Optimizing the MPEG-4 Encoder - Advanced Diamond Zonal Search

Alexis M. Tourapis [‡], Oscar C. Au [‡], Ming L. Liou [‡], Guobin Shen [‡], Ishfaq Ahmad [†]

Department of Electrical and Electronic Engineering [‡],
Department of Computer Science [†],
The Hong Kong University of Science and Technology,
Clear Water Bay, Kowloon, Hong Kong.
Email: {alexis, eeau, eeliou, eesgb, iahmad} @ust.hk

Abstract

Motion Estimation (ME) is an important part of the MPEG-4 encoder, due to its significant impact on the bitrate and the output quality of the encoded sequence. Unfortunately this feature occupies a significant part of the encoding time especially when using the straightforward Full Search (FS) algorithm. The Diamond Search (DS) was recently accepted as a fast motion estimation algorithm for the MPEG-4 VM. In this paper we propose a new algorithm named Advanced Diamond Zonal Search (ADZS), which is significantly faster than DS (in terms of number of checking points and total encoding time) and gives similar, if not better, quality (in terms of PSNR) of the output sequence. This is more obvious in the high bit rate cases. Our experiments verify the superiority of the proposed algorithm.

1. Introduction

MPEG-4 delivers new capabilities and functionalities to the world of multimedia with emphasis on interactivity, which though usually means real time processing.

Key parts of efficient video coding have always been motion estimation and compensation. By using motion estimation and compensation techniques, we are able to exploit the temporal correlation that exists between frames of video sequences and thus achieve high compression.

In MPEG-4 the technique of block matching motion estimation is the one used due to its simplicity. The current frame is first divided into square blocks of pixels. Then for each one of these blocks we try to find a block in a reference frame that is the closest to it, according to a predetermined criterion. This block is used as a predictor for the current one and the displacement between them defines a motion vector associated with the current block.

The distortion measure used is the sum of absolute errors (SAE or SAD) because it does not require any multiplication and gives similar performance as the mean square error (MSE). If a maximum displacement of p pixels/frame is allowed, then we will have $(2p+1)^2$ locations to search for the best match of the current block. The algorithm that examines all these locations is called the brute force exhaustive search (or full search (FS)). As previous experiments have shown [5]-[8], this algorithm could use a significant part of the computational power of the encoder, which could reach up to even 80% and higher. Unfortunately such an amount is extremely inappropriate for real time applications such as in the case of MPEG-4, as previously mentioned.

This research was partially supported by a grant from the Hongkong Telecom Institute of Information Technology and a grant from the Industry Department of Hong Kong Government.

In an effort to reduce the complexity in the MPEG-4 encoder, the Diamond Search (DS) Algorithm [9]-[10] was proposed and initially adopted in the standard. The algorithm was able to reduce complexity without, in most cases, significantly affecting the quality of the video stream. Unfortunately it was found that the algorithm had poor performance in some cases, especially cases with relatively large global motion.

In this paper we propose a new algorithm, which not only can further decrease the computational complexity of the MPEG-4 encoder compared to the DS algorithm, but is also more robust and can achieve better performance in terms of quality for cases with large global motion. The algorithm is actually an improvement of our previous work in [1]-[4].

2. Advanced Diamond Zonal Search (ADZS)

Most if not all of the existing motion estimation algorithms do not take into consideration, even implicitly, the bits required to encode the motion vectors, and thus are not exactly optimal as far as the overall system performance is concerned. They usually speed up the motion estimation process at the cost of significantly lower quality. Our proposed algorithm, named Advanced Diamond Zonal Search (ADZS) solves this problem by defining diamond shaped zones around a center, searching one zone at a time starting from the innermost zone, and going outward until a good enough motion vector is found. The SAD is used as the distortion measure. Several thresholds are used in determining whether a motion vector is good enough. This algorithm takes advantage of the center-biased property of motion vectors by favoring inner zones (smaller motion vectors). Huge speed up is possible when it stops searching at an inner zone.

The algorithm is actually an improvement of the algorithm of DZS-ER, previously proposed in [7]. Zonal algorithms have been proved of being able to provide great flexibility at great speed and with good performance.

There are several reasons why we are proposing the Diamond Zonal pattern instead of the Circular one, as proposed in [1]. First of all, the diamond pattern is much more regular than the circular, which makes the algorithm easier and simpler to implement, especially for hardware. It was also found that motion vectors are coded in a pattern more similar to a diamond in terms of bits [4], and thus the pattern can help slightly in the overall performance of the estimation. Finally, the zones defined using the diamond pattern, contain fewer checking points than the ones designed using the circular pattern, something that can significantly increase speed up (approximately 50% increase).

As was previously shown [2], thresholds and thresholds alone cannot always ensure algorithmic performance, especially for high activity or fast sequences. By setting their values too high, degradation of the video quality will be inevitable. For this reason, introducing the Half-Stop (HS) Criterion makes it possible, for such cases, to significantly enhance the speed up performance of our algorithm without reducing quality. This criterion considers the probability of finding a better match after examining a number of zones following the current best match.

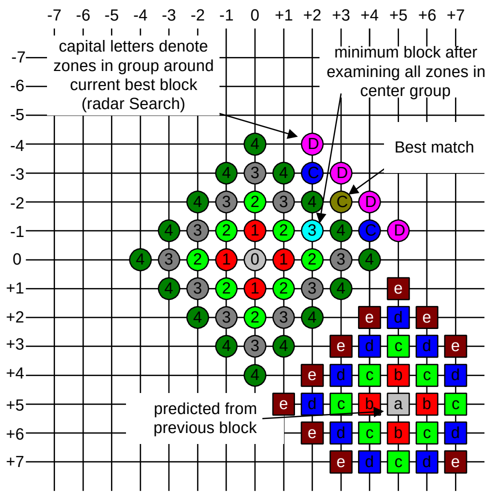

Fig. 1: Definition of the DZS-ER zones.

Finally with the Radar Zonal Search (RZS) [3] technique the algorithm is further improved, since it is possible to reduce the actual number of zones examined, especially considering that the outermost zones always contain more checking points. RZS could actually be considered as a local search technique, which tries to refine the final result in case all previous criteria (thresholds and HS) have failed. The combination of all three techniques (DZS, HS, and RZS) was presented in [7] and was named as Diamond Zonal Search with Embedded Radar (DZS-ER) algorithm (Fig. 1).

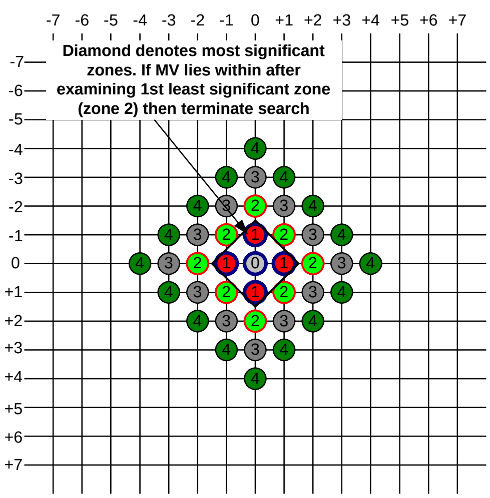

Fig. 2: Advanced Diamond Zonal Search (ADZS)

By considering that the innermost zones usually have a higher probability of containing the actual motion vector solution [11], we may further improve our algorithm. The algorithm is modified to give a higher priority to zones 0 and 1 around the current center (i.e. the prediction or (0,0)). After examining these two zones, and if the 2^{nd} zone is also examined but the best motion vector, up to this point, lies within zones 0 and 1, it is very likely that this motion vector could actually be the best motion vector, or at least a very good candidate. Thus we may terminate, instead of continuing, our search and select the current best match as our motion vector. It could be said that this technique is a special case of the HS criterion, with different parameters used for the two innermost zones (2 for 0^{th} and 1 for the 1^{st} zone). It is rather obvious that the new criterion can work in conjunction with the previously discussed criteria. The partitioning of the zones in this manner, when using diamond shaped zones is named Advanced Diamond Zonal Search (ADZS) (Fig. 2).

3. Algorithm for ADZS

Here is the algorithm for the proposed Advanced Diamond Zonal Search (ADZS) for estimating the motion vector MV of the current block. Note that a block B is considered to belong in diamond shaped zone i *if* $abs(B_x) + abs(B_y) = i$, where B_x and B_y correspond to the position of Block *B.*

Step 1: Set Last = False and MinZone = 0. Also set the following parameters:
- thresholding (ie. $thresa = 768$ & $thresb = 1792$),
- Half-Stop criterion (ie. $zsize = 3$),
- Num of zones (ie. $znum = pznum = 4$).

If block is an edge block, depending to the position, do the following:
- If block is on the first column, assume previous MV to be equal to (0,0).
- If block is on the first row, select previous MV as the prediction.
- If block is on the last column, assume above right MV to be equal to (0,0).

Compute the predicted MV by using the previous, above, and above-right MVs and by calculating their median.
If $MV_{predicted} = (0,0)$, go to Step 9.
If $floor(\ 0.5 + \sqrt{[(MVx_{predicted})^2 + (MVy_{predicted})^2]}) < 4$ set *pznum* for current block to *3*.

(DZS around predicted motion vector)

Step 2: Construct *pznum* diamond shaped zones around $MV_{predicted}$ in the search window. Set i =0.
Step 3: If (i - MinZone)> *zsize* goto Step 23.
Step 4: Compute SAD for each search point in zone i.
Let MinSAD be the smallest SAD up to this point.
Let MinZone be the zone where the smallest SAD has been found up to now.
Step 5: If (i = 2) and (Minzone!=2) goto Step 23.
Step 6: If MinSAD < *thresa* or LAST = true, goto Step 23.
Step 7: If *thresa*< MinSAD<*thresb*, set LAST = true.
Step 8: If i<*pznum*, set i= i+1and goto Step 3.

(DZS around (0,0))

Step 9: If LAST = true goto Step 23.
Else construct *znum* diamond shaped zones around (0,0) in the search window. Set i=0, MinZone=-2.
Step 10: If (i - MinZone)> *zsize* goto Step 23.
Step 11: Compute SAD for each search point in zone i.
Let MinSAD be the smallest SAD up to this point.
Let MinZone be the zone where the smallest SAD has been found up to now.
Step 12: If (i = 2) and (Minzone!=2) goto Step 23.
Step 13: If MinSAD< *thresa* or LAST = true, goto Step 23.

Step 14: If *thresa*< MinSAD< *thresb*, set LAST = true.
Step 15: If i<*znum*, set i= i+1and goto Step 10.

(DZS around best location – Embedded Radar)

Step 16: If LAST = true goto Step 23.
Construct 4 diamond shaped zones around the best location found until now. Set i=1, MinZone=-1.
Note that if location is previously examined, then it is not necessary to examine it again.
Step 17: If (i - MinZone)> *zsize* goto Step 23.
Step 18: Compute SAD for each search point in zone i.
Let MinSAD be the smallest SAD up to this point.
Let MinZone be the zone where the smallest SAD has been found up to now.
Step 19: If (i = 1) and (Minzone!=1) goto Step 23.
Step 20: If MinSAD< *thresa* or LAST = true, goto Step 23.
Step 21: If *thresa*< MinSAD< *thresb*, set LAST = true.
Step 22: If i<4, set i= i+1 and goto Step 17.

(Final step. Use best MV found.)

Step 23: **T**he motion vector is chosen according to the block corresponding to MinMAD.

By performing an optional local half-pixel search, we can refine this result even further.

4. Simulation Results

The proposed algorithm was embedded in the MPEG-4 VM encoder and was tested under several cases. A more detailed analysis of our experiments can be found in [8]. The algorithm was compared versus the FS and DS algorithms.

Table 1 shows in detail the results of our simulations. It demonstrates the average PSNR values, complexity and total encoding time for each algorithm. Different bitrates and frame rates for each sequence were chosen. For the first 4 sequences, Q2 rate control (a VM5 rate control algorithm) with the IPPP… scheme was used, where as for *tennis* and *foreman* we have used the TM5 rate control algorithm, with *M=1* and *N=15* (IPP…IP…). Search areas of (–16, +15.5) and (–32, +31.5) were used for all cases. The *Line-SAD* corresponds to a small optimization preexisting in the MPEG-4 encoder, where, if the partial calculation of the SAD exceeds the current minimum, the SAD calculation stops and we proceed to process the next candidate. Complexity in terms of *Line-SAD* is also included since it reflects more accurately the actual performance of each algorithm. Columns named *SC16/32* and *SL16/32* correspond to the ratios of complexity between FS and Fast Algorithms for Checking points and Line-SAD respectively.

It is evident that the ADZS algorithm has similar or superior performance to DS, in either speed-up or PSNR, for all cases presented. For the first two cases (*container* and *silence*), and the *tennis* sequence, even though PSNR is very similar, the speed up of ADZS is almost double of that of DS. For *news*, the average PSNR is slightly smaller, but the difference in video quality was not actually visible. Still the speed-up of ADZS is again significant. In *coastguard*, a sequence with significant scene variation (water in coastguard), even though the speed up of both algorithms is relatively similar, the PSNR value of ADZS is much higher than that of DS. Finally, in *foreman*, the algorithm yields higher PSNR and is much faster than DS. Apparently, ADZS can achieve slightly larger speedup or slightly better PSNR.

The total timing required by the encoder for all simulations is also shown in our table. From the results, it is evident that even though fast motion estimation algorithms may be significantly faster than FS, the total encoding time is non-proportionally reduced (only 2 to 5 times). This is expected since other portions of the MPEG-4 encoder also require substantial computation.

Even though we have set default parameters for the algorithm, it is possible to give more flexibility to the system by allowing the user to select different values for the above thresholds and criteria, or even to disable or enable the different options. This allows the user to achieve different tradeoffs between speed and quality, depending on the application. The current thresholds have been selected after performing extensive simulations and tests under various testing conditions [6]. More powerful, adaptive techniques for the selection of these parameters are currently under development, which can enhance the performance of the algorithm even further.

5. Conclusion

In this paper we have presented a new, efficient motion estimation algorithm. Our results demonstrate its superiority versus the DS algorithm. The algorithm can significantly reduce the complexity of the MPEG-4 encoder vs. the FS algorithm without sacrificing quality. It is also possible to refine performance by modifying the different parameters of the algorithm, or by introducing adaptive techniques.

6. References

[1] 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.

[2] 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. 121-124, Apr 21-23, 1999.

[3] 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.

[4] A.M. Tourapis, O.C. Au, M.L. Liou, and G. Shen, "An Advanced Zonal Block Based Algorithm for Motion Estimation," *1999 IEEE International Conference on Image Processing (ICIP'99) Proceedings*, section 26PO3.1, Kobe, Japan, October 1999.

[5] A.M. Tourapis, O.C. Au, and M.L. Liou, "The Second Status Report of Core Experiment on Fast Block-Matching Motion Estimation using Half Stop Circular Zonal Search," in *ISO/IEC JTC1/SC29/WG11 MPEG9/m4239*, Roma, Italy, December'98.

[6] A.M. Tourapis, O.C. Au, and M.L. Liou, "Status Report of Core Experiment on Fast Block-Matching Motion Estimation using Half Stop Zonal Search with Adaptive Search Area," in *ISO/IEC JTC1/SC29/WG11 MPEG99/m4580*, Seoul, Korea, March'99.

[7] A.M. Tourapis, O.C. Au, M.L. Liou, and G. Shen, "Status Report of Core Experiment on Fast Block-Matching Motion Estimation using Diamond Zonal Search with Embedded Radar," in *ISO/IEC JTC1/SC29/WG11 MPEG99/m4917*, Vancouver, Canada, July'99

[8] A.M. Tourapis, O.C. Au, M.L. Liou, and G. Shen, "Status Report of Core Experiment on Fast Block-Matching Motion Estimation using Advanced Diamond Zonal Search with Embedded Radar," in *ISO/IEC JTC1/SC29/WG11 MPEG99/m4980*, Melbourne, Australia, October'99

[9] 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.

[10] 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. 369-377, Aug. 1998.

[11] 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, Aug. 1994, pp. 438-42.

Table 1: PSNR, complexity, and total encoding time of FS, DS, and ADZS

Sequence	Size	BR	FR	SA	Algorithm	PSNR-Y	PSNR-U	PSNR-V	bits	Check Pt.	SC16	SC32	Line-SAD	SL16	SL32	User	System	Total	ST16	ST32
Container	QCIF	10	7.5	16	FS	29.81	37.54	36.60	98792	7501824			68302357			121.6	1.01	122.6		
					DS	29.76	37.43	36.58	99752	96969	77	280	980438	70	221	38.51	0.86	39.37	3.11	7.73
					ADZS	29.78	37.49	36.67	98960	42840	175	634	432575	158	500	37.88	0.68	38.56	3.18	7.89
				32	FS	29.72	37.55	36.57	98912	27142090			216387987			303.3	0.97	304.2		
					DS	29.74	37.48	36.69	98912	97030	77	280	983232	69	220	38.42	0.83	39.25	3.12	7.75
					ADZS	29.79	37.57	36.63	99136	43148	174	629	433649	158	499	37.83	0.84	38.67	3.17	7.87
Silence	QCIF	24	10	16	FS	30.82	35.21	36.60	238560	10036224			83127826			152.5	1.45	154		
					DS	30.92	35.29	36.73	239024	146141	69	248	1563459	53	163	51.9	1.28	53.18	2.9	6.77
					ADZS	30.95	35.38	36.78	239032	97658	103	372	1203564	69	211	51.25	1.28	52.53	2.93	6.85
				32	FS	30.90	35.29	36.63	238992	36311715			254215255			358.5	1.49	360		
					DS	30.93	35.34	36.76	239752	146243	69	248	1564111	53	163	52.18	1.07	53.25	2.89	6.76
					ADZS	30.85	35.25	36.73	239808	98459	102	369	1208703	69	210	51.54	1.04	52.58	2.93	6.85
News	CIF	112	15	16	FS	34.05	38.04	38.93	1118416	60420096			485966112			903.2	3.84	907		
					DS	34.02	38.10	38.97	1119584	832799	73	276	7859307	62	204	307.8	4.27	312.1	2.91	7.21
					ADZS	33.86	37.99	38.99	1119736	292760	206	786	3506010	139	457	302.6	5.22	307.8	2.95	7.31
				32	FS	34.03	37.93	38.85	1115936	229998933			1601397125			2246	3.94	2250		
					DS	33.99	38.07	38.99	1119336	835773	72	275	7924857	61	202	307.4	4.09	311.5	2.91	7.22
					ADZS	33.85	38.02	38.92	1115144	293583	206	783	3508813	139	456	302.8	4.37	307.2	2.95	7.33
Coastguard	CIF	112	10	16	FS	27.03	38.87	41.65	1112576	40144896			436958221			767.5	4.7	772.2		
					DS	26.44	38.79	41.46	1113232	811384	49	188	10979746	40	136	217.5	5.49	223	3.46	9.50
					ADZS	27.07	39.10	41.65	1112400	837308	48	183	10636965	41	140	215	5.64	220.6	3.50	9.61
				32	FS	27.06	38.64	40.99	1112656	153000000			1494261189			2114	5.21	2119		
					DS	26.47	38.77	41.59	1117360	818603	49	187	11072882	39	135	218.4	5.21	223.7	3.45	9.48
					ADZS	27.06	38.99	41.60	1115176	845089	48	181	10749119	41	139	215.6	5.84	221.5	3.49	9.57
Foreman	CIF	512	15	16	FS	34.51	40.25	41.47	5121912	56770560			466055393			877.4	1.54	878.94		
					DS	34.07	39.96	41.17	5121848	1207842	47	179	15329481	30	96	321.5	1.63	323.13	2.72	6.53
					ADZS	34.41	40.19	41.45	5121776	730942	78	296	9488139	49	155	314.7	1.75	316.45	2.78	6.67
				32	FS	34.84	40.56	41.75	5121960	216106380			1469593841			2109	1.64	2110.6		
					DS	34.09	39.97	41.17	5121920	1248552	45	173	15932054	29	92	322.7	1.85	324.55	2.71	6.5
					ADZS	34.40	40.20	41.46	5121784	731238	78	296	9479108	49	155	313.5	2.35	315.85	2.78	6.68
Tennis	SIF	1024	30	16	FS	34.98	41.89	41.01	10240968	94617600			841202773			1582	2.50	1584.5		
					DS	34.92	41.81	40.93	10241200	1383397	68	259	12729904	66	221	523	2.22	525.22	3.02	7.69
					ADZS	34.95	41.83	40.96	10240880	541962	175	661	6059421	139	465	514.8	2.46	517.26	3.06	7.8
				32	FS	35.00	41.91	41.02	10241208	358185240			2817021417			4034	2.6	4036.6		
					DS	34.90	41.81	40.92	10241192	1386257	68	258	12787826	66	220	522.9	2.27	525.17	3.02	7.69
					ADZS	34.91	41.82	40.95	10241080	544757	174	658	6085037	138	463	513.7	2.15	515.85	3.07	7.83