A New Motion Estimation Algorithm for High Definition Videos and its Hardware Design

Pargles Dall'Oglio, Cássio Cristani, Marcelo Porto, Luciano Agostini
{pwdalloglio, crcristani, agostini}@inf.ufpel.edu.br, msporto@inf.ufrgs.br

Group of Architectures and Integrated Circuits - GACI
Federal University of Pelotas - UFPEL

Abstract

This paper presents a new algorithm focusing in high quality fast motion estimation for high definition video coding and a hardware design for the random search step. This algorithm provides more efficiency to avoid local minima falls in fast motion estimation due to random search exploration. The random strategy doesn't have data dependences to the hardware implementation and allow the maximum parallelization to achieve high performance. The developed algorithm is called Iterative Random Search (IRS). *It evaluates the central region of the search area, and also chooses, randomly, many other candidate blocks in the search area. This approach gives to the algorithm the possibility to avoid local minima falls, increasing the quality results, especially for high definition videos. The evaluation results shows that the algorithm gets the better quality compared to other fast algorithms and the proposed architecture for the random step can process high definition videos (HD 1080p) in real time (at 30 frames per second).*

1. Introduction

The number of electronics devices that handles with high definition videos increase each year. Most of these devices demand for solutions that aim to small hardware, low power and high quality. However, the amount of data to represent a high definition video without compression is very high, exceeding the transmission, processing and storage capacity of these devices, especially portable devices.

The video coding basically tries to identify and remove the redundant information presented in the digital video. The redundancy contained between neighbor frames (called temporal redundancy) is explored by the motion estimation (ME) process.

There are many fast algorithms to exploit the information redundancy to reduce the computational complexity. However, fast algorithms are vulnerable to local minima falls with the increase of the video resolution. This characteristic generates significant losses in the visual quality could be achieved in comparison with Full Search algorithm (FS) [1]. Because of that, is necessary the research and development of new algorithms to approach the results obtained with FS.

The high definition videos require efficient coding solutions, mainly for real time applications (30 frames per second). A current personal computer can code and decode a HD 1080p (Full High Definition) video in software in real time. However, the digital TV set-top-box, for example, does not have a last generation general purpose processor to accomplish this task with the necessary performance.

Fast algorithms have not easy hardware implementation, some characteristics as data dependencies and not regular memory access can be very complicated to deal with. In other hand, FS algorithm requires lots of hardware resources to achieve high performance. In this context, a fast algorithm, that can be easily implemented in hardware, is very important for real time ME in high definition videos.

This paper presents a new algorithm for fast motion estimation targeting high quality when processing high definition videos and a hardware design for the random search step of the IRS algorithm. This algorithm provides an efficient way to avoid local minima falls in the fast ME for high definition videos, due to the random search exploration. The IRS was applied to ten HD 1080p video sequences and the test results show that gets better quality compared to other fast algorithms.

This paper is organized as follow: the motion estimation is explained in section 2. In section 3 is presented the RS ant the IRS algorithm. In section 4 the propose architecture is explained. The comparisons with other architectures from literature are shown in section 5. In section 6 the conclusion and future works are presented.

2. Motion Estimation

Motion Estimation (ME) represents 80% of the total computational complexity of current video coders [1]. However, ME is responsible for most of the gains achieved in compression process.

A digital video consists in a sequence of pictures (frames), are needed between 24 and 30 frames per second [2] to generate a moviment sensation. Each frame are divided into blocks, each block are formed by a set of pixels. The pixels are the less information of a video, represents a brightness or a color intensity.

Neighbors pictures usually are similar because the rate sampling nearly 30 frames per second, and higher when used special cameras, for example the transmission of slow motion replays in the last soccer worldcup and the next worldcup in Brazil, able to save thousands pictures per second. This similarity between frames is defined temporal redundancy.

The ME is a stage responsible to reduce (or eliminate) the temporal redundancy. For this process, ME uses a search window (or search area) for compare blocks for a current frame to a reference frame (before frame or after frame). To find the smallest difference or the better similarity (best matching) between blocks, a subtraction is performed between current block and compared block.

There are several similarity metrics which can be used in this search process. In this paper, the criteria used to compare the similarity is the Sum of Absolute Differences (SAD) [1]. For the block with highest similarity (lowest value of SAD) a motion vector is generated to identify this block. So, the quality and the cost of each motion vector are directly connected to the used ME algorithm.

High definition videos have very homogeneous regions and in consequence have more blocks with identical values. The best blocks of these regions can be considered local minims while the best block of the all search window can be considered global minima. The increase in video resolution requires a proportional increase in the search, because it, often does not find the global minimum in the center, preventing fast algorithms that start your search in the center, find the global minimum.

The algorithms are divided in two classes: optimum and sub-optimum (fast) algorithms. The costs of these algorithms is measured by the amount of candidate blocks compared (CBC) and the quality is measured by the PSNR (Peak Signal-to-Noise Ratio) [3] in this work.

The Full Search Algorithm is the only witch always finds the global minima (the best matching) because it evaluates all candidate blocks in a search window. However, its computational cost is extremely high. The sub-optimum algorithms use some heuristics to speed up the search. These heuristics try to get good quality results with low complexity, but are susceptible to local minima.

3. The RS and IRS algorithm

The random search step of the IRS algorithm divides the search area into four regions to randomize the initial search locations. This is done to assure a better distribution of the selected blocks. The random selection as a strategy to avoid local minima falls and obtain gains in quality when coupled with others search strategies when high definition video compression is performed, while preserving a low computational complexity.

The Iterative Random Search (IRS) algorithm uses a random step and also applying an evaluation process around the central region of the search area to guarantee good results when the video has little movement, where the best candidates are near to the search area center. Additionally, the IRS algorithm also explores a final iterative refinement in best result for each quadrant, obtained in the random search.

The random stage can generate different results. Nevertheless, an analysis of the average deviation showed that this variation is insignificant, so this will not be considered in this paper.

The efficiency of random algorithms is directly influenced by the number of blocks (N value) and the size of the search area. As higher is the N value, higher is the probability to find the optimal result. However, the impacts in computational cost must be evaluated when N value is modified. A larger search area usually produces better PSNR results. It happens because new regions can be explored and more candidate blocks can be considered. However, the increase in the search area also causes an increase in the computational cost, since more blocks must be compared.

The IRS algorithm was developed in C and it was evaluated though software simulations with ten HD 1080p video sequences [4]. These simulations were done with several combinations of N values and search area size to IRS algorithm and was possible to find the best values for N and for search size to 16 and 96x96 respectively.

The results presented in Tab. 1 represent the average of the 10 test video sequences to PSNR and CCBs results for IRS, Full Search (FS) [1], Uneven Multi Hexagon (UMH) [5], Diamond Search (DS) [1], Hexagon Search (HS) [6], Three Step Search (TSS) [7], and Four Step Search (FSS) [8].

Tab.1 – Comparative results of PSNR

Algorithm	**PSNR (dB)**	**CBC x 10^6**
FS	35.90	15,040.51
IRS	**34.45**	**91.82**
UMH	34.42	311.96
DS	33.02	48.07
HEX	32.79	32.52
FSS	32.40	58.03
TSS	30.94	43.51

The PSNR difference between IRS and FS is only 1.45 dB, in other hand, the number of CCBs is decreased in more than 169 times. The IRS algorithm obtained the better PSNR result among all fast algorithms analyzed, showing the efficiency of random selection with a central evaluation and also has no data dependencies among its five iterative processes allowing greater parallelism and improved performance in hardware.

4. Proposed Hardware Architecture

The proposed architecture to random search step of the IRS algorithm uses block size 16x16 and a pixel sub-sampling rate of 4:1. Fig. 1 presents this architecture and its components. The reference memory witch contains the search area reduced to 32x32 bytes because the sub-sampling. A local memory bank, which contains a 8x8 bytes memory for each N random candidate block (represented by LM in the Fig. 1). A group of Processing Units (PU) (contains N PUs, one for each random candidate block). The COMP is a comparator responsible to find the lowest SAD between the two iterative steps, generating the final Motion Vector.

Random generators is implemented in hardware using a Linear Feedback Shift Register (LFSR) [9] with a huge number of bits (about 32) and storing the less significant bits of the random values generated by it. The first five bits and the next five bits inform the memory address for x and y position respectively, to each random candidate block.

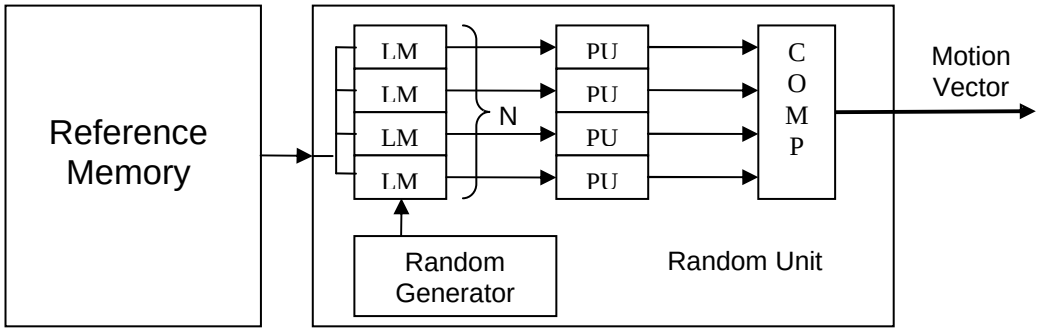

Fig. 1 – RS Architecture.

In the first cycle the reference memory is empty, so it takes one cycle per line to fill up the memory with data from an off-chip memory, which contains the reference area. In parallel with this operation, the current block memory is filled with data from the original frame (with no compression) and the random unit generates and stores all random positions in its registers.

After the first line of reference memory is filled it starts to be read line by line and if the line contains the random position the appropriate local memory (LM) is filled with a part of this line. This process is ready when the reference memory is completely read and in this moment all local memories will be filled. The next step is to read every LM line by line and process this data in the PUs, calculating the SAD. After the SADs are processed, the comparator finds the best position among the randomized positions.

5. Comparative Results

Tab. 2 presents the comparisons with related works published in the literature and using the parameters: number of Pus, PU Size, Memory Size, Cycles per Block and PSNR. The work [10] proposes an architecture to Multi Point Diamond Search (MPDS) uses the same block size and pixel sub-sampling level than that used in this work. The work [11] proposes an architecture for the DS algorithm and generates samples for fractional motion estimation and it also has a motion compensation for luminance samples integrated. That work uses the same block size than used in this work.

Tab.2 – Comparisons among architectures

	Number of PUs	**PU Size**	**Memory Size**	**Cycles per Block**
This Work	16	8	1.024	120
Sanchez [10]	45	16	12.000	560
Liang [11]	9	16	2.040	184

The proposed random architecture uses about 12 times less hardware compared to [10] as shown in Tab. 2. That work use more hardware because the MPDS algorithm uses five DS instances but obtain a better video quality.

The memory size of this architecture is larger than [11], however this work only needs to fill the reference memory once while that work has data dependencies for each iteration of DS algorithm and needs to fill the memory at each iteration.

This work does not have frequency results, since this architecture was not mapped to a FPGA or a standard cells technology. However, it is possible to evaluate the number of cycles necessary to process one input block. This works needs less cycles than that used by [11], because the random unit generates the random number in parallel when the memory is being filled, so this step does not require additional time. As the work [11] needs 184 cycles to process one block and is capable to process HD 1080p videos in real time in the worst case, this architecture should achieve a processing rate capable the same type of videos in real time. As the work [10] is about five times slower than [11], this work should also be five times faster than [10].

6. Conclusions and Future Works

This work presented a new algorithm for ME focused on high definition videos, called IRS, and also proposes a hardware architecture to implement the random step of this algorithm. The presented algorithm introduces randomness to motion estimation as a strategy to avoid local minima, achieving better video quality. Another advantage is the inexistence of data dependencies among the random blocks, allowing greater parallelism and improving the performance in the hardware.

A hardware architecture was proposed to perform the random step of the IRS algorithm. Analyzing this architecture, it is possible to conclude that it is able to process HD 1080p videos in real time (at 30 frames per second).

The comparative results show that the proposed architecture has low hardware resources utilization and can generate low power consumption because need a low frequency to generate all vectors of a high resolution frame.

7. References

[1] P. Kuhn. Algorithms, Complexity Analysis and VLSI Architectures for MPEG-4 Motion Estimation. Boston: Kluwer Academic Publishers, 1999.

[2] R .Gonzalez; R. Woods. 2004. Digital Image Processing using MATLAB, Prentice Hall, Upper Saddle River, NJ, 2004.

[3] I. Richardson. H.264 and MPEG-4 Video Compression : Video Coding for Next-Generation Multimediade Chichester: John Wiley and Sons, 2003.

[4] Xiph.org: Test media, available at <http://media.xiph.org/video/derf/>, May, 2011.

[5] X. Yi; J. Zhang and N. Ling. Improved and simplified fast motion estimation for JM. JVT-P021, July 2005.

[6] C. Zhu, X. Lin and L. Chau, Hexagon-Based Search Patten for Fast Block Motion estimation. IEEE Transactions on Circuits and Systems for Video Technology, pp.349-355, May 2002.

[7] X. Jing, L. Chau, An efficient three-step search algorithm for Block motion estimation. IEEE Transactions on Multimedia, Vol. 6, No. 3, 2004, pp. 435-438.

[8] O. Tasdizen, et al., Dynamically Variable Step Search Motion Estimation Algorithm and a Dynamically Reconfigurable Hardware for Its Implementation. IEEE Transactions on Consumer Electronics, Vol. 55, No. 3, 2009, pp. 1645-1653.

[9] H. Beker . Cipher Systems: The Protection of Communications. John Wiley & Sons Inc, 1983.

[10] G. Sanchez, et al. High Efficient Motion Estimation Architecture with Integrated Motion Compensation and FME Support. IEEE Second Latin American Symposium on Circuits and Systems, 2011.

[11] W. Liang, et al. A cryptographic algorithm based on Linear Feedback Shift Register. IEEE ICCASM, 2010.