Transactions Letters

An Effective Variable Block-Size Early Termination Algorithm for H.264 Video Coding

Libo Yang, Keman Yu, *Member, IEEE*, Jiang Li, *Senior Member, IEEE*, and Shipeng Li, *Member, IEEE*

***Abstract*—The H.264 video coding standard provides considerably higher coding efficiency than previous standards do, whereas its complexity is significantly increased at the same time. In an H.264 encoder, the most time-consuming component is variable block-size motion estimation. To reduce the complexity of motion estimation, an early termination algorithm is proposed in this paper. It predicts the best motion vector by examining only one search point. With the proposed method, some of the motion searches can be stopped early, and then a large number of search points can be skipped. The proposed method can work with any fast motion estimation algorithm. Experiments are carried out with a fast motion estimation algorithm that has been adopted by H.264. Results show that significant complexity reduction is achieved while the degradation in video quality is negligible.**

***Index Terms*—Early termination, H.264, motion estimation, variable block-size.**

I. Introduction

RECENT years have seen rapid development of video coding techniques. Generally, compression performance is improved along with an increase of computational cost. H.264, as the newest joint standard of the ITU-T Video Coding Experts Group (VCEG) and the ISO/IEC Moving Picture Experts Group (MPEG), has achieved a significant improvement in coding efficiency compared to previous standards, such as MPEG-1/2/4 and H.261/H.263. However, its complexity is too high to be widely applied in real-time applications.

In general, motion estimation (ME) contributes most of the complexity of a video encoder. There are two ways to reduce the computational cost. The first one is to speed up the algorithms themselves. For ME, numerous fast algorithms have been proposed, such as hexagon-based search (HBS) [5], enhanced predictive zonal search (EPZS) [3], and hybrid unsymmetrical-cross multihexagon-grid search (UMHexagonS) [1]. The other way is to terminate the ME calculation early. By predicting the blocks whose discrete cosine transform (DCT) coefficients will quantize to zeros, some methods [2], [6], [4] effectively reduce the computation of ME. On the other hand, a significant portion of blocks have a zero motion vector (MV) after ME. The zero-motion detection (ZMD) algorithm proposed in [4] detects such blocks by comparing their sum of absolute difference (SAD) with a predefined threshold and then skips the remaining search points.

However, the aforementioned early termination methods are all developed for previous coding schemes, such as H.263. They cannot be applied to H.264 any longer. This is because compared to H.263, where only two block sizes (16×16 and 8×8) are used, seven block sizes varying from 16×16 to 4×4 are used in H.264.

Extending the concept of ZMD, we propose a variable block-size best motion detection (VBBMD) algorithm for H.264 video coding. The proposed method differs from ZMD in three aspects: 1) the thresholds used in VBBMD are obtained based on the detection accuracy of different block sizes, respectively, while complexity reduction is also considered, which makes the threshold decision more reasonable; 2) the predicted motion point, rather than the zero motion point, is checked in VBBMD, so that more search points can be skipped; and 3) dual thresholds are utilized for the block size of 16×16 in VBBMD. The lower threshold is used to skip the motion search of smaller block sizes. Our method can work with any motion search algorithm and no additional computation is required. The entire discussion is focused on inter-frames (P-frames) and integer pixel motion search only.

The rest of this paper is organized as follows. Section II describes the early termination algorithm and its improvement. Simulation results are shown in Section III. Finally, we conclude the paper and give future directions in Section IV.

II. Early Termination Algorithm

A. Variable Block-Size Zero Motion Detection

Generally, for most sequences, there exist a significant number of blocks that have a zero MV after ME, as shown in Table I. In this table, mode 1–7 represent seven inter prediction modes with different block sizes varying from 16×16 to 4×4 in H.264, as shown in Fig. 1. It can be seen that 30.71% to 98.03% of blocks have zero motion in different video sequences with low and high motion activities. If we can predict the zero-motion blocks (ZMB), we can stop ME early and eliminate a portion of the computation. Extending the concept of ZMD, we develop a variable block-size ZMD (VBZMD) algorithm.

Manuscript received July 23, 2004; revised December 9, 2004. The work presented in this paper was carried out at Microsoft Research Asia. This paper was recommended by H. Gharavi.

L. Yang is with the Department of Information and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: lb_yang@hotmail.com).

K. Yu, J. Li, and S. Li are with the Microsoft Research Asia, Beijing 100080, China (e-mail: kmyu@microsoft.com; jiangli@microsoft.com; spli@microsoft.com).

Digital Object Identifier 10.1109/TCSVT.2005.848306

TABLE I
ZMB RATES (%) OF SEVEN MODES (QP = 32)

Sequence	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Akiyo	98.03	97.51	97.18	97.28	97.14	96.47	97.04
Salesman	96.80	95.92	95.69	95.15	94.78	94.13	94.33
News	91.90	91.72	91.98	92.35	91.93	91.31	91.78
Silent	85.01	84.65	84.15	85.01	84.75	84.23	85.06
Coastguard	32.44	33.51	32.80	30.91	31.40	30.71	31.06
Foreman	43.97	44.15	43.57	44.25	44.81	43.73	45.42

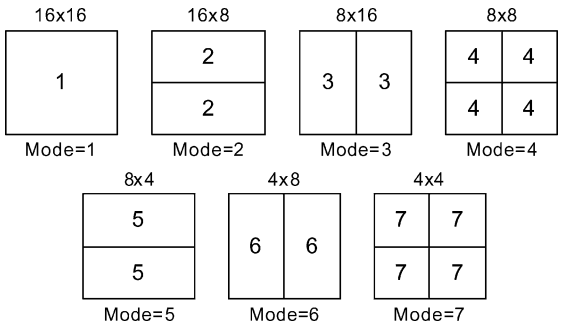

Fig. 1. Seven prediction modes in H.264.

In the previous ZMD method, when the SAD of a block is smaller than a given threshold, the block can be regarded as a ZMB. In H.264, the cost function J, rather than SAD, is used as the measure of prediction error in selecting the best matching block

$$J(m,\lambda) = \mathrm{SAD}(s, c(m)) + \lambda R(m-p) \tag{1}$$

where $m = m\,(m_x, m_y)^T$ is the current MV, $p = (p_x, p_y)^T$ is the predicted MV, and λ is a Lagrangian multiplier. $R(m-p)$ is the number of bits to code the MV.

If a block has zero motion, it is likely to have a small cost at $\mathrm{MV}(0,0)$. Therefore, we define thresholds $\mathrm{THZ}_i\ (i = 1, \ldots, 7)$ for seven block sizes, respectively. During ME, $\mathrm{MV}(0,0)$ is first examined. If the cost at $\mathrm{MV}(0,0)$ satisfies (2), the block is regarded as a ZMB, and then the remaining searches can be skipped

$$J_i < \mathrm{THZ}_i, \quad \text{for } i = 1, \ldots, 7. \tag{2}$$

How to determine the thresholds for different block sizes is the key to ZMD in H.264. It is clear that the larger the thresholds are, the more ZMBs are detected and the more search points can be skipped. However, more blocks are incorrectly selected at the same time, which results in more significant loss in image quality. Therefore, there is a tradeoff between performance and complexity. In practice, preventing loss in video quality may be more important than a minor increase in complexity. Therefore we regard detection accuracy as the guide in determining thresholds. Here the accuracy represents the probability of a block being ZMB when its cost is smaller than a threshold. Our method is to obtain several candidate sets of thresholds based on different accuracy rates in experiments and select an optimal set which provides a good tradeoff between quality and complexity in practice. Experiments on many sequences show that with the same threshold, both accuracy rates and detection rates in low motion scenes are higher than those in large motion scenes. As the Foreman sequence can represent scenes that possess relatively large motion, we can select thresholds based on Foreman and apply them to other sequences. Table II shows the candidate thresholds we obtain.

In our method, if the zero-motion cost of a block is smaller than the corresponding threshold, we regard $(0,0)$ as the best MV of the block and jump to the next block in the same macroblock. If the current block is the last block in a macroblock, we jump to the next mode. After all modes are examined, the mode with the minimal cost is selected as the best mode as any ME algorithm for H.264 does. In mode 1, since one macroblock contains only one block, if the zero-motion cost is small enough, it is likely that not only $(0,0)$ is the best MV, but also mode 1 is the best mode of this macroblock. Thus, we define an additional lower threshold THS for mode 1. If the zero-motion cost is smaller than THS, the motion search of the entire macroblock is stopped. The entire procedure of VBZMD is outlined as follows.

```
Assume the current macroblock is MB(x, y)
For each mode i (i = 1, ..., 7) of the MB(x, y)
  For each block in the macroblock
     Calculate the cost at (0, 0)
     If cost < THZ_i
       Set MV = (0, 0)
       If mode = 1 and cost < THS
         Set best mode = 1
         Exit two loops
       End if
       Exit one loop
     Else
       Do the normal motion search process
     End if
  End for
End for
If best mode is not set
  Set best mode to be the mode with the minimal cost
```

The H.264 encoder we use is version 6.1e of the reference JVT software [7]. We combine the proposed method with the fast ME algorithm UMHexagonS [1] that has been adopted by the H.264 standard. The fast ME method can reduce the complexity of integer-pixel ME by up to 90% in comparison to full search, whereas the calculations can be further reduced by using the proposed method. Because the threshold of 800 corresponds to a relatively high accuracy rate of 78% in mode 1 and it seems to work well in experiments, we choose it as THS. As shown in Table III, for the *Akiyo* sequence that represents

TABLE II
CANDIDATE THRESHOLDS FOR DIFFERENT ACCURACY RATES OF FOREMAN ($QP = 32$)

Accuracy	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7
80%	600	550	570	400	315	300	250
75%	960	650	660	480	360	350	270
70%	1650	920	890	600	410	400	300
65%	2270	1200	1160	700	470	460	350
60%	2800	1500	1430	880	530	530	400

TABLE III
PERFORMANCE OF VBZMD WITH CANDIDATE THRESHOLDS ($QP = 32$)

Sequence	Method	Accuracy	PSNR (dB)	PPMB	Bit-rate (kbps)
Akiyo	Fast ME		35.22	63.86	14.52
	Fast ME +VBZMD	80%	35.18	29.02	14.65
		75%	35.20	21.41	14.67
		70%	35.19	7.96	14.70
		65%	35.18	5.17	14.59
		60%	35.17	4.17	14.67
Foreman	Fast ME		32.90	160.48	79.34
	Fast ME +VBZMD	80%	32.87	124.97	79.00
		75%	32.86	112.85	79.55
		70%	32.82	88.75	80.49
		65%	32.77	64.77	82.19
		60%	32.75	49.39	84.73

small motion scenes, up to 93.47% of search points per macroblock (PPMB) are reduced while the average peak signal-to-noise (PSNR) degradation is not more than 0.05 dB and the increases of bit rates are marginal. Even for the *Foreman* sequence, which possesses larger facial motion and camera panning, calculations are also obviously saved and the PSNR loss is still slight. Here the number of search points is calculated at the macroblock level. For example, one search point in mode 4 with the block size of 8×8 equals to $1/4$ PPMB in this way. Here, the threshold set corresponding to the accuracy rate of 65% provides a good tradeoff between performance and complexity. The PSNR degradations on *Akiyo* and *Foreman* sequences are

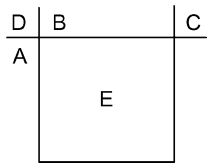

Fig. 2. Current and neighboring blocks.

TABLE IV
RATES OF PREDICTED VECTOR BLOCKS (%)

Sequence	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5	Mode 6	Mode 7
Akiyo	98.05	97.51	97.23	97.37	97.27	96.61	97.24
Salesman	96.84	95.71	95.46	95.40	95.11	94.54	95.03
News	91.80	91.70	91.66	92.83	92.71	92.22	93.16
Silent	85.09	85.04	84.57	86.32	86.80	86.36	88.28
Coastguard	58.68	62.19	62.41	64.32	66.17	67.09	69.04
Foreman	54.19	56.26	55.46	57.79	60.00	59.15	63.15

0.04 and 0.13, respectively, whereas the computation savings are 91.9% and 59.64%, respectively.

Intuitively, PSNR associated with high accuracy should be larger than PSNR associated with low accuracy. However, in this table, it can be seen that the PSNR values for *Akiyo* sequence with accuracy of 75% and 70% are larger than that with accuracy of 80%. This is possible because larger thresholds not only reduce more computation but also reduce the bits required to code MVs, and then the saved bits improve the overall PSNR.

B. Variable Block-Size Best Motion Detection

Although the VBZMD algorithm has achieved fairly good performance on reducing computation, we can further improve it to achieve better performance. In H.264, the median value of the left, top and top-right (or top-left) neighboring blocks' MVs is used as the prediction of the current block's MV, as illustrated in Fig. 2. Since the MVs of the neighboring blocks are usually correlated, the predicted vector is likely to be the best MV after ME. For the convenience of description, we define the blocks whose best MV is exactly the predicted vector as predicted vector blocks (PVB). As shown in Table IV, the PVB rates of different sequences are consistently larger than the ZMB rates shown in Table I. Particularly, for scenes with relatively large motion, such as *Coastguard* and *Foreman* sequences, PVB rates are higher than ZMB rates by 23%–122%. On the other hand, another experiment shows that for most sequences, more than 97% of ZMBs are also PVBs at the same time. The observations inspire us to use the predicted vector to predict the best MV instead of the zero MV. In addition, in H.264, the difference between the current MV and the predicted vector, rather than the current MV itself, is encoded and transmitted. Therefore, using the predicted vector to predict the best MV can save bits that are used to code MVs.

TABLE V
PERFORMANCE OF VBBMD WITH THE OPTIMAL THRESHOLDS ($\mathrm{QP} = 32$)

Sequence	Method	PSNR (dB)	PPMB	Bit-rate (kbps)
Akiyo	Fast ME	35.22	63.86	14.52
	Fast ME+VBZMD	35.18	5.17	14.59
	Fast ME+VBBMD	35.18	4.17	14.59
Foreman	Fast ME	32.90	160.48	79.34
	Fast ME+VBZMD	32.77	64.77	82.19
	Fast ME+VBBMD	32.77	30.12	82.34

Based on the preceding VBZMD, we propose a VBBMD algorithm. The key idea is to compare the cost at the predicted vector with a given threshold. If the cost is smaller than the threshold, the predicted vector is regarded as the best MV, and then the remaining search points can be skipped.

Similarly, how to select the thresholds for different block sizes is still the key issue. Using the same approach in VBZMD, we can obtain several candidate sets of thresholds in terms of different accuracy rates. The procedure of VBBMD is almost the same as that of VBZMD, only except for checking the predicted vector instead of the zero MV. We encode different video sequences with the candidate threshold sets and select an optimal set. The results show that the set of thresholds (2500, 1500, 1400, 920, 625, 570, 500) with accuracy of 66% provides a good tradeoff. Moreover, the lower threshold THS for mode 1 is also 800 because it works well. Table V shows the results of VBBMD with the optimal threshold set. Compared to the results of VBZMD in the same table, we can see that more search points are further reduced, while the PSNR and bit rates remain almost the same. It is verified that VBBMD is more effective than VBZMD.

In the foregoing descriptions, the thresholds are determined when the quantization parameter (QP) is set to 32. However, experiments show that different QP values result in different thresholds. This is because the cost value J is related to QP. In (1), J is directly related to λ, while λ is obtained from a lookup table indexed by QP. Therefore, we need to adjust the thresholds according to the value of QP. When a block is regarded as a PVB, the maximal difference between the current MV m and the predicted MV p in (1) is $3/4$ pixels, which can be encoded with 10 bits. Thus, we approximately set $R(m-p)$ to 10. We define the thresholds for $\mathrm{QP} = 32$ to be the principal thresholds, then the thresholds for other QP values can be set as the principal thresholds plus $(\lambda_{\mathrm{QP}} - \lambda_{32}) \times 10$.

III. SIMULATION RESULTS

We apply VBBMD to a H.264 reference encoder [7] with the UMHexagonS fast ME [1]. We set the motion search range to 32, and set the number of reference frames to 1. The RD optimization is disabled and the CAVLC entropy coding is enabled in our experiments.

TABLE VI
PERFORMANCE OF VBBMD IN DIFFERENT CONDITIONS

QP	Performance	Akiyo	Sales-man	News	Silent	Coast-guard	Fore-man
28	ΔPSNR (dB)	-0.05	-0.02	-0.06	-0.04	-0.05	-0.12
	ΔPPMB (%)	94.12	90.22	83.30	81.95	87.67	81.76
	ΔBit-rate (%)	0.04	1.60	1.33	2.42	3.72	4.60
32	ΔPSNR (dB)	-0.03	-0.02	-0.06	-0.04	-0.06	-0.13
	ΔPPMB (%)	93.39	91.07	83.48	83.46	86.37	80.61
	ΔBit-rate (%)	1.03	0.80	1.26	2.95	4.63	3.43
36	ΔPSNR (dB)	-0.05	-0.06	-0.08	-0.04	-0.06	-0.13
	ΔPPMB (%)	92.63	90.43	82.17	83.72	77.46	76.10
	ΔBit-rate (%)	0.66	0.37	1.41	-0.07	3.00	3.45

To examine the effectiveness of the proposed method in different experimental conditions, we choose six sequences with motion activities varying from small to large. They are *Akiyo, Salesman, News, Silent, Coastguard*, and *Foreman* sequences. All sequences are in quarter common intermediate format (QCIF) and encoded at 30 frames/s. All frames except for the first frame are encoded as P-frames. In addition, in order to examine the performance at different bit rates, three QP values, 28, 32, 36, are used in our experiments. As described in Section 2.2, the thresholds used in VBBMD are adjusted according to QP.

For the convenience of comparison, we show the average PSNR gains ($\Delta\mathrm{PSNR}$), the average reduction rates of PPMB ($\Delta\mathrm{PPMB}$) and the increasing rates of bit-rate (ΔBit-rate) in Table VI. From the results we can see that the proposed methods are stable and work well in different conditions. Compared to using the fast ME algorithm only, our method further reduces 76.10% to 94.12% of the integer-pixel search points, while the PSNR degradation is only 0.06 dB on average and the bit rates are just slightly increased.

IV. CONCLUSION

In this paper, an early termination method VBBMD is proposed for H.264 video coding to help existing fast motion search algorithms by further reducing the amount of computation. It predicts the best MV by examining only one search point. With this method, some of the motion searches can be stopped early and the computation associated with these searches can therefore be reduced. In the method, the thresholds are decided based

on the detection accuracy of different block sizes while complexity reduction is also considered, which makes the thresholds decision more reasonable. Experimental results show that our proposed method achieves significant complexity reduction, while the degradation in video quality is negligible.

Future directions may include making the thresholds adaptive to the quantization level and the motion level, and applying the early termination method to subpixel ME.

REFERENCES

[1] Z. B. Chen, P. Zhou, and Y. He, "Fast integer pel and fractional pel motion estimation for JVT," presented at the JVT-F017, 6th Meeting, Awaji, Japan, Dec. 5–13, 2002.

[2] A. Yu, R. Lee, and M. Flynn, "Performance enhancement of H.263 encoder based on zero coefficient prediction," in *Proc. ACM Multimedia*, Nov. 1997, pp. 21–29.

[3] A. M. Tourapis, "Enhanced predictive zonal search for single and multiple frame motion estimation," in *Proc. VCIP*, Jan. 2002, pp. 1069–1079.

[4] J. F. Yang, S. H. Chang, and C. Y. Chen, "Computation reduction for motion search in low rate video coders," in *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, Oct. 2002, pp. 948–951.

[5] C. Zhu, X. Lin, L. P. Chau, K. P. Lim, H. A. Ang, and C. Y. Ong, "A novel hexagon-based search algorithm for fast block motion estimation," in *Proc. ICASSP*, vol. 3, May 2001, pp. 1593–1596.

[6] X. Zhou, Z. H. Yu, and S. Y. Yu, "Method for detecting all-zero DCT coefficients ahead of discrete cosine transform and quantization," *Electron. Lett.*, vol. 34, no. 19, pp. 1839–1840, Sep. 1998.

[7] JVT Reference Software Version 6.1e. [Online]. Available: http://bs.hhi.de/~suehring/tml/download/old_jm/