# FAST AND ACCURATE MOTION ESTIMATION ALGORITHM BY ADAPTIVE SEARCH RANGE AND SHAPE SELECTION

*Toru YAMADA, Masao IKEKAWA, and Ichiro KURODA*

Media and Information Research Laboratories, NEC Corporation
1753, Shimonumabe, Nakahara-ku, Kawasaki 211-8666, JAPAN

## ABSTRACT

This paper presents a fast and accurate motion estimation algorithm. To obtain accurate motion vectors while minimizing computational complexity, we adjust the search range for each frame and each block to suit the motion level of the video. An appropriate search range for each frame is determined on the basis of motion vectors and prediction errors obtained for the previous frame. At each block, the search range is determined on the basis of the search range of its frame and of the motion vector values of all adjacent blocks for which those values have already been obtained. With our algorithm, since narrow search ranges are chosen for areas in which little motion occurs, computational complexity can be reduced without degrading estimation accuracy. Since wide search ranges are chosen for areas of significant motion, good video-quality encoding can be maintained. In the encoding of an SDTV size video, the addition of range adjustment results in a reduction in the computational complexity of motion estimation of roughly 65%, while maintaining the same video quality.

## 1. INTRODUCTION

Video coding standards, such as MPEG-2, use motion compensation to reduce inter-frame redundancy. In this motion compensation, only prediction errors which differ from a reference image are coded. The motion vectors needed to create reference images are obtained by means of motion estimation, a process for locating points at which prediction errors will be minimum.

A full search (FS) algorithm, which searches all positions in a search range, is optimal in terms of estimation accuracy, but its computational complexity is quite high. To reduce this computational complexity, various fast motion estimation algorithms have been proposed, including the three-step search(TSS)[2], the new three-step search(NTSS)[3], and the four-step search(FSS)[4] algorithms. These algorithms are based on the assumption that prediction error increases in monotonical proportion to the distance of search points from points of minimum prediction error. However, estimation accuracy with these algorithms is likely to be unsatisfactory since that assumption is often untrue [5].

Another approach to reducing computational complexity is to adjust the search range size to suit the motion level of a video. Among the various methods proposed for adjusting search ranges[6]-[11], those in [6] and [7], a list of search range candidates is previously prepared, and a single range is chosen from it on the basis of either prediction error values or of the motion vector values previously obtained for adjacent blocks. Both of these methods suffer, however, from the fact that the number of search-range candidates is insufficient to reflect meaningfully the large number of variations in actual video motion. By way of contrast, in [8], search ranges are calculated directly, not chosen from candidate lists. Unfortunately, however, horizontal and vertical search ranges cannot be calculated independently. As a result, a wide search range in both horizontal and vertical directions will be used even when, for example, video motion is great only in the horizontal direction alone. While the method proposed in [9] also adjusts search range at each block, since it modifies only ±1 pixel from the search range for the previous block, it cannot adapt to sudden motion change. The methods proposed in [10] and [11] differ in that they minimize search range by shifting the search start point, but they are of limited value because their performance depends on the reliability of each start point prediction.

Since motion in a single frame will vary from block to block, it is necessary to adjust the search range at each block, and since it will not be constant from scene to scene in a video sequence, it is also necessary to adjust the search range at each frame. Furthermore, since horizontal motion and vertical motion occur independently, it is also necessary to adjust horizontal and vertical search ranges independently. This paper presents an algorithm able to do all these things. The search range for each frame is determined on the basis of the accuracy of motion estimation achieved in the previous frame. The smaller the number of correct vectors obtained for a previous frame, the wider the search range chosen for the frame that follows. Conversely, so long as a large number of correct vectors are obtained for a previous frame, and so long as none of them is especially large, a narrow search range can be chosen. More specifically, search range is determined on the basis of both the sum of the absolutes of the motion vectors and the sum of the prediction errors for the previous frame. While the search range chosen for any given block will not exceed the search range chosen for its frame, that range may be reduced from this maximum on the basis of the motion vector values of those adjacent blocks for which such vectors have already been obtained. This makes it possible to adjust optimally both the area and shape of each search range.

The proposed method achieves both fast and accurate motion estimation. In the encoding of an SDTV size video, the addition of such range adjustment results in a reduction in the computational complexity of motion estimation of roughly 65%, while maintaining the same video quality.

The subsequent sections of this paper are organized as follows: Section 2 discusses the proposed algorithm's method of determining a search range for each frame, and it presents a performance evaluation; Section 3 discusses the proposed

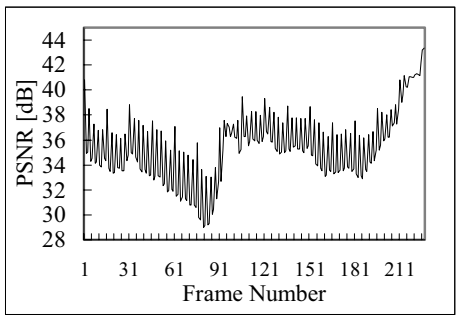


Fig.1 PSNR per frame with a fixed ±32 pixel search range (sports)

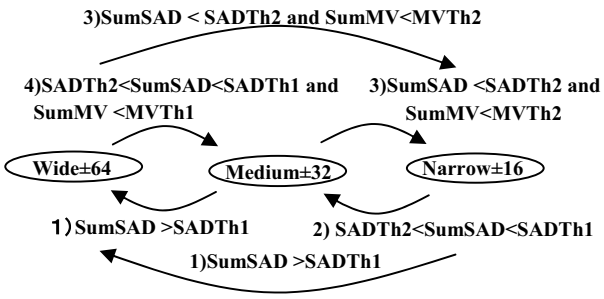


Fig.2 Search range selection for individual frames.

algorithm's method of determining a search range for each block, and it presents a further performance evaluation; Section 4 summarizes our work.

## 2. SEARCH RANGE DECISION FOR EACH FRAME

### 2.1. Proposed Algorithm

In motion estimation, motion vectors that exceed the search range cannot be detected, and when this happens, since sufficient motion compensation efficiency cannot be obtained, video quality will be degraded in the encoding process. Figure 1 shows PSNR per frame when an SDTV size video was encoded at 4 Mbps CBR and a fixed ±32 pixel search range was adopted. The video sequence, from a sports program, included both large and small motions. Hereafter, we refer to this sequence as "sports." As may be seen in Fig.1, the PSNR from the 40th frame to the 95th frame is degraded. This portion of the sequence contained very large motions, for which a ±32 pixel search range was insufficient, resulting in deterioration in both motion compensation efficiency and PSNR. While such PSNR degradation might be avoided with a wider search range, the unneeded computational complexity created by this range at scenes with only small motions would be wasteful.

To avoid this problem, we have developed a method for making adaptive search range decisions at each frame. Search range is modified on the basis of the motion estimation results for the previous frame. The search range for any given frame is chosen from among three candidates, i.e., ±64, ±32, and ±16 pixels. These values are coincident with the maximum values of describable motion vectors, as determined by an f_code of the MPEG standard.

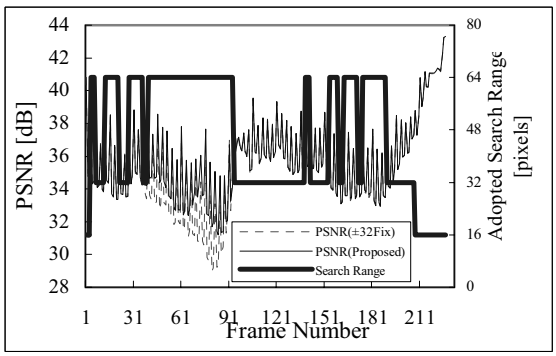


Fig.3 PSNR per frame with search ranges adjusted at each frame (sports)

In search range selection, the sum of absolute vector values (SumMV) and the sum of prediction errors (SumSAD) are first calculated. To do this, the prediction error for each block is calculated on the basis of the sum of the absolute difference (SAD) between that block and a reference block. When this SumSAD exceeds a certain threshold value, the largest search range (±64 pixels) will be chosen for the next frame. When both SumSAD and SumMV are smaller than certain threshold values, the narrowest search range (±16 pixels) will be chosen for the next frame. That is, each search range is determined by comparing SumSAD and, at times, SumMV with predetermined threshold values, as may be seen in Fig.2. In this figure, the thresholds for SumMV are MVTh1 and MVTh2 (MVTh1 > MVTh2), and the thresholds for SumSAD are SADTh1 and SADTh2 (SADTh1 > SADTh2)

### 2.2. Performance Evaluation for the Proposed Algorithm

We incorporated the proposed method into a software MPEG-2 encoder and compared its video quality with that for a conventional approach. We encoded "sports" for the same conditions as those which resulted in the graph seen in Fig.1, and then calculated PSNR per frame. On the basis of results obtained in a preliminary experiment, we determined (MVTh1, MVTh2)=( $15\times10^4$ , $10\times10^4$ ) and (SADTh1, SADTh2)=( $35\times10^5$ , $25\times10^5$ ) to be suitable thresholds for SDTV size videos. Figure 3 shows the PSNR per frame. The dotted line indicates values for the same fixed ±32 pixel search range as was used for the results in Fig.1. The thick line indicates search ranges chosen with the proposed method. As may be seen in Fig.3, the wide search range (±64 pixels) chosen for the scene from the 40th frame to the 95th frame resulted in improved PSNR.

We next evaluated the relationship between computational complexity and average PSNR, both for our method and for fixed range values. To do this, we employed MPEG-2 encoder software that fully optimizes the processor's Intel architecture so as to achieve fast encoding. We also employed a two-step hierarchical search for fast and accurate estimation. In this search, a full search was first executed on half-resolution images obtained by sub-sampling. Next, refinement of the obtained candidate vectors was executed on the basis of a narrow range full search executed on original resolution images.

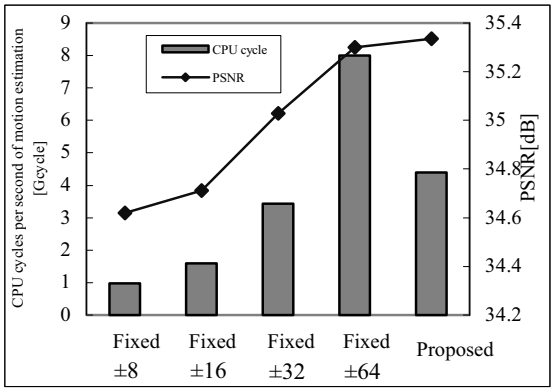


Fig.4 Comparison of CPU clock cycles and average PSNR

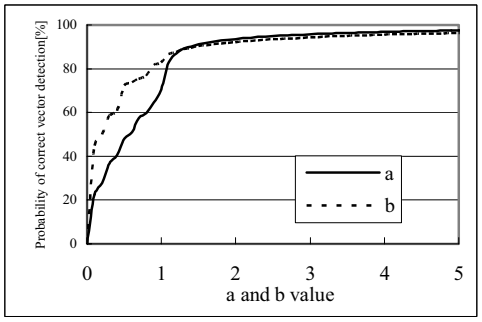


Fig.6 Probability of correct motion vector detection

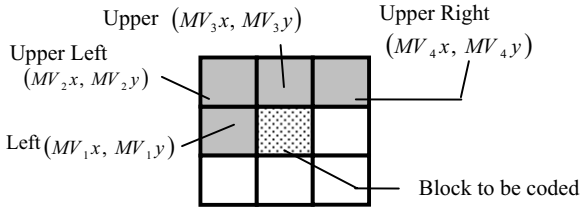


Fig.5 Adjacent blocks used for search range decisions

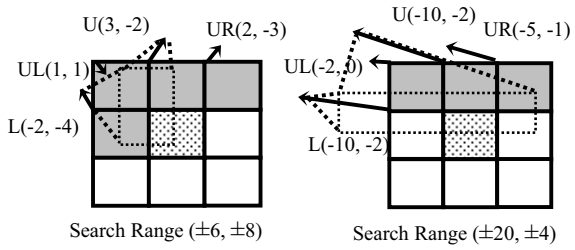


Fig.7 Examples of search range decisions (U:Upper, UL:Upper Left, L:Left, UR:Upper Right)

Figure 4 shows average PSNR values for "sport" with respect to CPU clock cycles per second of motion estimation on a 3.2GHz Pentium4 PC. With the proposed method, since a wide search range (±64 pixels) was chosen for scenes with large motions, more CPU cycles were needed than were needed with a fixed ±32-pixel search range. Computational complexity was 45% less, however, than that with a fixed ±64-pixel search range. Further, as may be seen in Fig.4, PSNR with the proposed method was as high as that with a fixed ±64-pixel search range.

## 3. SEARCH RANGE DECISION FOR EACH BLOCK

### 3.1. Proposed Algorithm

Since motion is not constant across a frame, varying from block to block, computational complexity can be further reduced by narrowing the search range for certain blocks below the maximum represented by the range for the frame. In the proposed method, a search range for each block is obtained in the manner described below.

A search range $(\pm SRx, \pm SRy)$ is directly calculated on the basis of a function of any adjacent motion vectors (left, upper left, upper, or upper right) which have already been obtained. In this paper, we use the function defined as:

$$SRx = a \max\left(|MV_1x|, |MV_2x|, |MV_3x|, |MV_4x|\right) \quad (1)$$

$$SRy = b \max\left(|MV_1y|, |MV_2y|, |MV_3y|, |MV_4y|\right) \quad (2)$$

where $a$ and $b$ are constant values, $(MV_1x, MV_1y)$ is the motion vector of the left block, $(MV_2x, MV_2y)$ is the motion vector of the upper left block, $(MV_3x, MV_3y)$ is the motion vector of the upper block, and $(MV_4x, MV_4y)$ is the motion vector of the upper right block (See Fig.5). The search range for the frame is used as a maximum search range for each block; that is, the search range for the block will not exceed the frame search range.

When $a$ and $b$ values are small, search ranges will also be small, as will computational complexity. In this case, however, it will not be possible to obtain optimal motion vectors in scenes with large motions. On the other hand, when $a$ and $b$ values are large, search ranges will also be large. In this case, while it will be possible to obtain optimal motion vectors in scenes with large motions, much computational complexity will be required. That is, it is necessary to determine optimal $a$ and $b$ values for a desired balance between range and complexity. To do this, we first examined the probability that correct vectors might be detected for various $a$ and $b$ values, defining a correct vector as one which would be obtained by a full-search algorithm with fixed a ±64-pixel search range. We examined the probability that the correct vectors would be included search ranges obtained by means of (1) and (2). In this experiment, we used eight standard sequences ("ballet", "bus", "carousel", "cheer", "flower", "football", "mobile", and "tennis"). Figure 6 shows the result of our experiments. As may be seen, when $a$ and $b$ are 2.0, 94% of the correct vectors can be obtained. Since this probability does not change drastically when $a$ and $b$ are greater than 2.0, we use $a$=$b$=2.0. Figure 7 (a) shows an example of search range decisions at $a$=$b$=2.0. Search range is calculated as twice the maximum absolute value of the four motion vectors in adjacent blocks. With our proposed method, since search range is calculated independently in the horizontal and vertical directions,

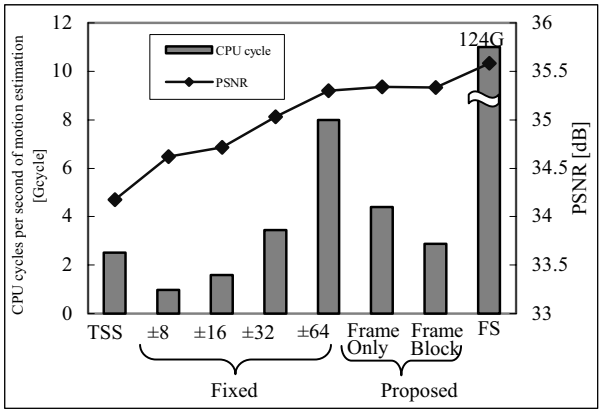


Fig.8 Comparison of CPU cycles and PSNR (sports)

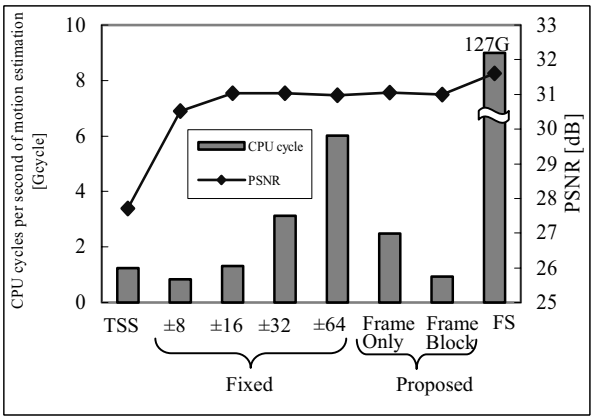


Fig.9 Comparison of CPU cycles and PSNR (bus)

obtained search ranges change not only in size but also in shape (horizontal and vertical ratio). For example, as may be seen in Fig.7 (b), when adjacent blocks have a large vector in the horizontal direction, a horizontally wide search range is obtained. In general, since there is a correlation between motions in adjacent blocks, we can predict that the current block to be coded will also contain large motion in the horizontal direction, and our method results in a search range chosen to suit this prediction. Additionally, the choice here of a narrow search range in the vertical direction helps avoid the creation of unneeded computational complexity.

### 3.2. Performance Evaluation of the Proposed Algorithm

In order to determine the relationship in our proposed method between average PSNR and computational complexity, we incorporated it into the previously described software MPEG-2 encoder and encoded "sports." Figure 8 shows CPU clock cycles per second of motion estimation vs. average PSNR both for our method and for fixed ranges. With our method, since the calculated search range may often be narrower than the maximum search range for any given block, CPU clock cycle can be kept relatively low, as compared to those for large fixed ranges: 65% less than those for a fixed ±64-pixel search range, while maintaining the same video quality. With the proposed algorithm, we can achieve almost as low computational complexity as that of the TSS, while at the same time achieving a roughly 1.2dB higher PSNR.

We also conducted a similar comparative evaluation of computational complexity vs. average PSNR for "bus." Figure 9 shows the results. Since "bus" does not include large motions, it does not need a ±64-pixel search range. Good PSNR can be achieved with a ±16-pixel search range. The proposed method calculates optimal search ranges narrower than ±16-pixels, and computational complexity can be reduced by 30% less than that for a fixed ±16-pixel search range, which indicates that the proposed method can also be effective for small-motion video sequences.

## 4. CONCLUSION

In this paper, we have proposed an adaptive search range decision algorithm for use with both frames and individual blocks. It offers both fast and accurate motion estimation. Simulation results show that when an SDTV size video is encoded, the computational complexity of motion estimation can be reduced by roughly 65% from that for a fixed wide search range, while maintaining the same video quality.

## 5. REFERENCES

[1] ISO-IEC/JTC1/SC29/WG11, "Generic coding of moving pictures and associated audio," IS13818-2, 1994.

[2] T. Koga, et al., "Motion compensated interframe coding for video conferencing," *Proc. National Telecommun. Conf.*, pp. G5.3.1-5.3.5, 1981.

[3] R.Li, et al., "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol.4, no.4, pp.438-442, Aug. 1994.

[4] L.M.Po and W.C.Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol.6, no.3, pp.313-317, June 1996.

[5] H.S.Oh and H.K.Lee, "Adaptive adjustment of the search window for block-matching algorithm with variable block size," *IEEE Trans. Consumer Electronics*, vol.44, no.3, pp.659-666, Aug. 1998.

[6] Y.H.Choi and T.S.Choi, "Fast motion estimation techniques with adaptive variable search range," *IEICE Trans. Fundamentals*, Vol.E82-A, No.6, pp.905-910, June 1999.

[7] K.Ramkishor and S.Krishna, "Spatio-temporal correlation based fast motion estimation algorithm for MPEG-2," *Proc. of Asilomar Conf. on Signals, Systems, and Computers*, Vol.1, pp.220-224, Nov. 2001.

[8] P.I.Hosur, "Motion adaptive search for fast motion estimation," *IEEE Trans. Consumer Electronics*, vol.49, no.4, pp.1330-1340, Nov. 2003.

[9] C.H.Lin, et al., "DSRA: A block matching algorithm for near-real-time video encoding," *IEEE Trans. Consumer Electronics*, Vol.43, No.2, pp.112-122, May 1997.

[10] L.Luo, et al., "A new prediction search algorithm for block motion estimation in video coding," *IEEE Trans. Consumer Electronics*, Vol.43, No.1, pp.56-61, Feb. 1997.

[11] K.L.Chung and L.C.Chang, "A new predictive search area approach for fast block motion estimation," *IEEE Trans. Image Processing*, Vol.12, No.6, pp.648-652, June 2003.