A FAST RANDOM BASED MOTION ESTIMATION SEARCH ALGORITHM

[1]Cássio Cristani, [1]Pargles Dall'Oglio, [1,2]Marcelo Porto, [2]Sérgio Bampi, [1]Luciano Agostini.

[1] GACI, Federal University of Pelotas – UFPel
[2] Institute of Informatics, Federal University of Rio Grande do Sul - UFRGS

ABSTRACT

The efficiency of motion estimation is directly related to the used search algorithm. In this paper, a new fast motion estimation (ME) algorithm is presented. This algorithm is called Galaxy Random Search (GRS) and it uses random search as strategy to avoid local minima falls in high definition videos, with a good tradeoff between quality and computational cost. The GRS algorithm has basically two steps, the first one is the evaluation of the central region of the search area, and the second one is the random evaluation. These two steps can be easily paralyzed. Thus, it is an interesting alternative for multi-core or hardware implementations of video codecs. The developed algorithm was applied to ten HD 1080p video sequences, and the average quality results was superior to other fast ME algorithms. Compared to full search (FS) algorithm, the GRS can reduce the number of compared candidate blocks in more than 200 times.

1. INTRODUCTION

Motion estimation (ME) is the most important stage of current videos encoders. This step represents 80% of the total computational complexity of current video coders [1]. On the other hand, ME is where the video coder achieves the biggest coding gains. The ME aims to identify and to reduce the temporal redundancy between neighbor frames.

The performance of motion estimation is directly connected to the used search algorithm. The algorithms are divided in two classes: optimum and sub-optimum (fast) algorithms. The Full Search Algorithm (FS) [2] is the only one which represents the optimum quality result, since it evaluates all the candidate blocks in a search range. However, its computational cost is extremely high. The sub-optimum algorithms use some heuristics to speed up the search convergence. The use of these heuristics makes possible a large reduction in the number of compared candidate blocks (CCB), compared to FS. However, these heuristics turns fast algorithms susceptible to local minima falls. Thus, when the local minimum is not found, the quality of motion estimation may suffer great losses in relation to the optimum result. Though, current search algorithms aim to ally low cost in computational complexity and high quality.

In this paper, a new fast motion estimation algorithm, called Galaxy Random Search (GRS) is presented. This algorithm uses a random search of some candidate blocks in the search area as a strategy to avoid local minima falls. The GRS also explores important characteristics in the motion estimation such as locality (exploring the central region in the search area) and also iterative refinement (providing the convergence to more similar region).

The paper is organized as follow: the motion estimation is explained in section 2. In section 3 is presented the GRS algorithm. In this section, the quality results and computational cost of the GRS algorithm are also evaluated. The comparisons with other algorithms from literature are shown in section 4. In section 5 the conclusion are presented.

2. MOTION ESTIMATION

The video compression is based on the elimination of redundant data. The ME is responsible to find a correlation between frames, mapping the temporal redundancy in motion vectors. These vectors indicate the spatial displacement of a block in the current frame, in relation with a reference frame. For each block of the current frame a motion vector is generated, indicating the best matching position in the search area from the reference frame. The ME algorithm defines how the search for the best matching is done. The computational cost and quality results of the ME depends on directly of the ME algorithm efficiency. There are several similarity metrics to compare the block from the current frame and the candidate block in the search area. In this paper, the Sum of Absolute Differences (SAD) [3] is used.

The ME algorithm should have a good tradeoff between computational cost and quality, and this is not a simple task. The efforts to developing new ME algorithms are being a relevant theme to scientific community nowadays.

3. THE GRS ALGORITHM

The motion estimation aims to find the temporal redundancies in digital videos, reducing the volume of information, with small, or even imperceptible, losses in visual quality.

The ME algorithms should be fast and presents low computational complexity. The Galaxy Random Search algorithm (GRS) uses the randomness as way to avoid local minima falls. The random search of candidate blocks in the search area is a strategy to transpose SAD peaks, which is an insurmountable barrier only with a traditional iterative refinement. However, the random

search applied in an isolate way does not consider several important characteristics of motion estimation. The GRS algorithm does an iterative search in the central region of the search window, which guarantees good results when the video presents low motion activity, where good vectors are found next to the center region.

Figure 1 presents a flowchart of GRS algorithm. In this flowchart, both steps are made sequentially. However, they can be made in parallel, since there are no data dependences between them. That way, a hardware implementation can explore this characteristic to increase the global performance of the video encoder.

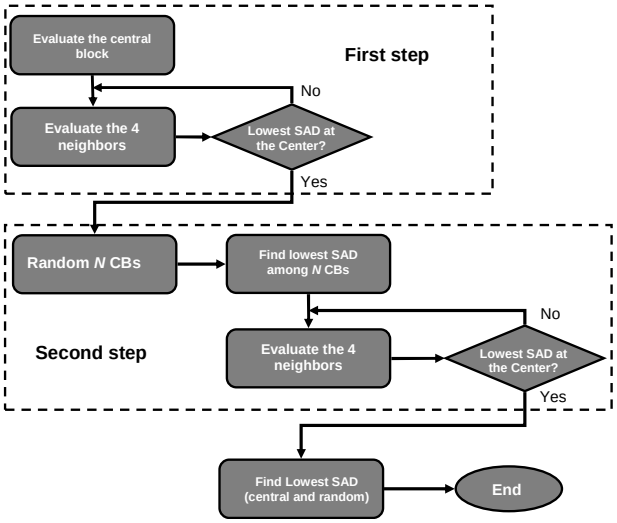

Figure 1 - Flowchart of GRS algorithm

In the first step the algorithm explores the central region of the search window. Candidate blocks from the center of the search window tend to present good results of similarity. It happens, because considering the current block, the candidate block located in the same spatial position of the reference frame, tends to represent the same frame region. The GRS algorithm explores the central block, and its four neighbors as well. The SAD calculation for these blocks is done and if the lowest SAD is found at the center, the search is finished. When the lowest SAD is found in one of the four neighbors, an iterative refinement is started, considering the lowest SAD position as the new center, and evaluating its three neighbors. This iterative refinement is repeated until the lowest SAD is found at the center. The iterative refinement gives the GRS algorithms the possibility to converge to regions which present smaller SAD in the search window.

The refinement used in the GRS is the same refinement used in the final step of Diamond Search (DS) algorithm [4], called Small Diamond Search Pattern (SDSP) [4]. Nevertheless, this approach, used in an isolate way, presents great susceptibility to local minima falls.

In the second step, the GRS algorithm evaluates N random candidate blocks in the search window. This stage uses a random function to choose the N candidate blocks. This strategy aims to increase the probability of avoiding local minima falls. The N candidate blocks are evaluated, and the one with the lowest SAD is chosen. This region is explored by a final iterative refinement which tries to identify blocks with fewer residues. This refinement is identical to that one applied in the center, in the first step.

Finally, a comparison is made between the best blocks from the first and second steps. From those, it is selected the one which presents the best similarity, and the motion vector is generated for this candidate block.

Figure 2 presents a search example of the GRS algorithm, with N=6. In this figure it is possible to observe both steps presented in this algorithm.

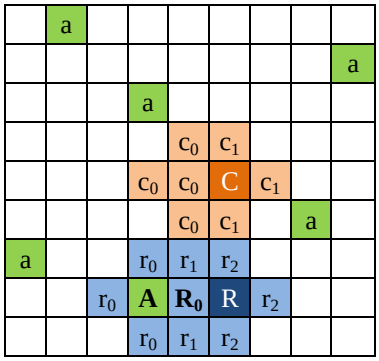

Figure 2 – The GRS algorithm with N=6.

In the first step, the GRS does the evaluation in the center of the search window, indicating by the blocks “c” in Figure 2, highlighted in orange. The central block and their neighbors (blocks “c_0” in Figure 2) are compared. The iteration begins due to the fact that the right neighbor presents the lowest SAD. The next three neighbors (blocks c1 in Figure 2) are evaluated, and the process stops because the center presents the lowest SAD result. Therefore, the block indentified with “C” is the best block obtained from the first step.

In the second step, the GRS algorithm evaluates N random candidate blocks (N=6 in this example), marked with “a” in Figure 2, and highlighted in green. The candidate block identified with “A” in Figure 2 is the block that presented the smaller SAD among the six candidate blocks which were evaluated. This block is used as center for the final refinement. This final refinement is highlighted in blue in Figure 2. In the first refinement stage, four neighbors are evaluated (blocks “r_0” in Figure 2). The iterative process begins because the right neighbor (block “R_0” in Figure 2) has the lowest SAD result. Therefore, block “R_0” becomes the new center and the algorithm iterates with the evaluation of their three neighbors (blocks “r_1” in figure 2). Again, the best result was also found in right neighbor. In the second iteration, after comparing the center and their neighbors (blocks “r_2”in Figure 2), the best result in the center is found (block “R” in Figure 2), and the searching ends. This central block is the best block found in the second step.

The last stage of GRS is the identification of the best block among the central evaluation result (block "C") and the final refinement (block "R"). The block with smaller SAD is the block that will be used, and for this block a motion vector will be generated.

One characteristic from GRS algorithm is the possibility of the generation of different motion vectors for different executions in the same video. This occurs due to the random step, which can choose different candidate blocks to a new execution on the same search window. However, one evaluation from the average deviation showed that this variation is too small, about to be despised.

3.1. Quality Evaluation

The GRS algorithm was implemented in C language and evaluated in software. These evaluations were made to the first two hundred frames in ten 1080p videos samples [5]. The chosen videos were: blue_sky, man_in_car, pedestrian_area, rolling_tomatoes, rush_hour, sun_flower, tractor, traffic, station2 and riverbed.

A determinant factor in GRS quality is the number of random candidate blocks (N). Greater values of N tend to generate better quality results, since a higher number of candidate blocks will be evaluated.

Figure 3 presents a graphic with quality results from GRS algorithm, considering the variation of the N value and the search window size. In this figure, the block size was 16x16 pixels. The search area varies in the horizontal axis and the quality (PSNR) varies in the vertical axis. Analyzing the graphic, it is possible to realize that as the number of random candidate blocks increases, the quality is also increased, considering a specific search window. The quality increasing observed between curves is practically proportional. That way, considering only the quality results, the best result, for a specific search area, will be achieved from the biggest N value.

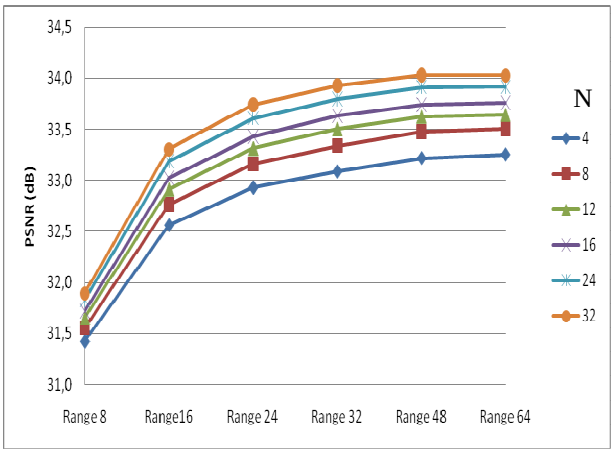

Figure 3 – GRS quality evaluation for different search ranges and N values.

The range defines how many pixels for each side of a current block belong to the search window. In the graphic presented in Figure 3 it is possible to observe that for a fixed N value, the quality increases according to the increasing of the search range. This happens because when the search area is increased, the algorithm has new regions in the frame to search better candidate blocks. However, the quality gains tend to stabilize for a range of [-48, +48] or higher. From ranges equal or higher than [-48, +48], the algorithm does not use the regions far from the center, especially by the iterative processes satisfying the stop condition before the end of the given search window.

3.2. Computational Cost Evaluation

The impacts, caused by the variation of the N values and the search window size can also be perceptible through computational cost evaluation. Table 1 shows the quality results and the computational cost for the GRS algorithm, considering the variation of the N value. The quality increase is measured according to the achieved PSNR and the computational cost is measured in number of compared candidate blocks (CCB).

The results presented in Table 1 considers a search range of [-64, +64]. The difference in PSNR between the smallest and the greatest N value is 0.77 dB. The number of CCBs grows for higher N values. The increasing is about 6% among N=4, 8, 12 and 16. However, for higher N values the increasing is about 13%. The best tradeoff, between quality and computational complexity, is obtained for N =16. In this case, the quality gain upon N = 4 is about 0.5 dB, with an increasing in the number of CCBs of 34% (73.2x106 CCBs).

Table 1 – Evaluation of the number of N influence in PSNR and CCBs

N Value	PSNR (dB)	# CCBs (x10^6)
4	33.26	52.80
8	+0.25	+6.97
12	+0.39	+13.73
16	+0.50	+20.40
24	+0.66	+33.63
32	+0.77	+46.78

Table 2 shows the average quality and computational costs results considering the search window range variation for N=16. The GRS algorithm presents a small increasing in the number of candidate blocks with the growing of the search window range. However, as cited before, the quality gains tend to stabilizes for a search range of [-48, +48]. The PSNR gain is only 0.01dB comparing the ranges [-48, +48] and [-64, +64].

Table 2 – Evaluation of the search range influence in PSNR and CCBs

Search Range	PSNR (dB)	# CCBs (x10^6)
8	31.84	76.87
16	+1.35	+6.51
24	+1.77	+8.45
32	+1.96	+9.15
48	+2.07	+9.51
64	+2.08	+9.56

The use of a higher search window implies in an additional cost of memory. In this case, the increase in the range from, 48 to 64 pixels does not bring a significant improving in quality. The surplus charge to memory is avoided and the quality loss are practically null when a range of [-48, +48] is used. This range is used to be considered the one with the best tradeoff between quality and computational cost. Considering the range [-48, +48], the GRS algorithm achieves a PSNR gain of 2.07 dB in comparison to the [-8, +8] range. The number of CCBs presents an increase of 12%.

4. COMPARATIVE RESULTS

The GRS algorithm was compared to some well know ME algorithms: Full Search (FS), Diamond Search (DS), Three Step Search (TSS) [6] and Four Step Search (FSS) [7]. Table 3 presents the average PSNR, number of compared candidate blocks (CCBs) by each algorithm, considering the search range of [-48, +48]. The GRS algorithm can improve the quality results obtained by the other evaluated fast algorithms. The GRS algorithm achieves a quality gain of 0.72dB in relation to DS algorithm. Compared to FSS, the gain is about 1.34 dB, and 1.8 dB in comparison to TSS algorithm. This quality gain is related to the efficiency to avoid local minima falls of the GRS algorithms, especially in high resolution videos.

Analyzing the computational cost, the GRS algorithm can reduce about 200 times the number of CCBs, compared to FS algorithm. However, compared to the other fast algorithms, the GRS algorithm presents an increase in the number of CCBs. The GRS algorithm presents two times more CCBs than DS algorithm. When compared to FSS, the number of CCBs grows about 26%, and 68% in relation to TSS algorithm.

Table 3 – Comparison among GRS, FS, DS, FSS and TSS algorithms.

Algorithm	PSNR (dB)	# CCBs (x10^6)
FS	35.89	14662.60
GRS	33.74	73.16
DS	33.02	48.07
FSS	32.40	58.03
TSS	30.94	43.51

Even presenting higher number of CCBs than the other fast algorithms, the GRS algorithm achieves better quality than all other algorithms and it can reach a similar or better processing rate, since both steps of the GRS algorithm can be done in parallel (there are no data dependences between these steps).

5. CONCLUSIONS

This paper presented a new motion estimation algorithm called Galaxy Random Search (GRS). This algorithm uses a random search as strategy to avoid local minima falls. GRS also explores two important characteristics in the motion estimation process: (1) the exploration of the central region of the search window since this exploration helps to achieve good results for low motion activity videos, where good candidate blocks are frequently found close to the center; and (2) the use of a final iterative refinement, giving the GRS algorithms the possibility to converge to better similarity areas.

The GRS algorithm was evaluated with ten HD 1080p video sequences. The number of random blocks and the size of search area used by the GRS algorithm were explored. The search range with the best tradeoff between quality results and computational cost is [-48, +48] pixels. The GRS algorithm presented a better result in terms of quality, among all compared fast algorithms. The PSNR gains compared to the DS algorithm is about 0.72 dB. The number of CCBs of the GRS algorithms is higher than all other fast algorithms, but it is 200 times lower than the CCBs of FS. However, the processing rate of the GRS algorithm can be equal or even superior to those algorithms, due to the possibility of parallelization of its two steps, since there are no data dependences between then.

6. REFERENCES

[1] GONZALES, R, WOODS, R, 2004. Digital Image Processing using MATLAB, Prentice Hall, Upper Saddle River, NJ, 2004.

[2] PURI, A. et al. Video Coding Using the H.264/MPEG-4 AVC Compression Standard. **Elsevier Signal Processing: Image Communication,** [S.l.], n. 19, p.793–849, 2004.

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

[4] Zhu, S., MA, K., A New Diamond Search Algorithm for Fast Block-Matching Motion Estimation. In: *IEEE Transactions on Image Processing*, Vol. 9, No. 2, 2000, pp. 287-290.

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

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

[7] Tasdizen, O., 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.