FAST MULTI-FRMAE MOTION ESTIMATION ALGORITHM WITH ADAPTIVE SEARCH STRATEGIES IN H.264

Xiang Li, Eric Q. Li, and Yen-Kuang Chen

EE department of Tsinghua Univ., Beijing
Intel China Research Center, Intel Corporation, Beijing
Architecture Research Lab, Intel Corporation, Santa Clara

ABSTRACT

In the new H.264/AVC video coding standard, motion estimation takes up a significant encoding time especially when using the straightforward full search algorithm (FS). In this paper, a fast flexible multi-frame motion estimation algorithm with adaptive search strategies (FMASS) is presented. With special considerations on the multiple reference frames and block modes, several techniques, i.e., adaptive search strategies for single frame and flexible multi-frame selection have been utilized to significantly improve the speed-up performance in H.264. Extensive simulations show that it can minimize the matching points by more than 1190 times compared with FS. In addition, the output quality of the encoded sequences losses only 0.053dB in terms of PSNR on average. Fast speed-up performance and unnoticeable quality losses make the proposed algorithm outperform most of other famous algorithms proposed in recent years, such as ARPS3, MVFAST and UMHexagonS, among which the latter two have been already accepted by MPEG-4 and JVT respectively.

1. INTRODUCTION

H.264/AVC [1] is the state-of-the-art video compression standard recently developed by the ITU-T/ISO/IEC Joint Video Team (JVT). The new standard is aimed at high-quality coding of video contents at very low bit-rates. Compared with H.263+/MPEG-4 advanced simple profiles, up to 50% of bit-rate reduction can be achieved [2].

H.264 uses the same hybrid block-based motion compensation and transform coding model as those existing standards. Moreover, a number of new features and capabilities, such as multi-frame (up to five frames) motion estimation and seven block modes, have been introduced in H.264 to efficiently improve the coding performance. However, the complexity of H.264 encoder increases tremendously with these new features. Particularly, the integer pixel motion estimation module is the most time-consuming one, which consumes more than 90% execution time in H.264 encoder.

Several famous motion estimation algorithms have been proposed to reduce the motion estimation complexity, such as Block-Based Gradient Descent Search [3], Unrestricted Center-Biased Diamond Search [4], HEXagon-Based Search [5] and MVFAST (Motion Vector Field Adaptive Search Technique) [6]. Though successfully used in the previous video coding standards, they are not particularly designed for H.264 and their performances are not very satisfying with H.264. Recently, several fast motion estimation algorithms were proposed for H.264, such as UMHexagonS (Unsymmetrical-cross Multi-Hexagon-grid Search) [7-8] and ARPS-3 (unequal-arm Adaptive Rood Pattern Search) [9]. UMHexagonS was accepted by JVT, claiming that more than 90%~95% computations can be saved on average compared with the fast full search algorithm in JVT reference software with a fairly good PNSR performance. However, the speed-up of UMHexagonS is not very outstanding, much slower than most of the classic algorithms, such as MVFAST.

In this paper, we propose a novel fast integer pixel motion estimation algorithm with adaptive search strategies for H.264 encoder. Two major features, viz. "Flexible Multi-frame search criterion" and "Adaptive Search Strategies for single frame integer pixel motion estimation" have been characterized. Our proposed algorithm is much faster than any other algorithm mentioned above while maintaining an unnoticeable quality loss in terms of PSNR compared with FS.

The paper is organized as following: Section 2 describes the proposed algorithm in detail. Section 3 shows the simulation results and some discussions. Finally section 4 concludes the whole paper.

2. ALGORITHM DESCRIPTION

In this section, the framework of the proposed algorithm will be described first. In the following, we will discuss the two major features in detail. Finally we will summarize the whole algorithm.

2.1. Framework of the Proposed Algorithm

Since we have to search seven block modes and five reference frames for each macroblock, the framework of the proposed algorithm looks like some loops. The motion search starts from the 16x16 block mode to the 4x4 block mode in a hierarchical descend order for each macroblock. Within each block mode, the reference frames are searched from the most recent (defined as ref0) to the least recent frame (defined as ref4). Adaptive motion search strategies will be applied on each reference frame. After that, decisions on whether to make further searches on ref3 and ref4 will be made according to the flexible multi-frame search criterion. In case the whole motion search process finishes, we will get the best motion vector (MV), reference frame and some other corresponding information.

2.2. Adaptive Search Strategies for Single Frame Motion Estimation

For single frame motion search, we use a MV distribution model to select a best strategy in integer pixel motion estimation. First, a hierarchical MV prediction is used. If most of the predictors are entirely same, then stop directly; otherwise make further searches with cross or hexagon pattern according to the MV distribution model. In order to quicken the whole process, early termination and zero-block detection techniques are also employed.

2.2.1. MV Probability Distribution Model

In general, the MVs, especially the MV differences after median prediction, approximately comply with symmetrical exponential distribution. Assuming the independence of MVs in X and Y axis, the combined distribution probability can be defined as:

$$P_{MV}(x,y) = P_X(x)P_Y(y) \tag{1}$$

Where $P_X(x)$ and $P_Y(y)$ are the probability distribution of MVs in X and Y dimension respectively. Both of them comply with the exponential distribution defined in Equation (2).

$$P(n) = \begin{cases} \lambda, n = 0 \\ \frac{1}{2}\lambda\beta^{|n|}, n \neq 0 \end{cases} \tag{2}$$

Considering the physical meaning of probability distribution, (2) should satisfy the constraint that the sum of probability in search window is 1, namely

$$\sum_{n=-W}^{W} P(n) = 1 \tag{3}$$

Where W denotes the size of the search window.

Meanwhile, we can define the median absolute value of MVs as

$$MV_{median} = \sum_{n=-W}^{W} P(n)(|n|+1) \tag{4}$$

According to (2) – (4), we can get

$$\begin{cases} \lambda = \dfrac{1}{MV_{median}} \\ \beta = 1 - \dfrac{1}{MV_{median}} \end{cases} \tag{5}$$

Eventually, the probability distribution model of MVs in single frame can be derived from MV_{median}. Since MV_{median} varies comparatively slowly for the whole video sequence, it will be used as a key parameter for the future frames prediction.

Studies [10] show that MVs are more likely distributed in a diamond-shaped area. More theoretically, it is possible to find a diamond region where more than 99% MVs are located according to this model. The edge length of this adaptive diamond is derived from the actual MV data, which is very similar to MV_{median}. Here we denote this diamond edge length as ADL. It will be used to select the appropriate search strategies in the following procedures.

2.2.2. Hierarchical MV Prediction

Considering the high correlations among MVs and the spatial/temporal properties of H.264, a hierarchical MV prediction criterion is employed to find best initial search points. Four types of predictors are used, where the first two are spatial and the other two are temporal predictors.

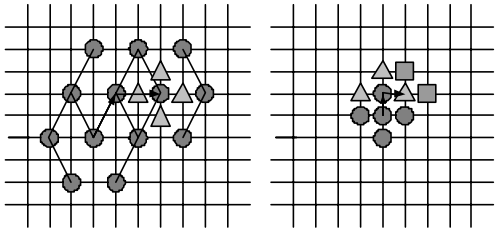

Fig. 1. (A) Hexagonal Search (B) Cross Search

- MVs from the surround blocks of same mode (top, top-right and left)
- MVs from uplayer blocks mode on the same position (e.g. 16x16 is the uplayer block mode of 16x8 and 8x16).
- MVs from directly temporal predictor
- MVs from accelerated temporal predictor

The latter two can be defined as (6) and (7) respectively, where (6) denotes the MV from the corresponding block of the recent reference frame and (7) represents the scaled MV from previous reference frames.

$$MV_{DTemp} = MV_{curfrm-1} \tag{6}$$

$$MV_{ATemp} = MV_{curref-1} \times \frac{curref+1}{curref} \tag{7}$$

If the number of same predictors is above a threshold, the search process of current reference frame terminates immediately. Otherwise, all predictors will be searched to find a best match. Subsequently, we compare the best match position with the original point, choose a better one, take it as the initial search point and mark it as "prediction hits".

2.2.3. Search strategies based on the probability model

If prediction does not hit, a more precise search pattern will be selected between cross search and hexagon search according to the MV distribution model.

First, we calculate the ADL based on the MVs of the previous frame and take this ADL as the basic prediction parameter (ADL_{Pred}) for the current frame. After that, we adjust the ADL_{Pred} according to the MVs from the surrounded blocks of same mode as defined in (8).

$$ADL_{Pred} = \begin{cases} 4, if\ Average(|MV_{Top}|, |MV_{TopRigh}|, |MV_{Left}|) > 4 \\ ADL_{Pred}, otherwise \end{cases} \tag{8}$$

Hereafter, if we predict that MV locates in a small area, e.g., ADL_{Pred} is smaller than *4*, cross search will be used. Otherwise we will choose hexagon search method.

2.2.4. Early termination and Zero-block detection

In order to further quicken the process of integer pixel motion estimation, a simple early termination and zero-block detection techniques are also used. For early termination, we take the minimum matching difference of the top, top-right and left blocks of the same mode as the adaptive threshold. If the matching difference of current block is lower than the threshold, the search process terminates immediately (refer to [11] for more discussions). In addition, similar technique of zero-block detection [7] is also used in our algorithm.

2.3. Flexible Multi-reference Frame Search Criterion

Multi-reference frame prediction is a distinguished feature to substantially improve the coding efficiency in H.264. However, the probability that ref3 and ref4 are used as the best reference frames is very low, less than 3% in common. Since these two reference frames are far away from the current frame, the MVs are always larger than other reference frames, which may yield more search points and low search efficiency. Thus, we propose a flexible multi-frame search criterion, which can automatically determine whether to search ref3/ref4 or not.

First, we can obviously notice that ref3/ref4 will be more likely selected with conditions of vibration or smooth moving. Suppose a scene of shaking one's head. There may be some past frames very similar to the current one, and ref3/ref4 has larger possibility to be selected as the best reference frame. However, with other conditions, such as fast moving, static scenario and etc, skipping ref3/ref4 is a better choice.

Fig.2 shows a representative example of MVs of a straight-line moving object. It can be easily observed that the MVs are in proportion and Ref0 has a much larger probability to be the best reference frame.

Therefore, we can derive (9) according to the previous mentioned analysis, where T_{MV} is a threshold of difference (equal to 4 in our simulation).

$$\begin{cases} \left|2\times MV_{ref0} - MV_{ref1}\right| < T_{MV} \ \ and \ \left|3\times MV_{ref0} - MV_{ref2}\right| < T_{MV} \\ Cost_{ref0} < Cost_{ref1} \quad and \quad Cost_{ref0} < Cost_{ref2} \end{cases} \quad (9)$$

If the MVs and matching costs satisfy (10), ref3/ref4 will be skipped. Otherwise we will continue searching the remaining two frames. With this criterion, we can further minimize the computations of integer pixel motion estimation with only a negligible loss in PSNR.

2.4. Summary of the proposed algorithm

The proposed algorithm can be described as follows:

Loop1 Search Mode (from 16x16 to 4x4);
Loop2 Search Ref (from ref0 to ref4);
S1 Get ADL_{Pred};
S2 Perform the hierarchical MV prediction, if the number of the same predictors is larger than the threshold, take the predictor as the MV and go to S6;
S3 Do the zero-block detection on the initial point, if the block is zero-block, take the initial point as the MV and go to S6;
S4 If prediction hits, refine the MV with CS and early termination technique, then go to S6;
S5 Select a better strategy between cross search and hexagon search according to ADL_{Pred}, after that make further searches together with early termination and zero block detection techniques;
S6 If the current reference frame is ref2, make decisions on continuing searching ref3/ref4 (go to Loop2) or terminating the search process on the current block mode (go to Loop1), otherwise search the next reference frame (go to Loop2).

3. SIMULATION RESULTS AND DISCUSSIONS

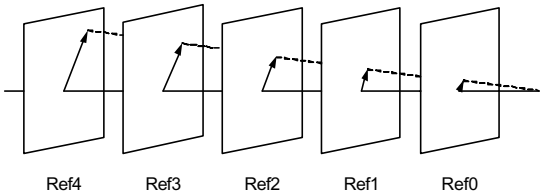

Fig. 2. MVs of a moving object

Our proposed algorithm FMASS was implemented into the JVT reference model JM6.1e of H.264. There are 19 test sequences used in our simulation covering from QCIF to CIF resolution. Additionally, we choose high-motion, median-motion as well as low-motion sequences to make the experiments more comprehensive. E.g., Bus, Football and Stefan are sequences with large motions. Carphone, Foreman and Salesman are those with medium motions, and Clare, Grandmother are sequences with low motions or still sequences. The coding settings comply with the specification provided by [12], i.e., five reference frames, seven block modes, a search window of [-32, 32] and CABAC entropy coding. Apart from the first frame, all the other frames are coded as P frames and the total frame rate is 30 Hz. For comparison use, we implemented three famous fast algorithms which were proposed recently: MVFAST, ARPS-3 and UMHexagonS[1]. It should be noted that MVFAST was accepted by MPEG4 in 2000 and UMHexagonS was recently accepted by JVT in March, 2003. Table I[2] summarizes the PSNR losses (Y component) and speed improvement of fast algorithms compared with FS at the same bit rate. Where the PSNR is calculated by the AVSNR [12] specified by JVT. According to table I, though the average PSNR loss of UMHexagonS is the lowest, our proposed algorithm also shows a comparatively optimal performance. Compared with FS, the maximal loss in PSNR of FMASS is 0.1 dB, and the average loss is only about 0.05dB, which is almost negligible for fast motion estimation algorithms. On the other hand, it can be easily observed that FMASS substantially outperforms other fast algorithms in terms of speed-up performance. Compared with FS, the average block matching points of FMASS is only about 17.7 points, less than 50% of MVFAST and 3% of UMHexagonS. Moreover, it is also faster than another famous algorithm PMVFAST (also accepted by MPEG-4), the matching points of which is about 70% of MVFAST [11]. As a matter of fact, FMASS is much faster than other fast algorithms in execution time as well, more than 510 times faster than FS and 16 times faster than UMHexagonS. We should note that the matching points and the executing time are not in linear proportion, since the fast algorithms introduce more condition judgments which may lead to a high penalty in speed especially for fast speed CPUs.

Since the computations of integer pixel motion estimation have been tremendously reduced with our algorithm, the sub-pixel motion estimation module becomes the most time consuming one in the encoder now. As a matter of fact, we have also developed a

[1] The algorithm used in the simulation of UMHexagonS is based on [7]. The final version is about twice as faster as this one with a little PSNR losses. Please refer to [8] for more details.

[2] The platform used in simulation is Pentium-4 3.0G with 800M FSB, WinXP, DDR 512M.

TABLE I SIMULATION RESULTS:
PSNR(dB), Points (Average Search Points per block), Time (Average Matching Time (ms/MB))

Sequences	FS		MVFAST			ARPS-3			UMHexagonS			FMASS		
	Points	Time	PSNR	Points	Time	PSNR	Points	Time	PSNR	Points	Time	PSNR	Points	Time
Carphone_qcif	21125	188.3	-0.137	40.38	0.473	-0.135	39.75	0.490	-0.044	762.6	5.828	-0.107	21.43	0.405
Claire_qcif	21125	187.7	-0.096	16.98	0.250	-0.074	28.08	0.370	-0.018	535.6	4.048	-0.048	7.65	0.245
Container_qcif	21125	187.8	-0.042	22.83	0.308	-0.051	26.58	0.360	-0.014	777.3	5.850	-0.055	7.48	0.243
Foreman_qcif	21125	186.4	-0.103	52.68	0.585	-0.100	53.15	0.603	-0.004	795.6	6.133	-0.071	23.85	0.438
M&D_qcif	21125	188.6	-0.026	27.83	0.348	-0.012	29.53	0.390	0.006	693.1	5.183	-0.034	9.15	0.263
News_qcif	21125	186.3	-0.079	24.25	0.313	-0.057	29.53	0.385	0.001	725.2	5.438	-0.043	9.58	0.265
Salesman_qcif	21125	188.7	-0.028	24.23	0.315	-0.021	27.13	0.370	-0.006	818.7	6.155	-0.014	7.65	0.250
Stefan_qcif	21125	186.5	-0.267	49.38	0.558	-0.350	51.20	0.593	-0.041	774.4	6.023	-0.072	24.40	0.453
Bus_sif	21125	173.4	-0.258	52.58	0.540	-0.258	56.23	0.553	-0.009	791.2	5.465	-0.025	19.90	0.360
Flower_sif	21125	171.1	-0.084	47.83	0.500	-0.109	53.25	0.535	-0.017	778.8	5.318	-0.089	17.88	0.330
Football_sif	21125	170.4	-0.142	63.00	0.608	-0.123	66.78	0.615	-0.021	782.2	5.355	-0.048	37.10	0.510
Tennis_sif	21125	172.8	-0.123	43.15	0.455	-0.136	45.35	0.465	-0.003	808.6	5.533	-0.041	17.70	0.333
Foreman_cif	21125	165.7	-0.239	60.10	0.600	-0.138	63.38	0.585	-0.023	782.3	5.368	-0.099	29.63	0.443
Stefan_cif	21125	165.5	-0.473	52.93	0.553	-0.407	58.65	0.565	-0.050	753.8	5.215	-0.073	29.30	0.440
Container_cif	21125	163.2	-0.054	29.65	0.335	-0.054	28.20	0.333	-0.006	773.4	5.175	-0.074	8.70	0.243
Coastguard_cif	21125	162.8	-0.066	53.50	0.555	-0.018	53.83	0.530	0.008	792.5	5.435	-0.013	22.03	0.363
Mobile_cif	21125	163.5	-0.422	49.00	0.530	-0.026	45.33	0.475	0.001	804.1	5.528	-0.032	15.00	0.310
Salesman_cif	21125	162.4	-0.035	28.85	0.333	-0.027	29.55	0.340	-0.011	815.2	5.445	-0.027	9.50	0.248
Tempete_cif	21125	162.7	-0.087	47.75	0.508	-0.021	42.50	0.450	0.000	784.0	5.350	-0.041	18.38	0.343
Average	21125	175.5	**-0.145**	41.41	0.456	**-0.111**	43.58	0.474	**-0.013**	765.7	5.465	**-0.053**	17.70	0.341
Aver Speed-up	1.0	1.0	--	**510.1**	**384.9**	--	**484.8**	**370.2**	--	**27.59**	**32.11**	--	**1194**	**514.5**

fast algorithm for sub-pixel motion estimation, which is able to further improve the speed-up performance without any losses in PSNR. However, we could not discuss more about it due to the page length limitation.

4. CONCLUSION

H.264/AVC is the state-of-the-art video compression standard developed by JVT. Though the new introduced features improve the coding performance efficiently, the complexity of the integer pixel motion estimation of H.264 increases tremendously as well. In this paper, we propose a fast flexible multi-frame motion estimation algorithm based on adaptive search strategies. With more considerations on the multiple reference frames and block modes, the proposed algorithm reduces the complexity of the integer pixel motion estimation significantly. With the Intel Pentium-4 3.0GHz system, we have achieved more than 510 times speed-up over FS on average with unnoticeable losses in PSNR. Furthermore, the proposed algorithm is much faster than many other famous fast algorithms in various conditions. Since the average matching points of the proposed algorithm for five reference frames is reduced to less than 20 points, it almost approaches the lowest bound of the fast algorithms and more focus should be laid on other computation intensive modules in future.

REFERENCES

[1] Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC) - Joint Committee Draft. Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, 7th Meeting: Pattaya, Thailand, 7-14 March, 2003.

[2] Gary J. Sullivan, Thomas Wiegand, Thomas Stockhammer, "Using the Draft H.26L Video Coding Standard for Mobile Applications", *Proc. IEEE International Conference on Image Processing (ICIP2001)*, Thessaloniki, Greece, Sep. 2001, invited paper.

[3] Lurng-Kuo Liu, Ephraim Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, pp, 419-422, Aug.1996.

[4] Jo Yew Tham, Surendra Ranganath, Maitreya Ranganath, and Ashraf Ali Kassim, "A novel unrestricted center-biased diamond search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 8, pp. 369-377, Aug. 1998.

[5] Ce Zhu, Xiao Lin, and Lap-Pui Chau, " Hexagon-Based Search Pattern for Fast Block Motion Estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 349-355, May 2002.

[6] P. I. Hosur and K. K. Ma, "Motion vector field adaptive fast motion estimation," *in Proc. 2nd Int. Conf. Information, Communications and Signal Processing (ICICS'99)*, Singapore, Dec. 7-10, 1999.

[7] Zhibo Chen, Peng Zhou, Yun He, "Fast Integer Pel and Fractional Pel Motion Estimation for JVT", JVT-F017r1.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 6th meeting, Awaji, Island, JP, 5-13 December, 2002

[8] Zhibo Chen, Peng Zhou, Yun He, "Fast Motion Estimation for JVT", JVT-G016.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 7th Meeting: Pattaya II, Thailand, 7-14 March, 2003

[9] Kai-Kuang Ma, Gang Qiu, "Unequal-arm Adaptive Rood Pattern Search for fast Block-matching Motion Estimation in the JVT/H.26l", *Proc. IEEE International Conference on Image Processing (ICIP2003)*, Barcelona, Spain, Sep. 2003.

[10] 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. IEEE Int. Sym. on Circuits & Systems (ISCAS-2000)*, Geneva, Switzerland, May, 2000.

[11] Alexis M. Tourapis, Oscar C. Au, Ming L. Liou, "Highly efficient predictive zonal algorithms for fast block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, pp. 934-947, Oct. 2002.

[12] G. Sullivan, "Recommended Simulation Common Conditions for H.26L Coding Efficiency Experiments on Low-Resolution Progressive-Scan Source Material", VCEG-N81.doc, ITU-T VCEG, 14th Meeting: Santa Barbara, CA, USA, 24-27, Sep., 2001.