MACROBLOCK SKIPPING ALGORITHMS FOR HIGH DEFINITION H.264/AVC VIDEO CODING IN THE BASELINE PROFILE

Susanna Spinsante, Ennio Gambi and Damiano Falcone
Dipartimento di Elettronica, Intelligenza artificiale e Telecomunicazioni
Universitá Politecnica delle Marche, via Brecce Bianche 12, I-60131 Ancona, Italy

Keywords: High Definition, macroblock skipping, complexity reduction.

Abstract: This paper discusses different macroblock skipping algorithms to be applied in the H.264/AVC Baseline profile, in order to facilitate the adoption of High Definition video coding in real time applications. Moving from Standard to High Definition video coding, there is six times as much data to process: this motivates the search for suited Mode Decision strategies, to reduce complexity while preserving an acceptable video quality for the final user. The proposed schemes permit to speed up significantly the Mode Decision procedure, by forcing the selection of the SKIP mode over each frame, without affecting significantly the final quality.

1 INTRODUCTION

In the framework of the H.264/AVC video coding (ITU-T, 2005), a huge amount of fast Mode Decision strategies have been proposed (see, for example, (Richardson, 2006)), with the aim of optimizing the process in terms of computational resources and time, without affecting the final perceived video quality. Macroblock (MB) skipping algorithms usually play a prominent role in fast Mode Decision schemes, as they permit to increment the number of MBs encoded as SKIP, that require the lower amount of computational resources. However, most of these proposals for MB skipping and fast Mode Decision address the Standard Definition (SD) video formats, whereas not so much work has been carried out in the context of High Definition (HD) video coding.

The H.264/AVC standard foresees the so-called Fidelity Range Extension (FREXT) profile to support specific applications, like content distribution, studio editing and post processing. FREXT has also been addressed as the most proper profile for the management of HD coding, but, given its extremely high number of available options and its complexity, it cannot be used in constrained environments, like those related to the provisioning of real-time services (*e.g.* videoconferencing). On the contrary, the aim of this paper is to provide algorithms that can help introducing HD video coding in constrained environments, by selecting the H.264/AVC Baseline profile. As a matter of fact, the Baseline profile seems the only one able to meet requirements on the time delay, even if not specifically conceived for supporting HD coding. Based on such premises, this paper begins showing that the Baseline profile can be actually used for HD coding, instead of the FREXT profile, without a significant quality degradation, but with savings in encoding time. Then, several MB skipping strategies are presented and tested, that could be included in the Baseline profile to permit further reduction of the processing time, thus making real time HD coding more feasible, without significantly affecting the final delivered video quality. HD formats provide more visual information than any Standard Definition (SD) format; the increased pixel count inherent in the HD formats supports better picture quality and makes viewing images on larger screens clearer and easier to watch. In videoconferencing, this enhances the overall viewing experience and eliminates meeting fatigue. Colors are more vibrant and realistic, and movements are sharp and smooth. The higher amount of available information, with respect to SD formats, makes it reasonable to test the applicability of skipping strategies, to reduce the computational burden of the encoder, without affecting the perceived final quality.

Spinsante S., Gambi E. and Falcone D. (2007).
MACROBLOCK SKIPPING ALGORITHMS FOR HIGH DEFINITION H.264/AVC VIDEO CODING IN THE BASELINE PROFILE.
In *Proceedings of the Second International Conference on Signal Processing and Multimedia Applications*, pages 59-66
DOI: 10.5220/0002135600590066

The paper is organized as follows: Section 2 presents the results of a performance comparison between Baseline and FREXT profiles, when applied to encode different HD video sequences. Section 3 provides an overview of Mode Decision and SKIP in H.264/AVC, and discusses a number of macroblock skipping algorithms, with the aim of drawing some basic ideas, and tailoring them for use with HD sequences. Section 4 presents the performances obtained by testing the new solutions within the Baseline profile of the Reference Software version JM11.0; finally, the last section provides some remarks that conclude the paper.

2 APPLICABILITY OF THE BASELINE PROFILE FOR HD CODING

Before HD encoding/decoding, several video communication services, like videoconferencing, adopted the Common Interchange Format (CIF), and other formats such as 4CIF and 16CIF, with "full resolution" being defined as 16CIF. Now, for HD videoconferencing, the ITU-T recommends the formats listed in Table 1.

Table 1: ITU-T recommended HD formats.

Format	frames per second	Resolution (16:9)
1080p	24,30	1290x1080
720p	24,30,60	1280x720

The progressive scan format (denoted by suffix p) is an alternative to interlacing that improves picture quality on larger screens, reduces jagged pictures, and smoothes movement on large monitors.

In this section, a comparison between the Baseline and the FREXT profiles, applied to encode a series of High Definition video sequences, is presented. The two profiles are described in detail in the standard and the related documents; only the features of interest are consequently reminded in the following. Comparison is based on the evaluation of the computational time required to encode the HD sequences, and the resulting Peak Signal to Noise Ratio (PSNR) of the Luma (Y) component. As the computational time depends on the test platform adopted, it has been expressed in relative units (r.u.). The selected encoder configuration is as follows:

- Total number of frames: 100
- Frame rate: 30 fps
- Hadamard transform: Not Used
- Image format: 1280x720p
- Search range: 16
- Number of reference frames:1, P-slice reference:1
- Quantization Parameter: 28
- Entropy coding: CAVLC (Baseline), CABAC (FREXT)
- RD Optimization: Enabled
- Number of B frames: 0 (Baseline), 49 (FREXT)

A subset of the results obtained over a number of different video sequences is reported in Table 2, for the Shields, Car, and Park Run HD sequences, selected as representatives for their motion features. As shown, for each sequence, the resulting PSNR values obtained with the two profiles differ by less than 0.3 dB, but the total encoding time required by FREXT is more than 1.6 times that required by Baseline. This confirms the idea that it is possible to adopt Baseline also for HD coding, when dealing with real time services, because the final quality remains acceptable, whereas a significant reduction of the coding time is achievable. The same conclusions can be drawn by performing a subjective (*i.e.* visual) comparison among the sequences encoded with the two profiles.

Table 2: Quality and computational time comparison between Baseline and FREXT profiles, for different HD video sequences.

	PSNR Y (dB)		Time (r.u.)	
Sequence	Baseline	FREXT	Baseline	FREXT
Shields	35.02	35.03	1.07	1.68
Car	37.26	37.48	1	1.60
Park Run	32.84	32.92	1.04	1.88

The above comparison shows that the adoption of the Baseline profile for HD coding can be a viable solution for introducing HD in real time services. However, the problem of making the coding process more efficient, in order to meet strict time constraints, still holds, as the total time required by the Baseline profile to encode a HD sequence remains, typically, almost three times larger than that required for encoding its corresponding CIF version.

Among the actions that can be performed to make a video encoder work faster, a prominent role can be played by the adoption of MB skipping algorithms that can force the selection of the SKIP mode during the Mode Decision process. The peculiarities of the HD format (higher resolution and redundancy), and the limitation of the human visual perceptivity, can be taken into account to design *ad hoc* fast strategies.

3 MACROBLOCK SKIPPING ALGORITHMS FOR FAST MODE DECISION

One of the key functions in contributing to the computational time required by a video encoder is the Mode Decision process. H.264 Mode Decision (Ivanov, 2006) utilizes a Lagrangian rate-distortion cost function J when computing mode costs for available MB segmentation options:

$$min\{J(s,c,MODE|QP,\lambda_{MODE})\} \quad (1)$$

$$J(s,c,MODE|QP,\lambda_{MODE}) = SSD(s,c,MODE|QP) + \lambda_{MODE} \cdot R(s,c,MODE|QP) \quad (2)$$

where QP is the MB quantization parameter, λ_{MODE} is the Lagrangian multiplier, s and c represent the original and the reconstructed MBs, respectively. $MODE$ is the mode chosen from the set of potential modes (SKIP, 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4, Intra 4×4, Intra 16×16). $R(s,c,MODE|QP)$ is the transmitted bit rate associated with $MODE$ and QP, SSD is the Sum of the Squared Differences. The Mode Decision algorithm implemented in the Reference JM encoder sequentially calculates J for all segmentation options. The best coding mode, *i.e.* the one found only by an exhaustive evaluation of all the available modes, is that giving the minimum rate-distortion (RD) cost, which can be defined according with a Sum of Absolute Differences (SAD) or SSD criterion.

In order to make a SKIP decision, after computing all the modes, JM performs the following additional checks:

- the best mode selected is Inter 16×16
- the reference frame is the previous one
- the Motion Vector (MV) found is equal to the predicted MV
- all the quantized transform coefficients are equal to zero

The exhaustive search, necessary to detect the best coding mode, can be a big burden for the encoder, especially when it has to deal with HD sequences and their frame formats. On the other hand, experimental tests show that, typically, more than half the MBs in each frame of a sequence could be encoded as SKIP, without any relevant loss in the final quality. Even if not motivated by an analytical approach, this empirical consideration is commonly accepted as true, and adopted as a guideline in the design of fast Mode Decision techniques. Taking into account the fact that the SKIP mode is also the one having the lowest computational complexity, as it does not encode any motion or residual information, it seems reasonable to optimize the SKIP mode selection during the Mode Decision process, to get a significant reduction in the overall encoding time. While in the case of CIF and QCIF sequences this could lead to an unacceptable impact on quality, the larger amount of information available in HD formats should compensate the risk of quality loss. These considerations inspire the MB skipping algorithms proposed in the paper: they rely on some basic ideas, borrowed from the technical literature, that are briefly exposed in the following.

3.1 Macroblock Skipping Strategy based on Skip Mode Distortion Estimation

The algorithm presented in (Richardson, 2004) aims at predicting MBs that are likely to be skipped by the encoder. MBs that can be skipped with little or no increase in distortion are not coded, resulting in substantial computational savings, without significantly affecting RD performance. Correct prediction of skipped MBs can save significant computational resources, since all of the subsequent processing of the MB (Motion Estimation, transform and quantization, entropy coding) can be avoided. For each MB, the algorithm estimates the increase in distortion at the decoder, if the MB were skipped, compared to the distortion if the MB were coded. The distortion is approximated by an energy measure, the Sum of Absolute Errors (SAE), which is typically computed by a video encoder, so it does not require further operations. Provided that:

$$SAE_{MB} = \sum_{i,j} |a_{i,j} - b_{i,j}| \quad (3)$$

being $a_{i,j}$ an original Luma sample value, $b_{i,j}$ the corresponding decoded Luma sample value, and i, j ranging from 0 to 15, SAE increases with the decoded image distortion. Introducing SAE_{diff}, the difference in SAE between a skipped MB and a coded MB, as an estimate of the increase in distortion due to skipping a MB, we can say that a low value of SAE_{diff} implies little advantage in coding and transmitting the MB, whereas a high value of SAE_{diff} implies a significant benefit in coding the MB, compared with skipping it:

$$SAE_{diff} = SAE_{SKIP} - SAE_{noSKIP} \quad (4)$$

SAE_{SKIP} is the sum of absolute errors between the uncoded MB and the luma values in the same position, in the reference frame. It is an approximate measure of the distortion if the MB is skipped, since

the decoder copies the luma samples from the reference frame. SAE_{noSKIP} is the SAE of the decoded MB (compared with the original, uncoded MB) and is an approximate measure of distortion, if the MB is not skipped. SAE_{SKIP} is calculated as the first step of a Motion Estimation algorithm in the encoder and is readily available at an early stage of processing of each MB. SAE_{noSKIP} is not normally calculated during coding or decoding, and cannot be calculated if the MB is actually skipped. Therefore, a model for SAE_{noSKIP} is used, to estimate SAE_{diff}. More specifically, given a MB at position i in frame n, the value of SAE_{noSKIP} is set equal to the SAE of the most recent available decoded MB in position i. Experimental results have shown this is a good predictor for SAE_{noSKIP}; the encoder has to compute and store this value for each coded MB. Older values of SAE_{noSKIP} for position i are replaced when a new MB is coded.

The MB skipping model compares SAE_{diff} to a threshold: when lower, the current MB is skipped, otherwise it is encoded. The threshold controls the proportion of skipped MBs: a higher threshold results in an increased number of skipped MBs, but also an increased distortion, due to incorrectly skipped MBs.

3.2 Active Macroblock Skipping based on Flexible Macroblock Order

The MB skipping strategy described in (Beesley, 2005) exploits one of the error resilience tools available in H.264/AVC, the Flexible Macroblock Order (FMO), that allows a picture to be partitioned into one or more slices, so that every MB in a missing slice is likely to have all of its neighbours, from a correctly decoded slice, available for concealment purposes at the decoder. According with the proposed method, MBs selected for skipping are deliberately removed during the encoding stage, in the knowledge that they can be effectively concealed during the decoding phase. This provides a significant reduction in bit stream size, with a resulting effect on quality that, obviosuly, depends on the concealment options activated at the decoder.

The proposed algorithm computes individual MB PSNR values, along with the PSNR values should each MB be concealed using weighted pixel value averaging, *i.e.* one of the several concealment strategies available. The resulting PSNR values are then compared: if the concealed MB has an improved PSNR over its decoded equivalent, then the MB is marked for possible removal from the encoded bit stream. Actually, the MB is really removed only in the case all its neighbouring MBs are not removed, because they are necessary to perform concealment at the decoder. The selection of the removable MBs is performed by means of a list of all possible candidates, ordered by their bit stream size, so that the MBs with the largest potential savings are given the highest priority.

3.3 A Fast Algorithm for Inter-mode Selection

In (Yu, 2004), the authors present an improved version of the so-called Modified Fast Inter-mode selection (MFInterms) algorithm, to provide a more efficient prediction of Mode Decision. The strategy includes temporal similarity detection, and the detection of different moving features within a MB.

The basic idea exploited by the suggested algorithm is that a mode having a smaller partition size may benefit detailed areas, whereas a larger partition size is more suitable for homogeneous areas. Generally, a homogeneous MB is more likely to require examination of fewer inter-modes, compared to a highly detailed MB. Two measurements are included in the algorithm, targeted at MBs encoded with SKIP mode, and MBs encoded by the inter-modes with larger decomposed partition size (greater than 8×8 pixels): the temporal similarity between two MBs, and the motion consistency of a MB. Macroblocks coded with the SKIP mode can be easily detected by comparing the residue between the current MB and the previously encoded MB with a threshold, as follows:

$$S_{residue} = \sum_m \sum_n |\mathbf{B}_{m,n,t} - \mathbf{B}_{m,n,t-1}| \qquad (5)$$

$$T(S_{residue}) = \begin{cases} 1, & S_{residue} < Th_{ASV} \\ 0, & S_{residue} > Th_{ASV} \end{cases} \qquad (6)$$

where $S_{residue}$ is the sum absolute difference between $\mathbf{B}_{m,n,t}$ and $\mathbf{B}_{m,n,t-1}$, which represent current and previous MBs, respectively. The temporal similarity is implemented by means of an adaptive spatially varying threshold, Th_{ASV}, which depends on a constant and the sum absolute difference of four nearest encoded neighbours. This is motivated by the fact that, generally, the skipped MBs tend to occur in clusters, like in a part of a static background; a MB undergoes temporal similarity detection if one of the encoded neighbours is a skipped MB.

Besides the temporal similarity computation, the proposed algorithm suggests checking the motion vector of each 8×8 block decomposed from a highly detailed MB. If consistency among motion vectors exists, the inter modes with partition size greater than 8×8 are checked, otherwise, all possible inter modes are searched.

3.4 Macroblock Skipping Algorithms for HD Coding

The different approaches for MB skipping, previously reviewed, have been applied to HD sequences, with the aim of improving the efficiency of the Mode Decision process, and reducing the required coding time in the Baseline profile. In order to make the following discussion clearer, the algorithm discussed in 3.1 will be referred to as Alg_1, the algorithm discussed in 3.2 will be referred to as Alg_2, finally the algorithm discussed in 3.3 will be referred to as Alg_3.

The implementation of Alg_1, in the case of HD coding, is based on a *SAE* computation performed on the Luma samples only. The SAE_{noSKIP} value is determined by subtracting the Luma samples of the last non skipped MB, in a given position, from the Luma samples of the same MB in the original frame. As expected, the *SAE* value gets higher for those MBs belonging to high motion parts of the frame. The threshold used to control the amount of skipping in a frame is fixed to a constant value, determined by experimental observations over many different sequences.

In order to apply the basic idea of Alg_2 to HD frames, they are virtually divided into 9 areas, located by the first two MB rows and columns, and by the last two MB rows and columns, as shown in Figure 1.

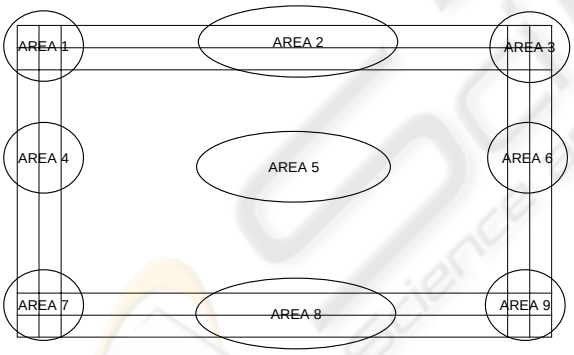


Figure 1: Nine areas in a 1280×720 HD frame.

This structure allows to have a high number of available reference MBs in the central part of the frame (AREA 5), where it is more probable to find high motion MBs, *i.e.* MBs for which the SKIP coding option is less probable to occur. All the MBs included in AREA 5 have also up to four available neighbouring MBs, so that decision on their coding mode can be performed by considering two controlling conditions:

- the number of skipped MBs in the last frame available as a reference;
- the dominant coding mode of the neighbouring MBs, in the current frame, according with a majority rule.

The MBs located in AREA 5 can also be subjected to the Mode Decision strategy presented in Alg_3, because all their neighbouring MBs are available, as shown in Figure 2. The adaptive threshold depends on the residues of the neighbouring MBs, and on a constant C, which will assume different values:

$$Th_{ASV} = C \cdot (S_{N1}, S_{N2}, S_{N3}, S_{N4}) \tag{7}$$

being S_{Ni} the residue of the i-th neighbour.

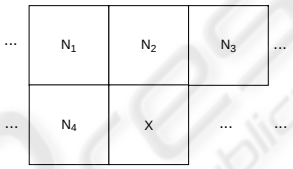


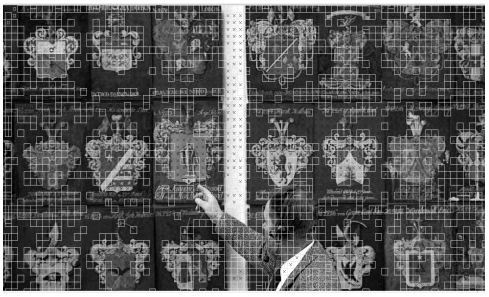
Figure 2: Relative positions of four nearest encoded neighbours of the current MB.

The MB skipping strategies discussed up to this point have been integrated within the Baseline profile of the Reference Software JM11.0, for testing purposes on the HD sequences Car, Shields, and Park Run. The Car sequence presents a dominant static component, the sky in the background, which is mainly encoded as SKIP, whereas the part of the frame showing a highway with passing cars is mainly encoded as INTER (*i.e.* predicted). The proposed schemes try to increase the number of skipped MBs within this region. The Shields sequence is somehow more "complicated", because it is very rich in details, so that increasing the number of SKIP MBs could determine an unacceptable loss in quality. Finally, the Park Run sequence is a good representative of sequences rich in motion, as it features a moving subject, together with a movement of the camera (panning) tracking it. Sample frames of the three sequences are shown in Figure 3.

4 PERFORMANCE EVALUATION OF THE MB SKIPPING ALGORITHMS ON HD SEQUENCES

In order to test the performances provided by the MB skipping algorithms previously discussed, at first they have all been applied to encode each sequence in

a)



b)

c)

Figure 3: Sample frames of the: a) Car, b) Shields, and c) Park Run sequences.

the Baseline profile, then, the one showing the best behavior has also been tested with the Rate Distortion Optimization (RDO) option activated, and with a frame rate constrained at 15 fps. The experimental results are reported in the following.

4.1 Tests on the Car Sequence

Table 3 shows the variation of the total encoding time, and the quality loss, with respect to the Reference Software, in the case of the Car sequence, for each algorithm and some variants, *i.e.* different threshold values for Alg_1, different amounts of neighbor MBs for Alg_2, different values for the constant C in Alg_3, and combinations of Alg_1 and Alg_2.
The joint adoption of $Alg_1(1)$ and $Alg_2(2)$ provides the best trade-off between quality loss and coding time reduction; in the case of HD sequences, a quality loss of up to 0.2 dB is considered acceptable. For this

Table 3: Experimental results on the Car sequence.

Algorithm	Δ Coding Time (%)	Δ PSNR Y (dB)
Alg_1 (1)	-21.1	-0.79
Alg_1 (2)	-30.3	-2.94
Alg_2 (1)	-50.1	-6.97
Alg_2 (2)	-29.3	-3.4
Alg_2 (3)	-31.5	-3.46
Alg_3 ($C = 0.5$)	-11.4	-0.33
Alg_3 ($C = 0.85$)	-12.6	-0.64
$Alg_1(1) \wedge Alg_2(2)$	-19.2	-0.16
$Alg_1(1) \vee Alg_2(2)$	-53.5	-7.92

case, a comparison between the Reference Software and the $Alg_1(1) \wedge Alg_2(2)$ option is provided in Figure 4, showing the average PSNR values of the Luma component over each frame.

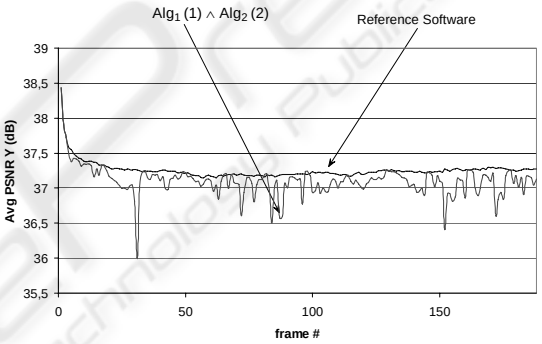


Figure 4: Frame-by-frame PSNR Y comparison between Reference Software and $Alg_1(1) \wedge Alg_2(2)$ option, for the Car sequence.

Fluctuations in the PSNR Y values, due to the $Alg_1(1) \wedge Alg_2(2)$ option, with respect to the Reference Software, can be explained by considering that some MBs, corresponding to the moving cars, are skipped anyway, because the majority of their neighbours, belonging to the static highway in the background, gets skipped. This phenomenon causes a decrease in the average PSNR Y values, that becomes quite high (over 1 dB) in some frames (30, 155).

The same coding option, $Alg_1(1) \wedge Alg_2(2)$, has been tested with RDO activated and a target frame rate of 15 fps. The results obtained are reported in Table 4. The adoption of the proposed skipping algorithm provides remarkable improvements, with respect to the Reference Software, by reducing the required coding time. This is a valuable issue when dealing with time-constrained systems.

Table 4: Performance comparison between $Alg_1(1) \wedge Alg_2(2)$ and Reference Software, with RDO activated, and a target frame rate of 15 fps (Car).

Option	Δ Coding Time (%)	Δ PSNR Y (dB)
RDO	-14.08	-0.16
15 fps	-9.83	-0.16

4.2 Tests on the Shields Sequence

Following the same approach used for testing the MB skipping algorithms on the Car sequence, Table 5 reports the variation of the total encoding time, and the quality loss, with respect to the Reference Software, in the case of the Shields sequence.

Table 5: Experimental results on the Shields sequence.

Algorithm	Δ Coding Time (%)	Δ PSNR Y (dB)
Alg_1 (1)	-12.5	-0.48
Alg_1 (2)	-13.2	-1.27
Alg_2 (1)	-22.2	-0.17
Alg_2 (2)	-13.8	≈ 0
Alg_2 (3)	-13.9	≈ 0
Alg_3 ($C = 0.5$)	-13.3	-0.28
Alg_3 ($C = 0.85$)	-18.3	-1.27
$Alg_1(1) \wedge Alg_2(2)$	-9.6	≈ 0
$Alg_1(1) \vee Alg_2(2)$	-19.4	-0.83

The best behaviour, in this case, is obtained by applying Alg_2 (1): the very low motion of the camera permits to skip a good amount of MBs in some parts of the frame that are quite homogeneous. This way, it is possible to perform an efficient Mode Decision on the basis of the neighbouring MBs. Figure 5 shows the frame-by-frame trend of the average PSNR of the Luma component, computed by applying Alg_2 (1), with respect to the Reference Software.

There is a good match between the average PSNR Y values determined by the algorithm, and those provided by the Reference Software, thanks to the fact that the sequence under test is a low motion one, and it does not present sudden variations of the scene. The $Alg_2(1)$ coding option has been tested with RDO activated and a target frame rate of 15 fps. The corresponding performances are reported in Table 6; they show not a remarkable improvement, with respect to the Reference Software, meaning that the sequence is handled similarly by the two encoder implementations.

4.3 Tests on the Park Run Sequence

The last series of results obtained by testing the proposed algorithms are related to the Park Run HD sequence. The variations of the total encoding time, and the quality loss, with respect to the Reference Software, for this last sequence under test, are reported in Table 7.

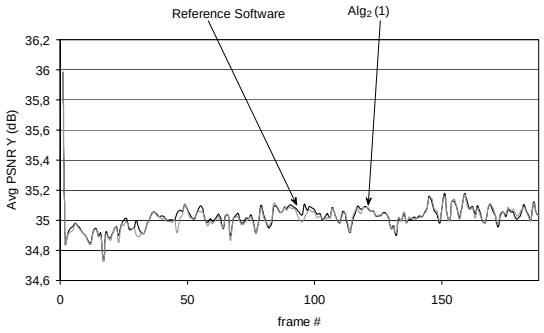


Figure 5: Frame-by-frame PSNR Y comparison between Reference Software and Alg_2 (1) option, for the Shields sequence.

Table 6: Performance comparison between $Alg_2(1)$ and Reference Software, with RDO activated, and a target frame rate of 15 fps (Shields).

Option	Δ Coding Time (%)	Δ PSNR Y (dB)
RDO	-1.23	≈ 0
15 fps	-2.97	≈ 0

Table 7: Experimental results on the Park Run sequence.

Algorithm	Δ Coding Time (%)	Δ PSNR Y (dB)
Alg_1 (1)	-9.8	-5.49
Alg_1 (2)	-13.2	-10.17
Alg_2 (1)	-9.5	≈ 0
Alg_2 (2)	-9.8	≈ 0
Alg_2 (3)	-9.8	≈ 0
Alg_3 ($C = 0.5$)	-8.2	-0.22
Alg_3 ($C = 0.85$)	-16.6	-3.48
$Alg_1(1) \wedge Alg_2(2)$	-6.3	≈ 0
$Alg_1(1) \vee Alg_2(2)$	-7.5	-5.6

The Alg_2 (3) coding option provides a limited reduction of the total encodig time, without affecting the final quality of the sequence. Many of the other algorithms tested, though ensuring a stronger reduction of the coding time, cause an unacceptable penalty of the final quality; we could generalize this result to high motion sequences, being Park Run a good representative of them. For the algorithm providing the best performances, a frame-by-frame comparison of the average PSNR Y values is proposed in Figure 6.

The same MB skipping algorithm has been tested also with RDO activated, and a target frame rate of 15 fps. The corresponding performances are reported in Table 8. The application of the proposed skipping al-

Table 8: Performance comparison between Alg_2 (3) and Reference Software, with RDO activated, and a target frame rate of 15 fps (Park Run).

Option	Δ Coding Time (%)	Δ PSNR Y (dB)
RDO	-0.32	≈ 0
15 fps	-2.07	≈ 0

gorithm provides slight improvements in constrained systems, with a small reduction of the total encoding time.

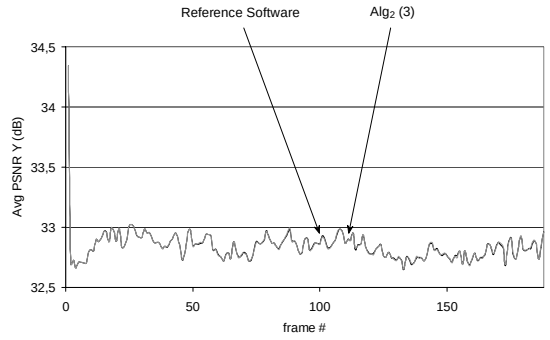


Figure 6: Frame-by-frame PSNR Y comparison between Reference Software and Alg_2 (3) option, for the Park Run sequence.

The last comparison presented in Figure 7 shows the reduction of the coding time provided by some of the proposed algorithms, for each sequence under test. As a general remark, MB skipping algorithms have a stronger effect on sequences presenting sudden variations of the scene, like Car, whereas their performances get similar to those of the Reference Software, when dealing with static or low motion sequences, like Shields.

5 CONCLUSION

This paper discussed different MB skipping algorithms to increase the efficiency of the Mode Decision process, when applying the Baseline profile of the H.264/AVC standard for the real time coding of HD sequences. The algorithms rely on three basic strategies and their combinations. The results presented in the paper show that different skipping schemes should be applied, according with the main foreseen features of each sequence, to get the best trade off between coding time reduction and quality loss. The most remarkable improvements are usually obtained for sequences presenting sudden variations of the scene, or a high motion degree.

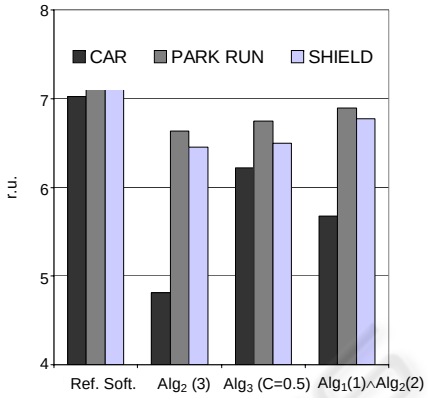


Figure 7: Coding time variations due to some of the proposed MB skipping algorithms, for each HD sequence under test.

REFERENCES

International Telecommunication Union. (2005). Advanced Video Coding for generic audiovisual services. In *Telecom. Standardization Sector of ITU*. ITU Ed.

C. Kannangara, I. Richardson, M. Bystrom, J. Solera, Y. Zhao, A. MacLennan and R. Cooney. (2006). *Low-Complexity Skip Prediction for H.264 Through Lagrangian Cost Estimation*. In *IEEE Trans. Circ. Sys. Video Techn. Vol. 16, No. 2, pp. 202-208, February 2006*. IEEE. 2006.

Y. Ivanov and C. J. Bleakley. (2006). *Skip prediction and early termination for fast mode decision in H.264/AVC*. In *ICDT 2006, 2006 International Conference on Digital Telecommunications*. IEEE. 2006.

I. Richardson and Y. Zhao. (2004). *Video Encoder Complexity Reduction by Estimating Skip Mode Disortion*. In *ICIP04, 2004 International Conference on Image Processing*. IEEE. 2004.

S. Beesley, A. Armstrong, C. Grecos, and D. Parish. (2005). *Active Macroblock Skipping in the H.264 Video Coding Standard*. In *5th IASTED International Conference Visulization, Imaging, and Image Processing*. IASTED. 2005.

A. Yu, and G. Martin. (2004). *Advanced Block Size Selection Algorithm for Inter Frame Coding in H.264/MPEG-4 AVC*. In *ICIP04, 2004 International Conference on Image Processing*. IEEE. 2004.