HIGH PERFORMANCE MOTION PREDICTION ALGORITHM FOR H.264/AVC ENCODING

Ronaldo Husemann[1], Humberto Frohlich[2], Ricardo Kirtchner[3], Valter Roesler[4], Altamiro Amadeu Susin[5]

Av. Bento Gonçalves, 9500, Block IV, Prédio 72, Sala 233 - Porto Alegre, RS -Brazil

Instituto de Informática – UFRGS – Phone: +5551 3308-6167

{[1]rhusemann, [4]roesler}@inf.ufrgs.br, {[2]frohlich, [3]kuronno}@gmail.com,[5]altamiro.susin@ufrgs.br

ABSTRACT

The ITU-T H.264 can be considered at the present time as the most efficient video coding standard. This codec uses distinct and complex combined techniques in order to encapsulate raw videos in highly compressed streams. Unfortunately the improved H.264's coding efficiency increases significantly the encoder computational complexity, difficultying real-time operation. Particularly the motion estimation algorithm is an important algorithm used to remove temporal redundancy within image sequences. Considering its importance, in this paper, we propose an optimized high performance motion estimation algorithm. The proposed algorithm, based in a small diamond topology zonal search, has been designed to operate with orthogonal linear vectors specially aligned with a subsampled memory, in order to reduce the number of operations and data manipulating. Our proposal was implemented in the H.264 JM software reference indicating significant gains in terms of performance.

Index Terms— Motion estimation, H.264 video encoder.

1. INTRODUCTION

In beginning of 1980 the ITU-T organization introduced the H.261 video encoder standard mainly aiming teleconference applications. This standard had been used as base for the development of distinct subsequent video encoder solutions like MPEG-1, H.262/MPEG-2 (main standard for DVD and digital TV devices), H.263, H.263+, MPEG-4, and others [1].

In March 2001 the organizations ITU-T and MPEG published jointly a new emerging international video encoding specification entitled as H.264/MPEG-4 AVC (Advanced Video coding) [2].

Comparison analysis among all the cited video codecs points to that the H.264 specification has the better coding efficiency. Due the H.264's improved coding efficiency the new codec can be considered until now as the state of the art in video encoding [1].

All these H.264 coding advances only could be obtained due to several internal improvements like: DCT (Discrete Cosine Transform) supporting smaller sizes block (4x4 and 8x8 pixels), intra prediction algorithm (allowing reuse of spatial data information inside a frame), flexible slice description (arbitrary macroblock orderind and slice size), quarter-sample accuracy, block data partitioning in distinct configurations (4x4, 4x8, 8x4, 8x8, 8x 16, 16x8 and 16x16), additional in-loop deblocking filter (improving subjective video quality), adaptive entropy methods as CAVLC (Context Adaptive Variable Lenght Coding) and CABAC (Context Adaptive Binary Arithmetic Coding) and others [3].

Unfortunately the inclusion of all these distinct algorithms impacts increasing significantly the overall encoder complexity. Figure 1 represents a H.264 encoder block diagram structure In fact the increased complexity represents a relevant difficult for the adoption of a H.264 encoder solution in devices with limited processing power and mainly when real-time operation are desired.

Particularly inside a video encoder there are modules (represented in the Figure 1 as shadowed blocks) responsible by the motion prediction algorithm, also called as motion estimation. This algorithm must to identify redundant information from each specific macroblock in relation to other positions in the same frame (intra prediction) or in relation to one or more previous frames (inter prediction), which consumes a great number of computational operations and data structure handling [4].

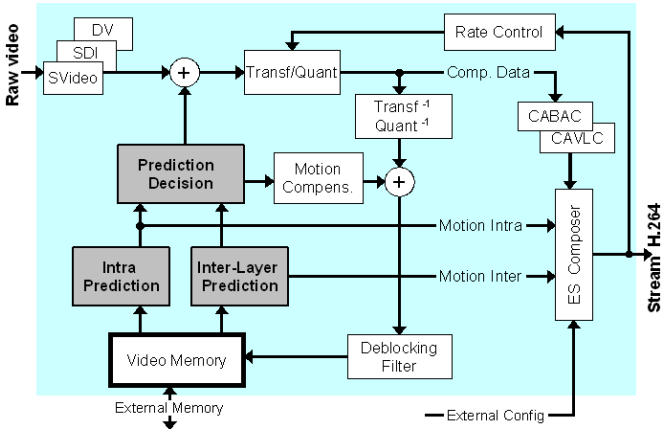

Figure 1. Simplified H.264 video encoder model.

In this paper we propose an optimized motion estimation algorithm, which was designed to improve encoding performance, based in a subsampled small diamond zonal search technique. The integrated scheme uses orthogonal linear data vectors specially aligned with memory to reduce complexity and data manipulating in a practical implementation.

Our proposal has been implemented in the JVT (Joint Video Team) H.264 reference code version 16. Experimental results confirm significant gains in terms of performance.

This paper is organized in the following way: Section 2 describes the relevant motion estimation works, Section 3 presents the proposed approach, Section 4 depicts the performed experiments and its results and finally Section 5 presents some conclusions.

2. MOTION PREDICTION WORKS

As already commented the motion estimation algorithm is responsible for identifying of spatial (intra) or temporal (inter) pixel data similarity among frames, allowing to reuse whole macroblocks from a reference frame. When an equivalent data macroblock is located in the reference frame it is registered in form of motion vectors (bidimensional offset between macroblock positions). The new frame can be reconstructed using motion vector information. The difference between real frame and the reconstructed one is called residual error and must be included in the elementary stream to allow compensation in the decoder side.

In order to identify pixel data redundancies each macroblock passes for intensive computation mechanisms, where must be computed the similarity values of the specific target macroblock for different positions in the reference frames. Distinct similarity calculation method could be used (SAD, HAD, MSE, etc) [5].

The most known motion estimation mechanism is called as full search method. It receives this name because looks for a similar macroblock inside the whole reference window search area in, testing all possibilities (full search). After perform all tests it can determine the best motion vector (lowest similarity error). The variation called Fast Full Search performs the macroblock searches in a spiral topology aiming to speed up efficiency for slow videos (when the motion vector registers small values).

Full search is a very accurate procedure, however is very slow due the great number of performed tests. In order to reduce processing time several sparse search motion estimation method were created in the last years. Basically these solutions were designed to search for similarity block only testing some specific regions, called regions of interest (ROI), in the window search area. Due the reduction of the number of possibilities this approach can register some error distortions when compared with full search method, but, in the other hand, reduces strongly the number of operations.

Sparse search algorithms can use square geometric search topology (four positions are tested for each step) as the examples Three-Step search (TSS)[6], the Four-Step Search (4SS) [6] and the Diamond Search (DS) [5].

The UMHEX algorithm [7] adopts optimized hexagonal search topologies (six tested positions for each step) to improve efficiency in the best motion vector determination.

Recent motion estimation algorithms started to incorporate a zonal search approach, where the specific window search ROI for each macroblock is dynamically adjusted considering previous calculated motion vectors (called as predictors). An example is the PMVFast [4] method, which tests all macroblocks around the predictors, following the lowest similarity value until locate the threshold value.

3. PROPOSED APPROACH

As already commented, the H.264/AVC encoder high complexity difficult its use in real-time applications. To reach this goal, we propose an innovative motion estimation architecture, specially designed to increase performance.

3.1. Block Search Pattern

PMVFAST algorithm adopts a block search pattern which combines two structures, large and small diamond, but only the second one allows pixel resolution [4]. We suggest a simplified solution based only in a small diamond structure, which is firstly centered on the previous candidate, refining around it in order to select another better solution.

In our approach, during the first stage five neighbor candidates should be analyzed in a cross topology (center, right, left, up and down). The search process terminates when the central candidate registers the lowest similarity. In other cases, the small diamond pattern should be moved in the direction of the most similar block. Observing Figure 2 we can note that when moving in any direction only three new values (black positions) must be computed.

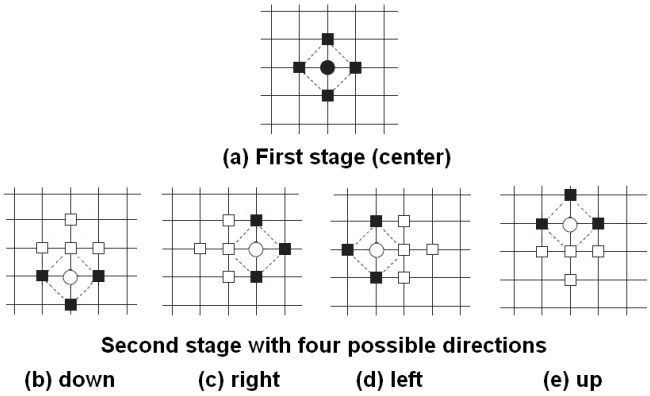

Figure 2. Small diamond search pattern of the proposal.

3.2 Similarity error calculation

The overall motion prediction processing time is strongly dependent of the adopted method of similarity error calculation because this method should be performed several times each step. In order to speed up the performance in our proposal we suggest using a 4:1 subsampling SAD scheme.

The SAD (Sum of Absolute Differences) is a traditional similarity error computing method largely used in image compressing. In fact most of the high performance processors (Intel SSE2, AMD 3Dnow and IBM Cell) provide specific SIMD (Single Instruction Multiple Data) operators which are available to speed up this operation.

Particularly for Intel and AMD platforms we found an optimized hardware architecture able to perform a complete SAD operation over a 64-bits array (when eight samples of 8-bits are processed simultaneously) in only one machine cycle. Considering that a normally extensive operation of whole macroblock SAD calculation should be implemented in few clocks. So our solution is based in internal orthogonal 64-bit vectors, called as linear vectors. Each linear vector is particularly aligned for one the main directions (horizontal and vertical respectively) demanded by the proposed diamond pattern (Figure 2).

By using these linear vectors a particular block shift can be implemented only by updating the respective vector data pointer. For example when a right shift in needed we just increment the horizontal linear vector data pointer. If a left shift is needed the same data pointer is decremented. Similar approaches occur to vertical vector during up and down shifts. The size of horizontal linear vector is the width of the window search while the vertical linear vector size is the window search height. So we have total flexibility of moving inside the window search area.

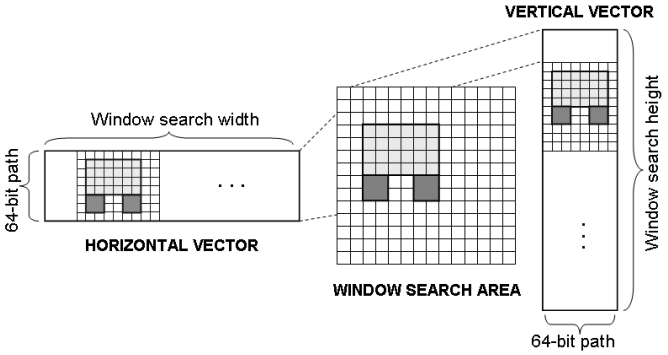

Figure 3. Proposed 64-bits linear vectors.

Additionally to the algorithm adaptation to SIMD architectures we perform a preprocessing 4:1 subsampling method. The idea is to reduce the number of samples computed to consequently improve the total processing time. This approach was proposed by Liu in [8]. In his work, Liu performed several practical experiments considering distinct subsampling configurations (1:1, 2:1 and 4:1) and concluded that a 4:1 sub-sampling could be used to obtain relevant performance gains, producing however some quality losses in the final image.

When a macroblock is submitted to a 4:1 subsampling process only one pixel is sampled for each 2x2 pixel block. The generated data structure is 75% smaller than the original one, reducing the numbers of operations, but it limits motion resolution. In other words, after a 4:1 subsampling the minimal distance between two samples is two pixels (dimension of 2x2 blocks). The obtained motion vectors lose resolution impacting in increase distortion.

In our proposal we proposed an alternative approach, where each image is subsampled, but produce distinct consecutive images each one containg part of the original reference data. Particularly for a 4:1 subsampling method each reference image will be divided in four subsampled images. In Figure 4 we can found an example of this proposed scheme when applied in a whole macroblock. The original reference macroblock, which contains 256 pixels (16x16 size) will be transformed in four subsampled blocks each one containing 64 pixels (8x8 size). To help understanding the pixels related to each subsampled image has the same color in a gray scale. For example the black pixels represent the pixels located in even lines and columns. The resulting blocks are called as REF_EE (resulting from even line and column), REF_EO (even line and old column), REF_OE (odd line and even column) and REF_OO (odd line and column).

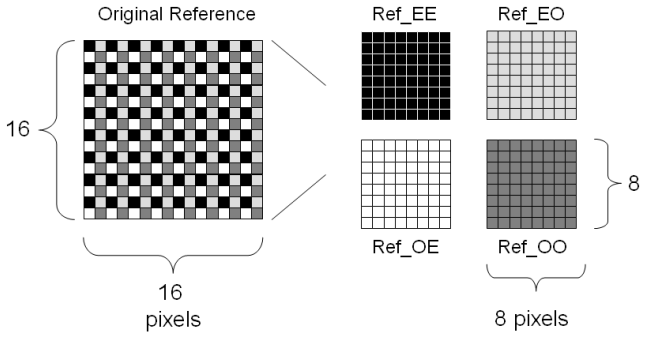

Figure 4. Subsampling method for a 16x16 macroblock.

By using these resulting subsampled images to determine error similarity instead of the original macroblock, we reduce each calculation complexity to a quarter of the total number of operations without affecting motion vector resolution. Our proposed subsampling scheme preserves pixel resolution. When the algorithm needs to determine similarity to a neighbor position we just need to jump to the consecutive subsampled image.
The implementation of this technique in our proposal involves the use of distinct complementary memories, each one containing part of the subsampled block. So we can support pixel resolution, even when we use subsampled images. Independently of the pixel block position our algorithm does not access external memory, but only points to the corresponding subsampled memory.

4. PRACTICAL RESULTS

Aiming to measure the performance gains of our proposal we have implemented it in ANSI C, incorporating it as an alternative motion estimation method of the software JM v.16.2, the H.264 AVC reference software designed by the JVT [9].

Using it we developed several practical experiments in order to compare our proposal implementation with other JM solutions.

Basically four scenarios were tested:

- Fast Full search: default JM encoder algorithm;
- UMHEX: fast motion prediction, based on an hexagonal pattern approach;
- PMVFAST: fastest JM zonal search algorithm;
- Proposed: implementation of our proposal.

The experiments here described consider two video sequences (Harbour and Crew), for distinct bitrates configurations (1.5M until 3Mbps). Both videos present the same characteristics (4CIF resolution, 30fps and 100 frames), but varying motion activities.

The parameters analyzed in this validation are the final video quality and the specific motion prediction execution time. Table 1 contains the obtained video quality results.

Table 1. Results of final quality in terms of PSNR (dB)

Video	Scenario	Bitrate (kbps)			
		1500	2000	2500	3000
Harbour	Fast Full Search	29.97	30.99	31.81	32.54
	UMHEX	29.96	30.99	31.81	32.54
	PMVFAST	29.94	30.96	31.79	32.54
	Proposed	**29.83**	**30.85**	**31.68**	**32.44**
Crew	Fast Full Search	35.95	36.81	37.46	37.98
	UMHEX	35.94	36.78	37.45	37.97
	PMVFAST	35.91	36.77	37.44	37.96
	Proposed	**35.67**	**36.56**	**37.28**	**37.82**

The obtained results comparing the algorithm performances are presented in Table 2.

Table 2. Results of execution time (s)

Video	Scenario	Bitrate (kbps)			
		1500	2000	2500	3000
Harbour	Fast Full Search	791.2	789.7	789.4	790.8
	UMHEX	4.89	4.69	4.54	4.39
	PMVFAST	1.81	1.77	1.72	1.72
	Proposed	**0.76**	**0.74**	**0.74**	**0.74**
Crew	Fast Full Search	790.1	790.1	789.9	789.8
	UMHEX	4.52	4.38	4.29	4.23
	PMVFAST	2.13	1.97	1.99	2.01
	Proposed	**1.01**	**0.99**	**0.97**	**0.96**

Video quality comparison of both sequences is depicted in Figure 5. The registered losses are very small (less than 0.3dB).

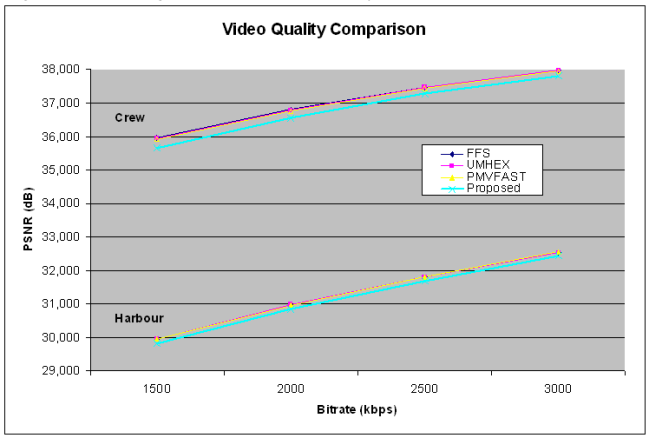

Figure 5. PSNR for luminance results.

Figure 6 and 7 present the comparison of performance registered for sequences Harbour and Crew sequence. FFS results were omitted to help comparison between the rapid algorithms.

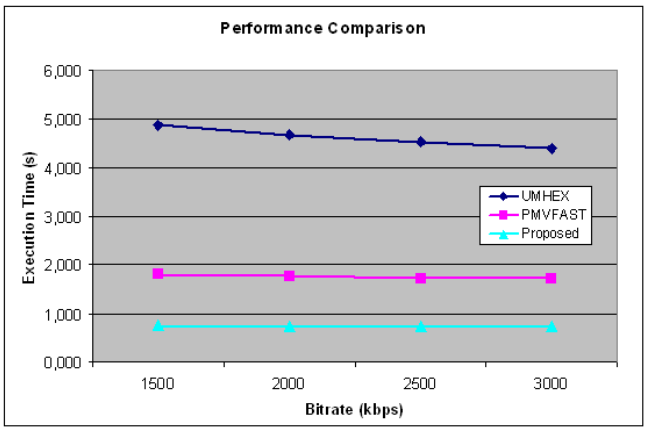

Figure 6. Performance results for the Harbour sequence.

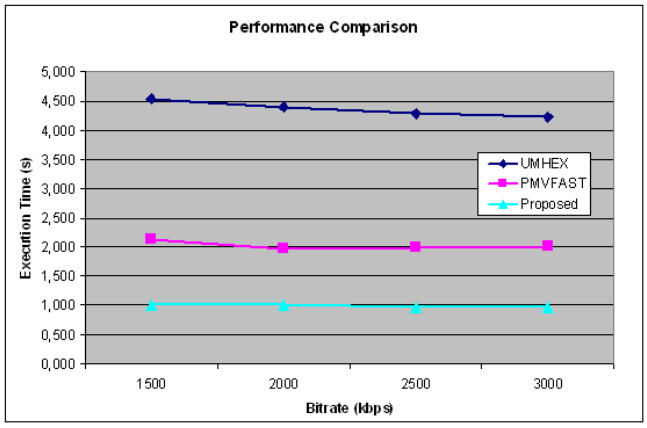

Figure 7. Performance results for the Crew sequence.

5. CONCLUSIONS

We proposed in this paper an optimized algorithm, based in a small diamond topology zonal search, which uses orthogonal 64-bits linear vectors aligned with 4:1 subsampled memories to increase performance. The proposed algorithm was incorporated in the H.264 JM software reference. The results show significant performance gains with small impacts in terms of video quality. Particularly for the Harbour sequence the mean registered quality loss was less than 0.15dB while for Crew sequence it was less than 0.3dB. For both sequences the experiments registered for our proposal performances, in term of execution time, more than 2 times bigger that the fastest JM motion prediction solution.

6. REFERENCES

[1] Richardson, I.E.G. *H.264 and MPEG-4 Video Compression*, UK: Wiley & Sons, 2003.

[2] G. J. Sullivan and T. Wiegand, "Video compression from concepts to the H.264/AVC standard," Proc. IEEE, pp. 18-31, Jan.2005.

[3] R. Kordasiewicz and S. Shirani On Hardware Implementations Of DCT and Quantization Blocks for H.264/AVC. The Journal of VLSI Signal Processing, Volume 47, Number 3, June 2007 , pp. 189-199.

[4] Tourapis A. M., Au 0. C. and Liou M. L. "Predictive Motion Vector Field Adaptive Search Technique (PMVFAST) - Enhancing Block Based Motion Estimation," in Proc. of Visual Com. and Image Processing, pp.883-892, Jan. 2001.

[5] Shan Zhu and Kai-Kuang Ma. A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. IEEE Trans.on Image Processing. V. 9, N. 2, February 2000.

[6] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," IEEE Trans. Circuits System Video Technol., vol. 6, pp. 313–317, June 1996.

[7] Z. Chen et. al, "Fast integer pel and fractional pel motion estimation for JVT, JVT-F017r1," JP, 5-13 Dec., 2002.

[8] Liu, B.; Zaccarin, A. "New fast algorithms for the estimation of block motion vectors". In: IEEE Trans on Circuits Systems for Video Technology, V. 3, N. 2, April 1993, pp 148-157

[9] ISO/IEC JTC1/SC29/WG11, "Working draft of reference software for advanced video coding". 2009.