME MOTION ESTIMATION



(a) (b)

Figure 9: 3-Dimensional diamond and cube patterns used in (a) 3D-EPZS and (b) EPZS^3

Similar to other zonal algorithms, EPZS can also be implemented in an N-dimensional framework, as it was presented in [10]. Such an algorithm can, for example, find an implementation in Multi-frame motion estimation, which is already used in standards such as H.263++ and H.26L. A three dimensional version of the EPZS algorithm can be seen in Figure 9, where

for the case of multi-frame motion estimation, the 3rd dimension is time. All the above predictors could be again selected in similar fashion as in the case of Single Frame Motion Estimation, with the only difference that motion vectors now have three parameters, where as thresholds are again calculated in similar fashion. The implementation of EPZS could be named as 3D-EPZS for the case of the diamond pattern, where as when using the square pattern, the algorithm is named as EPZS^3 (EPZS cube), since the pattern used now resembles more closer to a cube. An example of a possible search case using 3D-EPZS is shown in Figure 10.

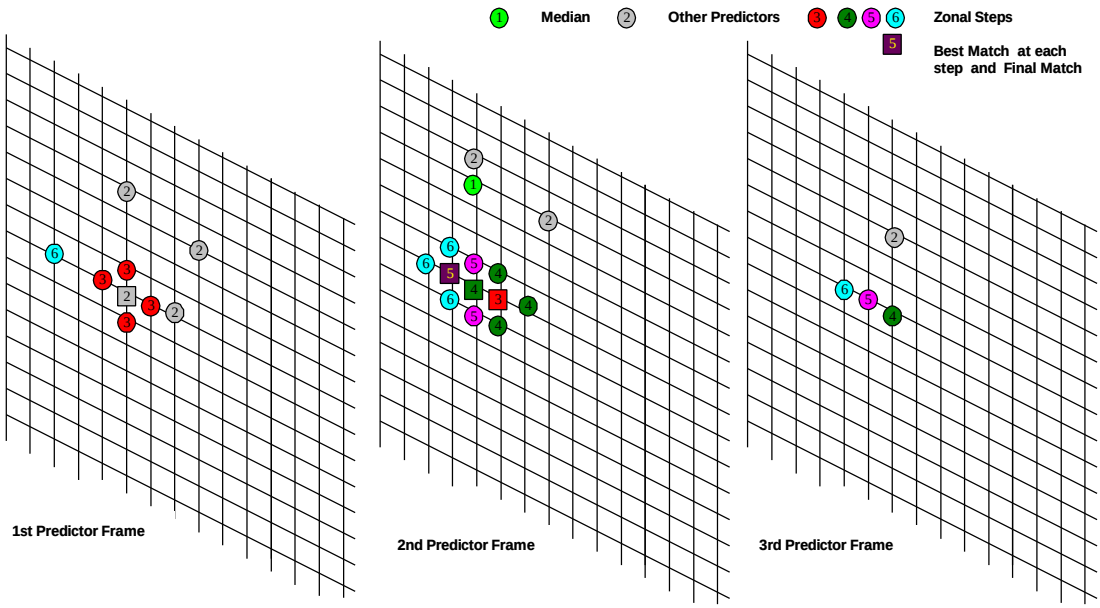


Figure 10: Implementation example of 3D-EPZS

5. SIMULATION

The proposed EPZS algorithm using either the diamond or square patterns 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* [11]. We have also simulated and evaluated EPZS^3 against FS and 3D-PDS, which is also a 3-dimensional zonal algorithm that was also presented in [10].

In Table 2 EPZS and EPZS^2 are compared in terms of PSNR and checking points versus the FS, DS, MVFAST, and the PMVFAST algorithms. It is obvious from the results that the proposed algorithm of EPZS is significantly faster than all these algorithms, especially when the diamond pattern is used, while achieving similar and in most cases significantly better PSNR. EPZS^2, even though is a bit slower, can achieve better PSNR in almost all cases. On the average, for the sequences examined in this test, EPZS is roughly 4.2, 1.8, and 1.2 times faster whereas its PSNR is approximately 0.83dB, 0.20dB, and 0.03dB higher than DS, MVFAST, and PMVFAST, respectively. In comparison EPZS^2 is around 3.1 and 1.3 times faster than DS and MVFAST, where as it is 1.14 times slower than PMVFAST. It also can achieve PSNR of 0.84dB, 0.21dB, and 0.04dB higher than these algorithms respectively. For the cases examined, EPZS and EPZS^2 are, respectively, about 242 and 176 times faster than FS for Search Area (SA) 16, 647 and 496 times for SA 32, and 3569 and 2613 times for SA 64 while having an average PSNR loss of only 0.03dB and 0.02dB versus FS. We should note that in MPEG-4, SA 16 implies that the search area examined is (-16, +15) around the center. The superiority of these algorithms in terms of encoded video quality can also be seen from Figures 10 and 11.

In Table 3 the simulation results using the EPZS^3 algorithm are shown compared to 3D-PDS. We have selected for our simulations a search area of ±8 and a temporal window of 5 frames. Simulation was performed on the sequences *Akiyo* and *Foreman* at 10fps. The selected algorithms were simulated in MATLAB™, and only PSNR and checking points were considered in the evaluation. It is rather obvious from this table that using the EPZS^3 algorithm, not only have we managed in significantly reducing the number of checking points examined compared to 3D-PDS but also to enhance quality. In particular, for both sequences the PSNR has increased by about 0.1dB where as the number of checking points is reduced by 59% and 36% for *Akiyo* and *Foreman* respectively. This implies that the algorithm has great potential not only for single frame motion estimation, but also for the case of multiple frame motion estimation.

Finally, similar to other zonal algorithms, EPZS and EPZS^2 have very smooth motion fields and can basically capture the true motion inside a sequence. This property is very useful and important for many applications such as temporal interpolation, transcoding, deinterlacing, error concealment, and even video object segmentation.

6. CONCLUSION

We have proposed in this paper a new highly efficient block-based motion estimation algorithm named as the *Enhanced Predictive Zonal Search (EPZS).* The algorithm essentially enhances the previously proposed PMVFAST algorithm by considering additional highly likely predictors, and by improving the thresholding calculation. Furthermore, due to the improved efficiency of the prediction, the algorithm only uses a single refinement pattern (either diamond or square), thus significantly reducing the implementation complexity of the algorithm. Our simulation results demonstrate the superiority of EPZS versus most if not all other previous fast motion estimation algorithms in both terms of speed up and visual quality, where as it can essentially produce similar if not better visual quality to that of the Full Search algorithm at an insignificant fraction of the complexity. We have also presented an implementation of the algorithm in multi-frame motion estimation named as EPZS^3, which also has superior performance than other similar algorithms. Finally, like all other zonal algorithms, EPZS is capable of generating a smooth motion vector field, which could be used for several applications including temporal interpolation, deinterlacing, error concealment etc.

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. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. A.M. Tourapis, O.C. Au, and M.L. Liou, "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation," *in proceedings of Visual Communications and Image Processing 2001 (VCIP-2001)*, pp.883-892, San Jose, CA, January 2001.
8. A.M. Tourapis, O.C. Au, and M.L. Liou, "New Results on Zonal Based Motion Estimation Algorithms - Advanced Predictive Diamond Zonal Search," *in proceedings of 2001 IEEE International Symposium on Circuits and Systems (ISCAS-2001)*, vol.5, pp.183–186, Sydney, Australia, May 6-9, 2001.
9. "Optimization Model Version 1.0", *ISO/IEC JTC1/SC29/WG11 MPEG2000/N3324*, Noordwijkerhout, Netherlands, March 2000.
10. A.M. Tourapis, H.Y. Cheong, O.C. Au, and M.L. Liou, "N-Dimensional Zonal Algorithms. The Future of Block Based Motion Estimation?, " *in proceedings of the 2001 IEEE International Conference on Image Processing (ICIP'01)*, Thessaloniki, Greece, October 2001.
11. 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).

Table 2: Simulation results of proposed algorithms in MPEG-4

Sequence	Format	BR	SA		Motion Estimation Algorithms					
					FS	***EPZS***	***EPZS²***	**PMVFAST**	**DS**	**MVFAST**
Hall monitor	QCIF	10	16	PSNR	30.35	*30.40*	*30.34*	30.34	30.32	30.32
				Speed Up	1	*458*	*310*	378	77	239
Mother & Daughter	QCIF	24	16	PSNR	34.80	*34.78*	*34.80*	34.78	34.78	34.76
				Speed Up	1	*345*	*242*	311	74	206
Coastguard	QCIF	48	16	PSNR	28.88	*28.93*	*28.89*	28.90	28.73	28.83
				Speed Up	1	*232*	*171*	158	58	101
Coastguard	CIF	112	16	PSNR	27.03	*27.17*	*27.15*	27.14	26.44	27.05
				Speed Up	1	*173*	*138*	114	49	97
			32	PSNR	27.06	*27.22*	*27.22*	27.19	26.46	27.10
				Speed Up	1	*686*	*547*	431	187	370
Foreman	CIF	112	16	PSNR	30.04	*29.94*	*29.99*	29.95	29.58	29.91
				Speed Up	1	*115*	*89*	106	43	85
			32	PSNR	30.37	*30.19*	*30.26*	30.24	29.63	30.18
				Speed Up	1	*436*	*338*	386	158	307
		512	16	PSNR	34.51	*34.57*	*34.60*	34.56	34.07	34.43
				Speed Up	1	*147*	*115*	139	47	98
			32	PSNR	34.84	*34.87*	*34.90*	34.87	34.09	34.70
				Speed Up	1	*578*	*443*	526	173	366
		1024	16	PSNR	35.47	*35.53*	*35.55*	35.54	34.97	35.37
				Speed Up	1	*225*	*168*	203	55	126
			32	PSNR	35.53	*35.59*	*35.60*	35.59	34.97	35.41
				Speed Up	1	*886*	*658*	763	208	474
Basket	CCIR 625	4000	64	PSNR	26.71	*26.54*	*26.55*	26.49	26.01	26.40
				Speed Up	1	*2529*	*1947*	1913	584	1326
		9000	64	PSNR	30.78	*30.64*	*30.66*	30.57	29.97	30.46
				Speed Up	1	*2643*	*2013*	2037	590	1366
Bus	CCIR 525	4000	64	PSNR	29.78	*29.64*	*29.64*	29.53	27.15	29.10
				Speed Up	1	*3757*	*2785*	2655	627	1524
		9000	64	PSNR	34.03	*33.98*	*33.99*	33.91	31.18	33.45
				Speed Up	1	*4051*	*2984*	2923	640	1570
Flower Garden	CCIR 525	4000	64	PSNR	28.33	*28.22*	*28.22*	28.21	27.88	28.13
				Speed Up	1	*4131*	*2934*	3580	729	1423
		9000	64	PSNR	33.17	*33.11*	*33.12*	33.07	32.76	32.99
				Speed Up	1	*4302*	*3017*	3704	730	1426
Stefan	CCIR 525	4000	64	PSNR	30.77	*30.57*	*30.58*	30.51	28.71	30.02
				Speed Up	1	*3461*	*2676*	2473	662	1751
		9000	64	PSNR	34.97	*34.89*	*34.88*	34.81	33.26	34.43
				Speed Up	1	*3613*	*2772*	2594	668	1813

Table 3: Simulation results of EPZS[3]

Sequence	Format	FR	SA		Motion Estimation Algorithms		
					FS	**3D-PDS**	**3D-EPZS²**
Akiyo	CIF	10	8	PSNR	38.407	38.974	39.039
				Checking Points	1972580	141717	58234
Foreman	CIF	10	8	PSNR	28.838	28.837	28.932
				Checking Points	1972580	245385	157001

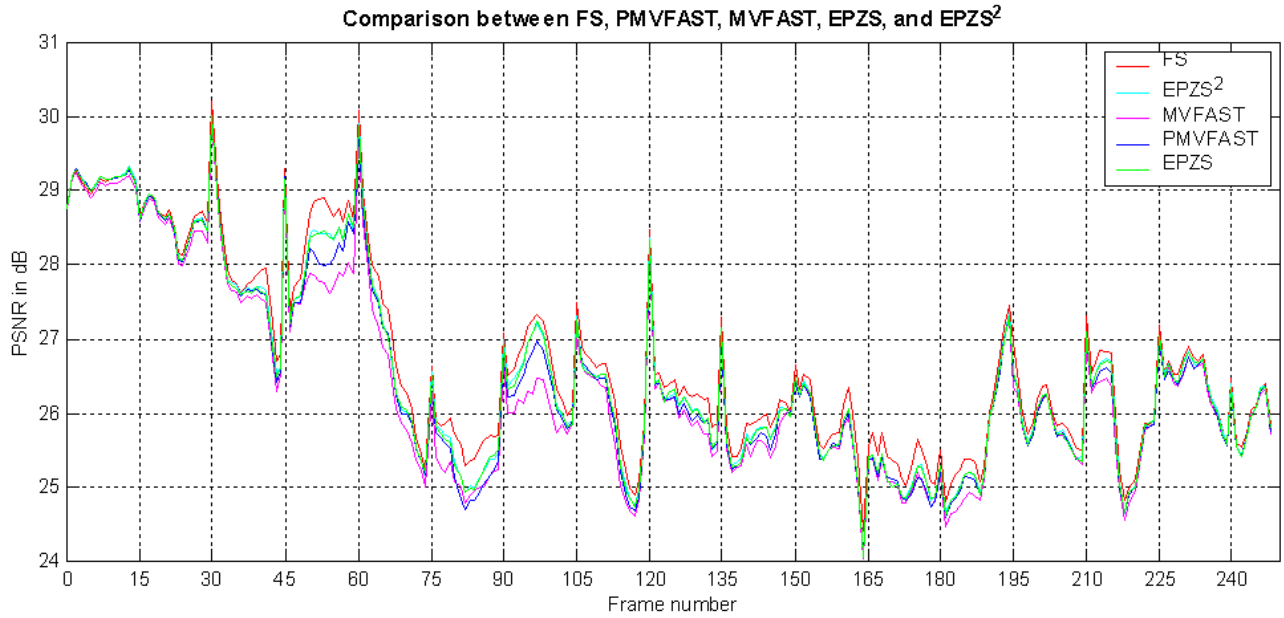


Figure 11: Per frame PSNR for *basket* at 4000kbps

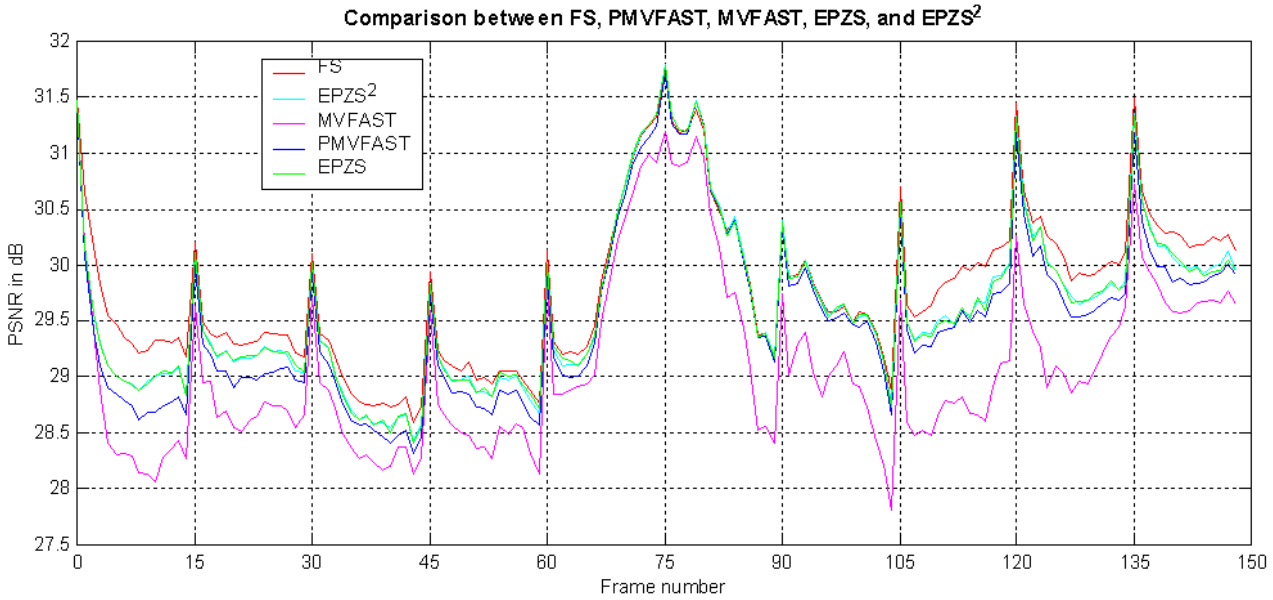


Figure 12: Per frame PSNR for *bus* at 4000kbps