# On New Sorting-Based Lossless Motion Estimation Algorithms

D. QUAGLIA, M. PERGA, B. MONTRUCCHIO, P. MONTUSCHI

Dipartimento di Automatica e Informatica
Politecnico di Torino
Corso Duca degli Abruzzi, 24 - I-10129 Torino
ITALY

*Abstract:* - Block motion estimation represents a cpu-intensive task in video encoding and many fast algorithms have been developed to improve the searching and matching phases. A milestone within the lossless approach is partial distortion elimination (PDE/SpiralPDE) in which distortion is the difference between the block to be coded and the candidate prediction block. In this paper we show that contributions to distortion can be reliably estimated using the Taylor series expansion. The approximation method is then used to derive eight new PDE–based algorithms in which the matching order depends on the magnitude of the estimated distortion contributions. Exhaustive comparisons using several, widely different, video sequences show that these algorithms reduce the total encoding time by up to 20% with respect to SpiralPDE, while the computation for motion estimation is reduced by about 30%. The proposed algorithms are also compared with other PDE–based lossless approaches known in literature and there is a significant gain over all of them.

*Key-Words:* - Full search, lossless motion estimation, fast block matching

## 1 Introduction

In video encoding, motion compensation is used to improve the efficiency of the prediction from past or future frames. Motion estimation is the process of evaluating similarities between adjacent frames [1–3]. The class of "block–matching algorithms" is the most frequently used, especially in coding schemes based on the discrete cosine transform (i.e., MPEG-1 [4], MPEG-2 [5], MPEG-4 [6], H.261 [7], H.263 [8], and H.264 [9]). For each block being coded in the current frame, block–matching methods find the most similar area in the frame used for prediction. The displacement between these two blocks is the motion vector for all pixels of the given coded block. The process of motion estimation consists of two main sub–tasks: 1) for each block being coded, finding the most similar candidate block in the frame used for prediction (*searching process*), and 2) for each candidate block, measuring the match between the two blocks (*matching process*), e.g., with the sum of absolute differences (SAD). The matching process is nested into the searching process.

The most accurate block matching method is the Full Search that compares the block being coded with every possible candidate block in the search window; Full Search motion estimation requires up to 70% of the encoding time, thus negatively affecting the performance of hardware and software encoders. A well–known class of techniques to speed–up the matching process is named Partial Distortion Elimination (PDE) [10]: since the SAD between two blocks is obtained by progressively adding partial pixel–to–pixel differences, then a block comparison can be interrupted if the partial sum becomes greater than the minimum value of SAD already found for another candidate block in the search window. In the PDE approach, the order in which blocks are tested during the searching process, affects the speed of the whole estimation; in fact, if a good prediction is found early, then many more successive tests have a tighter distortion bound and may be quickly skipped. For example, SpiralPDE [11] improves the basic PDE algorithm using a spiral outward trajectory starting from the center of the search window according to the statistical distribution of the optimum motion vectors.

In the PDE approach, performance can be significantly improved if the pixels providing the largest contributions to SAD are checked first, and this can be best accomplished by estimating the actual value of pixels and their contribution to SAD. In [12] four different estimation algorithms are proposed. The first adapts the checking order in $16 \times 16$ blocks according to the pixel–level gradient magnitude, the second applies a top–to–bottom scan of each of the $8 \times 8$ blocks based on the order of their mean gradient magnitude, the third uses an adaptive matching scan of $8 \times 8$ blocks based on the order of gradient magnitude, and the fourth (P4) is a top-to-bottom scan of $4 \times 4$ blocks based on the order of their mean gradient magnitude. P4 gets the best results. The selection of the pixel order during the matching phase is also the basis of later works [13, 14]. In particular, in [14] pixels are checked according to the value

of the gradient computed along to the so–called Hilbert path.

In this paper we show that contributions to SAD can be reliably estimated from local information using the Taylor series expansion. Then we derive ten methods for SAD estimation and we compare their performance on several, widely different, test sequences. This work extends [15] since it provides an exhaustive comparison of the algorithms derived from the Taylor series expansion. Among the techniques, FFSSG and FFSSDG reduce the total encoding time by up to 20% with respect to a reference lossless method, the Spiral PDE [11], while the computation for motion estimation is reduced by about 30%. The proposed algorithms are also compared with other PDE–based lossless approaches known in literature and there is a significant gain over all of them.

The paper is organized as follows. Section 2 theoretically analyzes SAD behavior using the Taylor series expansion and describes ten estimation methods. Section 3 reports the experimental results for the proposed algorithms. Finally, conclusions are drawn in Section 4.

# 2 Analysis and estimation of the SAD

Block matching methods find the most similar candidate block in the frame used for prediction; to measure the match between the two blocks the most frequently used criterion is the sum of absolute differences (SAD) which is traditionally defined as

$$SAD(p,v) = \sum_{i=1}^{N}\sum_{j=1}^{N} |f_t^p(i,j) - f_\tau^{p+v}(i,j)| \quad (1)$$

where $p$ is the position of the macroblock being coded, $v$ is the candidate motion vector, $N$ is the block width/height, $f_t^p(i,j)$ is the luminance intensity of the pixel $(i,j)$ in the block with position $p$ in the frame at time $t$ and $f_\tau^{p+v}(i,j)$ is the luminance intensity of the pixel $(i,j)$ in the candidate prediction area situated at position $p+v$ in the frame at time $\tau$.

The PDE approach uses the partial sum of differences to eliminate impossible candidates before the complete calculation of the SAD. As shown in (2), the partial sum of differences $SAD_k(p,v)$ is computed until it becomes equal to or greater than the minimum SAD already found (with another candidate vector).

$$SAD_k(p,v) = \sum_{i=1}^{k}\sum_{j=1}^{N} |f_t^p(i,j) - f_\tau^{p+v}(i,j)| \geq SAD_{min}(p) \quad (2)$$

In particular, in (2) matching process is performed row by row and the test is performed after every row. If this condition becomes true for $k \leq N$, the candidate vector $v$ is rejected without further computations.

The order in which pixels within a block are picked up to compute the SAD, affects the speed of the motion estimation; in fact, if the highest contributions to SAD are found early, then the distortion bound may be reached after a small number of differences and the partial sum can be stopped.

Equation (1) can be rewritten as (3) in which the top-to-bottom row-by-row matching order is replaced by the generic order $S$ (being a summation, SAD does not change with this order). In (3) and in the following equations we omit $(i,j)$ after $f_t^p$ since $S(k)$ provides the same meaning.

$$SAD(p,v) = \sum_{k=1}^{N\times N} |f_t^p(S(k)) - f_\tau^{p+v}(S(k))| \quad (3)$$

The PDE approach shown in (2) can be rewritten as in (4) in which $m$ is the number of differences needed to reach $SAD_{min}$.

$$\begin{aligned} SAD_m(p,v) &= \sum_{k=1}^{m\leq N\times N} |f_t^p(S(k)) - f_\tau^{p+v}(S(k))| \\ &\geq SAD_{min}(p) \end{aligned} \quad (4)$$

In this context, matching optimization consists in finding a matching order $S$ such that the highest contributions to SAD are checked first and, therefore, the average number of differences needed to reach $SAD_{min}$ is kept small. SpiralPDE can be considered the simplest version of this approach since it uses a top-to-bottom row-by-row scan as shown in (2); for this reason we use SpiralPDE as the comparison term of our algorithms. To further improve matching, it is necessary to have much more information about the pixel values of the candidate blocks; in this work we detail the relationship between the pixel values in a block and those in the adjacent blocks (i.e., for other candidate vectors).

## 2.1 Taylor series expansion of the distortion

Using the notation of (1) let $d(v,i,j)$ be the distortion function, which depends on the candidate vector and the pixel (i,j) as follows:

$$d(v,i,j) = |f_t^p(i,j) - f_\tau^{p+v}(i,j)| \quad (5)$$

We are interested in estimating the behavior of $d(v,i,j)$ with different values of the candidate vector $v$. If the value of $d$ is known for a given vector (e.g., for the null vector 0), then we can approximate its value in the neighborhood by means of the Taylor series expansion reported in (6).

$$d(v,i,j) \cong d(0,i,j) + \nabla_v d(0,i,j)\cdot(v-0) \quad (6)$$

Equation (6) suggests that the magnitude of the differences in the center of the search window together with their

**Table 1**. The set of algorithms derived from Taylor series expansion.

| Name | Approximation used to evaluate $d(v,i,j)$ |
|---|---|
| FFSSL | $f_t^p(i,j)$ |
| FFSSD | $d(0,i,j)$ |
| FFSSG | $\nabla_v f_t^p(i,j)\cdot(v-0)$ |
| FFSSGoD | $\nabla_v d(0,i,j)\cdot(v-0)$ |
| FFSSGL | $f_t^p(i,j)+\nabla_v f_t^p(i,j)\cdot(v-0)$ |
| FFSSDGoD | $d(0,i,j)+\nabla_v d(0,i,j)\cdot(v-0)$ |
| FFSSDG | $d(0,i,j)+\nabla_v f_t^p(i,j)\cdot(v-0)$ |
| FFSSGoDL | $f_t^p(i,j)+\nabla_v d(0,i,j)\cdot(v-0)$ |
| FFSSDL | $f_t^p(i,j)+d(0,i,j)$ |
| FFSSGGoD | $\nabla_v f_t^p(i,j)\cdot(v-0)+\nabla_v d(0,i,j)\cdot(v-0)$ |

differential term can provide some information about pixel differences for other candidate vectors (e.g., vector $v$). For example, given two points at position $(i',j')$ and $(i'',j'')$ (in the coordinate system of the block) we have that

$$d(v,i',j') \cong d(0,i',j') + \nabla_v d(0,i',j')\cdot(v-0) \quad (7)$$

and

$$d(v,i'',j'') \cong d(0,i'',j'') + \nabla_v d(0,i'',j'')\cdot(v-0) \quad (8)$$

Since we aim at finding the matching order in which the highest contributions to SAD are checked first, we have to estimate the greater between $d(v,i',j')$ and $d(v,i'',j'')$ through the behavior of this function for other candidate vectors. In particular, if in the center of the search window

$$d(0,i',j') > d(0,i'',j'') \quad (9)$$

and

$$\nabla_v d(0,i',j') > \nabla_v d(0,i'',j'') \quad (10)$$

then, according to (7) and (8), we assume that

$$d(v,i',j') > d(v,i'',j'') \quad (11)$$

In other words, the matching order that minimizes the number of computations in the center of the search window may also be effective for other candidate vectors.

Considering (5), we further approximate (6) through the Taylor series expansion of the luminance $f_t^p(i,j)$ as follows

$$d(v,i,j) \cong f_t^p(i,j) + \nabla_v f_t^p(i,j)\cdot(v-0) \quad (12)$$

## 2.2 The proposed algorithms

We have considered various ways of approximate $d(v,i,j)$ by combining the terms of (6) and (12). The name of each algorithm and the approximation used are reported in Table 1. Luminance values and differences are considered for all pixels in the center of the search window and positions are sorted in decreasing order of the value of the estimate.

**Table 2**. Video sequences used in tests.

| Seq. N. | Name | Frames | Type |
|---|---|---|---|
| 1 | Carphone | 382 | QCIF |
| 2 | Claire | 494 | QCIF |
| 3 | Container | 300 | QCIF |
| 4 | Foreman | 300 | QCIF |
| 5 | Glasgow | 749 | QCIF |
| 6 | Grandmother | 869 | QCIF |
| 7 | Miss America | 150 | QCIF |
| 8 | Mother & Daughter | 960 | QCIF |
| 9 | News | 300 | QCIF |
| 10 | Salesman | 448 | QCIF |
| 11 | Silent | 300 | QCIF |
| 12 | Suzie | 150 | QCIF |
| 13 | Trevor | 150 | QCIF |
| 14 | Bream 0 | 300 | CIF |
| 15 | Bream 1 | 300 | CIF |
| 16 | Foreman | 299 | CIF |
| 17 | Hall | 300 | CIF |
| 18 | Bus | 150 | CIF |
| 19 | Flower Garden | 250 | CIF |
| 20 | Mobile & Calendar | 250 | CIF |
| 21 | Tempete | 250 | CIF |
| 22 | Mobile & Calendar | 80 | CCIR-601 |
| 23 | Table Tennis | 82 | CCIR-601 |

The sequence of sorted positions is then used as the matching order for the other candidate vectors of the search window (which are tested in spiral order as in SpiralPDE). We use $16\times16$ blocks and comparisons with $SAD_{min}(p)$ are performed every eight positions.

The proposed algorithms require a sorting phase in which a vector of 256 positions must be sorted by the value of the distortion estimate which is in the range (0,255). This fixed range allows the use of a fast sorting technique with linear complexity based on *counting sort* [16]; according to this technique each key value is used as the address in a sparse vector which is then compacted using fast copying routines.

# 3 Experimental results

The proposed algorithms have been compared with other methods using 23, widely different, standard sequences. In Table 2 for each of them are reported an identification number (used in the Figures), the commonly–used name, the number of frames and the image size (QCIF, CIF, CCIR-601). Chosen sequences cover several motion possibilities, ranging from slow motion (e.g., Claire and Grandmother) to large motion (e.g., Foreman). For some sequences tests have been done using more than one image size.

Tests have been performed using a modified version of

the standard MPEG-2 encoder [10] with search window $15 \times 10$ for QCIF and $30 \times 20$ for both CIF and CCIR-601. The size of the search window is the same for both P and B pictures.

The performance of the proposed algorithms are compared with SpiralPDE, which is a traditional PDE approach, and P4 [12] which achieves the best results among sorting–based algorithms known in literature.
Simulation results are reported using two metrics:

- the mean number of checked pixels per block used to compute the partial distortion. In particular, in Fig. 2 and 5 the number of pixels is expressed as the fraction of checked pixels with respect to SpiralPDE;
- total cpu time needed to encode the sequence. Since encoding time varies widely with the sequence, we report the fraction of encoding time with respect to SpiralPDE.

While the mean number of checked pixel is a measure for the motion estimation phase only, total encoding time reflects the actual gain achieved by the algorithm in a not optimized implementation. In fact, both the proposed algorithms and P4 require additional computational effort with respect to SpiralPDE because of the gradient evaluation (not present in all the algorithms) and the sorting phase. In all the Figures, results are reported as a function of the sequence number defined in Table 2; to provide a reference point for all the Figures, results about FFSSG are always reported (thick line).

First, we show results for the algorithms using only single terms of (6) and (12), i.e., FFSSL, FFSSG, FFSSD, and FFSSGoD (Fig. 1, 2, and 3). Fig. 1 reports the mean number of checked pixels per block for FFSSG, FFSSL, FFSSGoD, and FFSSD compared with P4 and SpiralPDE. Fig. 2 reports the reduction of the number of checked pixels with respect to SpiralPDE for FFSSG, FFSSL, FFSSGoD, FFSSD, and P4. Fig. 3 reports the reduction of the encoding time with respect to SpiralPDE for FFSSG, FFSSL, FFSSGoD, FFSSD, and P4.

Then, we have considered all the other algorithms obtained combining in all ways two terms of (6) and (12). Such algorithms are FFSSGL, FFSSDGoD, FFSSDG, FFSSGoDL, FFSSDL, and FFSSGGoD (Fig. 4, 5, and 6). Fig. 4 reports the mean number of checked pixels per block for FFSSG, FFSSGL, FFSSDGoD, FFSSDG, FFSSGoDL, FFSSDL and FFSSGGoD algorithms. Fig. 5 reports the reduction of the number of checked pixels with respect to SpiralPDE for FFSSG, FFSSGL, FFSSDGoD, FFSSDG, FFSSGoDL, FFSSDL, and FFSSGGoD. Fig. 6 reports the reduction of the encoding time for each sequence with respect to SpiralPDE for FFSSG, FFSSGL, FFSSDGoD, FFSSDG, FFSSGoDL, FFSSDL, and FFSSGGoD.

The exhaustive comparison of the algorithms based on Taylor series expansion shows that FFSSG provides the best results reducing the total encoding time by up to 20% with respect to SpiralPDE, while the computation for motion estimation is reduced by about 30%. The comparison between FFSSG and FFSSGoD experimentally shows the validity of the approximation of the gradient of distortion with the gradient of luminance; this clarifies the use of such approximation in [15].

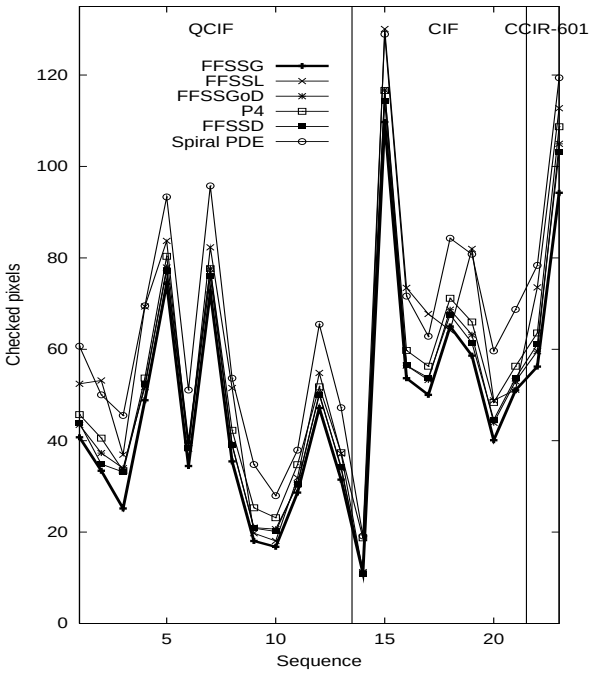


**Fig. 1**. Mean number of checked pixels per block as a function of the sequence. FFSSG, FFSSL, FFSSGoD, and FFSSD are compared with P4 and SpiralPDE.

# 4 Conclusions and future works

We have presented a general method to estimate SAD contributions in block–matching motion estimation algorithms, based on the Taylor series expansion. The approximation method has been used to derive ten PDE–based algorithms (eight new) in which matching order depends on the magnitude of the estimated SAD contributions. Using several, widely different, video sequences, we have compared the performance of the proposed algorithms with SpiralPDE, which is a traditional PDE approach, and state–of–the–art sorting–based algorithms known in literature. Experimental results show that FFSSG and FFSSDG reduce the total encoding time by up to 20% with respect to SpiralPDE, while the computation for motion estimation is reduced by about 30%. Future work will aim to apply the proposed approximation method to the even more demanding motion compensation scheme of H.264.

*References:*

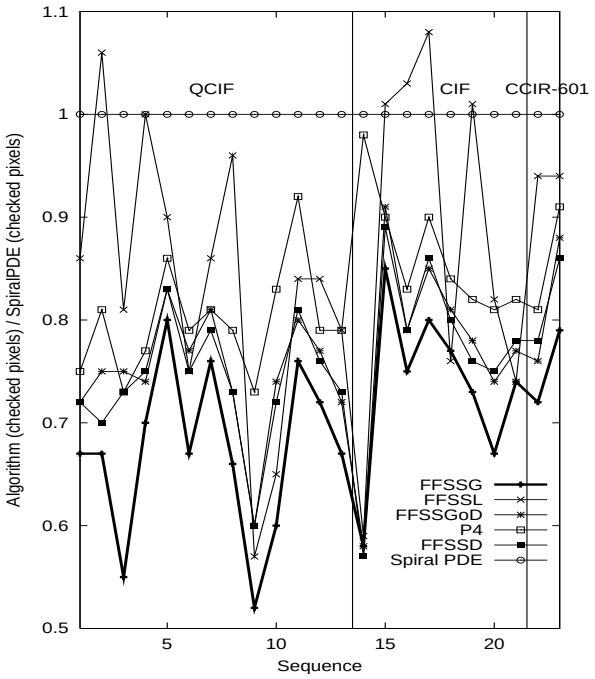


**Fig. 2**. Reduction of the number of checked pixels with respect to SpiralPDE as a function of the sequence. FFSSG, FFSSL, FFSSGoD, and FFSSD are compared with P4.

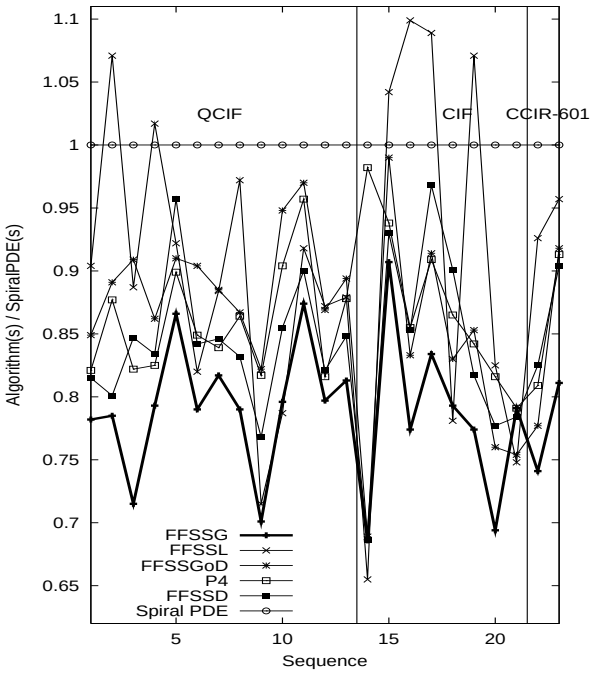


**Fig. 3**. Reduction of the encoding time with respect to SpiralPDE as a function of the sequence. FFSSG, FFSSL, FFSSGoD, and FFSSD are compared with P4.


[1] J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images - a review," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 917–935, August 1988.

[2] Frédéric Dufaux and Fabrice Moscheni, "Motion estimation techniques for digital TV: A review and a new contribution," *Proceedings of the IEEE*, vol. 83, no. 6, pp. 858–876, June 1995.

[3] Christoph Stiller and Janusz Konrad, "Estimating motion in image sequences," *IEEE Signal Processing Mag.*, vol. 16, no. 4, pp. 70–91, July 1999.

[4] ISO/IEC, "MPEG-1 coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mb/s," *ISO/IEC 11172*, 1993.

[5] ISO/IEC, "MPEG-2 generic coding of moving pictures and associated audio information," *ISO/IEC 13818*, 1996.

[6] ISO/IEC, "MPEG-4 - information technology - coding of audio-visual objects - part 2: Visual," *ISO/IEC 14496-2*, 2000.

[7] International Telecommunications Union, "Video codec for audiovisual services at p×64 kbits," *ITU-T Rec-ommendation H.261*, 1993.

[8] International Telecommunications Union, "Video coding for low bitrate communication," *ITU-T Recommendation H.263*, 1998.

[9] ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC, "Advanced video coding for generic audiovisual services," *ITU-T*, May 2003.

[10] S. Eckart and C. Fogg, "ISO/IEC MPEG-2 software video codec," *Proc. SPIE*, vol. 2419, pp. 100–118, 1995.

[11] Digital Video Coding Group, *ITU-T recommendation H.263 software implementation*, Telenor R&D, 1995.

[12] Jong Nam Kim and Tae Sun Choi, "Adaptive matching scan algorithm based on gradient magnitude for fast full search in motion estimation," *IEEE Trans. Consumer Electron.*, vol. 45, no. 3, pp. 762–772, August 1999.

[13] W.G. Hong and T.M. Oh, "Sorting-based partial distortion search algorithm for motion estimation," *Electronics Letters*, vol. 40, no. 2, pp. 113–115, 22nd January 2004.

[14] Yui-Lam Chan and Wan-Chi Siu, "An adaptive partial distortion search for block motion estimation," in *ICASSP 2003*, 6–10 Apr. 2003, pp. 153–156.

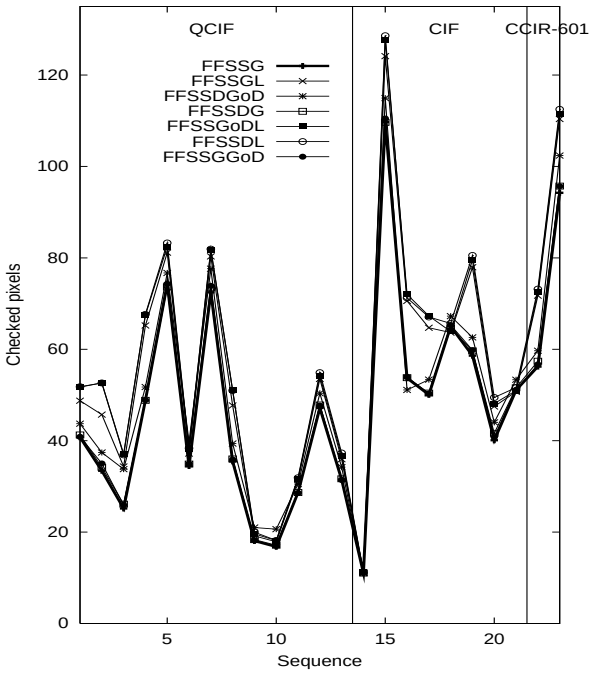


**Fig. 4**. Mean number of checked pixels per block as a function of the sequence. FFSSG (thick line) is reported as reference.

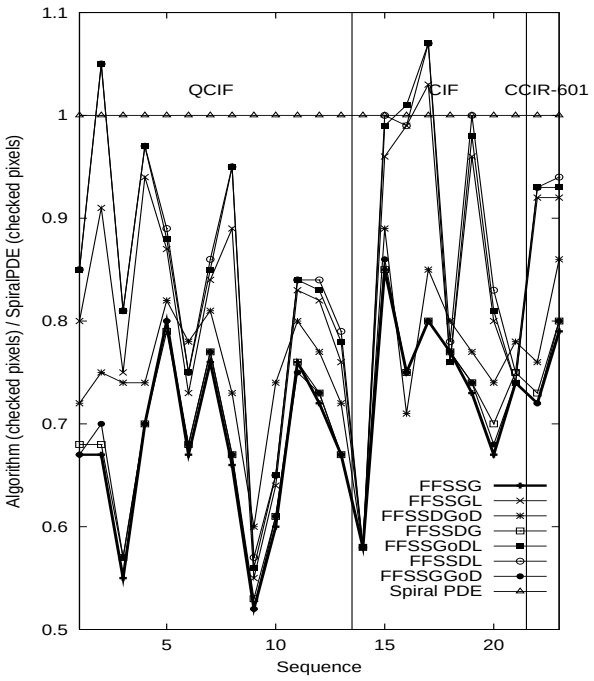


**Fig. 5**. Reduction of the number of checked pixels with respect to SpiralPDE as a function of the sequence. FFSSG (thick line) is reported as reference.

[15] B. Montrucchio and D. Quaglia, "New sorting–based lossless motion estimation algorithms and a partial distortion elimination performance analysis," *to appear in IEEE Trans. Circuits Syst. Video Technol.*, 2004 - preprint available at http://multimedia.polito.it/montrucchio-quaglia-FFSSDG.html.

[16] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest, *Introduction to Algorithms*, MIT Press and McGraw-Hill, 1990.

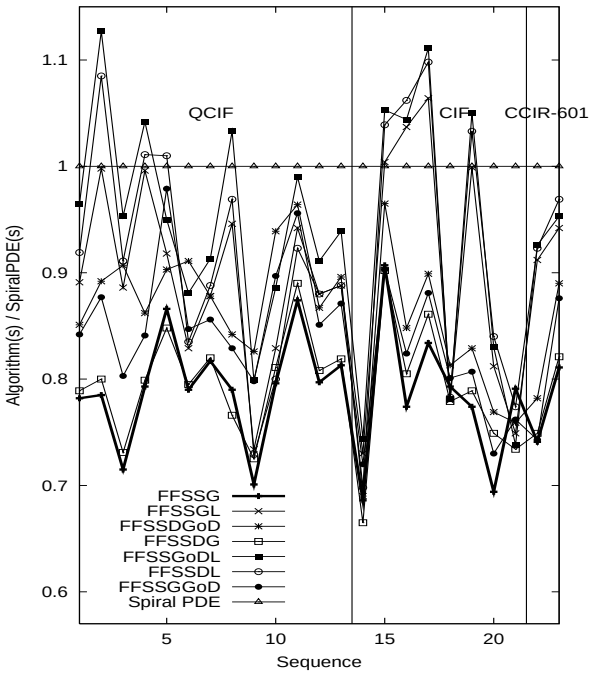


**Fig. 6**. Reduction of the encoding time with respect to SpiralPDE as a function of the sequence. FFSSG (thick line) is reported as reference.