SEARCH WINDOW SIZE DECISION FOR MOTION ESTIMATION ALGORITHM IN H.264 VIDEO CODER

Gianluca Bailo, Massimo Bariani, Ivano Barbieri, Marco Raggio

Department of Biophysical and Electronic Engineering
University of Genova
Via Opera Pia 11 A,
16146 Genova,
ITALY

ABSTRACT

In this paper, we propose an innovative algorithm for reducing the complexity of the Motion Estimation (ME) module used in standard video compression applications. The idea is based on applying a Motion Detection algorithm for the decision of search window sizes in the Motion Estimation. The Motion Detection algorithm is based on blob coloring [1] over sub-sampled binary images generated by block wise threshold image difference. This solution decreases the encoder complexity thanks to the reduction of SAD (sum of absolute difference) calculations especially in low complexity sequences like video-surveillance and video-telephony ones. The selected video compression algorithm for validating the proposed approach is H.264 [2] that represent the state of the art of video codec.

1. INTRODUCTION

Today there are new high band request from actual multimedia service and efficient video codec are needed in order to obtain a sensible reduction of video streams high band occupation.

Recently, the Joint Video Team (JVT), formed by VCEG (Video Coding Experts Group) of ITU-T and MPEG (Moving Pictures Experts Group) of ISO/IEC, has developed a new hybrid video coding standard, formerly known as H.26L, and presently indicated as ITU-T recommendation H.264 or MPEG-4 part 10 standard [2]. Compared with previous standards, H.264 introduces many new features in all the aspects of the video encoding process. Firstly it can use 7 blocks configuration (see Figure 1). If all blocks are active, exhaustive motion estimation is performed and better quality and compression performance are achieved.

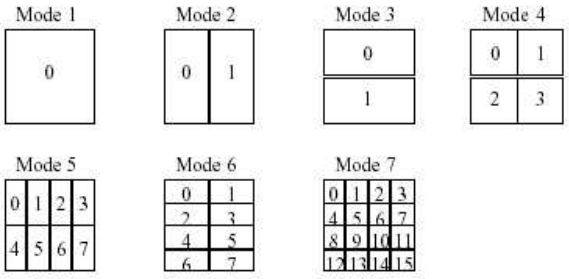

Figure 1 - Block configurations for ME

Another H.264 key feature is the multiple frame reference; in this case motion estimation is performed N time (where N is the number of reference frames) in order to improve quality and compression. Anyway the computational complexity is increased N times respect to the single motion estimation application.

H.264 achieves 50% bit-rate save at same quality compared with existing video coder standards as H.263 [3], 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) [4].

The high compression rate together with the good quality obtained by the H.264 standard make it suitable for a large variety of applications (video-surveillance, video telephony on mobile terminals). Moreover the introduction of new features like the new frame types called SP-frames and SI-frames [5] allow implementing bit-stream switching and "VCR functionalities" (random access, fast forward, fast backward, etc.) in a more efficient way [5]. Therefore, Video-Surveillance operations like sequence retrieval, switching between video stream coming from different cameras and navigation in video-streams can be better performed using H.264 algorithm.

However, typical Video-Surveillance applications have low power constrains (especially in the video encoder part) in order to work on embedded systems and mobile

0-7803-8554-3/04/$20.00 ©2004 IEEE.

terminals. This requirement implies the need to dramatically reduce the complexity of the H.264 video encoder.

Algorithm analysis shows that the ME and the mode-decision modules is the most complex in the H.264 encoder (especially when Rate-Distortion Optimization is used) [6].

This is mainly due to the great number of SAD calculation in ME in few frames (e. g. 1500 million in the for 50 frames of standard sequence foreman with 5 reference frame and all block configuration activated). There are many algorithms that perform a reduction of the number of SAD calculation based on spatial and temporal correlation of motion vector [7]

In the following, we will introduce a different approach, based on motion detection and blob coloring algorithms. 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. This approach is useful for each configuration feature of the H.264 codec.

2. PROPOSED ALGORITHM

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 only when the motion detection algorithm identifies motion and, specifically in the region where the motion is detected. In particular, motion detection evaluates a parameter that is usually a constant, the *search window size*. Motion detection evaluates where and when to set larger window for motion estimation, otherwise minimum-size window will be used.

The blocks diagram in Figure 2 explains where the motion detection module works in window search estimation.

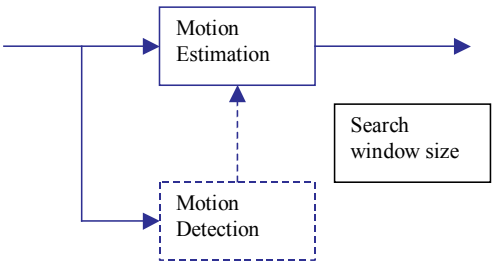

Figure 2 - Algorithm position in H.264

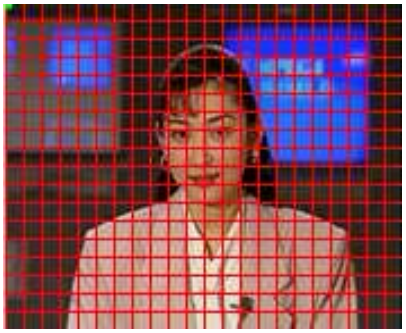

Figure 3 - 8x8 blocks

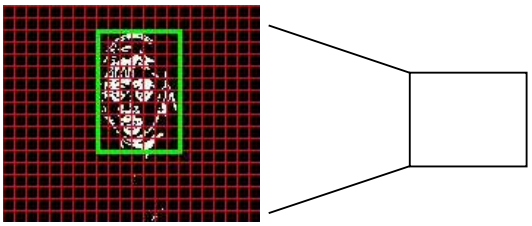

Figure 4 - block sub-sampling

This algorithm consist of several steps:

Step 1: images are divided into 8x8 blocks (Figure 3)

Step 2: image difference with threshold is calculated. The difference is evaluated between the current frame and a reference frame (the latest intra-frame). The result is a binary image of black and white pixel (a threshold is applied).

Step 3: in each block the sum of the white pixels is evaluated. If the resulting sum value is greater than a fixed threshold the status of the related block becomes activated, else it is not activated.

Step 4: now we have a set of activated and not activated blocks. Each block becomes a pixel of a reduced-size binary image (see Figure 4).

Step 5: blob coloring [1] is performed over this sub-sampled binary image obtained from step 1, 2, 3, and 4. The result is an array of vectors, each one composed by two coordinates identifying motion object boundary (bold rectangles in Figure 5).

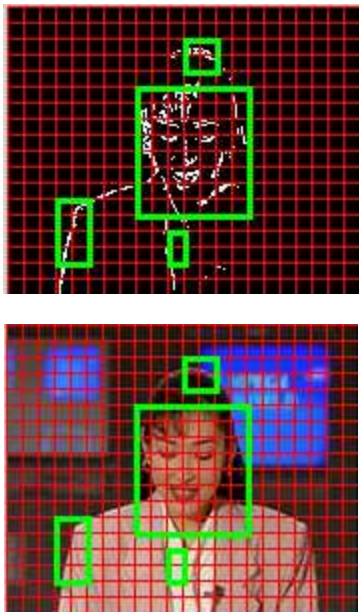

Figure 5 - object boundary in bold

Step 6: with information obtained from the blob-coloring application step, we can define an approximate motion image (Figure 6). This is an approximation of blob coloring output result, because we need as little as possible complexity in motion detection algorithm.

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 0 0 0 1 1 1 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0
0 0 0 0 1 1 1 0 0 0 0 0
0 0 1 1 1 1 1 1 0 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 0 0

Figure 6 - Motion Images Example

Step 7: during video coder motion search, 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.

3. RESULTS

Our approach was integrated within version jm60a [8] of the reference JVT software. The presented tests have been performed using five standard sequences in CIF and QCIF format. *Akiyo*, *Silent*, *News*, and *Wall* 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 different kind of sequence, containing more motion. In the following tests, we encoded the sequences at 30fps. The CAVLC entropy coder and Hadamard were used for all tests, with quantization values of 28, 32, 36, and 40. The H.264 activated block configurations are 8x8 and 16x16. The complete results are shown in the following tables (from Table 1 to Table 8). The performance is measured in terms of time employed to compress the test sequences and the compression rate in terms of compressed sequence size. These values are compared to the reference software and the difference in percentage is shown in the below tables. The quality is evaluated using the SNR Y, measured by the jm6.0a test model [8] (in the below tables differences are shown in decibel).

From these results we observe that our proposed scheme can simplify the encoder complexity. The proposed approach achieves from 50% to 60% encoding time reduction in typical video-surveillance and video-telephony sequences. Anyway, also the worst-case *foreman* shows a good computational time reduction (about 25%). It should be noted that the H.264 coding efficiency is only slightly decreased.

CIF 28	Time %	Total bit %	SNR Y (db)
Hall	-67.02	+0.03	-0.01
Silent	-60.16	+0.03	-0.01
News	-59.52	+0.05	-0.02
Foreman	-26.18	+0.05	-0.02
Akiyo	-65.40	0.00	0.00

Table 1 – Cif: Quantization 28

CIF 32	Time %	Total bit %	SNR Y (db)
Hall	-67.58	+0.06	-0.02
Silent	-60.39	+0.03	-0.01
News	-60.14	+0.06	-0.02
Foreman	-26.98	+0.09	-0.03
Akiyo	-65.74	0.00	0.00

Table 2 - Cif: Quantization 32

CIF 36	Time %	Total bit %	SNR Y (db)
Hall	-67.69	+0.06	-0.02
Silent	-60.18	+0.03	-0.01
News	-59.92	+0.09	-0.03
Foreman	-25.95	+0.12	-0.04
Akiyo	-66.03	+0.03	-0.03

Table 3 - Cif: Quantization 36

CIF 40	Time %	Total bit %	SNR Y (db)
Hall	-68.05	+0.03	-0.01
Silent	-60.02	+0.03	-0.01
News	-60.35	+0.10	-0.03
Foreman	-26.23	+0.20	-0.06
Akiyo	-66.10	+0.06	-0.06

Table 4 - Cif: Quantization 40

QCIF 28	Time %	Total bit %	SNR Y (db)
Hall	-62.29	0.00	0.00
Silent	-59.58	+0.03	-0.01
News	-60.02	0.00	0.00
Foreman	-41.57	+0.06	-0.02
Akiyo	-62.95	0.00	0.00

Table 5 - Qcif: Quantization 28

QCIF 32	Time %	Total bit %	SNR Y (db)
Hall	-62.59	0.00	0.00
Silent	-59.86	+0.06	-0.02
News	-60.04	+0.06	-0.02
Foreman	-42.14	+0.15	-0.02
Akiyo	-63.48	0.00	0.00

Table 6 - Qcif: Quantization 32

QCIF 36	Time %	Total bit %	SNR Y (db)
Hall	-62.63	0.00	0.00
Silent	-59.58	+0.03	-0.01
News	-60.63	+0.03	-0.01
Foreman	-42.10	+0.16	-0.05
Akiyo	-63.57	0.00	0.00

Table 7 - Qcif: Quantization 36

QCIF 40	Time %	Total bit %	SNR Y (db)
Hall	-62.91	-0.03	-0.01
Silent	-59.92	-0.04	-0.01
News	-60.94	-0.04	-0.01
Foreman	-42.27	-0.57	-0.16
Akiyo	-63.78	0.00	0.00

Table 8 - Qcif: Quantization 36

4. CONCLUSION

An innovative algorithm for ME complexity reduction is presented. The proposed algorithm has been validated on the H.264 encoding of standard CIF and QCIF sequences. Best performances are achieved in particular situation like video-surveillance and video-telephony. In this situations performance are remarkable during long time running applications where exhaustive motion estimation is made only in rarely situation (video surveillance case).

The motion detection algorithm has a very low complexity due to sub-sampled image blob coloring application and it can work in real time. Moreover the motion detection algorithm is present in most of the video-surveillance applications; therefore our approach does not have any overhead in this kind of applications.

As a future work, we will apply the described algorithm to H.264 features scalability (blocks scalability).

5. REFERENCES

[1] D. H. Ballard and C. Brown, *Computer Visio*n, Prentice Hall, New Jersey, 1982.

[2] ISO/IEC 14496-10, ITU-T Rec.H.264, Joint Video Specification, October 2002.

[3] ITU-T Recommendation H.263, "Video coding for low bitrate communication", Feb. 1998

[4] 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

[5] 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

[6] 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

[7] 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

[8] JVT Reference Software version jm 6.0a, Bs.hhi.de/~suehring/tml/download/