Adaptive Early Jump-Out Technique for Fast Motion Estimation in Video Coding

Ho-Chao Huang Yi-Ping Hung Wen-Liang Hwang
Institute of Information Science
Academia Sinica
Taipei, Taiwan, R.O.C.

Email: jet@smart.iis.sinica.edu.tw
hung@iis.sinica.edu.tw
whwang@iis.sinica.edu.tw

Abstract

An adaptive early jump-out technique for speeding up the block-based motion estimation is proposed. By using the new technique, we can speed up the full range search several times without losing the picture quality significantly. The proposed technique can also be embedded into almost all the existing fast motion estimation algorithms to speed up the computation further. Since the proposed technique can be embedded into the existing motion estimation algorithms, it can be applied to almost all the standard video codecs, such as the MPEG coder, and improve the coding speed of such codecs significantly. Our technique has been tested on the H.261 and the MPEG-I codecs, and the coding speed does improve significantly.

Keywords: Early Jump-Out, Motion Estiation, Video Compression

1 Introduction

Motion estimation is one of the major parts of video coding standards[1, 2] and multimedia coding systems. The high computational cost of motion estimation is the major problem of the coders. Many fast motion estimation algorithms, such as the three-step search[3], the 2-D logarithmic algorithm[4], the conjugate direction search[5] and the inter-block correlation algorithms[6, 7], have been proposed for solving the computation problem of motion estimation.

In this paper, the adaptive early jump-out technique for motion estimation algorithms is proposed. The early jump-out (EJO) technique has been applied for speeding up corner detection and template matching in [8, 10, 11], and has achieved very impressive results. However, since the characteristic of motion estimation is quite different from those of corner detection and template matching, both the mathematical early-jump-high models[8, 10] and the trained early-jump-high sequence[11] are unsuitable for motion estimation. An adaptive early jump-out threshold sequence

training algorithm is proposed. The adaptive early jump-out technique has the capability to learn the characteristic of the image sequence on-line. The experimental results show that the adaptive early jump-out technique can speed up the full range search algorithm without losing the picture quality significantly. Moreover, the proposed technique can also be applied to almost all the existing fast motion estimation algorithms, such as the three-step search, the conjugate direction search and the inter-block correlation search, and still gains the improvement on the computational speed. Our technique has been tested on the H.261 and the MPEG-I codecs, and the coding speed does improve significantly.

This paper is organized as follows. Section 2 briefly reviews the most commonly used block-based motion estimation algorithms and the early jump-out techniques. The proposed adaptive early jump-out technique is presented in section 3. Section 4 describes the order for searching and the order for computing the match error. The experimental results are shown in section 5. Finally, the conclusion and future work are stated in section 6.

2 Preliminary Reviews

In this section, we briefly review the motion estimation algorithms and the early jump-out techniques. In subsection 2.1, four classes of the commonly used block-based motion estimation approaches are presented. The early jump-out techniques for corner detection and template matching are described in subsection 2.2.

2.1 Block-Based Motion Estimation Algorithms

Most video coding standards use motion compensation to exploit temporal redundancy in the video. Decoders construct a predicted block of pixels from pixels in a previously transmitted image. Motion within the images usually implies that the pixels in the reference image, usually the previous image, will be in a different position from the pixels in the current image, and the displacement, motion vector, is estimated by the video encoder and encoded in the bitstream. The predicted image block is usually a good estimate of the current image block, and it is usually more efficient to transmit the motion vector plus the difference between the predicted block and the current block, than to transmit a description of the current block by itself.

The motion estimation of the video encoder determines the best motion vectors. Using a search strategy the encoder attempts to match the pixels in an image block with those in the reference image. The vector corresponding to the best match is reported after the search is completed.

In order to determine which one is the best motion vector, a match criterion must be defined. Two matching criteria, the mean squared error (MSE) and the mean absolute error (MAE), are the most common choices. Following are the definitions of the MSE and the MAE respectively.

$$MSE(\Delta x, \Delta y) = \frac{1}{n \times n} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} [S_c(x+j, y+k) - S_r(x+\Delta x+j, y+\Delta y+k)]^2 \qquad (1)$$

$$MAE(\Delta x, \Delta y) = \frac{1}{n \times n} \sum_{j=0}^{n-1} \sum_{k=0}^{n-1} |S_c(x+j, y+k) - S_r(x+\Delta x+j, y+\Delta y+k)| \quad (2)$$

where $S_c(x, y)$ is the current image block located at position (x, y), and $S_r(x + \Delta x, y + \Delta y)$ is the corresponding block in the reference image, at position $(x + \Delta x, y + \Delta y)$. Here, $(\Delta x, \Delta y)$ is the possible motion vector in matching operation. The most similar block is found to be the minimum value of match error. The MAE needs fewer computations than the MSE does, but it may cause a little degradation on the prediction quality. For the sake of algorithms design and comparison, the MSE criterion is used in this paper.

After the match criterion is selected, some kind of search strategy must be chosen. Several kinds of search strategies are described in the following paragraphs.

Full Range Search Algorithm The full range search algorithm is the brute force block matching search algorithm, and has been shown to be able to produce the best motion compensated images. The following is a brief description of the full range search algorithm.

First, the current image is partitioned into n by n sized blocks. In most video coding standards, such as the MPEG and H.261, the size of image block is 16 by 16 pixels. For each current image block, a $(2M + 1) \times (2M + 1)$ search range in the reference image centered at the corresponding position of the current image block is defined. That is, the possible values of both Δx and Δy in (1) and (2) are $[-M, M]$. The current image block is compared to each possible image block within the search range, i.e., the reference image block, of the reference image by using a selected match criterion. Then the reference image block with the minimum match error is attached to the current image block, and the corresponding $(\Delta x, \Delta y)$ is the best motion vector. For each image block the number of block matching with full range search is $(2M + 1) \times (2M + 1)$, and the number of multiplications is $(2M + 1) \times (2M + 1) \times n \times n$. For a video with image resolution of 352 by 288, 110398464 multiplications are needed for motion compensating an image, if the image block size is 16 by 16 and the search range is $[16, 16] \times [16, 16]$, for example.

Three Step Search Algorithm The basic assumption of most fast motion estimation algorithms is that the reference image block with motion vector $(\Delta x_1, \Delta y_1)$ closer to the best motion vector (mx, my) has a smaller match error than that of a reference image block with farther motion vector $(\Delta x_2, \Delta y_2)$.

The three step search is a typical 2-D search algorithm. In three step search grids of 9 displacements are examined, and the search continued based on a smaller grid centered on the position of the best match. Usually, the grids are reduced in size by a factor of 2 at each step.

An example of three step search algorithm is illustrated in figure 1. The first grid has a spacing of 4 pixels. The first step examines pixels at shifts of 0, 4, or -4 pixels in each direction, marked 1 in figure 1. The best position is used as the center point of the second grid. Assume the best position is the pixel marked 1 directly to the left of the center pixel. The second grid has a spacing of 2 pixels. The second step examines pixels at shifts of 0, 2, or -2 pixels in each direction from

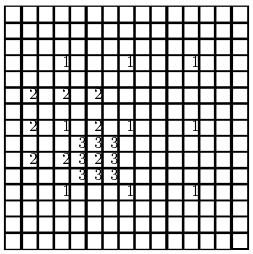


Figure 1: An example of three step search algorithm

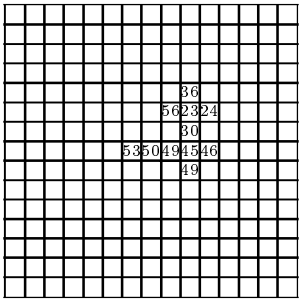


Figure 2: An example of conjugate direction search algorithm

the center of the new grid, marked 2 in the figure. The best position is used as the center point of the third grid. Here, we assume it is the lower right pixels of the second grid. The third grid has a spacing of 1 pixel. The third step examines pixels at shifts of 0, 1, or -1 pixels in each direction from the center of the grid. The best position determines the motion vector.

Conjugate Direction Search Algorithm The next class of fast motion estimation algorithms is the 1-D approaching algorithms. The conjugate direction search (CDS) is the typical 1-D algorithm. In CDS the x-axis direction, the horizontal direction, is firstly examined to find the best match with a fixed y. Then the y-axis direction, the vertical direction, is searched to find the best match with the best x found in the previous step. Repeat the above steps until the best motion vector (mx, my) is obtained.

An example of conjugate direction search is illustrated in figure 2. All the values in figure 2 represent the match errors of the corresponding displacements. The initial match position is the point with error value 50. The first step searches on the x direction and finds the best position with error value 45. The second step searches on the y direction and the best displacement with error value 23 is found. The third step searches on x direction again, but can't find lower match error than 23. Finally the estimated motion vector $(2, 2)$ is transmitted and the match error is 23.

Inter-block Correlation Search Algorithm All the motion estimation algorithms stated above estimate the motion vector for each image block independently. Another class of fast motion estimation algorithms uses the correlation between image blocks to speed up the estimation process. In [6], a dynamic search range algorithm (DSRA) has been proposed. By considering the relation between neighboring blocks, the search area in the algorithm is adjustable. Due to the adaptation of the search area, the computational complexity can be largely reduced and the actual motion vectors can still be found.

2.2 Early Jump-Out Techniques

Early in 1972, Barnea and Silverman [8] introduced a class of sequential similarity detection algorithms for expediting the similarity detection between two structured data sets. Their contribution was to propose a monotonically-increasing threshold sequence algorithm where a threshold sequence could be defined such that if, at any accumulation stage in the computation of the MSE or the MAE, the partial result was greater than the corresponding threshold in the sequence, one could jump out of the similarity test. Recently, Cooper et al. applied this sequential algorithm to the dissimilarity test in corner detection, and called it the Early Jump-Out (EJO) technique [10].

We rewrite (1) into the one dimension form:

$$MSE(\Delta x, \Delta y) = \frac{1}{n \times n} \sum_{i=0}^{n \times n - 1} \Big[S_c(i) - S_{r(\Delta x, \Delta y)}(i) \Big]^2 \tag{3}$$

where $S_c(i)$ represents the value of i-th pixel in the current image block and $S_{r(\Delta x,\Delta y)}(i)$ represents the value of the i-th pixel in the reference image block with displacement $(\Delta x, \Delta y)$.

Let AE_j be the accumulated sum of squared error at step j. That is,

$$AE_j = \sum_{i=0}^{j} \Big[S_c(i) - S_{r(\Delta x, \Delta y)}(i) \Big]^2. \tag{4}$$

Let us define the early jump-out threshold sequence EJS_j, such that if the accumulated error AE_j is greater than EJS_j, the matching process is terminated and a predefined large error value is returned.

In order to apply the early jump-out technique to corner detection and template matching, several mathematical models, such as the Gaussian and the Exponential models, have been used to obtain the early jump-out threshold sequence [8, 10]. In [11], the authors have proposed a method for learning the EJO threshold sequence from training data, and have successfully applied it to the template matching problem with little performance degradation.

3 Adaptive Early Jump-Out Technique for Motion Estimation

This section describes the proposed adaptive early jump-out (AEJO) algorithm for fast motion estimation. Although the early jump-out techniques have been shown to have great helps on sav-

ing the computation for corner detection, template matching and feature detection, the following video coding properties make the mathematical models [8, 10] and the learning technique [11] for determining the EJO threshold sequence unsuitable for motion estimation in video compression.

Property 1 *For each current image block, the motion estimator in video coder has to choose exactly one "best-match" motion vector by comparing to the reference image.*

Unlike in the corner and feature detection problems, the search for best motion vector of image block is mandatory. That is, even no reference image blocks are similar to the current image block, the motion estimator still needs to choose a "most similar" reference block.

Property 2 *The error distribution of the best match depends on the characteristic of the image block. When the current image block is within an object having slow motion, the error value of the best match will be smaller, and when the current image block is within a rotated or blurred object or uncovered background, the error value of the best match will be usually larger.*

To deal with the motion estimation properties, the early jump-out technique should have the capability to learn the local error distribution of each image block. Following is the proposed AEJO algorithm for motion estimation.

Algorithm 1 *The Adaptive Early Jump-Out Motion Estimation Algorithm*

1. *Select a search strategy.*

   The early jump-out technique can be embedded in almost all search strategies, such as the full range search, the three step search and the inter-block correlation search algorithms.

2. *Select a match criterion.*

   In all experiments of this paper, the mean square error in equation (3) is used.

3. *Initialize the early jump-out threshold sequence EJS as a pre-set maximum error value.*

4. *For each current image block and each possible reference image block with displacement $(\Delta x, \Delta y)$:*

   (a) *Set the error value: $E = 0$.*

   (b) *Clear the flag for early jump-out event: EJO_{event}.*

   (c) *For $j = 0$ to $n \times n - 1$ do*

      i. *Compute the accumulated error:*
      $E = E + \left[S_c(j) - S_{r(\Delta x, \Delta y)}(j)\right]^2$.

      ii. *Compare with the EJS to detect whether the early jump-out event occurs or not:*
      If $E \geq EJS_j$ then set EJO_{event} and break the loop of (4c);

      iii. *Store the accumulated error: $AE_j = E$*

(d) If EJO_{event} is set: (i.e., jump-out occurs and this displacement $(\Delta x, \Delta y)$ is not a possible best match)

i. Continue matching with the next displacement.

(e) If EJO_{event} is not set: (i.e., jump-out did not occur and a new possible best match was found)

i. Set the motion vector $(mx, my) = (\Delta x, \Delta y)$.

ii. Update the early jump-out threshold sequence EJS (with algorithm 2 described below).

iii. Continue matching with the next displacement.

The next problem is to design an update algorithm for adjusting the early jump-out threshold sequence used in Algorithm 1, step 4(e)ii. The following rules can help on designing the update algorithm.

Rule 1 *Each early jump-out threshold value EJS_j should not be greater than $AE_{n\times n-1}$. That is, $AE_{n\times n-1}$ is the theoretic upper bound of new EJS_j.*

Since $AE_{n\times n-1}$ is the sum of square error of the possible best match, any time when the accumulated error is greater than $AE_{n\times n-1}$ implies that the reference image block in consideration should not be a possible best match.

Rule 2 *Each early jump-out threshold value EJS_j may not be less than AE_j. That is, AE_j is a reasonable lower bound of new EJS_j.*

Since AE_j is the accumulating error of the j-th step for the current possible best match, if the new EJS_j is less than AE_j, it implies the possible best match found will also be rejected by the new early jump-out threshold sequence and the result is unreasonable.

Rule 3 *$EJO_{n\times n-1}$ should be equal to $AE_{n\times n-1}$.*

Rule 3 can be deduced from rules 1 and 2. The update algorithm based on the above rules used in this paper is stated as follows.

Algorithm 2 *The Update Algorithm for the Adaptive Early Jump-Out Technique*

1. *For $j = 0$ to $n \times n - 1$:*
   $EJS_j = \frac{AE_j \times (EJO_{factor} - 1) + AE_{n\times n-1}}{EJO_{factor}}$,
   where EJO_{factor} is a parameter to control the behavior of the AEJO estimator.

When EJO_{factor} is set at a large value, EJS_j is near to AE_j, then it implies more computation saving but more image quality degradation. On the other hand, smaller EJO_{factor} implies a higher threshold sequence EJS. It makes the algorithm harder to jump out from the match loop, but can obtain better motion compensated image. Figure 3 illustrates the early jump-out threshold sequences with EJO_{factor} being set at 1, 4 and 8 respectively.

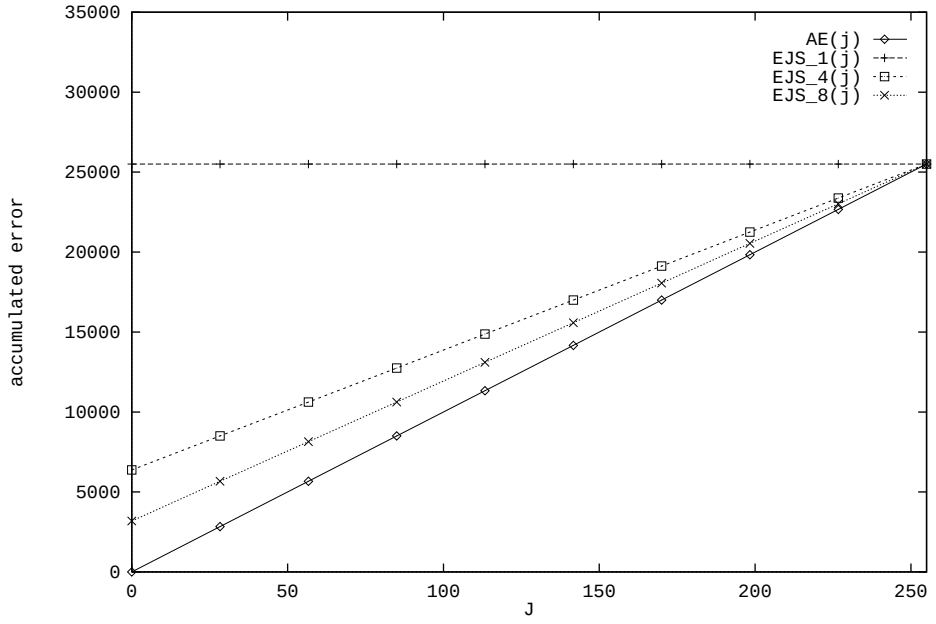


Figure 3: The early jump-out threshold sequences with EJO_{factor} being set at 1, 4 and 8 respectively.

4 Order for Searching and for Computing Match Error

In order to get better motion compensated image and save more computation, the search order and the match order (i.e. the order for computing the match error) should be well designed. These issues are discussed in the following subsections.

4.1 Search Order of Motion Estimation

Since the EJS is learned in the matching process on-line, a good match implies a lower EJS, and a lower EJS can save a lot of computation than a higher EJS. Thus to reorder the reference image blocks such that the most possible displacements are searched first will be of great help. Two search orders stated below are compared in this paper.

Algorithm 3 *The Raster Scan Searching Order (RSSO)*

The first search order used by AEJO is the raster scan search order. This is the search order used by most traditional motion estimation algorithms. Figure 4 shows an example when the full range search is used with search range $[-2, 2]$ *by* $[-2, 2]$.

Algorithm 4 *The Spiral Searching Order (SSO)*

The other one is the spiral search order. An example of the search orders for the full range search and the three step search are illustrated in figure 5.

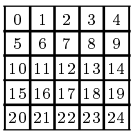


Figure 4: An example of the rater scan search order with search range $[-2, 2]$ by $[-2, 2]$

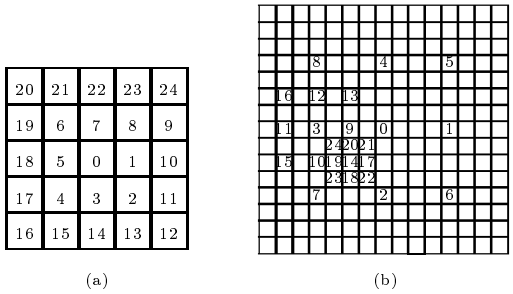


Figure 5: Examples of the spiral search order with (a) search range $[-2, 2]$ by $[-2, 2]$ for full range search and (b) search range $[-7, 7]$ by $[-7, 7]$ for three step search.

Since the motion vectors of most image blocks is near to zero, the match error of the displacement (0.0) usually much smaller than that of the left-top position, which is the first one of the raster scan search order. As the results shown in section 5, using the spiral search order can save much more computation than raster scan search order can do.

4.2 Order for Computing the Match Error

The next issue of AEJO motion estimator is the order for computing the match error (i.e. the match order). Three match orders are described as follows.

Algorithm 5 *The Raster Scan Match Order (RSMO)*

Similar to the raster scan search order stated in the previous subsection, the raster scan match order is also the common used one by almost all traditional motion estimation algorithms.

Algorithm 6 *The Spiral Match Order (SMO)*

The second one is the spiral match order. The spiral match order is similar to the spiral search order except that the spiral match order is defined for computing the accumulated error in a match. This match order is also similar to the accumulated order used in [10, 11].

Algorithm 7 *The Random Match Order (RMO)*

The third one is the random match order. In the random match order, the order of pixel pairs for computing the accumulated error is randomized.

Unlike in corner detection and template matching problems, the necessary information is not centralized at the central area of an image block. That is, the spiral match order can not speed up the motion estimation significantly. On the other hand, the random match order can help on gathering the information faster. As shown in the experimental results the random match order can slightly improve both the compensated image quality and the motion estimation speed.

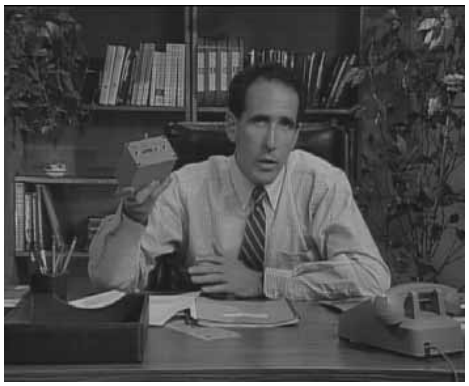
5 Experimental Results

In this section, several experiments for examining the performance of the AEJO are illustrated. In these experiments, three video sequences, the salesman (SAL), the miss America (MA) and the Susie (SU) sequences are used. The first images of three test video sequences are shown in figure 6. The salesman sequence is the most commonly used video test sequence which contains local fast motion objects. The miss America is a typical video phone/conferencing video sequence which contains several slow motion objects. The Susie sequence contains a large fast motion area. All three sequences contains 100 images. The image resolutions of SAL, MA and SU are 352 by 288, 352 by 288 and 360 by 240 respectively. The image block size and search range are 16 by 16 and $[-16, 16]$ by $[-16, 16]$ respectively. All performance values are evaluated by using the number of multiplications. The AEJO has the overheads of adjusting the EJO threshold sequence and comparing the accumulated error with the EJO threshold sequence, which are not counted in these experiments.

The first experiment is to decide which search order and match order is the best one and should be selected in the subsequent experiments. In this experiment EJO_{factor} is set at 16. Table 1 shows the results. Since the combination of the spiral search order (SSO) and the random match order (RMO) is the one requires the fewest number of multiplications, the indirected access of random match order makes it sometimes slower than the raster scan match order. Beside that, the (RMO, SSO) combination also produce the best compensated video quality, and that is why all the following experiments use the (RMO, SSO) combination. In this experiment we also observed that the spiral search order saved much more computation for slow motion videos (SAL and MA) than for fast motion video (SU).

The second experiment let EJO_{factor} vary to see the effect of this parameter. The result is shown in table 2 and EJO_{factor} varies from 4 to 64. The choice of EJO_{factor} is a tradeoff between the video quality and computational speed. The larger the EJO_{factor} value is, the higher computational speed and worse video quality will be obtained.

The third experiment is designed to apply the AEJO to several commonly used fast motion estimation algorithms: the three step search (TSS), the conjugate direction search (CDS) and the dynamic search range algorithm (DSRA). In all algorithms used here the EJO_{factor} value is set to 16, and the spiral search order and the random match order are used. Table 3 illustrates the experimental results. The video quality degradation due to applying the AEJO technique is usually within 0.05 dB and the computational saving is significant.



(a)

(b)

(c)

Figure 6: Video sequences used in the experiments: (a) salesman, (b) miss America and (c) Susie

Video	(RSSO, RSMO)	(RSSO, RMO)	(SSO, RSMO)	(SSO, RMO)
SAL (# of Mul.)	1168199325	944398360	198407027	107617367
SAL (dB of SNR)	35.0673	35.5032	35.356	35.5034
MA (# of Mul.)	1220425490	894931278	254161321	104590778
MA (dB of SNR)	36.0432	36.2816	36.2611	36.3198
SU (# of Mul.)	998889306	787502595	337930602	251342478
SU (dB of SNR)	34.1546	34.4864	34.2544	34.4976

Table 1: The experimental results of four combinations of search orders and match orders. In this table RSSO, SSO, RSMO and RMO indicate the raster scan search order, spiral search order, raster scan match order and random match order respectively.

Video	$EJO_{factor} = 4$	$EJO_{factor} = 8$	$EJO_{factor} = 16$	$EJO_{factor} = 32$	$EJO_{factor} = 64$
SAL (# of Mul.)	190476480	137946922	107617367	89131041	77810118
SAL (dB of SNR)	35.5499	35.5368	35.5034	35.4508	35.3679
MA (# of Mul.)	148693265	123330411	104590778	90104781	79392498
MA (dB of SNR)	36.3629	36.3504	36.3198	36.2691	36.1898
SU (# of Mul.)	616099702	389218320	251342478	170998226	125209555
SU (dB of SNR)	34.5651	34.547	34.4976	34.3934	34.2278

Table 2: The experimental results of various EJO_{factor}.

Algorithms	SAL (Mul)	SAL (SNR)	MA (Mul)	MA (SNR)	SU (Mul)	SU (SNR)
FRS	9884869632	35.5544	9884869632	36.3688	8143584768	34.5717
FRS-AEJO	107617367	35.5034	104590778	36.3198	251342478	34.4976
TSS	307270656	35.312	307542784	36.3636	254985472	33.4901
TSS-AEJO	23374169	35.2671	22299816	36.3174	42443736	33.4538
CDS	54577920	35.3467	51900672	36.3591	63627776	33.4184
CDS-AEJO	16994387	35.2916	16524635	36.3094	27980567	33.1774
DSRA	1122756864	35.5485	1283447808	36.3671	1811626496	34.5614
DSRA-AEJO	33768117	35.4951	32446080	36.318	104139216	34.4822

Table 3: The experimental result of applying the AEJO technique to several fast motion estimation algorithms.

Algorithms	SAL(%)	MA(%)	SU(%)
FRS-AEJO	7	6	18
TSS-AEJO	8	7	24
CDS-AEJO	7	6	20
DSRA-AEJO	7	7	18

Table 4: The estimation miss ratio of the motion estimation when applying the AEJO technique to a specific motion estimation algorithm. EJO_{factor} is set to 16.

The fourth experiment evaluates the estimation miss ratio of the motion estimation when applying the AEJO technique to a certain motion estimation algorithm. The result is illustrated in table 4. The estimation miss means that the motion vector estimated by combining with the AEJO technique is different from that estimated by the specific motion estimation algorithm alone. In this experiment, we can observe that the miss ratio is higher when estimating the motion vector of fast motion.

The last experiment is to apply the AEJO technique to the existing standard video codecs, the H.261 and MPEG-I coders. In H.261 the AEJO technique has been embedded in a full range search based motion estimator, and in MPEG-I coder the AEJO technique has been embedded to the three-step search based motion estimator. In both cases, the coding speed does improve significantly.

6 Conclusion and Future Work

An adaptive early jump-out technique for motion estimation in video compression is proposed in this paper. This technique has the capability to learn the local match error distribution on-line, and the early jump-out threshold sequence is updated based on the error distribution. The proposed technique can be embedded into almost all the existing motion estimation algorithms and can speed up the computation further. By using the new technique the computation of motion estimation can be greatly reduced without losing video quality significantly.

Although the adaptive early jump-out technique can save tremendous amount of computation, it has significant overhead for updating the early jump-out threshold sequence and deciding whether to jump out from the matching process or not. Furthermore, the spiral search order and random match order produce some overhead too. Our future work is to design a more efficient threshold sequence updating rule to reduce the overhead such that the proposed technique can be much more attractive. Another future work is to design an efficient search algorithm which can take the advantages of the proposed technique to speed up the motion estimation further. Finally, it is also possible to apply the adaptive early jump-out technique for hardware implementation.

Acknowledgments

Thanks to the Communications and Multimedia Laboratory, National Taiwan University, Taipei, Taiwan, ROC, for providing the H.261 and the MPEG-I codecs. We also thank Prof. Ja-Ling Wu, Mr. Den-Yuan Hsiau and Mr. Yeung-Chyuan Weng for their helps on testing the proposed technique on the H.261 and the MPEG-I codecs.

References

[1] *MPEG standard draft ISO-IEC/JTC1 SC29 on 22*, November 1991.

[2] 1989 CCITT Study Group XV, TD 35. Draft revision of recommendation h.261: Video codec for audiovisual services at p x 64 kbits/s. *Image Communication*, 2:221–239, August 1990.

[3] T. Koga, K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro. Motion compensated interframe coding for video conferencing. In *Proc. Nat. Telecommun. Conf.*, pages 65.3.1–65.3.5, New Orleans, LA, USA, December 1981.

[4] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans. Commun.*, COM-29:1799–1808, December 1981.

[5] R. Srinivasan and K. R. Rao. Predictive coding based on efficient motion estimation. *IEEE Trans. Commun.*, COM-33(8):888–896, August 1985.

[6] C.-H. Lin and J.-L. Wu. Fast motion estimation algorithm with adjustable search area. In *SPIE Symposium on Visual Commun. and Image Processing*, volume 2501, Taipei, Taiwan, ROC, 1995.

[7] C. H. Hsich, P. C. Lu, J. S. Shyn, and E. H. Lu. Motion estimation algorithm using interblock correlation. *IEE Electron. Letter*, 26(5):276–277, March 1990.

[8] D. I. Barnea and H. F. Silverman. A class of algorithm for fast digital image registration. *IEEE Trans. Comput.*, 21:179–186, 1972.

[9] Duda and Hart. *Pattern Classification and Scene Analysis.* John-Wiley & Sons, 1973.

[10] J. Cooper, S. Venktesh, and L. Kitchen. Early jump-out corner detectors. *IEEE Trans. on Patt. Anal. Mach. Intell.*, 15:823–828, 1993.

[11] A.-T. Tsao, Y.-P. Hung, C.-S. Fuh, and H.-Y. M. Liao. On learning the threshold sequence for the early jump-out template matching. In *TAAI Artificial Intelligence Workshop*, pages 186–193, Taipei, Taiwan, R.O.C., September 1995.