SEARCH WINDOW ESTIMATION ALGORITHM FOR FAST AND EFFICIENT H.264 VIDEO CODING WITH VARIABLE SIZE BLOCK CONFIGURATION

Gianluca Bailo, Ivano Barbieri, Massimo Bariani, Marco Raggio

Department of Biophysical and Electronic Engineering University of Genova
Via Opera Pia 11 A, 16146 Genova, ITALY
phone: +39 0103532037, fax: + 39 0103532036, email: bailo@dibe.unige.it
web: www.esng.dibe.unige.it

ABSTRACT

Today there are new high band requests from actual multimedia services, and efficient video coders are needed in order to obtain a sensible reduction of video streams band occupation, maintaining video quality. The new video coding standard H.264 seems to be answer to fulfill these requirements.

On the other hand, the H.264 encoder complexity is largely increased, if compared with previous standards, mainly because of the high computational cost of the Motion Estimation module.

In this paper, we propose an innovative algorithm for reducing the complexity of the Motion Estimation (ME) module. The main idea is to make dynamically modifiable a previously static H.264 coder parameter: the size of the search window in motion estimation. A Motion Detection module has been added to the video coder in order to compute the search window size. Detailed motion estimation is performed only when and where it is required.

In the proposed solution, a reduction in the number of calculated SAD (Sum-of-Absolute-Differences) allows decreasing the encoder complexity, especially in low complex sequences. H.264 has been used for validating the proposed approach; this paper shows the behavior of the Search Window Estimation (SWE) algorithm using different H.264 macroblock configurations.

1. INTRODUCTION

The wide range of multimedia applications based on video compression (video telephony, video surveillance, digital television) leads to different kind of requirements for a video-coding standard (image quality, compression efficiency).

Compared with previous standards, H.264 [1] introduces many new features in all the aspects of the video encoding process. H.264 saves 50% bit-rate maintaining the same quality if compared with existing video coder standards as H.263 [2], but the complexity of the encoder has been increased of more than one order of magnitude (while the decoder is increased by a factor of 2) [3]. The high compression rate together with the good quality obtained by the H.264 standard make it suitable for a large variety of applications. Several H.264 application areas require high power efficiency (especially in the video encoder part) in order to work on embedded systems and mobile terminals. This requirement implies the need to dramatically reduce the complexity of the H.264 video encoder. Algorithm analysis shows that mode-decision and ME modules are the most complex in the H.264 encoder (especially when Rate-Distortion Optimization is used) [5]. This is mainly due to the great number of SAD calculation in ME (e. g. 1500 millions in the first 50 frames of the standard QCIF sequence foreman, using 5 reference frame and all-blocks configuration activated). There are many algorithms performing a reduction of the number of SAD calculation based on spatial and temporal correlation of motion vector [6]. In the following, we will introduce a different approach, based on our motion detection (MD) algorithms explained in section 2. The algorithm proposed in this paper can significantly reduce H.264 ME computational cost and achieves the best performance in sequences with low complexity (where spatial movement is limited in time), as we can have in both video-surveillance and video-telephony environments. Section 3 describes how the MD is utilized inside the H.264 ME, this is the base idea of the Search Window Estimation (SWE) algorithm.

2. MOTION DETECTION ALGORITHM

The Motion Detection (MD) algorithm can be subdivided into several steps:

- Binary image difference evaluation
- Subsampling of binary image
- Blob coloring with minimum blob size threshold
- Merging overlapped blobs
- Background image evaluation

All steps are in cascade, the input of MD is the luminance image and the output is a set of information about motion area (blobs) like position or size.

2.1 Binary image difference evaluation

The image difference D(n) is evaluated by the difference between the actual image I(n) and a Background B(n). Then, a threshold value (*detection value*) is applied to the result of this operation in order to obtain Dth(n), the binary image that represents motion pixels into the image.
The inputs of this block are the actual image and the background; the output is a binary image with motion pixels. The formula is:

$$D(n) = I(n) - B(n)$$
$$Dth(n)=0 \text{ if } D(n) < \text{detection value},$$
$$Dth(n)=1 \text{ if } D(n) >= \text{detection value } n.$$

2.2 Subsampling of binary image

The image difference is subdivided into 8x8 blocks and then another threshold is applied to obtain a reduced size image difference (called subsampled image), where each block is represented by a single value.
Considering one single block the performed operations are the following: the number of pixels having value 1 is calculated, if this sum is over a fixed threshold (*block-motion threshold*) the correspondent pixel of the subsampled image is 1, else 0. For example if the 8x8-block (1,1) has the sum of pixel having value 1 over the *block-motion threshold*, the correspondent pixel (1,1) into the subsampled image has value 1. This step is performed because applying the blob-coloring algorithm to the whole image is heavy, but with little precision loose can be very fast over this subsampled image. Obviously, the greater is the *detection value* the smaller is the density of the motion pixels. Therefore, this parameter determines the sensitiveness of the motion detection algorithm (setting a great value implies the detection of considerable movements only).

2.3 Blob coloring with minimum blob size threshold

Blob coloring is a computer vision technique to obtain region growing and region separation for images.
We use this technique over binary subsampled image to separate motion regions and growing it. The regions coordinates are the output of this step. Blob coloring can be performed with the algorithm described in [7]. After the blob coloring, the resulting blobs sizes are filtered through a threshold that permits to delete too little blobs (4x4 pixels sizes are eliminated).

2.4 Merging overlapped blobs

The regions coordinates evaluated in the previous step are analyzed to find out overlapped regions. If there are two or more overlapped regions, a new region is created and it has size growing to max dimension of overlapped blobs. This operation reduces blob count and permit moving region separation.

2.5 Background image evaluation

The evaluation of the background image is very important for a good MD. In fact neither the first image nor the n-1 frame are valid backgrounds.
We use an algorithm based on adaptive background evaluation. This means that the background is evaluated at run-time and it can change.
The formula for background evaluation is:

$$B(n+1)= \alpha * F(n)+ (1 -\alpha) * B(n)$$

Where α is the *learning rate*. The learning rate establishes how fast the background can change. Since we evaluate blobs, we can use this information to see if there is need to put a fast learning rate or a slow one. Inside moving regions we can use slow rate, otherwise, where there is no motion the background is updated very fast. Therefore we use two learning rate, one fast and the other slow.

3. SEARCH WINDOW ESTIMATION

The proposed algorithm is based on the idea that exhaustive motion search is useful only in video sequences containing large motions and not in low complex sequences. H.264 can use its entire feature set if there is the real need to do it. The H.264 ME full-search can be applied just when the motion detection algorithm identifies motion and, specifically in the region where the motion is detected. In particular, the motion detection module is utilized to calculate a parameter that is usually a constant value, the *search window size*. Motion detection evaluates where and when to set a large window for the motion estimation; otherwise a minimum-size window will be used. Since this parameter strongly influences the number of calculation performed in the video coder, the dynamical setting of this value can save a lot of time in the compression process with minor effects on the quality of the produced video sequence.
The scheme in Figure 1 explains where the motion detection module works in the window search estimation. The information obtained by the MD algorithm (see Figure 2), is utilized to define an approximate motion image (Figure 3). This is an approximation of the MD output result, because we need as little as possible complexity in motion detection algorithm. This image is the interface between motion detection and H.264 motion estimation.

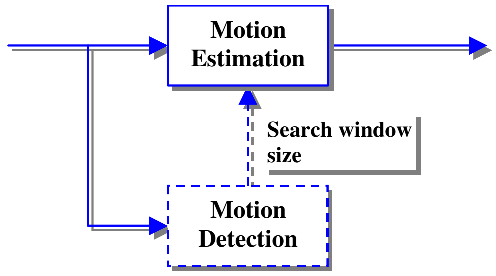

Figure 1 – Interfacing between MD and H.264

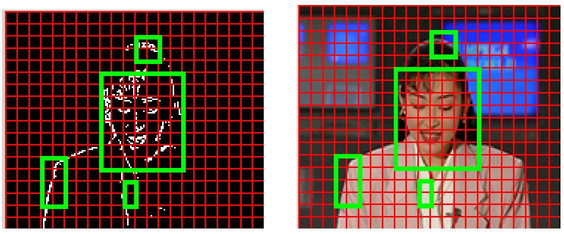

Figure 2– Image blocking and motion detection result

```
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0
0 0 1 1 1 1 1 1 0 0 0
0 0 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0
```

Figure 3- Motion Images Example

During the video coder motion search, the motion image is scanned in order to decide the search window size for every H.264 macro block. Larger window size will be applied for macro blocks where motion has been detected. Otherwise the video coder will use a minimum-size window.

4. RESULTS

The proposed algorithm has been validated with version jm60a [8] of the reference JVT software. The reported tests have been performed using three standard sequences in QCIF format. *Akiyo* and *hall* have been selected on the basis of application of interest (video-surveillance and video conference), while *foreman* has been utilized for comparison purpose in order to test the proposed algorithm with a more complex sequence. In the following tests, we encoded 100 frames of every test sequence at 30fps. The CAVLC entropy coder is used for all tests, with quantization values of 28 (Hadamard transform is not used).

The reported tests show algorithm performance for several H.264 block configurations. The selected configurations are numbered from 1 to 6 corresponding on the following activated blocks:

- 1: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8, 4x4
- 2: 16x16, 16x8, 8x16, 8x8, 8x4, 4x8
- 3: 16x16, 16x8, 8x16, 8x8, 8x4
- 4: 16x16, 16x8, 8x16, 8x8
- 5: 16x16, 16x8, 8x16
- 6: 16x16, 16x8

The results show that our proposed scheme can simplify the encoder complexity, maintaining good bit-save and quality performances for configurations 1, 2, 3, and 4, and very good performances for configurations 5 and 6. The main reason is because each element in the motion image (Figure 3) represents a 16x16 block of the original image. This means that we have a lack of precision in motion detection when handling small macroblocks.

The proposed algorithm is strongly influenced by the *detection value* threshold, as the estimation of detected movements depends on the *detection value* (using a very high value not all of the motions can be detected, otherwise a very low value can cause the detection of background noise as relevant motion). In the following tests we have set a value of 15 for the *detection value,* and 32 for the *block-motion threshold* (corresponding to 50% of a 8x8 block); these values are based on previous studies reported in [9].

Independently from the configuration type the proposed algorithm obtains a consistent reduction in encoding time (for *akiyo* and *hall* the whole encoding and MD is 3 times faster than the reference software). Table 1 and Table 2 show the above-described results for *akiyo* and *hall.*

As explained in previous sections, the proposed algorithm is suitable for applications as video-surveillance and video-telephony, where it can obtain the best performance improvements. Anyway we have tested our algorithm on more complex sequences as foreman. The test is very important because we achieve good performance using configurations 5 and 6, in particular using configuration 5 we obtain a better quality and compression rate than the standard (see Figure 4 and Figure 6). For this sequence the compression time is about 15-20% better than the standard (see Figure 5). This kind of sequences evidences the divergence from both the standard quality and compression rate when using small size blocks (configurations 1, 2, and 3) as can be noted in Figure 4 and Figure 6.

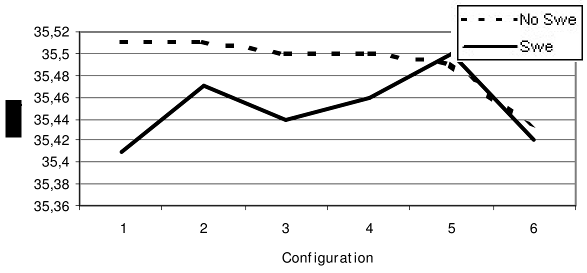

Figure 4 - Quality (PNSRY)

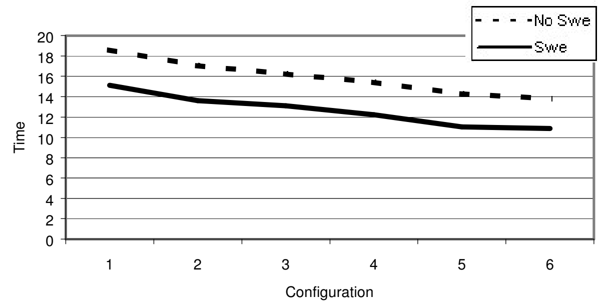

Figure 5 - Compression Time (sec)

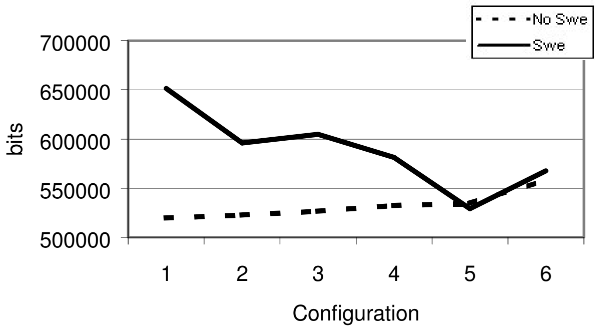

Figure 6 - Compression Rate (bits)

Conf	*Rate (bits)*		*PSNR Y*		*Time (sec)*	
	SWE	No SWE	SWE	No SWE	SWE	No SWE
1	211200	209648	38,28	38,28	6,561	18,171
2	215704	209424	38,27	38,28	6,001	16,67
3	211064	209768	38,27	38,27	5,518	15,922
4	211000	210632	38,27	38,27	5,141	15,189
5	211584	211568	38,26	38,26	4,33	14,11
6	219136	217912	38,22	38,23	3,861	13,578

Table 1 – Akiyo

Conf	*Rate (bits)*		*PSNR Y*		*Time (sec)*	
	SWE	No SWE	SWE	No SWE	SWE	No SWE
1	415144	346344	37,01	37,14	6,906	18,109
2	418232	346472	37,10	37,14	6,141	16,672
3	409480	354960	37,01	37,14	5,797	15,83
4	430032	355920	36,99	37,13	5,140	15,125
5	360120	359800	37,11	37,12	4,486	14,047
6	379208	374280	37,09	37,11	4,142	13,718

Table 2 – Hall

5. CONCLUSION

In this paper, the Search Window Estimation algorithm is presented and tested on a set of H.264 configurations. SWE is an innovative algorithm for ME complexity reduction based on motion detection. The proposed algorithm has been integrated in the JVT reference software jm60a [8] and validated using standard QCIF sequences. The SWE has been configured based on previous work [9]. Tests show significant results in video-surveillance and video-telephony standard sequences where the proposed approach allows performance improvement applying exhaustive motion estimation only in rarely situations. Future development will focus on H.264 image interpolation, varying the interpolation degree depending on the evaluated motion in particular regions by SWE.

REFERENCES

[1] ISO/IEC 14496-10, ITU-T Rec.H.264, Joint Video Specification, October 2002.
[2] ITU-T Recommendation H.263, “Video coding for low bitrate communication”, Feb. 1998
[3] S. Saponara, C. Blanch, K. Denolf, and J. Bormans, “The JVT Advanced Video Coding Standard: Complexity and Performance Analysis on a Tool-By-Tool Basis”, *Packet Video 2003*, Nantes, France, April 2003
[4] M. Karczewicz and R. Kurceren, “The SP- and SI-Frames Design for H.264/AVC”, *Ieee Transactions On Circuits And Systems For Video Technology*, Vol. 13, No. 7, July 2003
[5] P. Yin, H. C. Tourapis, A. M. Tourapis, and J. Boyce, “Fast Mode Decision And Motion Estimation For Jvt/H.264”, *International Conference on Image Processing - ICIP 2003*, Barcelona, Spain, September 2003
[6] R. Korada and S. Krishna, " Spatio-Temporal Correlation based Fast Motion Estimation Algorithm for MPEG-2," *35th IEEE Asilomar Conference on Signals, Systems and Computers*, California, November 2001
[7] D. H. Ballard and C. Brown, *Computer Vision*, New Jersey, Prentice Hall, pp 149 - 157, 1982
[8] JVT version jm6.0a, Bs.hhi.de/~suehring/tml/download/
[9] G. Bailo, M. Bariani, I. Barbieri, M. Raggio, “Dynamic Motion Estimation Search Window Size Calculation In H.264 Standard Video Coder”, *The Tenth International Conference on Distributed Multimedia Systems – DMS04*, San Francisco, USA, September 8-10, 2004