ADAPTIVE INTERPOLATION ALGORITHM FOR FAST AND EFFICIENT VIDEO ENCODING IN H.264

Gianluca Bailo, Massimo Bariani, Andrea Chiappori, Riccardo Stagnaro

Department of Biophysical and Electronic Engineering, University of Genova
Via Opera Pia 11 A, 16146 Genova, ITALY
phone: +390103532037, fax: +390103532036, emails: {bailo,bariani,chiappori,stagnaro}@dibe.unige.it

ABSTRACT

H.264/MPEG-4 AVC is the latest video-coding standard jointly developed by VCEG (Video Coding Experts Group of ITU-T and MPEG (Moving Picture Experts Group) of ISO/IEC. It uses state of the art video signals algorithms providing enhanced efficiency if compared with previous standards.

A ¼ pel displacement resolution is used in order to reduce the bitrate of the video signal in H.264.

In this paper, we propose an algorithm for the reduction of the interpolation computational time. The goal is to adapt the H.264 ¼ pel interpolation to the complexity of the video stream to encode, on the basis of our motion detection algorithm.

The proposed solution allows decreasing the overall encoder complexity both in low and high complex sequences. This paper illustrates the integration of our motion detection algorithm in the H.264 encoder. The obtained results are compared with the jm86 standard interpolation using different quantization values.

1. INTRODUCTION

The video coding standard H.264 [1] introduces many new features in all the aspects of the video encoding process. The compression efficiency has been highly improved maintaining the same video quality. Anyway, 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], if compared with previous standards such as H.263 [2]. The high compression rate together with the good quality obtained by the H.264 standard make it suitable for a large variety of applications. The H.264 encoding application area requires high power efficiency in order to work on embedded systems and mobile terminals. This requirement implies the need to reduce the complexity of the H.264 video encoder. The most time-consuming modules of the encoder are the motion estimation (ME) and the ¼ pel interpolation. The complexity of the ME is due to the great number of SAD operations; while the major part of interpolation calculations are due to the use of a 6-tap Wiener interpolation filter (with coefficients similar to the proposal of Werner [9]) and the averaging samples at full and half sample position.

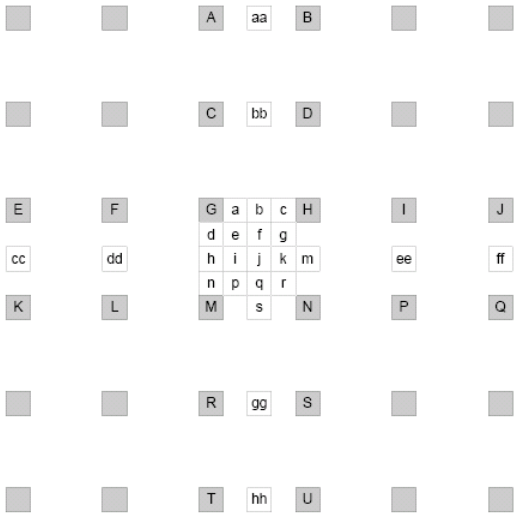


Figure 1 - Integer samples (shaded box with upper-case letters) and fractional samples (un-shaded box with lower-case letters) for ¼ pixel Interpolation [1].

The interpolation process is depicted in figure 1 and it can be divided into two steps. First, the horizontal or vertical Wiener filter is applied to calculate the half-pel positions, then, a bilinear filtering is applied to both the already calculated half-pel positions and the existing full-pel samples in order to compute residual quarter-pel positions.

This sequence of calculation is very heavy, and it represents the 18% of entire compression process. In the following, we will introduce the Local Dynamic Interpolation (LDI) approach tending to minimize calculation, based on our Motion Detection (MD) algorithm. The main idea is to utilize the complete ¼ pixel bilinear interpolation just in those areas where the MD module indicates the presence of motion. Otherwise, only the first part of the full interpolation (the Wiener filtering), together with a very simple ¼ interpolation, will be used.

The reference JVT software version is jm86 [6] and comparative tests are performed with standard fast motion estimation built in. The quality/compression ratio results very close to the standard full interpolation. Section 2 shows how our MD works. Section 3 shows how MD interacts

with the H.264 interpolation module. LDI algorithm results are also compared with standard one, as shows in Section 4.

2. MOTION DETECTION ALGORITHM

The new approach to MD algorithm can be subdivided into the following steps:

- Binary image difference evaluation
- Blob coloring with minimum object size threshold of entire difference image, with single pixel precision
- Merging overlapped blobs

Image difference is performed calculating the difference between two subsequent frames (unlike the work presented in [7] this approach avoid background estimation). The input of MD is the luminance image and the output is a set of information about motion area, like position and size.

2.1 Binary image difference evaluation

The inputs of this block are the current image and the background; the output is a binary image with motion pixels. The image difference D(n) is calculated by the difference between the current image I(n) and previously stored image I(n-1). Then a threshold value (*detection value*) is applied to D(n) in order to obtain Dth(n), the binary image that represents motion pixels into the image.
The formula for frame *n* is:

$$D(n) = I(n) - I(n-1)$$

Dth(*n*)=0 if D(*n*) < detection value.
Dth(*n*)=1 if D(n) >= detection value.

2.2 Blob coloring with minimum blob size threshold

Blob coloring is a computer vision technique to obtain region growing and region separation for images.
The input of this block is the difference image and the output is a set of moving object. We use this technique on the entire difference image, instead of using a subsampled 8x8 difference image as in [7], to separate motion regions and growing it. Blob coloring can be performed with the algorithm described in [5]. The regions coordinates are the output of this step. After the blob coloring, the resulting object set is filtered through a threshold that permits to delete the smallest moving regions, in order to avoid the errors due to image noise.

2.3 Merging overlapped blobs

At this point, there are a lot of blobs that may be overlapped. If we would obtain region growing, there is the need to check for overlapped regions and then fuse them together into a single region. The algorithm analyzes region coordinates for finding overlapping in the following way: if there are two or more overlapped regions, a new region having size growing to max dimension of overlapped blobs is created; otherwise the regions are maintained as original sets. This operation reduces object count and permit moving region separation. The input is a set of moving objects and the output is a reduced set of the same objects having enlarged size.

3. LOCAL DYNAMIC INTERPOLATION

The proposed algorithm is based on the idea that exhaustive interpolation is useful only when a video sequence contains large motions and not in low motion conditions, that is the usual situation for real video-surveillance sequences. The H.264 interpolation can be applied just when the motion detection algorithm identifies motion and, specifically, in the region where the motion is detected. In particular, the 6-tap Wiener filtering is always calculated, whereas the residual quarter-pel position, obtained by applying the bilinear filtering to half-pel and full-pel positions, is only computed in the motion regions (blobs) identified by the MD module. The motion detection evaluates where and when to apply the bilinear-filtering; otherwise a very simple interpolation is carried out (figure 2 shows LDI in the encoder scheme).
The information obtained by the MD algorithm, is utilized to define a Motion Image (MI). This is an approximation of the MD output result because we need as little as possible

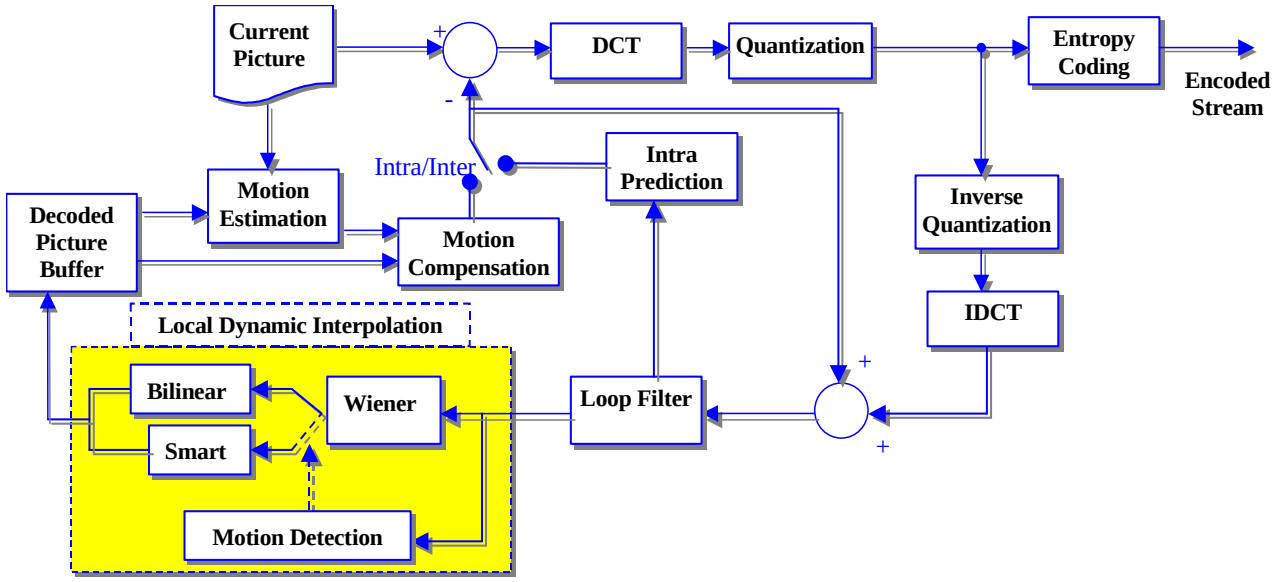


Figure 2 - the MD module controls the output of Wiener filter for choosing between smart or bilinear filtering, (input is the original size image, output is the ¼ pel interpolated image)

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

Figure 3 - MD interface; the motion image scanned for searching motion areas.

complexity in the motion detection algorithm. This virtual image is the interface between motion detection and H.264 interpolation.

During the video coding, the MI (Figure 3) is scanned in order to decide the area where bilinear interpolation is applied. Alternatively, a *smart* interpolation will be applied. The *smart* interpolation uses the pixels evaluated by Wiener filtering, replicating them to the near blank positions without other calculation.

4. RESULTS

The proposed algorithm has been validated on the reference JVT software version jm86 [6] implementing the H.264 video encoder, which include a fast ME (we used this option). The reported tests have been performed using standard sequences in QCIF format. *Hall, Salesman, Silent, Highway, and News* have been utilized in order to test the proposed algorithm with both simple and complex sequences (*Highway)*. In the following tests, we encoded 200 frames of every test sequence at 30fps. The selected block configurations are 16x16, 16x8, and 8x16; the CAVLC entropy coder is used for all the tests; Hadamard transform is not used. The RD optimization option in the reference encoder is turned off for all the experiments. The reported tests show the algorithm performance for different quantization values (Q inter) selected from 8 to 28. The results show that our approach can simplify the encoder complexity, maintaining high compression rate and good video quality. The proposed algorithm is strongly influenced by the *detection value* threshold: 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 the *detection value* to 15; this value is based on previous studies reported in [7] on the Search Window Estimation (SWE). For every sequence, two tests are performed in order to compare the following implementations: the jm86 Standard Fast with original interpolation algorithm (STDI), and the jm86 Standard Fast with LDI algorithm (LDI) [6]. Tests are performed on standard pc class workstation Pentium 4 3.0 Ghz equipped with 1024 MB main memory.

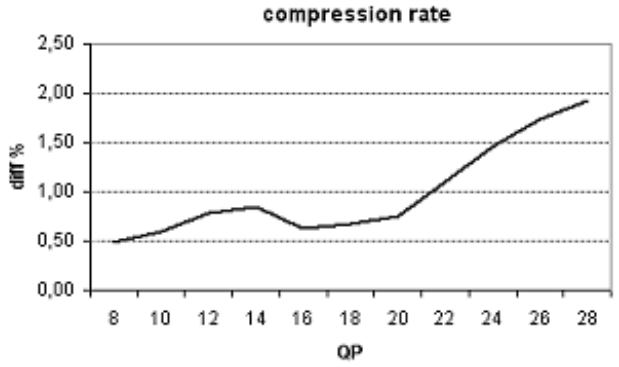
Table 1 shows the differences between the standard algorithm and the proposed approach for a given quantization value. The video quality is compared using the *PSNR*. The *compression* and the *interpolation time* entries represent the percentage increase (positive) or decrease (negative) of the LDI approach.

The reported interpolation time is the computational time of the interpolation module (summed with the MD module time in case of LDI test).

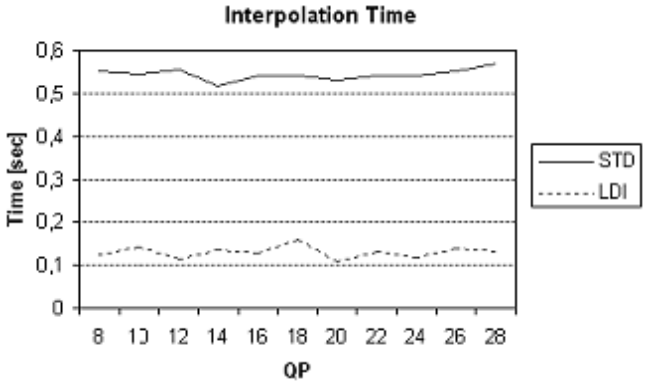
The proposed algorithm (LDI) obtains a consistent reduction in interpolation time for all the sequences: LDI is about 3 times faster than the reference software STDI. Noticeable fact is that there is little loss of bitrate/compression values. In particular graph 1 and graph 2 focus on the behavior of the *Hall* sequence (video surveillance – low complexity case). The difference in

Q Inter 20	*Psnry (dB)*	*Compression (%)*	*Interpolation Time (%)*
Hall	-0.01	-0.740	-76.4
Salesman	-0.03	-4.900	-72.8
Silent	-0.05	-5.217	-70.6
Highway	-0.09	-4.272	-74.0
News	-0.05	-5.174	-77.3

Table 1 - Comparative test for quantization value of 20



Graph 1 - Differential percentage bit rate for hall sequence



Graph 2 - Differential percentage time for hall sequence

bitrate and interpolation time between STDI and LDI is shown for several quantization values (8-28). Graph 1 shows the decrement in compression efficiency of the LDI, anyway the differences ranges from 0.5% to 2%, increasing with inter quantization (QP). The speed-up is the same in all the selected quantization range (see graph 2).

5. CONCLUSION

This paper presents an adaptive interpolation module for the video coding standard H.264. The proposed solution allows decreasing the overall encoder complexity both in low and high complex sequences, having small side effects either in image quality or in compression efficiency.

The integration of our Motion Detection module in the H.264 encoder is described and tested using the JVT reference software jm86 [6]. The reduction of the interpolation computational time has been measured using standard QCIF sequences.

The main idea of the Local Dynamic Interpolation (LDI) algorithm is to reduce the interpolation complexity using a partial bilinear interpolation in motion areas, and a simple interpolation in background areas.

Future developments will focus on a further reduction of the H.264 overall complexity by using the MD to drive the LDI together with the SWE. Since both the algorithms use the same MD module's output, the system efficiency will be even improved.

6. REFERENCES

[1] ISO/IEC 14496-10, ITU-T Rec.H.264, Joint Video Specification, Oct. 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] Y.L. Lai, Y.Y. Tseng, C.W. Lin, Z. Zhou, and M.T. Sun, "H.264 Encoder Speed-Up via Joint Algorithm/Code-Level Optimization," VCIP 2005, Visual Communications and Image Processing, Beijing, China, July 12-15, 2005.

[5] D. H. Ballard and C. Brown, Computer Vision, New Jersey, Prentice Hall, pp 149 - 157, 1982

[6] JVT version jm86, Bs.hhi.de/~suehring/tml/download/

[7] 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, Sept. 8-10, 2004

[8] Gianluca Bailo, Massimo Bariani, Ivano Barbieri, Marco Raggio. "Search window estimation Algorithm for Fast and Efficient H.264 Video Coding With Variable Size Block Configuration" - EUSIPCO2005 (Eurasip) Antalya, Turkey, Sept. 4-8, 2005.

[9] O. Werner, "Drift analysis and drift reduction for multiresolution hybrid video coding", Signal Processing: Image Commun., vol. 8, no. 5, July 1996.