ADAPTIVE BLOCK SUB-SAMPLING ALGORITHM FOR MOTION-ESTIMATION ON SIMD PROCESSORS

Munish Jindal and G. Nageswara Rao

Emuzed India Pvt. Ltd.,
munish@ieee.org and gnr@emuzed.com

ABSTRACT

Block-motion vector estimation is an integral part of commercial video compression standards. This paper describes an adaptive technique to select predefined sub-sampled block patterns to reduce the computation complexity of Block-Matching criterion (BM) in Motion Estimation (ME). The search-window texture complexity and 8x8 sub-block texture complexity are used to select the sub-sampling pattern for each sub-block. The proposed technique is suitable for both four motion-vectors (4MV) and one motion-vector (1MV) calculation of 16x16 macro-block (MB). The sub-sampling patterns are designed to completely exploit the 32-bit and 64-bit Single Instruction Multiple Data (SIMD) architecture available on media processors and General Purpose Processors (GPPs). The simulation results show that significant speed improvement in ME module can be gained with better PSNR compared to fixed block sub-sampling techniques.

1. INTRODUCTION

Motion Estimation (ME) is an essential and the most computation intensive process in standard based video-encoders [1]. It efficiently removes the temporal redundancy between frames by motion-compensating the blocks using Block-Matching Algorithms (BMA). There are several choices of matching criterion for BMA based upon parameters like Mean Square Error (MSE), Mean of Absolute Difference (MAD), and Sum of Absolute Difference (SAD). Among these SAD is most popular because it doesn't involve any multiplication or division operations.

The full-search algorithm (FSA) is the simplest BMA. It provides the optimal result by matching all possible candidates within the search window. The 8x8 sub-block motion vector search is centered on 16x16 macro-block motion vector with a search window of ±2 pixels. [3] However, FSA requires extreme computations and is not practical for real-time applications.

Research in last decade has led to the development of fast and efficient ME algorithms. The most famous fast ME algorithms reported in literature are Three Step Search (TSS), New Three Step Search (NTSS), Diamond Search (DS), Motion Vector Field Adaptive Search Technique (MVFAST) and Spatio-Temporal correlation based Search (STS) [4]-[8]. These fast ME algorithms reduce the computational complexity by reducing the number of search points inside the search window.

The computational complexity of ME algorithm can be further reduced by using Fast matching techniques (FMTs). FMTs reduce the computational complexity of ME by decreasing the number of pixels on which the cost function is calculated [9]-[11]. FMT is based upon the assumption that all pixels in a block move by the same amount; therefore motion-vector can be predicted by using only a fraction of the pixels lying inside the block. However, it has been shown that if less number of pixels are used to predict the match, it decreases the accuracy in motion estimation. Liu and Zaccarin had proposed efficient method based on pixel sub-sampling at the block level [10]. Their technique considers just one fourth of the total number of pixels forming the MB - where, four sub-sampling patterns are used depending upon search location for calculation of MAD. It is an effective technique for its quality but all positions in the search window, even if sub-sampled; need to be considered since this technique uses FSA. Therefore, it cannot be applied to search strategies such as MVFAST and STS. Chan had proposed an adaptive pixel decimation technique for the computation of BM criterion [10]. It considers 4x4 block and selects the sub-sampling patterns according to the value of luminance gradients. Chan's algorithm gives better results than Liu's algorithm, but it cannot be considered for a software implementation, as the computation complexity reduction is limited. Moreover, Chan's algorithm is not suitable for SIMD architectures.

In this paper, an adaptive algorithm has been proposed to predict SAD. The proposed technique is suitable for software implementation of video encoder on SIMD architecture based processors [2]. The proposed algorithm is suitable for all ME search strategies. It makes

0-7803-8554-3/04/$20.00 ©2004 IEEE.

use of texture complexity parameter of reference-frame macro-blocks and current-frame sub-blocks to select the predefined sub-sampled sub-block pattern. It doesn't increase the computational complexity because the texture complexity parameter of blocks is always calculated in an encoding process to make INTRA/INTER decision. The reduction in overall complexity of ME module makes the proposed algorithm practical for software implementation of MPEG-4 video encoder on media processors.

The organization of this paper is as follows. Section 2 briefly describes the overview of two types of adaptive pixel-decimation techniques available in literature. In Section 3, different sub-sampling patterns and texture complexity parameters are described. In Section 4, the proposed adaptive algorithm has been presented, which selects one of the predefined sub-block patterns based on the texture content of the sub-block and the texture content of the search window. Section 5 shows the simulation results of PSNR and computation reduction. The concluding remarks are given in Section 6.

2. ADAPTIVE PIXEL-DECIMATION TECHNIQUES

Adaptive pixel decimation techniques can be categorized into following two methods as follows:

2.1 Local Adaptive Pixel-Decimation

In these pixel-decimation techniques, the macro-block is divided into set of fixed sub-sampled patterns P1, P2, P3, etc. The fixed sub-sampled patterns P1, P2, P3, etc are adapted for calculation of SAD [10].

2.2 Global Adaptive Pixel-Decimation

This technique doesn't fix the macro-block pattern at the start of the BMA. Here, pixels are selected only when they have the features important in determining the match [12]. This technique is better than local adaptive techniques in terms of quality, but is impractical due to computational complexity involved in adaptation at the pixel level. Moreover, this technique cannot be implemented on SIMD architecture based processors.

This paper proposes a local adaptive pixel-decimation technique suitable for implementation on SIMD processors.

3. TEXTURE CLASSIFICATION

3.1 Block Sub-Sampling Patterns

In this section, a set of pixel patterns is defined. These patterns are designed to exploit SIMD architectures. The choice of patterns has been done considering an 8x8 sub-block that represents one fourth of the macro-block for motion vector calculation. Fig. 1.a shows patterns suitable for 32-bit SIMD architectures and Fig. 1.b shows patterns suitable for 64-bit SIMD architectures. In patterns P1, P2, and P3, groups of four consecutive pixels have

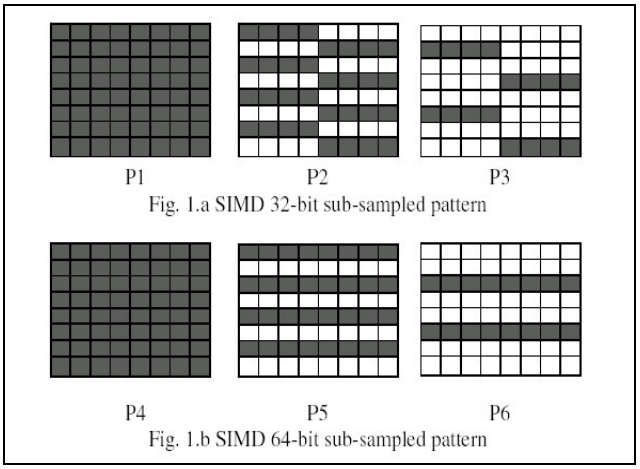

Fig. 1.a SIMD 32-bit sub-sampled pattern

Fig. 1.b SIMD 64-bit sub-sampled pattern

Fig. 1 Block Sub-Sampled Patterns

been considered to completely exploit the 32-bit SIMD architectures. This grouping also results in reduced memory accesses, since it is possible to load 4 adjacent pixels directly from memory into an unsigned 32-bit register.

3.2 Texture Classification of Block

The following parameter is used to classify the texture content of 8x8 sub-block:

$$B_V = \sum_{i=0}^{63} \left| B_i - \overline{B} \right|$$

The mean of the sub-block is denoted as $\overline{B}$.

The texture content of a macro-block is the sum of texture content of four 8x8 sub-blocks enclosed in it. The calculation of $\mathbf{B_V}$ is not an overhead for the proposed technique as it is always calculated in the video encoding process to make an INTRA/INTER mode decision for all macro-blocks. The above-mentioned parameter $\mathbf{B_V}$ can be changed based on encoder strategy for making INTRA/INTER mode decision, e.g. variance can be used against $\mathbf{B_V}$ to classify the texture content of block. The high value of $\mathbf{B_V}$ indicates high texture content in the block and low value of $\mathbf{B_V}$ indicates less texture content in the block.

3.3 Texture Classification of Search Window

The texture complexity parameter of reference–frame macro-blocks $\mathbf{RB_v}$ is used to classify the texture content of the search window. Let **B(cx, cy)** represent the macro-block at pixel location **(16cx, 16cy)** in the frame. The texture complexity ($\mathbf{S_{16x16}}$) parameter of search window for macro-block **B(cx, cy)** motion-vector calculation is modeled as:

$$S_{16\times16} = \frac{1}{2*(2a+1)*(2b+1)} \sum_{j=-b}^{+b} \sum_{i=-a}^{+a} RB_V(i+cx, j+cy)$$

where:

$$a: \left\lceil \frac{Horizontal\_Search\_Range}{16} \right\rceil$$

$$b: \left\lceil \frac{Vertical\_Search\_Range}{16} \right\rceil$$

The texture complexity ($\mathbf{S_{8x8}}$) parameter of search window for sub-blocks motion-vector calculation is modeled using the texture complexity parameter of reference-frame macro-blocks $\mathbf{RB_v}$ and current macro-block motion-vector as:

$$S_{8\times8} = \frac{1}{8}\begin{pmatrix} RB_V(i', j') + RB_V(i'+1, j') \\ + RB_V(i', j'+1) + RB_V(i'+1, j'+1) \end{pmatrix}$$

where:

$$i' = \left(cx + \left\lfloor \frac{mvx_{16x16}}{16} \right\rfloor \right)$$

$$j' = \left(cy + \left\lfloor \frac{mvy_{16x16}}{16} \right\rfloor \right)$$

The calculation of $\mathbf{S_{16x16}}$ and $\mathbf{S_{8x8}}$ is not computationally expensive as texture content parameter of reference-frame macro-blocks is available after encoding of the previous frame. The texture complexity parameters of search window ($\mathbf{S_{16x16}}$ and $\mathbf{S_{8x8}}$) and current sub-block ($\mathbf{B_V}$) give useful information for deciding the number of pixels to be used for better prediction of SAD. The higher parameter value indicates significant intensity variations; therefore more number of pixels should be considered for efficient prediction of SAD. Similarly, less number of pixels is enough to predict SAD for low texture parameter value. Moreover, less number of pixels is enough to predict SAD, when current sub-block texture and search window texture are similar. Thus, based upon the above-described texture classification, the pattern is selected out of P1, P2 and P3, as described in the next section.

4.PROPOSED ALGORITHM

Sub-sampling patterns are selected for each of the four sub-blocks of a macro-block from a set of fixed patterns P1, P2 & P3, depending upon the texture content of the sub-blocks and texture content of search window. Therefore, the proposed technique represents 3x3x3x3 = 81 possible texture patterns for a macro-block. The different sub-sampling patterns are chosen based upon the following pseudo code:

For MB motion vector calculation:

```
For all four sub-blocks (8x8):
If (4*BV8x8+ S16x16)>(5*Th1)
      Choose pattern P1 (64 pixels)
Else If (4*BV8x8+ S16x16)>(5*Th2)
If (2*Abs(BV8x8−S16x16))> Th2
      Choose pattern P1 (64 pixels)
Else
      Choose pattern P2 (32 pixels)
Else If (4*BV8x8+ S16x16)>(5*Th3)
If (2*Abs(BV8x8−S16x16))> Th3
      Choose pattern P2 (32 pixels)
Else
      Choose pattern P3 (16 pixels)
Else
      Choose pattern P3 (16 pixels)
```

For sub-block motion vector calculation:

```
If (2*BV8x8+ S8x8)>(3*Th4)
      Choose pattern P1 (64 pixels)
Else If (2*BV8x8+ S8x8)>(3*Th5)
If (2*Abs(BV8x8−S8x8))> Th5
      Choose pattern P1 (64 pixels)
Else
      Choose pattern P2 (32 pixels)
Else If (2*BV8x8+ S8x8)>(3*Th6)
If (2*Abs(BV8x8−S8x8))> Th6
      Choose pattern P2 (32 pixels)
Else
      Choose pattern P3 (16 pixels)
Else
      Choose pattern P3 (16 pixels)
```

The thresholds used for simulations are Th1=2048, Th2=1136, Th3=640, Th4=1792, Th5=704 and Th6=320.The proposed algorithm uses sum of absolute differences (SAD) for evaluation of block matching in view of its low computation cost. Note that here, SAD differs from its common definition that it is calculated over a set of selected pixels, instead of all pixels in sub-block. Therefore, the SAD is normalized by multiplication of 2 and 4 for SAD obtained by pattern P2 and P3 respectively. The thresholds used can be increased for quality-speed trade-off, with a gradual decrease in quality.

5. SIMULATION RESULTS

The simulation results are generated using prototype of MPEG-4 Simple Profile encoder for original sequences encoded at a constant bit-rate. There are 25 frames in a

Sequences	Pattern P1	Pattern P2	Pattern P3	Adaptive
Akiyo	44.89 (64)	44.87 (32)	44.80 (16)	44.80 (28)
Foreman	33.18 (64)	32.89 (32)	30.95 (16)	32.94 (37)
McDonald	29.62 (64)	29.18 (32)	28.14 (16)	29.44 (41)
Motorrace	27.48 (64)	27.16 (32)	26.20 (16)	27.29 (46)
Paris	31.07 (64)	30.93 (32)	30.27 (16)	31.01 (50)
Rowing	26.39 (64)	26.20 (32)	25.70 (16)	26.33 (49)
Rugby	25.88 (64)	25.63 (32)	25.25 (16)	25.74 (46)

Table 1. PSNR and Average number pixels per 8x8 sub-block for sub-sampled algorithms

Sequences	Pattern P1	Pattern P2	Pattern P3	Adaptive
Akiyo	82	90	95	91
Foreman	39	43	50	42
McDonald	36	41	48	39
Motorrace	35	40	47	38
Paris	53	58	62	55
Rowing	33	39	44	36
Rugby	33	39	43	37

Table 2. Number of frames per second on 200Mhz TM1300 processor

GOV (I, P, P, P, P, …). The search window size is ±15 pixels. The proposed algorithm shows better PSNR to complexity results for various ME techniques like STS, MVFAST, DS and TSS. STS motion-estimation algorithm is used for generating simulation results. To check the robustness of proposed algorithm, analysis has been done for low, medium & high motion and texture sequences at different bit-rates. All sequences are rectangular video with CIF format. Table 1 shows the PSNR values and the computation complexity reduction for test sequences encoded at bit rate of 384Kbps. It can be observed that the proposed algorithm is better than fixed sub-sampled techniques in terms of its quality even for high texture and high motion sequences. The proposed algorithm has been implemented on TriMedia TM1300 processor[1]. Table 2 shows results in terms of frames encoded per second on a 200MHz TM1300 processor. When fixed sub-sampled techniques are used, encoder can operate at certain fixed quality vs. speed trade-off points. However, using the proposed technique, the encoder can operate at any trade-off point suitable to its need. As shown in Fig. 2, the encoder can choose to operate anywhere on the curve by using appropriate thresholds. The thresholds are computed empirically to operate at the desired trade-off point.

6. CONCLUSION

In this paper an adaptive block sub-sampling algorithm has been presented, which is suitable for MPEG-4 SP encoder implementation on media processors supporting SIMD architecture. The results show that the proposed algorithm performs better in terms of quality for high texture and high motion sequences.

7. REFERENCES

[1] "Information Technology – Generic Coding for Audio-Visual Objects – Part 2: Visual," *MPEG-4 Standard, 14496-2, ISO/IEC Standard.*

[2] "TriMedia TM-1300 Data Book" - May 2000.

[3] "MPEG-4 Video Verification Model Version 18.0", ISO/IECF JTC1/SC29/WG11 N3908.

[4] R. Li, B. Zeng, and M. L. Liou, "A New Three-Step Search Algorithm for Block Motion Estimation," IEEE Trans. Circuits Systems for Video Technology, vol. 4, pp. 438–443, Aug. 1994.

[5] L. M. Po and W. C. Ma, "A Novel Four-Step Search Algorithm for Fast Block Motion Estimation," IEEE Trans. Circuits Systems for Video Technology, vol. 6, pp. 313–317, June 1996.

[6] S. Zhu and K. K. Ma, "A New Diamond Search Algorithm for Fast Block Matching Motion Estimation," IEEE Trans. Image Processing, vol. 9, pp. 287–290, Feb. 2000.

[7] "Optimization Model Version 2.0", ISO/IEC JTC1/SC29/WG11 N3675.

[8] K.Ramkishor and S.Krishna, "Spatio-Temporal Correlation Based Fast Motion Estimation Algorithm for MPEG-2", *IEEE Proceedings of the 35th Asimolar conference on Signals Systems and Computers*, November 2001, pp. 220-224.

[9] Bede Liu and Andre Zaccarin, "New Fast Algortihms for the Estimation of Block Motion Vectors", IEEE Trans. On CSVT, vol. 3, no. 2, pp. 148-157, April 1993.

[10] Yui-Lam Chan, Wai-Lam Hui and Wan-Chi Siu, "A Block Motion Vector Estimation using Patter based Pixel Decimation", IEEE International Symposium on Circuits and Systems, June 1997.

[11] F. Moschetti and Eric Debes, "A Fast Block Matching for SIMD Processors using Sub-Sampling", IEEE International Symposium on Circuits and Systems, May 2000.

[12] Yankang Wang, Yanqun Wang and Hideo Kuroda, "A Global Adaptive Pixel-Decimation Algorithm for Block-Motion Estimation", IEEE Trans. Circuits Systems for Video Technology, vol. 10, no. 6, pp. 1006–1011, September 2000.

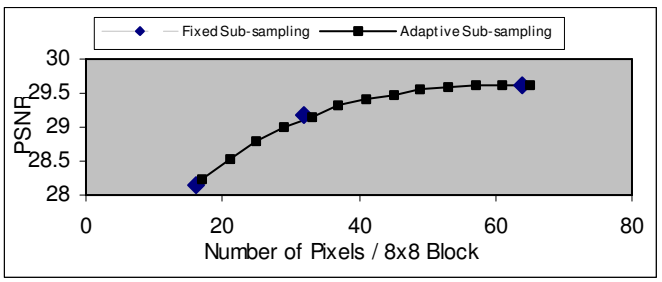

Fig 2. PSNR vs. pixels per 8x8 sub-block for McDonald sequence

[1] TM1300 is a trademark of TriMedia Technologies Inc.