A FAST VIDEO MOTION ESTIMATION ALGORITHM FOR THE H.264 STANDARD

P. Nasiopoulos, M. von dem Knesebeck

Department of Electrical and Computer Engineering
University of British Columbia, BC, Canada
2356 Main Mall, Vancouver BC, Canada
{panos, matthiask}@ece.ubc.ca

ABSTRACT

Video applications are becoming an essential component for mobile devices. H.264, the latest video-coding standard, shows significant potential in terms of bandwidth savings at the cost of substantially increased complexity compared to former standards. The computing power currently available on mobile devices is not sufficient to allow high quality real-time encoding using H.264. Our algorithm uses on average only 0.41% of the computational complexity of the full search method used by H.264, leading to a significant reduction in computational requirements and enabling real-time applications for mobile devices with the efficiency of H.264.

1. INTRODUCTION

Video applications are becoming an essential component of mobile devices. The requirement, however, of low power consumption coupled with limited processing power are challenges that prevent us from achieving real-time high quality applications. H.264 is the latest and most advanced video coding standard to date [1]. It yields a 50% improvement in compression efficiency compared to previous video standards such as MPEG-2 and MPEG-4. A number of different applications such as broadcasting and HD-DVD have already adopted H.264 as their new video codec. Although H.264 shows significant potential in terms of coding efficiency and thus bandwidth savings, its computational complexity causes a considerable challenge for real-time encoding on mobile applications. Fig. 1 represents the run time analysis of a typical H.264 encoder [2]. The encoder spends up to 52% of the overall computational time on Motion Estimation (ME), clearly the most time consuming process of video compression. For this reason, a lot of research has been done in this area and a number of different methods have been proposed for improving the speed of the motion estimation search.

The Full Search Method (FS) evaluates each displacement position within a limited search area (usually ±7 pixels) and chooses the point with the minimum distortion. Although this method will identify the match with the global minimum distortion (within the search area), it requires an unjustifiable amount of computational load.

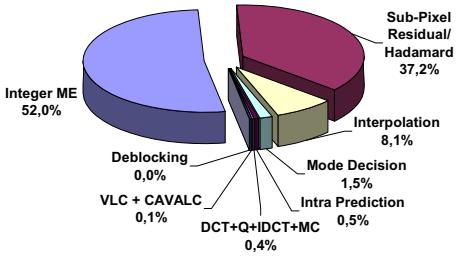

Fig. 1: Run time analysis of the H.264 encoder

In order to remedy this significant drawback, several fast search algorithms have been proposed reducing the number of search points to the ones with the highest probability. The most popular approaches are summarized in [3]. These fast methods are often center-biased, since the best match in natural video sequences is likely to be found within an area centered on the current macroblock position [4].

Among these, one of the most popular search algorithms is the Diamond Search (DS) method employing a diamond-shaped pattern and multiple refinement stages. This method reduces the computational load to 6% of the FS method [5].

A new approach that uses Sum of Absolute Differences and spatial and spatial-temporal search schemes (SADME) has recently been proposed by [6] which reduces the computational cost of the ME process drastically compared to all other existing techniques. Only 0.5% of the computational load is required compared to FS. Although SADME's performance evaluations confirm its suitability for real-time mobile applications, this method was specifically designed for H.263, a video standard with a motion estimation process much simpler than that supported by H.264.

In this study we present a new motion estimation method that uses concepts borrowed from the SADME method and is specifically designed for the H.264 standard. This new method aims at optimizing H.264's computational complexity in an effort to make it practical for mobile applications.

The paper is organized as follows: Section 2. gives a brief discussion of the ME procedure in H.264, Section 3. illustrates the proposed motion estimation method for H.264 and Section 4 elaborates on the experimental results. Conclusions are presented in Section 5.

2. NEW FEATURES OF H.264 MOTION ESTIMATION

It is well established that H.264 offers significant performance improvements over other existing video standards. These improvements are due to several new features, one of which is a much more flexible and efficient motion estimation process. Motion estimation improvements in H.264 include the number of reference frames, accuracy and block sizes. A summary of H.264's motion estimation features which are relevant to our implementation is presented in the following subsections.

2.1 Variable block-size selection

Variable motion estimation block sizes of 16x16, 16x8, 8x16 and 8x8 pixels are supported by H.264. In the case of 8x8, further partitions, which include 8x4, 4x8, or 4x4, might be used. Fig. 2 shows the various block sizes that are supported by H.264.

Blocks with more motion details can be encoded using smaller block sizes. This can improve the prediction and results in better compression rates, but at the cost of increasing computational complexity. Several variable block-size selection algorithms have been proposed to reduce the encoding time, such as the signal to noise ratio (SNR)-based algorithm [7, 8], adaptive threshold cost-based algorithm [7, 8], 3D recursive search scheme-based algorithm [7, 8], and the complexity measurement-based algorithm [9].

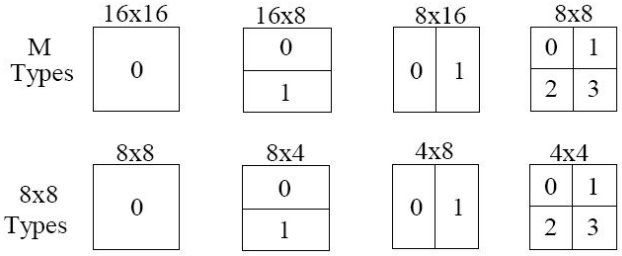

Fig. 2: Macroblock partitions for motion estimation

2.2 Motion Vector Prediction

Motion vectors (MVs) of neighboring macroblocks are often highly correlated. H.264 predicts an initial motion vector from the MVs of surrounding blocks in order to have a better starting point for the motion vector search. Then, only the difference between the predicted motion vector and the best-match motion vector is subsequently encoded in the bitstream. The challenge introduced by H.264, is that neighboring partitions may be different in size, since various block sizes are supported. Figure Fig. 3 illustrates a MV prediction pattern when all the neighboring partitions have the same size (e.g., 16x16) and Figure 4 shows a case with a current 16x16 block surrounded by blocks of various sizes.

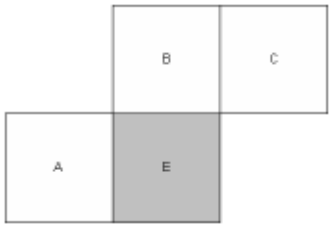

Fig. 3: Current (grey) and neighboring macroblocks (white) for MV prediction with same partition size

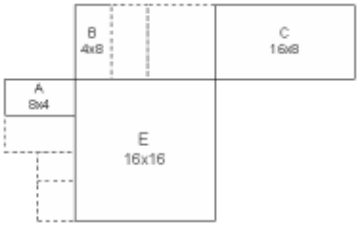

Fig. 4: Current and neighboring partitions with different partition sizes

The predicted MV is determined dependent on partition size and on the availability of nearby vectors [1]. As an example, the predicted MV in Fig. 3 is computed by taking the median of three surrounding MVs from areas A, B, and C.

2.3 Multiple reference picture motion compensation

While MPEG-2 and MPEG-4 use only one previous frame for motion compensation, H.264 supports multi-picture motion-compensated prediction. Fig. 5 illustrates this concept.

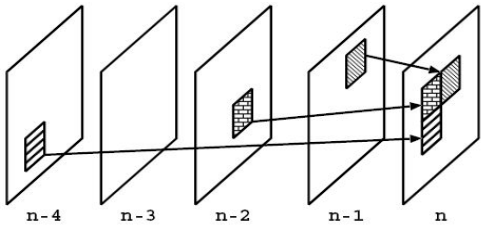

Fig. 5: Multi-frame motion compensation for H.264

It has been shown that using multiple reference frames for prediction can yield 5-20% in bit rate savings as compared to using only one reference frame [10]. However, both the encoder and decoder have to store multiple reference pictures in order to implement multi-frame motion-compensated prediction which consequently increases the encoding and decoding complexity.

Multiple reference frames are especially beneficial for situations with repetitive motion, uncovered background, camera shaking and sampling [10]. Many of these situations are pertinent for applications in mobile devices and should prove advantageous in related use-case scenarios.

3. OUR NEW MOTION ESTIMATION ALGORITHM FOR H.264

The SADME method is based on the distinct relationship between the sum of the absolute difference (SAD) for each macroblock and the corresponding motion vector (MV). SAD and MV values are highly correlated with each other; smaller SAD values indicate that the corresponding motion vector will also be smaller. SADME uses this correlation to avoid unnecessary steps in the motion estimation process and thus to reduce the computational load. This is achieved by separating the macroblocks into three categories with different search requirements, and optimizing the search schemes used for each category subsequently.

As a first step, we calculate the difference between the current and the previous frame by direct subtraction. The resulting difference frame is subdivided into 16x16 macroblocks and we calculate the SAD value for each of these macroblocks. In a second step we predict the current frame by using the motion vectors of the previous frame. We calculate the difference between the predicted frame and the current frame and compute the SAD values for each macroblock. Then, for each macroblock, the two SAD values are compared and since the smaller one will yield a better approximation for the motion vector, a new SAD array is generated using the smaller of the two values. This SAD array is the basis for classifying the macroblocks into three categories.

The thresholds that define these three categories are the mean (m) and the sum of the mean (m) plus the standard deviation (σ) of the SAD values, as shown in Table 1.

Category 1	**SAD >** *mean + std.dev.*
Category 2	*mean* < **SAD** <= *mean + std.dev.*
Category 3	**SAD** <= *mean*

Table 1: Category thresholds for macroblocks in SADME

One of H.264's main features is the use of variable motion estimation block sizes. Although every search starts with a 16x16 block, every other possible combination down to 4x4 blocks (e.g., 16x8, 8x16, 8x8, 8x4 and 4x8) is also searched. The block or combination of blocks that yields the smallest residual is chosen to represent that portion of the frame and the corresponding MVs are calculated. This is a departure from the standard macroblock size used in all previous video codecs. For simplicity reasons and reduction in complexity, our method uses only the 16x16 prediction from the previous macroblock to obtain the difference for the temporal case. This way the resulting SAD arrays, one from direct difference and the other from prediction, have the same size. We found that in the case of H.264, using the macroblock classification in 3 categories can significantly reduce the number of search paths performed by an H.264 encoder. Performance evaluations have shown that for macroblocks belonging to category 3, the search can be limited to the 16x16 block size. It turns out, with a very high degree of accuracy, that this is also the size that the encoder would chose at the end of its tree-structured search. For macroblocks in category 2, we stop the macroblock size search at 16x8 and 8x16; these levels preserve a very high level of accuracy for this category. Finally, for category 1, all possible block sizes are tested.

SADME reduces the computational load by estimating motion vectors using only a subset of the total 256 pixels available in each 16x16 macroblock. Two of these pixel subset patterns are shown in Fig. 6.

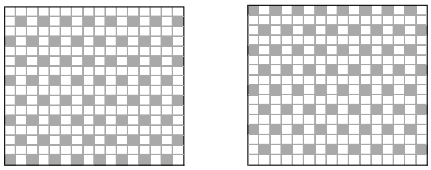

Fig. 6: Two pixel subsets covering ¼ of the MB pixels

We investigated the suitability of using subsets for the motion estimation process in H.264. Our tests have shown that, for 16x16 MBs, only one of the subsets is needed for accurately estimating the corresponding motion vector, resulting in 75% reduction in pixel-related computations. For smaller partitions, however, the accuracy of using only ¼ of the MB pixels is lacking the required accuracy. Instead, a combination of two pixel subsets is required for all other block sizes down to 4x4 pixels. The subset that yields the best results for our implementation is illustrated in Fig. 7. This approach cuts the number of pixel computations to half.

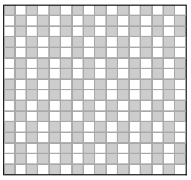

Fig. 7: Subset covering ½ of the MB pixels (combinat. from Fig. 6)

In SADME, the SAD values are sorted in ascending order and the search requirements for category 2 and category 3 are reduced by assuming that if the motion vectors of several successive macroblocks are (0,0), then the rest of the motion vectors are also (0,0). This approach, however, introduces two other drawbacks along with some unwanted distortion. One is additional time needed for sorting the SAD array. The second is that an initial search point has to be calculated for each MV (using a new scheme introduced by SADME) because of the spatially random order of the macroblocks within the array. Performance evaluations have shown that, for the case of H.264, the above assumption yields even higher levels of distortion than H.263 and MPEG. We also observed that the reduction in number of point calculations is less than 1%. For this reason, we depart from the SADME approach in our implementation by avoiding the sorting of the SAD values and macroblocks. Consequently, we entirely eliminate the additional calculations required for sorting and initializing MVs in the SADME method.

When multiple reference frames are used, our algorithm must be modified to take advantage of this temporal redundancy. First, for each macroblock of the present frame we calculate the difference between this macroblock and each of the corresponding macroblocks of all the previous reference frames. The frame that yields the smallest SAD value is chosen as the most appropriate frame for initial search. This SAD value is saved in one array along with information about the reference frame. In addition to the above direct difference approach, the information of the previous motion vector for the macroblock is used to determine which of the previous reference frame was used for deriving the value of that MV. Then, the difference between the present macroblock and the predicted macroblock of that specific frame is calculated and used in another SAD array; the information about the previous MV and the corresponding reference frame is also saved. For each macroblock, the smallest SAD value of the two values is used for generating the final SAD array. The rest of the process for estimating the motion vectors is the same as in the case of one reference frame (which has been described above). The Diamond Search pattern is used for estimating the motion vectors in all three categories.

4. EXPERIMENTAL RESULTS

The new motion estimation algorithm was implemented using the T264 software codec [11]. Although T264 is still in development, it features a distinguished encoding speed compared to other H.264 codecs such as JM [12].

Several test video sequences were encoded for our performance evaluations. The following parameter set was used: GOP: IPPP, I-frame interval: 15, Resolution QCIF, Ref.frames 5, QP 30.

Table 2 shows the average PSNR value for seven video streams, the size of test sequences before and after encoding, the resulting compression ratio and the number of pixel comparisons performed by H.264 using Full Search motion estimation and our method.

We observe that, for the same picture quality, our proposed algorithm is about 200 times faster than the full search (FS) method, using on average only 0.41% of its computational complexity.

5. CONCLUSION

We presented a novel motion estimation algorithm specifically designed for the H.264 standard. This algorithm takes advantage of special features of H.264 and the relationship between the Sum of Absolute Differences (SAD) value of the Macroblock (MB) and its corresponding motion vector (MV) as well as of temporal and spatial correlation present in video streams. Our method is computationally very efficient without sacrificing the quality of the video, as required for low bit rate video coding in battery powered mobile devices such as personal digital assistants and cellular telephones. For the same picture quality, the proposed algorithm is about 250 times faster than the full search method used by H.264.

6. REFERENCES

[1] I.E.G. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia, West Sussex, England: John Wiley & Sons, 2003.

[2] T. Chen, T. Huang and L. Chen, "Analysis and design of macroblock pipelining for H.264/AVC VLSI architecture," in Proc. of the Int. Symposium on Circuits and Systems, ISCAS '04, 2004, pp. 273-76.

[3] T. Kuo, T. Chan and H. Chen, "Efficient variable block size motion estimation for H.264 based on motion distribution likelihood," Proc. of SPIE, vol. 5960, pp. 19-29, Jul. 2005.

[4] R. Li, B. Zeng and M.L. Liou, "A new three-step search algorithm for block motion estimation," IEEE Transactions on Circuits and Systems for Video Technology, vol. 4, pp. 438-42, Aug. 1994.

[5] S. Zhu and K. Ma, "A new diamond search algorithm for fast block matching motion estimation," in Proc. of the Int. Conf. on Inform. Commun. and Signal Processing, 1997, pp. 292-6.

[6] H.-J. Lee, P. Nasiopoulos and V.C.M. Leung, "Fast Video Motion Estimation Algorithm for Mobile Devices," in IEEE Int. Conf. on Multimedia and Expo, 2005, pp. 370-73.

[7] A. Ahmad, N. Khan, S. Masud and M.A. Maud, "Selection of variable block sizes in H.264," in IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2004, pp. 173-76.

[8] A. Ahmad, N. Khan, S. Masud and M.A. Maud, "Efficient block size selection in H.264 video coding standard," Electron. Lett., vol. 40, pp. 19-21, Aug. 2004.

[9] A.C. Yu, "Efficient block-size selection algorithm for inter-frame coding in H.264/MPEG-4 AVC," in IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, 2004, pp. 169-72.

[10] Y. Huang, B. Hsieh, T. Wang, S. Chien, S. Ma, C. Shen and L. Chen, "Analysis and reduction of reference frames for motion estimation in MPEG-4 AVC/JVT/H.264," in IEEE Int. Conf. on Accoustics, Speech, and Signal Processing, 2003, pp. 145-48.

[11] T264 Open-Source Video Coding Framework, http://sourceforge.net/projects/t264/

[12] JM Reference Software v10.1, Aug. 2005.

Table 2: Comparison after encoding test sequences using original and modified T264 CODEC

Test sequence			Full Search Method in H.264			Proposed Method in H.264			
Name	Format	# of frms	kbits/s	# of cmp/MB*	PSNR	kbits/s	# of cmp/MB*	%of FS	PSNR
Foreman	QCIF	399	132.6	3,114,898	34.02	134.8	18,264	0.59%	33.99
Carphone	QCIF	382	145.8	3,113,268	35.15	148.1	14,680	0.47%	35.07
Salesman	QCIF	448	39.2	3,321,638	33.79	40.9	12,545	0.38%	33.75
Mother & Daughter	QCIF	960	68.9	3,122,341	35.15	70.2	10,612	0.34%	35.08
News	QCIF	299	98.3	3,115,169	35.23	100.1	10,427	0.33%	35.13
Akiyo	QCIF	299	48.4	3,115,169	37.00	48.8	9,557	0.31%	36.92
Suzie	QCIF	150	60.5	3,308,151	35.64	60.5	16,050	0.49%	35.54
Average			**84.8**	**3,172,948**	**35.14**	**86.2**	**13,162**	**0.41%**	**35.07**

* Number of pixel comparisons per Macroblock