Fast Multi-Frame Motion Estimation for H.264 and Its Applications to Complexity-Aware Streaming

Shu-Fa Lin, Meng-Ting Lu, and Homer Chen
Graduate Institute of Communication Engineering
National Taiwan University
Taipei, Taiwan 106, R.O.C.
{r92942088, b88901124}@ntu.edu.tw,
homer@cc.ee.ntu.edu.tw

Chia-Ho Pan
Computer & Communications Research Laboratories
Industrial Technology Research Institute
Hsinchu, Taiwan 310, R.O.C
chpan@itri.org.tw

Abstract—**JVT/H.264 achieves higher compression efficiency than previous video coding standards such as MPEG-4 and H.263. However, this comes at the cost of increased complexity due to the use of variable block-size, multiple reference frame motion compensation, and other advanced coding algorithms. In this paper, we present a fast motion estimation algorithm. This algorithm is based on an adaptive search strategy and a flexible multi-frame search scheme. Compared to the H.264 reference software JM 8.5, this algorithm achieves, on the average, a 522% reduction of encoding time, with negligible PSNR drop and bit-rate increase. Performance comparison results are described, and the architecture of a complexity-aware streaming server that applies the algorithm to provide differentiated services is discussed.**

I. INTRODUCTION

H.264 [1] is a new international video coding standard jointly developed by the ITU-T Video Coding Experts Group and the ISO/IEC Moving Picture Experts Group. It achieves more than 50% bit-rate reduction relative to the MPEG-2 standard [2] and represents the state-of-the-art video compression technology. The high coding efficiency is obtained through the use of many advanced coding tools such as variable block-size motion estimation/compensation, rate-distortion optimized mode decision, intra prediction, quarter-pixel motion compensation, multiple reference frames, etc.

The high coding gain of H.264 comes at the cost of increased complexity. A significant increase of the complexity is attributed to the use of motion estimation with variable block sizes and multiple reference frames. Our run time analysis indicates that the motion estimation alone consumes about 50% (for the case of 1 reference frame) and 70% (for the case of 5 reference frames) of the total encoding time. Therefore, for practical applications, fast motion estimation algorithms for H.264 are needed.

Recently, a number of fast motion estimation algorithms have been proposed. Ma et al. [5] observed the rood-shaped distribution of motion vector differences [1] and proposed an unequal-arm adaptive rood pattern search method (ARPS-3). It performs well on the computational and coding efficiencies for low and medium motion sequences. However, it can easily be trapped into local minima for high motion sequences due to the limited adaptivity of the arm length. Chen et al. [8] proposed a hybrid unsymmetrical-cross multi-hexagon-grid search strategy (UMHexagonS), which was adopted in the H.264 reference software. The PSNR performance of UMHexagonS is fairly close to full search. However, its computation cost is still too high.

In this paper, we present a fast motion estimation algorithm based on an adaptive search strategy and a flexible multi-frame search scheme. The core of our adaptive search strategy is formed by a combination of motion prediction, adaptive early termination, and dynamic search pattern to maintain the coding efficiency and reduce the computational cost. Our multi-frame search scheme increases the efficiency by skipping unnecessary reference frames.

This paper is organized as follows. Section II describes in detail the proposed method for fast multi-frame motion estimation. Section III shows the simulation results. Section IV describes the application of our motion estimation method to complexity-aware multimedia streaming, followed by a conclusion.

II. ALOGRITHM DESCRIPTION

The matching process visits the seven block sizes [1], from 16x16 to 4x4, in a hierarchical descending order. Like wise, in the case of multiple reference frames, the references frames are visited sequentially starting from the previous frame for each macroblock to be coded (see Fig. 2).

A. Adaptive Search Strategy

To increase the hit rate, five motion predictors are employed in our motion estimation algorithm, which

This work was supported in part by the National Science Council of Taiwan under contract NSC93-2752-E-002-006- PAE and by ITRI under contract 2F930202.

sequentially tests the five predictors and stops when an early termination condition described below is reached or when all five predictors are tested. When no early termination is detected at the end, a dynamic search pattern scheme is applied to refine the motion vector.

1) Motion prediction. Five motion predictors are used to find the best initial search point in the motion estimation process [5], [6] by exploiting the temporal and spatial correlations of motion vectors in a video sequence. The motion vectors referred to in this section are obtained with respect to the reference frame under consideration.

a) Median predictor: The median value of motion vectors of adjacent blocks on the left, top, and top-right of the current block is used to predict the motion vector of the current block to be coded.

$$V_{pred\text{-}median} = median\ (V_{left}, V_{top}, V_{top\text{-}right}), \quad (1)$$

where V_{left}, V_{top}, and $V_{top\text{-}right}$ are motion vectors of the left, top, and top-right blocks, respectively. The median operation is performed on each component of the motion vector.

b) Upper-layer predictor: The motion vector of the upper-layer block is used as the motion predictor for the lower-layer blocks. For instance, the motion vecotr of partition 0 of mode 2 is used as the motion predictor for partitions 0 and 1 of mode 4, as shown in Fig.1.

c) Spatial predictor: The motion vectors of adjacent blocks (left, top, and top-right) that have the same mode as the current block are used as the motion predictor of the current block. If no adjacent block has the same mode as the current block, the motion vector of the left block is used as the predictor.

d) Temporal predictor: The motion vector of corresponding blocks in the neighboring frames are used as the motion predictors of the current block.

e) Accelerator predictor: For high motion sequences, the object may move with increasing or decreasing acceleration [3]. To deal with such cases, an accelerator predictor is used and defined as follows:

$$V_{pred\text{-}accel} = V_{t\text{-}1} + (V_{t\text{-}1} - V_{t\text{-}2}), \quad (2)$$

where $V_{t\text{-}1}$ is the motion vector of corresponding block in the previous frame $t–1$, and $V_{t\text{-}2}$ is the motion vector of corresponding block in frame $t–2$.

When no early termination condition is reached, the motion predictor with the smallest Lagrangian cost [2] is chosen as the best motion prediction P_{best}.

2) Adaptive early termination. An early termination of motion prediction and refinement is applied when the video quality is good enough. By exploiting the correlations [6]-[8] of Lagrangian costs, a threshold to determine the video quality of the current block is derived based on the Lagrangian costs of its neighboring blocks with respect to the reference frame under consideration. Here, we set this threshold C_{pred} to be the smallest value among $C_{adjacent}$, $C_{pred\text{-}up}$, and $C_{pred\text{-}ref}$ defined in the following.

- $C_{adjacent}$: the smallest Lagrangian cost of the adjacent blocks (left, top, and top-right) that have the same mode as the current block.
- $C_{pred\text{-}up}$: a value derived from the smallest Lagrangian cost $C_{upper\text{-}layer}$ of the upper-layer block as follows

$$C_{pred\text{-}up} = (C_{upper\text{-}layer}/2)*0.9. \quad (3)$$

  Because smaller blocks are expected to have a better prediction, a multiplication factor 0.9 is applied in the equation.
- C_{pred_ref}: the predicted Lagrangian cost of the reference block in frame t (the reference frame under consideration). If there is no uncovered background or occluded object in a macroblock, the difference of Lagrangian costs of a block between different reference frames is small. Based on this property, C_{pred_ref} is computed by

$$C_{pred\text{-}ref} = C_{t'+1}, \quad (4)$$

  where $C_{t'+1}$ is the Lagrangian cost of the same block in frame $t'+1$ ($t' < t–1$), as shown in Fig.2. When $t'=t–1$, $C_{pred\text{-}ref}$ is set to a very large number.

In order to avoid using an extremely large threshold (and thereby ensure the video quality), C_{pred} is limited to $3*T_i$ (T_i is computed by Eq. (6)). The matching process for the current block is terminated when the Lagrangian cost (C_{curr}) of the current block is smaller than C_{pred}.

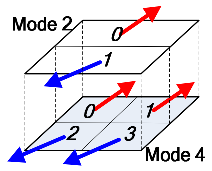

Figure 1. Example of the upper-layer predictor.

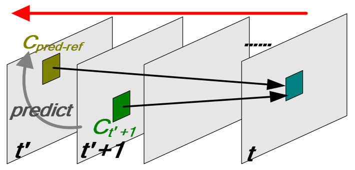

Figure 2. Example of the prediction for C_{pred_ref}.

3) Dynamic search pattern scheme. This scheme is applied for motion refinement. We set the search center at the position pointed to by P_{best}.

The mechanism of the dynamic search pattern selection is shown in Fig. 3. The refinement search pattern is selected from four candidates: large cross search pattern (Fig.4 (a)), hexagon search pattern (Fig.4 (b)), cross search pattern (Fig.4 (c)) , and adaptive rood search pattern (Fig.5). When $\Delta C = C_{curr} - C_{pred}$ is smaller than the threshold H_2, the best motion vector is expected to be close to P_{best}. In this case, the cross search pattern is used to refine the motion vector. Else if $\Delta C \leqq H_1$, the hexagon search pattern is used to refine the motion vector. If $\Delta C > H_1$, the refinement search pattern is selected between the rood search pattern and the large cross search pattern based on the magnitude of $V_{pred\text{-}median}$ (we set the threshold V_{th} to 7). For low and medium motion sequences, the rood search pattern often obtains the best motion vector accurately and rapidly. However, it does not perform well for high motion sequences. Therefore, the large cross search pattern is adopted to refine the motion vector for high motion sequences.

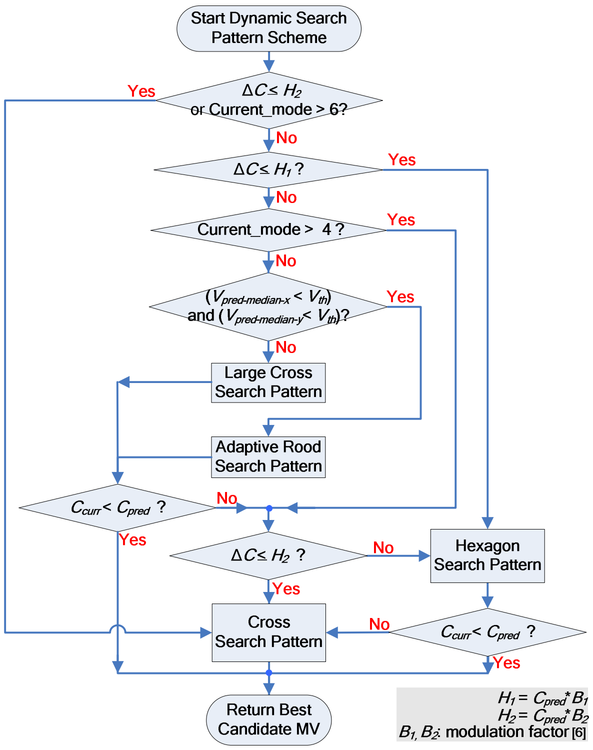
Start Dynamic Search Pattern Scheme
ΔC≤ H2 or Current_mode > 6?
Yes
No
ΔC≤ H1 ?
Yes
No
Current_mode > 4 ?
Yes
No
(Vpred-median-x < Vth) and (Vpred-median-y < Vth)?
Yes
No
Large Cross Search Pattern
Adaptive Rood Search Pattern
Ccurr < Cpred ?
No
Yes
ΔC≤ H2 ?
No
Yes
Hexagon Search Pattern
Cross Search Pattern
No
Ccurr < Cpred ?
Yes
Return Best Candidate MV
H1 = Cpred * B1
H2 = Cpred * B2
B1, B2: modulation factor [6]

Figure 3. The flowchart of the proposed dynamic search pattern selection.

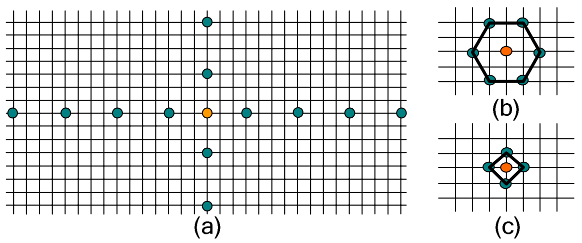

Figure 4. (a) Large cross search pattern.
(b) Hexagon search pattern. (c) Cross search pattern.

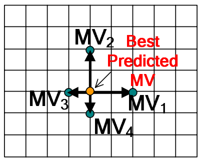

Figure 5. Adaptive rood search pattern.

B. *Flexible Multi-Frame Search Scheme*

With multiple reference frames, the computational cost can increase significantly [9]. We develop a flexible multi-frame search scheme that avoids unnecessary search operations.

A matching is good if the quantized transform coefficients of the motion-compensated block are all close to zeros [7]. A threshold is used in our algorithm. When the SAD of a motion-compensated block is smaller than the threshold, this block is considered a zero block. When a zero block is detected, the remaining modes and reference frames are skipped and not searched. The threshold T_7 is derived based on 4x4 blocks:

$$T_7 = (2^{qbits} - qp_{const})/Q_{coeff}[q_{rem}][0][0], \qquad (5)$$

where $qbits = 15 + QP/6$, $qp_{const} = (1{<<}qbits)/6$, $q_{rem} = QP$ mod 6, Q_{coeff} is the quantization coefficient table, and QP is the quantization parameter.

For the other modes of block sizes 16x16, 16x8, 8x16, 8x8, 8x4, and 4x8, the threshold is defined as:

$$T_i = T_7 * m_i, \qquad (6)$$

where m_i is a scale factor (in our experiment, m_i = 4, 2.5, 2.5, 2, 1.5, 1.5, for i = 1..6). Like the mechanism used in early termination, a cap is applied to the threshold. The cap is set to 512, 448, 448, 384, 384, 384, and 384, respectively, for i = 1..7. Furthermore, the Intra 4x4 prediction mode is skipped when a zero block is detected. This is because, in practice, the Intra 4x4 prediction mode is rarely selected as the best mode in such cases.

III. Simulation Results

Our proposed algorithm was implemented in the H.264 reference software JM 8.5. In our simulation, the search range of motion vectors was set to ±32 pels. Furthermore, RD optimization, CAVLC entropy coding, seven block modes, and five reference frames were turned on for all tests. Apart from the first frame, all the other frames were coded as P frames. The simulations were performed on a PC with a Pentium IV, 2.4 GHz CPU and a 768 MB RAM.

We divided the test sequences into two groups: slow and medium motion group (Mother & Daughter, Silent, Container, Mobile, Flower, and Tempete) and fast motion group (Stefan, Foreman, and Bus) to investigate the performance of our algorithm. Table 1 shows the performance comparisons against UMHexagonS and fast full search used in JM 8.5. Table 2 shows the average results of different QPs (16, 20, 24, 28, 32, 36, and 40).

On the average, our algorithm is 47.9 and 3.3 times faster than fast full search and UMHexagonS, respectively. The speedup is achieved with negligible bit-rate increase and PSNR drop. Note that the slight bit-rate increase of the sequences Foreman and Silent is inevitably caused by the flexible multi-frame search scheme.

TABLE I. PSNR COMPARISON (QP FIXED AT 28)

Sequence (CIF)	Search Method	PSNR-Y Gain	ME Speed-Up	Encoding Speed-Up	Bit-Rate Increase
Mobile	UMHexagonS	-0.00	11.02	3.35	-0.09%
	Proposed	-0.01	37.32	4.00	0.10%
Tempete	UMHexagonS	-0.00	11.67	3.55	0.07%
	Proposed	-0.01	39.41	4.48	0.40%
Flower	UMHexagonS	-0.00	11.48	3.30	-0.04%
	Proposed	-0.01	37.55	4.03	0.32%
M & D	UMHexagonS	-0.03	21.20	4.87	0.28%
	Proposed	-0.03	78.64	7.47	0.57%
Container	UMHexagonS	-0.01	21.91	4.65	0.36%
	Proposed	-0.05	52.00	5.48	1.03%
Silent	UMHexagonS	-0.02	18.30	4.33	0.37%
	Proposed	-0.02	46.05	5.14	1.57%
Foreman	UMHexagonS	-0.02	12.95	4.07	0.29%
	Proposed	-0.04	42.69	5.35	2.14%
Stefan	UMHexagonS	-0.02	11.45	3.63	-0.02%
	Proposed	-0.03	38.92	4.86	1.30%
Bus	UMHexagonS	-0.01	9.92	3.42	-0.08%
	Proposed	-0.01	30.10	4.25	0.51%

TABLE II. COMPARISON OF AVERAGE PSNR

Search Method	Average Bit-Rate Increase	Average PSNR-Y Gain	Average ME Speed-Up	Average Encoding Speed-Up
UMHexagonS	0.03%	-0.020	14.40	3.94
Proposed	1.66%	-0.039	47.89	5.22

IV. APPLICATIONS TO COMPLEXITY-AWARE MULTIMEDIA STREAMING

The new motion estimation method can be used in the complexity-aware streaming scenario illustrated in Fig. 6. Assume a content provider offers two kinds of real-time content, paid and free programs or commercials. It is reasonable for a customer to expect better quality when viewing paid programs. Because the server has to encode all input live videos, computational resources become the most critical factor instead of bandwidth. Therefore, it is desired to adjust the quality of free programs or commercials when a streaming server's load reaches a threshold. The control module in our architecture keeps monitoring the CPU usage. When the CPU usage reaches the predetermined threshold, it informs the codec module used by the free programs or commercials. The codec module will then appropriately switch to the less complex motion estimation method to release computation power. The released computation power can be used to serve more customers or guarantee the quality of the paid programs. Using this method, paid programs can be served with guaranteed quality while free programs or commercials are served with best-effort quality.

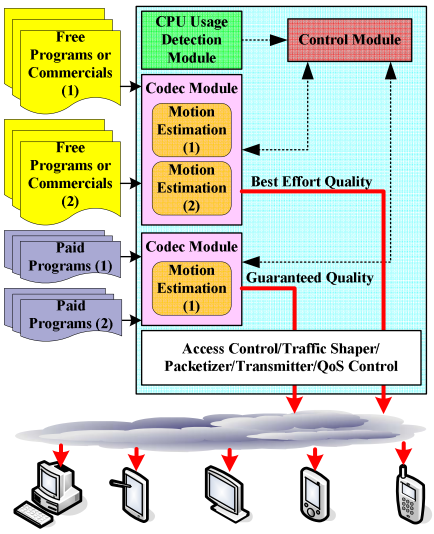

Figure 6. Complexity-aware streaming server.

V. CONCLUSIONS

In this paper, we have presented a multiple reference frame motion estimation algorithm based on an adaptive search strategy and a flexible multi-frame search scheme. Comparing to the H.264 reference software JM 8.5, our algorithm achieves, on the average, a 5.22 speed-up in the total encoding time, with negligible PSNR degradation and bit-rate increase. A complexity-aware streaming server architecture has also been presented to demonstrate a useful application of our algorithm for video streaming.

REFERENCES

[1] Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC), 2003.

[2] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of H.264 video coding standartd," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 13, no. 7, pp. 560-576, Jul. 2003.

[3] H.-Y, C. Tourapis, A. M. Tourapis, and P. Topiwala, "Fast motion estimation within the JVT codec," JVT-E023.doc, Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, 5th meeting, Geneva, Switzerland, Oct. 2002.

[4] C. Zhu, X. Lin, and L.-P. Chau, "Hexagon-based search pattern for fast block motion estimation," *IEEE Trans. on Circuits and Systems for Video Technology,* vol. 12, no. 5, pp. 349-355, May 2002.

[5] K. K. Ma and G. Qiu, "Unequal-arm adaptive rood pattern search for fast block-matching motion estimation in the JVT/H.26L," *Proc. IEEE Int'l Conference on Image Processing,* Sept. 2003.

[6] A. M. Tourapis, O. C. Au, and M. L. Liou, "Predictive motion vector field adaptive search technique (PMVFAST)- enhancing block based motion estimation," *Proc. Visual Communications and Image Processing*, San Jose, CA, pp. 883-892, Jan. 2001.

[7] Z. Chen, P. Zhou, and Y. 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, Japan, Dec. 2002.

[8] Z. Chen, P. Zhou, and Y. 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, Mar. 2003.

[9] Y.-W. Huang, et al., "Analysis and reduction of reference frames for motion estimation in MPEG-4 AVC/JVT/H.264," *Proc. IEEE Int'l Conference on Acoustics, Speech, and Signal Processing*, Apr. 2003.