Fast Motion Estimation for H.264/AVC using Dynamic Search Window

Sandro Moiron[1,2], Mohammed Ghanbari[2]

[1]Instituto de Telecomunicações, Portugal; [2]University of Essex, Colchester, United Kingdom

e-mails: smoiron@co.it.pt, ghan@essex.ac.uk

***Abstract* – The motion estimation is one of the most time consuming task in the encoding of inter predicted frames. Minimising the computational complexity of this task is mandatory in order to reduce the implementation cost, either in hardware or software. This paper proposes a dynamically adapted search algorithm for motion estimation to reduce the computational complexity. Moreover, a complementary refinement method is proposed to improve the objective quality of the video. Experimental results show that, when compared with the original fast full search, the proposed algorithm is able to achieve a significant reduction of the computational complexity up to 98%, introducing only a small quality penalty near 0.08dB and a bitrate increase up to 3%.**

I. INTRODUCTION

Motion estimation is one of the most important techniques used in video compression systems. It is widely used and recognised by its high efficiency, while reducing the temporal redundancy between successive frames. As a result, it plays an important role in most video standards such as MPEG *(Moving Pictures Experts Group)* and H.26X.

In the past, several fast search algorithms have been introduced to improve the efficiency in motion estimation [1] [2] [3]. Methods such as bidirectional prediction and sub pixel motion accuracy allowed to obtain better predictions [4]. The H.264/AVC [5] video compression standard goes a step forward into this direction by improving and refining further the efficiency of motion estimation. This is accomplished by introducing some novel tools which exploit the visual correlation and increase the compression efficiency of inter coded frames.

The variable block-size motion compensation with small block sizes introduces the motion estimation of small block partitions (from 16×16 up to 4×4). This improves the compression efficiency by allowing a more accurate estimation of the motion. The improvement is particularly noticeable in more detailed areas or with objects smaller than the block size 16×16. However, performing an individual motion estimation for each block partition significantly increases the number of calculations. Consequently, the computational complexity of the process is intensified. The second tool introduced was the multiple reference picture motion compensation. This technique allows the motion estimation using a list of up to 16 previously coded frames which increases the number of possibilities to find a good predictor. The computational complexity introduced by this tool is proportional to the number of reference frames used in the motion estimation process. However, while complexity increases linearly with the number of reference frames available, the quality improvement has a logarithmic trend. Overall, combining the new tools significantly improves the compression gain making the H.264/AVC the flag-ship in compression efficiency. Compared with the previous standards, the H.264/AVC is capable to obtain the same visual quality as its predecessor MPEG-2 [6] with 2 to 3 times lower bitrate [7].

The most popular algorithm for motion estimation is the block-based full search algorithm due to a number of reasons. Its implementation is straightforward and fits well with the rectangular frames used in video. Furthermore, it effectively combines with other techniques such as block-based image transforms. However, the high computational load associated with this algorithm represents 60-80% of the H.264/AVC encoder complexity. Therefore, a significant effort has been devoted in minimising the computational complexity of this algorithm. This is performed in order to reduce the implementation cost of hardware and software based encoders, as well as unshielding the possibility to use it in portable devices.

Previous work in this field can be divided into two different categories according to its strategy. The first category tries to reduce the search steps [1] [2], while the second tries to simplify the computation of the error criterion [3]. Both strategies have proven to achieve a significant reduction in the computational complexity. However, they lack in terms of flexibility to the content being coded by disregarding the current motion type of the movie sequence.

In this paper we introduce a new method to reduce the complexity of block based motion estimation. Unlike previous works that combine a fixed search window with a sparse motion search, this paper proposes the use of a dynamic search window combined with a refinement method. The search window size is defined according to the motion type of the video sequence and is able to assume rectangular shapes with variable dimensions and ratios. Experimental results show that this new method is able to achieve significant complexity reductions. When compared with the original block-based fast full search algorithm, the proposed method is able to reduce the computational complexity up to 98% introducing a minor reduction in the objective quality up to 0.08dB with a small bitrate increase of 3%.

In the next section the classical full search block-based motion estimation is introduced describing the process and metric involved in the process. Section III introduces the proposed method, discussing the methodology used to define the search window size and describing the issues arising from this implementation. An additional refinement method is also proposed and evaluated in section IV. Finally, section V presents the experimental results and section VI draws some conclusions.

The work of Sandro Moiron is supported by the *Fundação para a Ciência e a Tecnologia, Portugal* under the PhD Grant SFRH/BD/40117/2007 and by IT - Instituto de Telecomunicações.

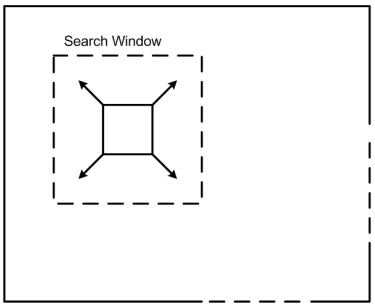

Figure 1 - Original search window.

II. Generic motion estimation

Despite the fact that many different implementations were proposed in the past years, the full search block matching is still the most popular strategy for motion estimation due to its high efficiency. It is frequently used in reference implementations of video encoders where the maximum quality is a requirement in order to measure the best compression gain that it can achieve. However, this has a huge computational cost and is not typically used in commercial encoders.

The motion compensation aims to reduce the temporal redundancy between two frames. It is implemented using motion estimation techniques that attempt to find the best position within the reference frame. This results in the minimum residual after the motion compensation. The process is usually performed in a block basis in order to simplify the implementation. Subsequently the image is divided into blocks of 16×16 pixels or smaller and subtracted from the reference frame in the best matching position. This block is commonly named predictor since it is the one that best predicts the motion occurred between the two frames. Due to the temporal proximity between the current and the reference frame, it is expected to find the corresponding block nearby the co-located position within the reference frame. As a result, the motion compensation is generally restricted to a predefined number of pixels around the original position of the block as illustrated in figure 1. This margin defines a search window that confines the motion estimation which would otherwise search for the best predictor within the full frame. Performing motion estimation along the full frame introduces a severe load due to the test of a wide range of possible positions. Furthermore, it would only introduce negligible quality gains.

The selection of the best prediction is defined according to a minimum residual criterion. Several metrics have been proposed for this purpose but the most frequently used is the *Sum of Absolute Differences (SAD)*. This metric computes the difference between the original block and the predicted block. In equation 1, the current pixels of macroblock (*MB*), belonging to the area (*A*), at spatial position (x, y) are subtracted from the macroblock prediction (*MB'*), which is displaced from (x, y) by the motion vector (v_i, v_j).

$$SAD = \sum_{(x,y)\in A} |MB_{x,y} - MB'_{x,y}(v_i, v_j)|. \qquad (1)$$

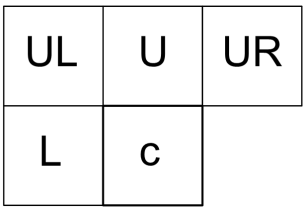

Figure 2 - Prediction blocks.

The SAD calculation is performed over the whole search window. In order to check all the search window positions, the most straightforward approach is to perform a raster scan, starting on the top left corner and scanning line by line until the bottom of the search window edge. However, there is a high probability for the best prediction to be located in the original position of the block as it happens for static areas. Therefore, a spiral scan centred in the original block position combined with the early termination technique is a most desirable strategy. When compared with the raster scan, the spiral scan guarantees the same quality with a fraction of the computational complexity. Moreover, the raster scan fast search like *Three Step Search (TSS)* [1] and the *Cross Search Algorithm (CSA)* [2], do have a poor performance for dynamic variations of motion among the objects in the frame.

The goal of search algorithms is to find for each block the best position in the reference frame.Transmitting one motion vector for each individual block can represent a significant number of bits due to the wide range of values that each motion vector can assume. Furthermore, motion vectors of neighbouring blocks are often highly correlated. As a result, the H.264/AVC determines the centre of the search window according to the motion of the neighbouring blocks; also known as motion vector predictor. This vector is defined by the median of the previously coded blocks (UL, U, UR and L) illustrated in figure 2. A more compact representation of the resulting motion vectors is obtained by the differential coding between the current and the predicted motion vector.

Overall, motion estimation with a predefined search window can efficiently remove most of the temporal correlation between two frames. However, the encoding complexity of each sequence is almost unaffected by the motion activity between frames. Sequences with higher motion activity are more likely to benefit from the use of a large search window. On the contrary, static sequences such as 'head and shoulders' do not benefit much from it. Therefore, using a fixed search window for all sequences may introduce a significant load and negligible quality improve. The following section proposes a method to adaptively select an appropriate search window size, in accordance with the temporal activity of the sequence.

III. Proposed Algorithm

This section describes an algorithm to reduce the computational complexity of the motion estimation routine through the use of an adaptive search window. This method defines the size of the search window based on the temporal mo-

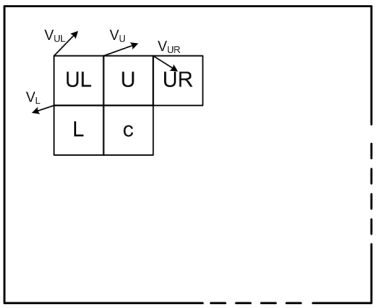

Figure 3 - Motion vectors from neighbouring blocks.

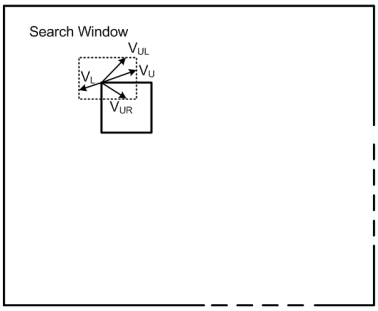

Figure 4 - Dynamic Search Window.

tion activity between frames. Furthermore, the proposed method is highly adaptive due to the capability to adjust the search window on a block basis.

In the previous section it was depicted that there is a high correlation of the motion activity between the neighbouring blocks. This results from the fact that blocks within a confined area are expected to have similar motion characteristics like direction and size. Figure 3 illustrates the motion activity of the surrounding predictors of block c where it can be verified that the motion vectors (V_{UL}, V_U, V_{UR} and V_L) tend to have similar direction and amplitude. Consequently, the proposed method defines the boundaries of the search window for the current block c, according to the motion characteristics of its neighbours. This method reduces significantly the size of the search window by exploiting the motion trend of the neighbours of the current block being encoded.

The motion vector diversity of the neighbours is analysed to estimate the area where the best prediction should be located. Afterwards, as it is illustrated in figure 3, the neighbouring motion vectors can be used to define the boundaries for the search window. Figure 4 shows that each one of these motion vectors act as a border delimiter for the motion estimation process, restricting the search range. The new search window is defined by the minimum and maximum offset imposed by of the motion vector of each neighbouring block. This is described in equation 2 where $SWmin_{(x/y)}$ is the x and y coordinates of the upper corner of the search window and $SWmax_{(x/y)}$ is the respective lower right corner. In this equation, $SWmin_{(x/y)}$ is equal to the minimum x and y coordinate values of all motion vectors in the input vector group; similar procedure applies to $SWmax_{(x/y)}$.

$$\begin{aligned} SWmin_{(x/y)} &= min\{V_{UL}, V_U, V_{UR}, V_L\} \\ SWmax_{(x/y)} &= max\{V_{UL}, V_U, V_{UR}, V_L\} \end{aligned} \quad (2)$$

Nevertheless, some issues apply to this method. The first row of blocks of each frame is encoded with the original full search window due to the lack of a sufficient number of neighbours. The second issue is the lower accuracy associated with the blocks belonging to the first column of the frame. These groups of blocks can only consider two previously coded blocks (U and UL), which might lead to a very restrictive search window and consequently a less accurate prediction. In order to minimise this inaccuracy an additional border, defined in the following section, is included to provide a minimum search range and find a better predictor. It should be highlighted that this will have a positive impact on the accuracy of all the following blocks.

IV. Refinement

This section proposes a refinement method to further improve the objective quality of the algorithm described in the previous section. This refinement is proposed as an additional method to be combined with the dynamic search window method.

The simple definition of the search window size based on the neighbouring blocks can, in particular cases, lead to a very restrictive search area. Although this is desirable from the point of view of the computational complexity, the reduction of the search window removes the flexibility required by motion estimation. The worst case happens when all the neighbours of the current block c have the same motion vector. Although this situation shows that the motion is consistent among all blocks in that area, the resulting search window size is zero because $SWmax_{(x/y)}$ becomes equal to $SWmin_{(x/y)}$. This situation limits the motion vector of the current block to a single value, blocking any attempt to find a better prediction that may reflect a different motion.

The method hereby proposed expands the search area by n pixels beyond each edge of the previously defined search window. Thus, even if the reduced search window is zero, the refinement margin will guarantee a minimum flexibility to accommodate variations of the motion. The selection of the number of pixels used in this refinement border is discussed in the following section. The analysis gives an overview of the objective quality improvements and the respective computational complexity increase.

V. Experimental Results

In order to test the performance of the proposed algorithm, the H.264/AVC reference software JM14.2 [8] was modified to accommodate the modifications. Three sequences, *(akiyo, bus* and *stefan)*, with 352×288 pixels were used to represent video content with slow, medium and high motion. These sequences were used to evaluate the performance and consistency for high and slow motion scenarios. All sequence were coded at 30 frames per second using the H.264/AVC in High Profile, variable bitrate, encoding scheme IPPPP, a single reference frame and without sub

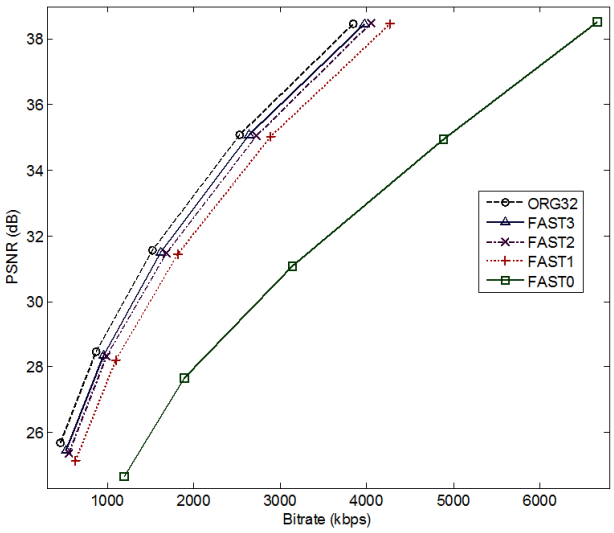

Figure 5 - RD performance for different refinement sizes on *bus* sequence.

pixel motion estimation. This test measures the objective quality *Peak Signal to Noise Ratio (PSNR)* and complexity reduction in terms of number of positions tested by the motion estimation routine (average search window size). Since the main focus of this work is the motion estimation algorithm, the presented complexity results only compare the area of the search window.

Figure 5 illustrates the quality performance achieved by the proposed dynamic search window algorithm when combined with the refinement method. Different border sizes were tested ranging from zero to three pixels, represented as *(FAST0/1/2/3)*, and compared with the default search algorithm used in the reference software *(ORG32)*. It can be verified that using a small refinement border significantly improves the objective quality. Further increasing the border size only introduce small quality improvements.

Table I presents the comparison results between the proposed algorithm and the default full search and window size used by the reference software. Each sequence is compared with its respective version encoded using the default algorithm. The results show an differential objective quality close to the original encoding (0.08dB) while introducing a small bitrate increase of 3% for *bus* and less than 1% for two other sequences. However, a complexity reduction near 98% is consistent for all the sequences tested. It can also be verified that sequences with high motion activity use an average search window bigger than sequences with small motion activity. This shows that the proposed algorithm is capable to adjust to the motion activity of each sequence.

VI. Conclusions

In this paper we presented a new method to reduce the computational complexity of the motion estimation. The proposed method is capable to dynamically adjust the search window on a block basis minimising the number of positions to be tested by the motion estimation algorithm. An additional refinement method is also proposed to be combined with the previous algorithm in order to further improve the objective quality results.

Table I
Performance comparison with the default full search of JM14.2 with a search window range of 32 pixels and quantiser equal to 28.

Method	△PSNR (dB)	△Bitrate (%)	Search Window (pixels)	Complexity Reduction (%)
akiyo				
FAST0	0.02	22.75	1.21	99.72
FAST1	0.00	0.18	1.74	99.52
FAST2	0.00	0.12	2.52	99.14
FAST3	0.00	0.06	3.39	98.56
bus				
FAST0	0.07	73.17	1.21	99.72
FAST1	0.00	10.73	2.02	99.40
FAST2	0.02	5.36	2.86	98.93
FAST3	0.01	3.16	3.81	98.24
stefan				
FAST0	0.35	27.76	1.22	99.72
FAST1	0.11	2.18	1.99	99.41
FAST2	0.09	1.15	2.87	98.93
FAST3	0.08	0.73	3.81	98.24

Experimental results show that the dynamic search window algorithm combined with the refinement method is able to achieve a significant complexity reduction up to 98% while introducing only a negligible objective quality penalty up to 0.08dB with 3% of bitrate increase.

References

[1] Reoxiang Li, Bing Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation", *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 4, no. 4, pp. 438–442, 1994.

[2] M. Ghanbari, "The cross-search algorithm for motion estimation", *Communications, IEEE Transactions on*, vol. 38, no. 7, pp. 950–953, 1990.

[3] W. Li and E. Salari, "Successive elimination algorithm for motion estimation", *Image Processing, IEEE Transactions on*, vol. 4, no. 1, pp. 105–107, 1995.

[4] T. Wiegand, G. Sullivan, G. Bjntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard", *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 557–559, July 2003.

[5] Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, "Draft ITU-T recommendation and final draft international standard of joint video specification ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC", JVT-G050 2003.

[6] ITU-T, *Recommendation H.262, Information Technology - Generic Coding of Moving Pictures and Associated Audio Information: Video*, Feb. 2000.

[7] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with H.264/AVC: Tools, performance, and complexity", *IEEE Circuits and Systems Magazine*, pp. 7–28, 1st Quarter 2004.

[8] http://iphome.hhi.de/suehring/tml/, *Suehring JM H.264/AVC*.