# New Sorting-Based Lossless Motion Estimation Algorithms and a Partial Distortion Elimination Performance Analysis

Bartolomeo Montrucchio, *Member, IEEE,* and Davide Quaglia, *Member, IEEE*

***Abstract*—In video encoding, block motion estimation represents a CPU-intensive task. For this reason, many fast algorithms have been developed to improve 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, (i) we analyze distortion behavior from local information using the Taylor series expansion and show that our general analysis includes other previous similar approaches. (ii) Then, we propose two full-search (lossless), fast-matching, block motion estimation algorithms, based on the PDE idea. The proposed algorithms, called fast full search with sorting by distortion (FFSSD) and fast full search with sorting by gradient (FFSSG), sort the contributions to distortion and the gradient values, respectively, in order to quickly discard invalid blocks. Experimental results show that the proposed algorithms outperform other existing full search algorithms, reducing by up to 20% the total CPU encoding time (with respect to SpiralPDE), while the computation strictly required by the motion estimation is reduced by about 30%. (iii) Finally, we experimentally find an operational lower bound (based on standard test sequences) for the average number of checked pixels in the PDE approach, which measures the performance of the searching and matching phases. In particular, SpiralPDE achieves performances very close to the searching phase bound, while there is still a remarkable margin on the matching phase. We then show that our algorithms, aimed at improving the performances of the matching phase, achieve interesting results, significantly approaching this margin.**

***Index Terms*—Distortion Taylor expansion, fast block matching, full search, lossless motion estimation, partial distortion elimination (PDE) bounds.**

## I. Introduction

In VIDEO encoding, motion compensation is used to improve the efficiency of the prediction from past or future frames. Motion estimation (comprehensive surveys can be found in [1]–[3]) is the process of evaluating movements between adjacent frames. The so-called "block-matching algorithms" are the most important of these estimation methods, especially in coding schemes based on discrete cosine transform. Pel-recursive, frequency-domain, and gradient-based motion estimation methods are less frequently used.

Manuscript received January 15, 2002; revised November 22, 2002. This work was supported in part by the Center for Multimedia Radio Communications, Politecnico di Torino, Torino, Italy. This paper was recommended by Associate Editor S. E. Lee.

The authors are with the Dipartimento di Automatica e Informatica, Politecnico di Torino, I-10129 Torino, Italy (e-mail: bartolomeo.montrucchio@polito.it; davide.quaglia@polito.it).

Digital Object Identifier 10.1109/TCSVT.2004.841689

For each test block in the current frame, the block-matching methods find the most similar candidate block in the frame used for prediction. The displacement between these two blocks is the motion vector for all pixels of the given test block.

The most accurate block-matching method is the full search (FS) that compares every possible candidate block in the search window with the test block; in this way, it produces accurate results, but it is computationally intensive.

The block-matching motion estimation is used in well-known video encoding standards, such as MPEG-1 [4], MPEG-2 [5], MPEG-4 [6], H.261 [7], H.263 [8], and H.264 [9]. In these standards, FS motion estimation requires up to 70% of the encoding time [10], thus negatively affecting the performance of the hardware and software encoders.

For this reason, several alternative and faster techniques have been developed, based on lossy and lossless algorithms, that try to reduce computational complexity by pruning the search space with respect to the FS method or reducing the number of pixels considered during block matching between candidate and test blocks (for example, by interrupting the comparison when the partial distortion between the two blocks is larger than a certain threshold).

Many of these techniques are also feasible for hardware implementation, even if a software real-time solution is now possible [10]. Video coding is becoming an integral part of the multimedia capabilities of today's PCs and the demand for video quality increases with CPU speed. Therefore, lossless algorithms may be adopted if their performance is not too far from the performance of lossy ones.

In order to speed up the motion estimation, we have to consider the pixel-to-pixel distortion between the block to be coded and the candidate prediction block. In this paper, we analyze distortion behavior from local information using the Taylor series expansion, and we propose two fast algorithms denoted as fast full search with sorting by distortion (FFSSD) and fast full search with sorting by gradient (FFSSG). These two algorithms are based on the sorting of distortion and gradient values on a pixel-by-pixel basis. They reduce the total encoding time by up to 20% with respect to a reference lossless method, the Spiral partial distortion elimination (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. We also experimentally analyze the potentialities of the lossless PDE technique in order to find out if it can be

TABLE I
NOTATION TABLE

| | |
|---|---|
| $p$ | position of the macroblock being coded |
| $v$ | candidate motion vector |
| $N$ | block width/height |
| $f_t^p(i,j)$ | luminance intensity of the pixel $(i,j)$ in the block $p$ of frame at time $t$ |
| $W$ | search window |
| $S$ | generic matching order |

further improved; the analysis shows that the improvement obtained with FFSSG algorithm is near to half the maximum operational improvement of the PDE technique.

The actual software implementation used for our experiments is based on the MPEG-2 reference encoder [12], [11].[1]

The paper is organized as follows. Section II introduces some background. Section III theoretically analyzes distortion using the Taylor series expansion and shows how such an analysis includes other previous methods; it also describes FFSSD and FFSSG algorithms in detail. Section IV reports the experimental results for the proposed algorithms and shows the potentialities of the PDE approach exploited by FFSSD and FFSSG. Finally, conclusions and future work are drawn in Section V.

## II. BACKGROUND

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). SAD is defined as

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

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 $\tau$.

Table I reports the notation used in whole paper. $S$ is a sequence of $N^2$ numbers representing the order in which the pixels of the blocks are picked during SAD computation. $S$ will be explicitly used in Section III, while in (1) the order adopted is the raster scan (top-to-bottom, left-to-right). Block matching methods search for the smallest $\mathrm{SAD}(p,v)$ by varying $v$ within a rectangular region called search window ($W$). The FS method computes SAD for every candidate vector in $W$. This method has been improved, both by pruning the search space $W$ (*fast searching techniques*) and by reducing the complexity of SAD computation (*fast matching techniques*). Both these different improvement techniques can be lossless or lossy, in the sense that coding efficiency is respectively equal or worse with respect to FS. Lossy techniques are typically faster than lossless at the expense of coding efficiency.

This classification, even if it is not the only one possible, is feasible for the purposes of this work. In fact, the two algorithms we are presenting are fast matching, lossless (using the PDE approach) block-matching methods.

In the following, we will provide an overview of both fast searching and fast matching improvement techniques. Lossy and lossless approaches will be described for each of these two methods. Many algorithms combine more than one technique (e.g., lossy fast searching and lossy fast matching) and could be inserted into more than one category. This classification is made upon the main characteristic, where useful, secondary characteristics are reported as well.

### A. Fast Searching

*1) Lossy:* There are many lossy motion estimation algorithms which reduce the number of tested points in the search window with some degradation of the prediction performance with respect to FS. Usually these algorithms exploit some general properties of natural images to reduce computational time [13]. One of these properties is that the block motion field of real-world video sequences usually varies slowly and, therefore, motion vectors have a center-biased distribution instead of a uniform distribution. Recent examples are the new three-step search (N3SS) [14], the four-step search (4SS) [15], the simple and efficient search (SES) [16], the block-based gradient descent search (BBGDS) [17], and the one-dimensional (1-D) gradient descent search [18]. Some genetic algorithms have been developed in order to avoid local minimum points [19]–[21]. Among adaptive algorithms, the adaptive search length (ASL) [22] varies the number of tested positions in the search space. In [23], two techniques are presented: an enhancement of the logarithmic search (also using fast matching lossy subsampling) and an approach for dynamically varying the size of the search window according to $\mathrm{SAD}(p,0)$. Some approaches exploit the spatial and temporal correlation between motion vectors to find a good starting point in the search space [24]–[28]. In [29], a diamond pattern is used to encode different diamond-shaped zones in the search window, so as to interrupt the search when a threshold is reached. A fuzzy approach is used in [30], and the gray prediction theory is adopted in [31]. In [32], search space is reduced with a very low image error; a simple threshold updating mechanism allows the algorithm to be easily realized in VLSI. Finally, in [33], algorithms proposing both fast searching and fast matching are presented. Partial SAD (as shown later in Section II-B1) is used to limit the number of candidate vectors in the search space.

*2) Lossless:* Lossless fast searching, as well as previous algorithms, are based on general properties, in particular of norms of blocks. An important example is the successive elimination algorithm (SEA) [34], which reduces the number of positions in the search space, thus decreasing the number of matching

[1]Available. [Online]. http://staff.polito.it/bartolomeo.montrucchio/FFSSDG or http://multimedia.polito.it/FFSSDG/

evaluations. In the exhaustive search, the SAD is computed for each point in the search window; the candidate having the minimum SAD value at the end of the search area is the selected motion vector. In SEA, distortion values are efficiently precomputed and by using some properties of the norm it is possible to avoid SAD computation for a high number of test blocks. Only a minimum percentage of test blocks needs the SAD evaluation. In [35], multiple matching criteria are used to compare reference and test blocks. If three fast matching criteria are not satisfied, SAD computation is avoided for that candidate block. In [36], SEA is used to implement a new rate-constrained exhaustive search with reduced computational complexity. In [37], the authors show how to combine SEA with the PDE approach, both in a lossless and lossy way. In [38], SEA is improved by introducing a method for the initial motion vector selection (since the efficiency of SEA largely depends on the initial candidate motion vector), and it is also extended to MPEG-4 by using both $16 \times 16$ and $8 \times 8$ blocks. SEA is also used and improved in [39] and [40] and for long-term memory motion-compensated prediction [41].

Multilevel SEA [42] partitions a block into several subblocks, then it uses sum norms of subblocks to generate a faster decision in order to discard as many candidates as possible. MSEA is further improved in the adaptive search order (ASO) [43]; in this case, the order of blocks used in MSEA is determined by using the predicted initial motion vector. Extended SEA (ESEA) [44] introduces tighter bounds for SAD and exploits the already calculated lower bounds during the calculation of the matching criterion (ESEA also extends SEA so as to substitute the SAD with the MSE).

### B. Fast Matching

*1) Lossy:* Fast matching approaches aim at reducing SAD computation time for each test block by testing only a subset of the pixels in the block. Important work on lossy fast matching is proposed in [45] and [46]; one of the techniques consists of pixel-decimation patterns during candidate-test block comparison. In [47]–[49], subsamples are taken on the most representative pixels. In [50], the order used for scanning the block is the so-called Hilbert sequence, which is shown to be better than the raster scan order. In this way, sorting, which is required in [47], is avoided. Sorting is also used in [51] in order to carry out decimation; in fact, all of the pixels within each macroblock are sorted in order, and the darkest and the brightest pixels are chosen as representative pixels. We will see how a similar ordering problem is specifically addressed in the algorithms proposed in this paper. A two-step search algorithm (TWSS) [52] combines the reduction of the search area with subsampling in the matching phase. In particular, a pattern is used in the block, and a subset of pixels (48 instead of 256 for the experiment in the paper) is used to approximate block-matching error. Also, the search area is decimated, and, since the combination of these two decimations could lead to an excessive loss in quality, more than one pixel in the search area is chosen during block-matching comparison. In this way, these pixels can be used for the second search phase. It is important to note that the number of pixels used for subsampling the block during block matching is chosen by analyzing a test sequence which shows that a uniform subsampling of the block is meaningful.

PDE is based on a simple idea. Since total distortion (e.g., SAD) between two blocks is obtained by progressively adding partial pixel-to-pixel differences, then a block comparison can be safely interrupted if the partial distortion becomes greater than the minimum distortion already found for another candidate block in the search window. PDE will be described in detail in Section II-B2. Even though PDE is a lossless fast matching approach, it can also be used in a lossy way. In [53], partial distortion and the current minimum distortion are normalized on the number of checked pixels before comparison. This method, called normalized partial distortion search (NPDS), has been improved in [54] by introducing progressive partial distortions at a very early stage so that computations can be further reduced. In [55], SAD computation is early abandoned once it becomes clear that, given partial SAD, total SAD will exceed the minimum distortion found thus far. In that work, PDE is based on a probabilistic criterion. Once again, a uniform partition pattern for pixel decimation is used. The idea of uniform partition has also been adopted for lossless fast matching (in [56]).

*2) Lossless:* The lossless fast matching approach can be very attractive, since it can be easily combined with a lossless fast searching approach (e.g., SEA) without any loss in quality. PDE [12], as already explained in Section II-B2, is a lossless approach. In particular, given a search window, the original PDE scans the window in a raster order, line by line, from top to bottom. In the same way during block matching, SAD is computed line by line from top to bottom. The comparison with the minimum SAD already found (to early discard candidates) is performed for every row (i.e., every 16 pixels). SpiralPDE [11] scans the search window using an outward spiral order. In fact, in most cases, the optimum motion vector is in the center of the search window and its SAD, taken as the upper bound, allows a large number of candidate blocks to be discarded. This outward spiral search strategy, even if simple, is very effective, as will be shown later in Section IV-B, when an analysis of PDE bounds will be performed.

In the original SpiralPDE, the matching order is raster scan even if there are no reasons why the first lines are more representative than others. If no assumptions are made on the frame content, the best way of selecting pixels for partial SAD computation is to use a uniform strategy [56]; while (as reported in Section II-B1) many lossy algorithms use uniform grid ordering, in [56], the Sobol's pseudorandom sequence is adopted [57]. This is because, even though the most uniform order is a grid one, grid uniformity is lost if it is interrupted. Instead, according to PDE theory, the Sobol's sequence can be interrupted at any moment.

A more sophisticated approach also has to consider the frame content. This has been done by Kim *et al.* in [58]–[61], where the matching order is computed using gradient values of the block to be coded. This matching order remains the same for all candidate blocks. Gradient is used because a relationship is derived between the gradient of the reference block and the matching distortion; specifically, it turns out that matching distortion is proportional to gradient magnitude of the reference block. Gradient is computed as reported in the simplest approximation of

[62]. Later in this paper we will derive a more general relationship that includes this one as a particular case.

In [58], four different algorithms are proposed. The first uses adaptive-matching scan of $16 \times 16$ blocks based on computed gradient magnitude, the second applies a top-to-bottom scan of each of the $8 \times 8$ blocks based on the order of gradient magnitude, the third uses an adaptive-matching scan of $8 \times 8$ blocks based on the order of gradient magnitude, and the fourth is a top-to-bottom scan of $4 \times 4$ blocks based on the order of gradient magnitude. P4 (the fourth) gets the best results. In [59], the same authors propose two other algorithms. The first is an adaptive-matching scan based on representative pixels, while the second (P2, which is the better of the two) uses an adaptive-matching scan by sorted rows/columns based on representative pixels. P4 appears to be the fastest, and it will be used in our comparison as the reference algorithm for Kim–Choi's work. Block-matching properties and a review of ideas presented in their previous papers are also shown in [63], while proposed algorithms are further detailed in [61] and in [60]; however, the P4 algorithm of [58] remains the fastest.

## III. Proposed Algorithms

In the motion estimation process, for each macroblock of the frame being coded (*test macroblock*), the most similar $N \times N$ area in the reference frame is used as its prediction and the displacement (*motion vector*) is coded in the bitstream together with the DCT-transformed residual error. The prediction block is usually searched in a rectangular area centered on the position of the test macroblock (*search window*) by evaluating a function known as *matching criterion*. Since DCT will be applied to the residual error between the test macroblock and its prediction, it is essential to adopt a matching criterion that leads to the most compact DCT coefficient configuration. It was shown in [64] that the sum of absolute differences (SAD) defined in (1) is a good tradeoff between effectiveness and simplicity (refer to Table I for the notation). Therefore, the motion estimation process for the macroblock at position $p$ can be formulated as finding $v'$ such that

$$\mathrm{SAD}(p, v') = \min_{v \in W} \mathrm{SAD}(p, v). \tag{2}$$

It is important to note that (1) refers to the *matching process* while (2) refers to the *searching process*; the matching process is nested into the searching process. Many motion estimation algorithms can be classified according to the way in which $(i, j)$ varies in (1) (*matching strategy*) and $v$ varies in (2) (*searching strategy*).

The PDE approach uses the partial sum of differences to eliminate impossible candidates before the complete calculation of the SAD. As shown in

$$\mathrm{SAD}_k(p, v) = \sum_{i=1}^{k} \sum_{j=1}^{N} |f_t^p(i, j) - f_\tau^{p+v}(i, j)| \geq \mathrm{SAD}_{\min}(p) \tag{3}$$

the partial sum of differences is computed until it becomes equal to or greater than the minimum SAD already found (with another candidate vector). In particular, in (3), the matching 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 (it is the rejection condition).

The searching strategy, that is, to say, the order in which blocks are tested during the searching phase, 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 skipped. For example, SpiralPDE 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.

Also, the matching strategy, that is, to say, 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 (4) 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 (4) and in the following equations, we omit $(i, j)$ after $f_t^p$ since $S(k)$ provides the same meaning:

$$\mathrm{SAD}(p, v) = \sum_{k=1}^{N \times N} |f_t^p(S(k)) - f_\tau^{p+v}(S(k))|. \tag{4}$$

The PDE approach shown in (3) can be rewritten as in

$$\mathrm{SAD}_m(p, v) = \sum_{k=1}^{m \leq N \times N} |f_t^p(S(k)) - f_\tau^{p+v}(S(k))| \geq \mathrm{SAD}_{\min}(p) \tag{5}$$

in which $m$ is the number of differences needed to reach $\mathrm{SAD}_{\min}$. The comparison with $\mathrm{SAD}_{\min}$ can be performed every $n$ differences. In this context, matching optimization consists of 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 $\mathrm{SAD}_{\min}$ 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 (3); for this reason, we use SpiralPDE as the comparison term of our algorithms. SpiralPDE is based on the implicit assumption that the first rows of the block should contain the highest contributions to SAD. As reported in Section II-B1, many algorithms avoid this arbitrary assumption by using a uniform grid or a pseudorandom uniform order (Sobol's order in [56]), suggesting that, without any knowledge about the distortion distribution, we have to assume that the highest contributions are uniformly spread over the block. To further improve matching, it is necessary to have much more information about the pixel values of the test block; in this study, we detail the relationship between the pixel values in a block and those in the adjacent blocks (i.e., for other candidate vectors).

### A. Taylor Series Expansion of the Distortion

Using the notation of (1) and (3), let $d(v,i,j)$ be the distortion, function of the candidate vector and the pixel (i, j) defined as follows:

$$d(v,i,j) = |f_t^p(i,j) - f_\tau^{p+v}(i,j)|. \tag{6}$$

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 approximate its value in the neighborhood by means of the Taylor series expansion, which is reported in

$$d(v,i,j) \cong d(0,i,j) + \nabla_v d(0,i,j) \cdot (v-0). \tag{7}$$

According to (7), the value of $d(v,i,j)$ is estimated from the sum of $d(0,i,j)$ and the differential term obtained through the inner product (denoted as $\cdot$) between the gradient vector of $d(0,i,j)$ and the difference vector $(v-0)$.

Equation (7) suggests that the magnitude of the differences in the center of the search window together with their differential term can provide some information about pixel differences for other candidate vectors (e.g., vector $v$). In particular, given two pixels at position $(i',j')$ and $(i'',j'')$ (in the coordinate system of the block) we have that

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

$$d(v,i'',j'') \cong d(0,i'',j'') + \nabla_v d(0,i'',j'') \cdot (v-0). \tag{9}$$

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

$$d(0,i',j') > d(0,i'',j'') \text{ and } \nabla_v d(0,i',j') > \nabla_v d(0,i'',j'') \tag{10}$$

then, according to (8) and (9), we assume that

$$d(v,i',j') > d(v,i'',j''). \tag{11}$$

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.

### B. FFSSD

The first proposed algorithm, called FFSSD, uses only the first term of (7) that is the distribution of differences (distortion) between the test block and the candidate area in the center of the search window. Differences are computed for all pixels in the center of the search window and positions are sorted in decreasing order of this value. 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). The matching process in the center of the search window, that is to say the one with the null candidate vector, requires the computation of all the differences in the $N \times N$ grid as in every PDE approach, but since the value of $\mathrm{SAD}_{\min}(p)$ has to be initialized, FFSSD involves no extra difference computation with respect to SpiralPDE. In FFSSD, the comparison with $\mathrm{SAD}_{\min}(p)$ is performed every eight pixels and this choice seems a good tradeoff (see also [56]) between the cost of comparisons and the number of useless differences (when the partial SAD is already greater than $\mathrm{SAD}_{\min}$). It is worth noting that FFSSD generalizes the approach followed by Ng and Zeng [51].

### C. FFSSG

When the macroblock being coded is quite similar to the area in the center of the search window, the first term of the Taylor expansion may become quite small and, therefore, it cannot be used to determine an optimized matching order (we could conclude that this is the best vector but we are performing an exhaustive full search). In other words, if all differences in the $N \times N$ grid are null, sorting is meaningless. In this case, we have to consider the second term in (7) that is the spatial gradient of $d(0,i,j)$ in the center of the search window. Moreover, according to (7), the importance of this term increases with the magnitude of the candidate vector $v$.

The second proposed algorithm, called FFSSG, is based on this idea; instead of using the value of differences (as in FFSSD), it gets an optimized matching order by sorting positions in decreasing order of the gradient of $d(0,i,j)$. Considering (6), we further approximate the gradient of $d(0,i,j)$ through the spatial gradient of $f_t^p(i,j)$ computed as the average of the gradient along eight directions, as shown in

$$|G[f(i,j)]| = \frac{1}{8} \sum_{m=-1}^{1} \sum_{n=-1}^{1} |f(i,j) - f(i+m, j+n)|. \tag{12}$$

The resulting value is between 0 and 255. This approximation is based on some preliminary experimental results and its general validity should be verified theoretically. In FFSSG, the comparison with $\mathrm{SAD}_{\min}(p)$ is performed every eight pixels as in FFSSD. The candidate vectors are tested in spiral order as in both FFSSD and SpiralPDE.

It is worth noting that FFSSG, in a more general framework, includes the approaches followed by Chan and Siu [47], Wang *et al.* [50], and Kim and Choi [58]–[61]. In particular, FFSSG does not use the first term of the Taylor's expansion of the distortion (7), just like in Kim's work [58]–[61] (these works use the Taylor's expansion to show the fact that the matching distortion at a certain position is proportional to the gradient magnitude of reference block in the current frame). The two algorithms are similar, but Kim's work uses a local-area-based-gradient sorting, while FFSSG works on pixel-based-gradient sorting. We found that a pixel-based approach can significantly improve performance, if the sorting operation is efficiently performed. On the other side, the use of eight directions for computing gradient, and of eight pixel-checking unit (for the eight pixel checking unit, previous results are reported in [56]) gives only a little effort versus the 16 pixel-checking unit and the simpler method for gradient computation used by P4.

### D. Sorting

FFSSD and FFSSG require a sorting phase in which a vector of 256 elements must be ordered according to key values be-

TABLE II
MOTION PICTURE SEQUENCES USED DURING TESTS

| Seq. N. | Name | Frames | Type | References using the sequence (**) |
|---|---|---|---|---|
| 1 | Carphone | 382 | QCIF | [63], [36], [3], [17], [10], [61], [59], [60], [58] |
| 2 | Claire | 494 | QCIF | [50], [42], [10], [61], [59], [60], [58], [56] |
| 3 | Container | 300 | QCIF | [56] |
| 4 | Foreman | 300 | QCIF | [33], [63], [44], [42], [36], [17], [10], [23] [61], [59], [60], [58], [41], [56] |
| 5 | Glasgow | 749 | QCIF | [56] |
| 6 | Grandmother | 869 | QCIF | [33], [63], [10], [61], [59], [60], [58], [56] |
| 7 | Miss America | 150 | QCIF | [33], [42], [36], [3], [17], [14], [18], [27], [10] [26], [29], [31], [30], [52], [56] |
| 8 | Mother & Daughter | 960 | QCIF | [33], [36], [10], [41], [56] |
| 9 | News | 300 | QCIF | [44], [56] |
| 10 | Salesman | 448 | QCIF | [50], [47], [33], [39], [42], [43], [34], [35], [37] [3], [17], [14], [18], [27], [10], [26], [31], [30] [52], [49], [56] |
| 11 | Silent | 300 | QCIF | [44], [56] |
| 12 | Suzie | 150 | QCIF | [43], [36], [10], [52], [56] |
| 13 | Trevor | 150 | QCIF | [63], [10], [61], [59], [60], [58], [56] |
| 14 | Bream 0 | 300 | CIF | [56] |
| 15 | Bream 1 | 300 | CIF | [56] |
| 16 | Foreman | 299 | CIF | [33], [63], [44], [42], [36], [17], [10], [23] [61], [59], [60], [58], [41], [56] |
| 17 | Hall | 300 | CIF | |
| 18 | Bus(*) | 150 | CIF | [22] |
| 19 | Flower Garden(*) | 250 | CIF | [51], [42], [38], [35], [2], [24], [21] |
| 20 | Mobile & Calendar(*) | 250 | CIF | [33], [38], [2], [27], [26], [21], [19] |
| 21 | Tempete(*) | 250 | CIF | |
| 22 | Mobile | 80 | CCIR-601 | [55], [28], [32], [30], [31], [56] |
| 23 | Table Tennis | 82 | CCIR-601 | [50], [51], [40], [2], [15], [16], [25], [18], [27] [26], [29], [13], [22], [32], [21], [30], [31], [19] [45], [48], [49], [46], [54], [53] |

(*) sequences used for testing in the MPEG/ISO standardization process.
(**) references can use the cited sequence with different image size; a small number of references also uses other sequences, not reported here.

tween 0 and 255 (differences and gradient values, respectively). This fixed range allows the use of a fast sorting technique with linear complexity based on *counting sort* [65]; according to this technique, each key value is used as an address in a sparse vector which is then compacted using fast copying routines.

## IV. EXPERIMENTAL RESULTS AND REMARKS

### A. FFSSD and FFSSG

In order to compare the performances of the proposed algorithms with other methods, we have used 23 different standard sequences. For each of them, Table II reports an identification number, the commonly used name, the number of frames, the image size, and the referenced papers that use that sequence. It is useful to note that the total number of frames used is of several thousands (about 8000), with QCIF ($176 \times 144$), CIF ($352 \times 288$), and CCIR-601 ($702 \times 576$) formats to show the performance the algorithms with different image sizes. Some sequences have also been tested with different resolutions. The sequences marked with (*) in Table II are ISO standard sequences for medium bit-rate video coding efficiency tests. Chosen sequences cover several motion possibilities, ranging from slow motion (e.g., Claire and Grandmother) to large motion (e.g., Foreman). As can be seen in Table II, the set of used sequences is (when possible) a superset of the test sequences usually used in several other papers.

We compare the proposed FFSSD and FFSSG with SpiralPDE, which is the reference PDE algorithm with top-to-bottom raster scan matching and spiral searching [11], SPD [56] to represent full search techniques using different matching order methods, and the P4 algorithm from [58], which performs a top-to-bottom scan of $4 \times 4$-sized blocks based on the order of gradient magnitude. Algorithms based on SEA have not been reported because they can be combined to all tested algorithms (as stated in [37]), and for this reason they are not in direct competition with FFSSD and FFSSG. The P2 algorithm [59] is not reported because its results are worse than those for P4 of [58] (as explained in Section II-B2).

The tests have been performed using a modified version of the Test Model 5 MPEG-2 encoder [12], and the code is available for download, as reported in Section I.

In our experiments, the block size is $16 \times 16$ pixels, while the width/height of the search area is $15 \times 10$ for QCIF and $30 \times 20$ for both CIF and CCIR-601 image formats. The search width/height is the same for both P and B pictures. SAD [see (1)] is used as the criterion for block matching.

Simulation results (all tested algorithms have been implemented) are reported in two different ways:

- checked pixels per block, i.e., the average number of pixels per block used to compute the partial distortion;
- total CPU time needed to encode the whole sequence with the modified MPEG-2 encoder.

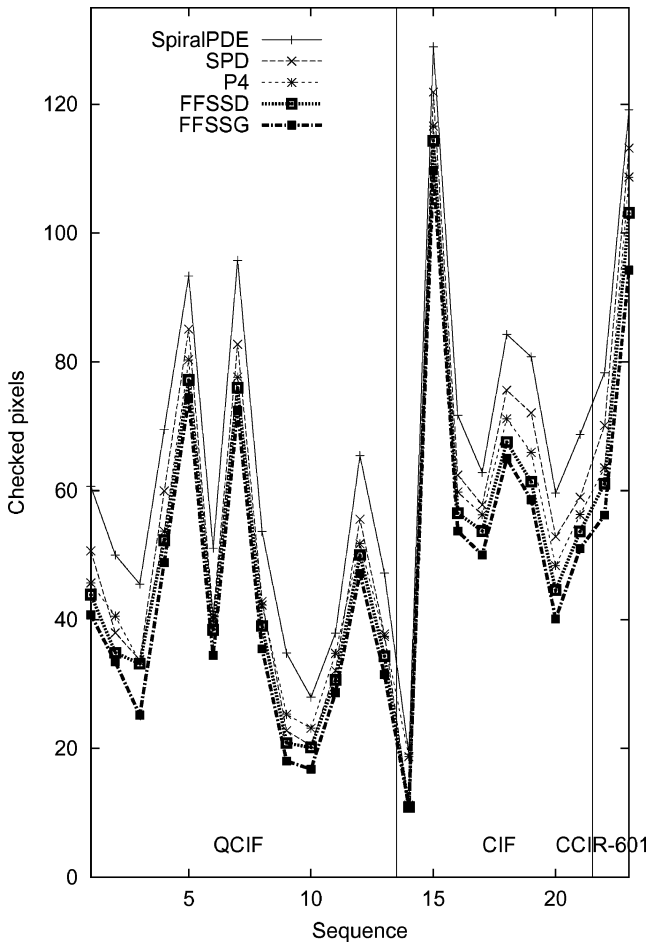


Fig. 1. Average number (for each sequence) of checked pixels per block for SPD, P4, FFSSD, FFSSG, and SpiralPDE.

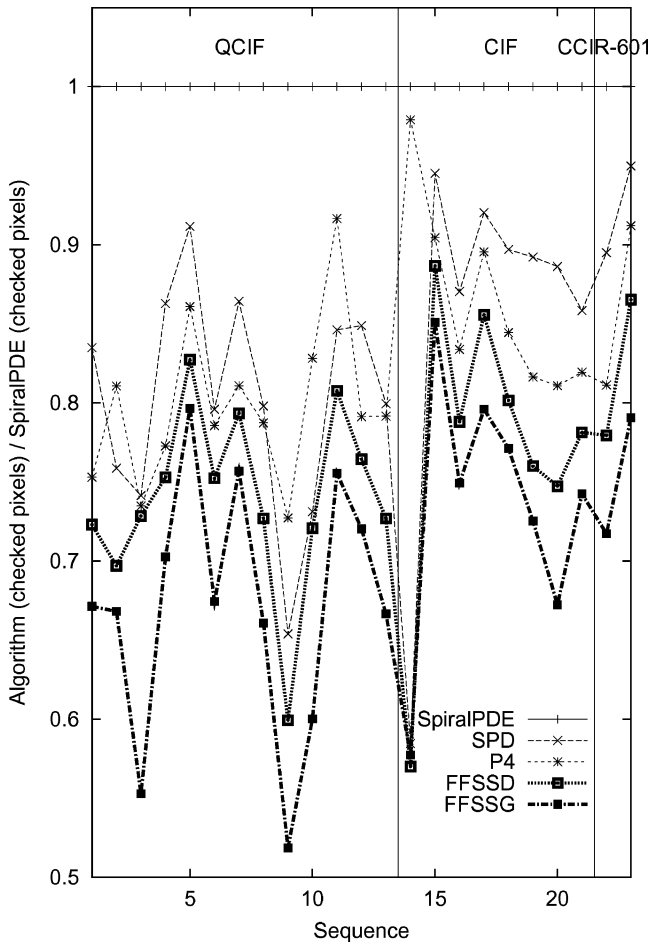


Fig. 2. Reduction of the number of checked pixels for each sequence with SPD, P4, FFSSD, and FFSSG. Reduction is expressed as the fraction of checked pixels with respect to SpiralPDE.

While pixel number is a measure for the motion estimation phase only, total CPU time reflects the actual gain the algorithm can provide in a not-optimized implementation. In particular, we have tried not to favor a particular algorithm in our implementation, since an optimized (software [10] or hardware) implementation may use the characteristics of the available hardware (e.g., CPU instruction sets such as MMX and VIS), and the choice of the hardware can also influence the choice of the algorithm. Our implementation can instead reveal the impact of the block-matching algorithm on a general-purpose architecture. In particular, all total encoding time results have been obtained in several tests on different i386–based machines (Intel Pentium III and AMD Athlon) in order to minimize small differences due to single test conditions; all tests have been performed using the Linux operating system.

It is also worth noting that there are additional computational costs in all algorithms except for SpiralPDE. In fact, P4 and FFSSG algorithms compute gradient for all pixels in the reference block; FFSSG gradient computation is more complex, since P4 uses the simple gradient implementation reported in [62]. FFSSD algorithm does not have such overheads, since the complete distortion computation with the central test block is always performed in every PDE approach, and no tests are done on the reference block; however, the sorting phase influences performance and, even though complexity is $O(n)$, sorting overhead is appreciable (the same happens in FFSSG). Further, P4 uses an ordering algorithm for the 16 subblocks. All algorithms (except for SpiralPDE and P4) make comparisons every eight pixels instead of every 16 pixels. The reason for this choice can be found in [56] and is based on the observation that the total CPU encoding time has the best reduction with eight-pixel comparisons with respect to 16–pixel comparisons. Moreover, for all tested algorithms (always with the exception of SpiralPDE), the access order of pixels in memory could be complex (e.g., with cache miss problems), and it could depend on the hardware architecture.

For all of these reasons, we also report the total CPU encoding time and, in particular, we give the reduction of this time, since it varies greatly with the sequence, the number of frames in the sequence, and the image format.

Fig. 1 reports, for each sequence of Table II, the average number of checked pixels per block for SPD, P4, FFSSD, FFSSG, and SpiralPDE. This number, which can vary from 1 to 256, changes greatly with the sequence (even if less than the encoding time), but from the Foreman sequence—that is reported both in QCIF (sequence number 4) and in CIF (sequence number 16) format—we can see that it does not vary significantly with the image format. Fig. 2 shows, for each

TABLE III
CHECKED PIXEL REDUCTION AND ENCODING TIME SPEEDUP (ON ALL SEQUENCES) FOR SPD, P4, FFSSD, AND FFSSG IN PERCENTAGES WITH RESPECT TO SPIRALPDE. MEAN VALUE AND STANDARD DEVIATION (IN BRACKETS) ARE REPORTED FOR EACH MEASURE

| | SPD | P4 | FFSSD | FFSSG |
|---|---|---|---|---|
| Checked pixel reduction | 16.75%(8.86) | 17.40%(6.06) | 24.1%(7.22) | 29.84%(8.13) |
| Encoding time speedup | 11.35%(6.28) | 13.04%(4.8) | 14.45%(5.32) | 20.8%(5.17) |

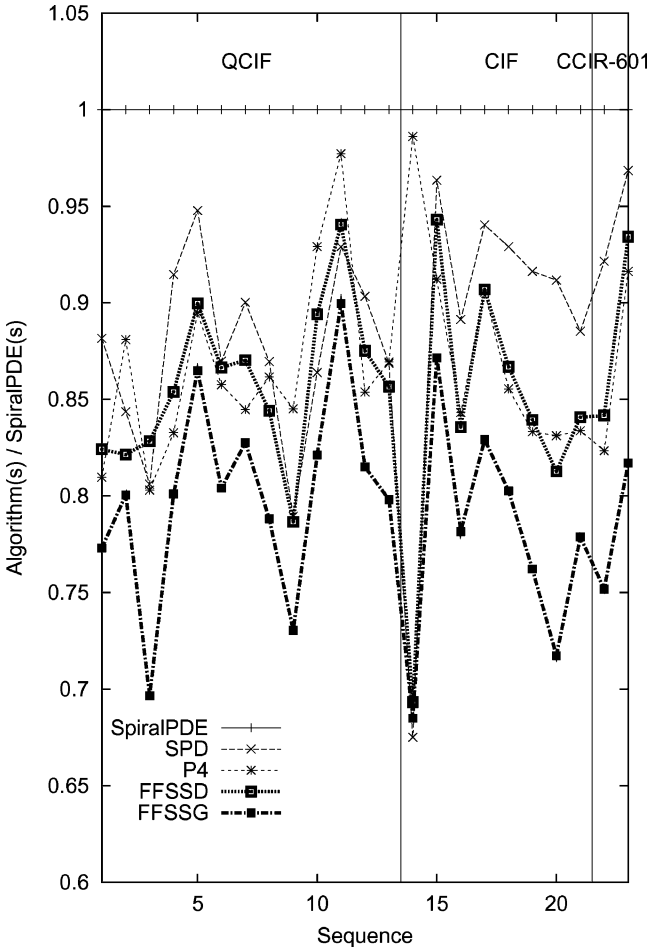


Fig. 3. Reduction of the encoding time for each sequence with SPD, P4, FFSSD, and FFSSG. Reduction is expressed as the fraction of CPU time with respect to SpiralPDE.

sequence, the average number of checked pixels per block for SPD, P4, FFSSD, and FFSSG normalized on SpiralPDE. The first row of Table III reports, for all algorithms, the mean value and the standard deviation (on all sequences) of the checked pixel reduction with respect to SpiralPDE. FFSSG is the fastest algorithm, with 29.84% improvement on SpiralPDE (P4 gets 17.40%) even if FFSSD also performs well (24.1%).

In particular, the significant improvement of FFSSG versus P4 (the two algorithms are similar) can then be explained with the fact that, in actual test sequences, the most significant pixels are not localized in small and well-defined areas, and this means that a matching order based on a single pixel granularity can be more efficient than a $4 \times 4$ block-based granularity. The same principle can explain the performance of SPD and FFSSD.

From the point of view of the total encoding time, Fig. 3 reports, for each sequence of Table II, the CPU times of SPD, P4, FFSSD, and FFSSG normalized on SpiralPDE. We can see that the FFSSG algorithm produces a good improvement with respect to the other algorithms, and the gain is constant for all sequences and image formats. FFSSD also produces a gain, but less than FFSSG. P4 has a worse CPU time gain, and specifically for some sequences it performs badly.

The second row of Table III reports, for all algorithms, the mean value and the standard deviation (on all sequences) of the CPU time speedup normalized on SpiralPDE. FFSSG shows the best performance, since it reports a 20.8% of total encoding time reduction with respect to SpiralPDE, while FFSSD provides 14.45% and P4 provides 13.04%.

### B. Experimental PDE Analysis

The purpose of this section is to find an operational lower bound (based on standard test sequences) for the average number of checked pixels per block in the PDE approach. Possible optimizations of PDE aim at the fast detection of the minimum SAD for the searching phase and the estimation of the largest contributions to SAD for the matching phase. Therefore, if the searching algorithm knew *a priori* the minimum SAD and the contribution of each pixel-to-pixel difference, it would achieve optimal performances.

The first test case (Search Wizard) finds the optimal speedup that can be obtained by improving only the searching phase (always with a full search strategy). For every macroblock, the algorithm initializes $\mathrm{SAD}_{\min}$ with the global minimum of distortion for that macroblock and then performs a traditional motion estimation. Since $\mathrm{SAD}_{\min}$ is the smallest possible, the greatest number of candidate vectors are rejected without performing the full SAD computation (lower bound for the average number of checked pixels per block). Some algorithms estimate the global minimum of distortion by predicting the best motion vector from the spatially neighboring macroblocks. Instead, in our case, the estimation is replaced by the actual minimum of distortion found with a preliminary motion estimation. In fact, here we are not interested in finding a fast algorithm but a lower bound for the number of checked pixels. The second test case (Match Wizard) finds the optimal speedup achievable by optimizing only the matching strategy. For every candidate vector, the algorithm computes the differences for all pixels in the block; then it counts the minimum number of contributions to reach $SAD_{\min}$. Finally, the third test case (Search & Match Wizard) combines both techniques in order to obtain the lowest operational bound for the PDE approach with the given set of test sequences. Such Wizards require, in order to be implemented, two successive phases: in the first, data about the optimal searching (matching) order are collected; in the second, such data are used to find the lowest operational bound.

The performance bounds of PDE are measured as the average number of pixels per macroblock used to probe the candidate

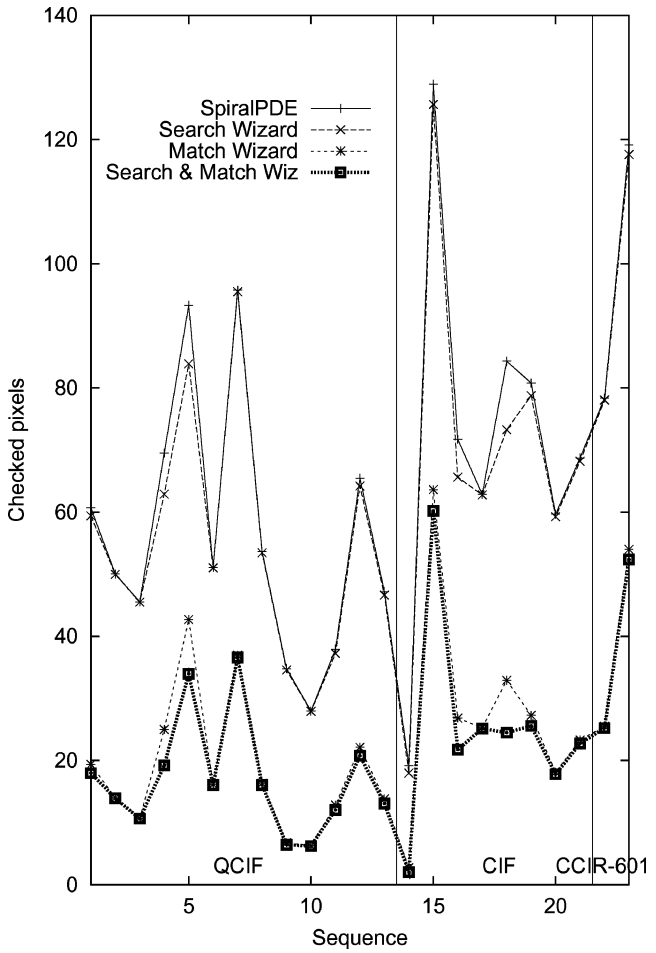


Fig. 4. Checked pixels (for each sequence) for Wizards and SpiralPDE.

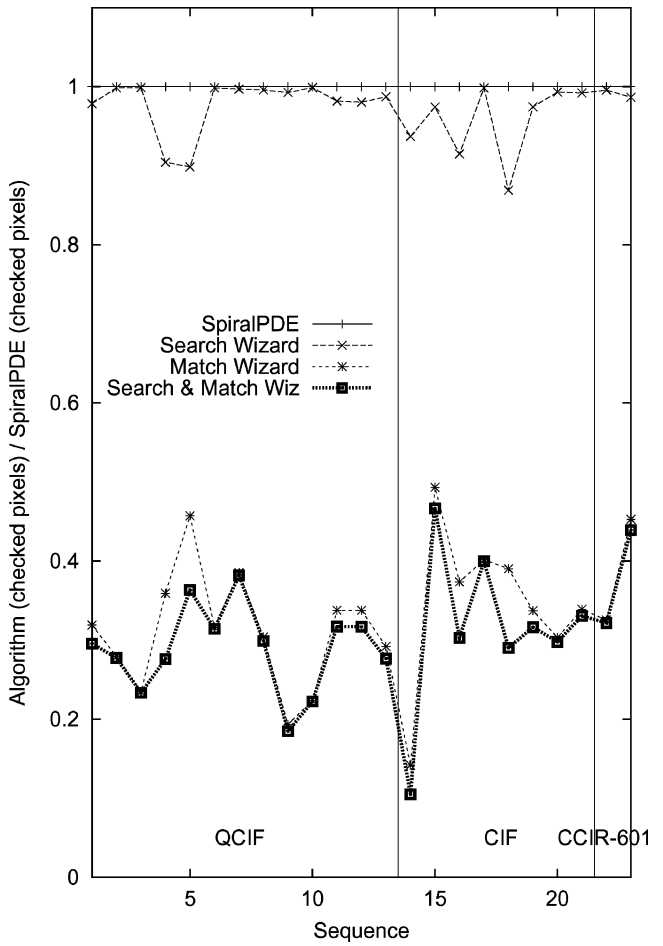


Fig. 5. Reduction of the number of checked pixels for each sequence with Wizards. Reduction is expressed as the fraction of checked pixels with respect to SpiralPDE.

vector; this metric is independent of the actual hardware architecture on which the algorithm is implemented. Encoding time is not considered because significant computational overheads are required to find the tight constraints explained above.

Fig. 4 reports the average number of checked pixels per block for the three test cases and for SpiralPDE with different video sequences (Table II contains a description of the sequences). The value reported for the Search & Match Wizard is the actual lowest bound for the average number of checked pixels per block in the PDE full-search lossless approach from an operational point of view (i.e., for the given set of test sequences). Fig. 5 shows the same information normalized on the performances of SpiralPDE for each tested sequence. Finally, Table IV reports the average and standard deviations of the percentage of speedup with respect to SpiralPDE for all tested sequences. It is worth noting that the improvement margin is extremely small for the searching process; instead, the matching process is a promising field of research that can lead to 70% of improvement of the motion estimation with respect to SpiralPDE.

### C. Performance Analysis

Finally, Fig. 6 compares the number of checked pixels (averaged on all tested sequences) for all algorithms. Values for Search & Match Wizard are taken from Section IV-B. The range of values is from 1 to 256. As can be seen in the figure, given the ideal Search & Match Wizard, the FFSSG algorithm features slightly less than half of the possible gain (in pixel) with respect to SpiralPDE. The other algorithms always perform worse.

TABLE IV
CHECKED PIXEL REDUCTION (ON ALL SEQUENCES) FOR WIZARDS IN PERCENTAGE NORMALIZED ON SPIRALPDE. MEAN VALUE AND STANDARD DEVIATION (IN BRACKETS) ARE REPORTED FOR EACH MEASURE

| Search Wiz. | Match Wiz. | Search&Match Wiz. |
|---|---|---|
| 2.82%(3.75) | 66.95%(8.15) | 69.43%(7.61) |

Table V reports the values of the normalized performance metric $\Delta x_{\%}$ defined as

$$\Delta x_{\%} = \frac{\text{pixels}_{\text{SpiralPDE}} - \text{pixels}_{x}}{\text{pixels}_{\text{SpiralPDE}} - \text{pixels}_{\text{Match\&SearchWizard}}} \cdot 100 \tag{13}$$

where $x$ represents the algorithm.

Results reported in Table V confirm that the FFSSG and FFSSD algorithms represent two efficient methods for fast full-search lossless motion estimation. Pixel reduction results and CPU total encoding times show that FFSSG is a promising method, even if there are still great possibilities before reaching the operational limits of PDE. Moreover, FFSSG and FFSSD can also be joined with algorithms based on SEA to further improve performances.

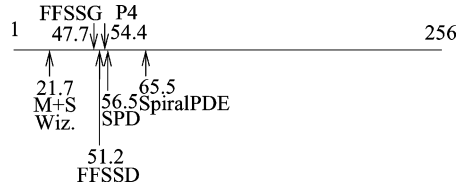


Fig. 6. Checked pixel (mean value) for all tested sequences for Search & Match Wizard, FFSSD, FFSSG, P4, SPD, and SpiralPDE.

TABLE V
$\Delta x_{\%}$ FOR SPD, P4, FFSSD, AND FFSSG ALGORITHMS. DATA ARE COMPUTED USING VALUES REPORTED IN FIG. 6

| SPD | P4 | FFSSD | FFSSG |
|---|---|---|---|
| 20.54% | 25.34% | 32.64% | 40.63% |

## V. CONCLUSION AND FUTURE WORK

In this paper, we have analyzed distortion behavior from local information using the Taylor series expansion. We have also shown that our analysis can comprise other similar methods (in particular, but not only [58]) based on matching strategy improvement. In particular, we have indicated that ordering by distortion of the central test block is a correct alternative to gradient methods, since distortion of this block represents the first order of the Taylor expansion of the distortion.

We have then presented two new full-search (lossless), fast matching, block-motion estimation algorithms, using a PDE approach, namely FFSSD and FFSSG. They aim at quickly discarding invalid candidate vectors by starting SAD computation from those pixel positions that give the highest contributions. Pixel positions are sorted according to some features of the pixels in the center of the search window. FFSSD sorts pixel positions according to the magnitude of pixel-to-pixel differences. FFSSG uses the gradient magnitude between adjacent pixels; gradient is computed on eight directions while previous algorithms used a simpler gradient formulation. Sorting is performed using a technique having linear complexity.

The proposed algorithms perform well in comparison with other existing FS algorithms. In particular, the computational gain of FFSSG with respect to the SpiralPDE algorithm has a mean value of 30%, with a peak value of about 40%, while total CPU encoding time is reduced by about 20%, with a peak value of about 25% (as reported in Table III). FFSSD and FFSSG are also independent of methods based on SEA and can be combined with them to improve global speedup.

Finally, we have presented an operational study (using test sequences) of the bounds and potentialities of the PDE technique. This study has indicated that SpiralPDE achieves performances which are very close to experimental lower bound in the search phase, while there is still an improvement margin on the matching phase. FFSSD and FFSSG achieve interesting results, significantly approaching this margin.

Future work will aim to further improve the matching strategy, especially in the directions depicted in Section III. In particular, we aim to obtain faster implementations of FFSSD and FFSSG to exploit the full potential of these algorithms.

## ACKNOWLEDGMENT

The authors would like to thank Prof. P. Montuschi for his comments and fruitful suggestions.

## REFERENCES

[1] J. K. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images–A review," *Proc. IEEE*, vol. 76, no. 8, pp. 917–935, Aug. 1988.
[2] F. Dufaux and F. Moscheni, "Motion estimation techniques for digital TV: A review and a new contribution," *Proc. IEEE*, vol. 83, no. 6, pp. 858–876, June 1995.
[3] C. Stiller and J. Konrad, "Estimating motion in image sequences," *IEEE Signal Process. Mag.*, vol. 16, no. 4, pp. 70–91, Jul. 1999.
[4] *MPEG-1 Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to About 1.5 Mb/s*, ISO/IEC 11 172, 1993.
[5] *MPEG-2 Generic Coding of Moving Pictures and Associated Audio Information*, ISO/IEC 13 818, 1996.
[6] *MPEG-4—Information Technology—Coding of Audio-Visual Objects—Part 2: Visual*, ISO/IEC 14 496-2, 2000.
[7] "Video Codec for Audiovisual Services at p × 64 kbits," International Telecommunications Union, ITU-T Recommendation H.261, 1993.
[8] "Video Coding for Low Bitrate Communication," International Telecommunications Union, ITU-T Rec-ommendation H.263, 1998.
[9] "Joint Final Committee Draft (jfcd) of Joint Video Specification (itu-t rec. h.264 – iso/iec 14 496-10 avc)," Joint Video Team (JVT) of ISO/IEC MPEG & ITU-T VCEG, Doc. JVT-D157, 2002.
[10] S. M. Akramullah, I. Ahmad, and M. L. Liou, "Optimization of h.263 video encoding using a single processor computer: Performance tradeoffs and benchmarking," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 8, pp. 901–915, Aug. 2001.
[11] "ITU-T Recommendation H.263 Software Implementation," Digital Video Coding Group, Telenor R&D, 1995.
[12] S. Eckart and C. Fogg, "ISO/IEC MPEG-2 software video codec," in *Proc. SPIE*, vol. 2419, 1995, pp. 100–118.
[13] Y.-L. Chan and W.-C. Siu, "An efficient search strategy for block motion estimation using image features," *IEEE Trans. Image Process.*, vol. 10, no. 8, pp. 1223–1238, Aug. 2001.
[14] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 4, pp. 438–442, Aug. 1994.
[15] L. M. Po and W. C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 3, pp. 313–317, Jun. 1996.
[16] J. Lu and M. L. Liou, "A simple and efficient search algorithm for block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 2, pp. 429–433, Apr. 1997.
[17] L.-K. Liu and E. Feig, "A block-based gradient descent search algorithm for block motion estimation in video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 4, pp. 419–422, Aug. 1996.
[18] O. T.-C. Chen, "Motion estimation using a one-dimensional gradient descent search," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 4, pp. 608–616, Jun. 2000.
[19] K. H.-K. Chow and M. L. Liou, "Genetic motion search algorithm for video compression," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 6, pp. 440–445, Dec. 1993.
[20] S. Li, W.-P. Xu, H. Wang, and N.-N. Zheng, "A novel fast motion estimation method based on genetic algorithm," in *Proc. ICIP*, Oct. 1999, pp. 66–69.
[21] M. Mattavelli and G. Zoia, "Vector-tracing algorithms for motion estimation in large search windows," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 8, pp. 1426–1437, Dec. 2000.
[22] M. R. Pickering, J. F. Arnold, and M. R. Frater, "An adaptive search length algorithm for block matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 6, pp. 906–912, Dec. 1997.
[23] S. Marlow, J. Ng, and C. Mc Ardle, "Efficient motion estimation using multiple log searching and adaptive search windows," in *Proc. Image Process. Applicat.*, Jul. 1997, pp. 214–218.
[24] B. C. Song and J. B. Ra, "A fast motion estimation algorithm based on multi-resolution frame structure," in *Proc. Int. Conf. Acoust., Speech Signal Process.*, Mar. 1999, pp. 3361–3364.
[25] J.-B. Xu, L.-M. Po, and C.-K. Cheung, "A new prediction model search algorithm for fast block motion estimation," in *Proc. Int. Conf. Image Process.*, Oct. 1997, pp. 610–613.

[26] Y.-T. Roan and P.-Y. Chen, "A fast search motion estimation method," in *Proc. Int. Conf. Syst., Man Cybernet.*, Oct. 2000, pp. 1568–1573.
[27] ——, "A fuzzy search algorithm for the estimation of motion vectors," *IEEE Trans. Broadcast.*, vol. 46, no. 2, pp. 121–127, Jun. 2000.
[28] J. Chalidabhongse and C. C. J. Kuo, "Fast motion vector estimation using multiresolution-spatio-temporal correlations," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 7, no. 3, pp. 477–488, Jun. 1997.
[29] A. M. Tourapis, O. C. Au, M. L. Liou, and G. Shen, "An advanced zonal block based algorithm for motion estimation," in *Proc. ICIP*, Oct. 1999, pp. 610–614.
[30] P.-Y. Chen and J. M. Jou, "A fast-search motion estimation method and its VLSI architecture," *IEEE Trans. Circuits Syst. II*, vol. 46, no. 9, pp. 1233–1240, Sep. 1999.
[31] J. M. Jou, P.-Y. Chen, and J.-M. Sun, "The gray prediction search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 6, pp. 843–848, Sep. 1999.
[32] V. G. Moshnyaga, "A new computationally adaptive formulation of block-matching motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 1, pp. 118–124, Jan. 2001.
[33] K. Lengwehasatit and A. Ortega, "Computationally scalable partial distance based fast search motion estimation," in *Proc. Int. Conf. Image Process.*, Sep. 2000, pp. 824–827.
[34] W. Li and E. Salari, "Successive elimination algorithm for motion estimation," *IEEE Trans. Image Process.*, vol. 4, no. 1, pp. 105–107, Jan. 1995.
[35] Y.-C. Lin and S.-C. Tai, "Fast full-search block-matching algorithm for motion-compensated video compression," *IEEE Trans. Commun.*, vol. 45, no. 5, pp. 527–531, May 1997.
[36] M. Z. Coban and R. M. Mersereau, "A fast exhaustive search algorithm for rate-constrained motion estimation," *IEEE Trans. Image Process.*, vol. 7, no. 5, pp. 769–773, May 1998.
[37] H. S. Wang and R. M. Mersereau, "Fast algorithms for the estimation of motion vectors," *IEEE Trans. Image Process.*, vol. 8, no. 3, pp. 435–438, Mar. 1999.
[38] Y. Noguchi, J. Furukawa, and H. Kiya, "A fast full search block matching algorithm for MPEG-4 video," in *Proc. ICIP*, Oct. 1999, pp. 61–65.
[39] T. M. Oh, Y. R. Kim, W. G. Hong, and S. J. Ko, "A fast full search motion estimation algorithm using the sum of partial norms," in *Proc. ICCE*, Jun. 2000, pp. 236–237.
[40] H. A. Mahmoud and M. Bayoumi, "A new block-matching motion estimation algorithm based on successive elimination," in *Proc. Int. Conf. Image Process.*, Sep. 2000, pp. 608–611.
[41] T. Wiegand, X. Zhang, and B. Girod, "Long-term memory motion-compensated prediction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 1, pp. 70–84, Feb 1999.
[42] X. Q. Gao, C. J. Duanmu, and C. R. Zou, "A multilevel successive elimination algorithm for block matching motion estimation," *IEEE Trans. Image Process.*, vol. 9, no. 3, pp. 501–504, Mar. 2000.
[43] L.-C. Chang, K.-L. Chung, and T.-C. Yang, "An improved search algorithm for motion estimation using adaptive search order," *IEEE Signal Process. Lett.*, vol. 8, no. 5, pp. 129–130, May 2001.
[44] M. Brünig and W. Niehsen, "Fast full-search block matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 241–247, Feb. 2001.
[45] B. Liu and A. Zaccarin, "New fast algorithms for the estimation of block motion vectors," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 3, no. 2, pp. 148–157, Apr. 1993.
[46] K. T. Choi, S. C. Chan, and T. S. Ng, "A new fast motion estimation algorithm using hexagonal subsampling pattern and multiple candidates search," in *Proc. Int. Conf. Image Process.*, Sep. 1996, pp. 497–500.
[47] Y.-L. Chan and W.-C. Siu, "New adaptive pixel decimation for block motion vector estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 6, no. 1, pp. 113–118, Feb. 1996.
[48] ——, "A new block motion vector estimation using adaptive pixel decimation," in *Proc. ICASSP*, May 1995, pp. 2257–2260.
[49] Y.-L. Chan, W.-L. Hui, and W.-C. Siu, "A block motion vector estimation using pattern based pixel decimation," in *Proc. ISCAS*, Jun. 1997, pp. 1153–1156.
[50] Y. Wang, Y. Wang, and H. Kuroda, "A globally adaptive pixel-decimation algorithm for block-motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 6, pp. 1006–1011, Sep. 2000.
[51] A. C. K. Ng and B. Zeng, "A new fast motion estimation algorithm based on search window sub-sampling and object boundary pixel block matching," in *Proc. ICIP*, Oct. 1998, pp. 605–608.
[52] F. H. Cheng and S. N. Sun, "New fast and efficient two-step search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 7, pp. 977–983, Oct. 1999.
[53] C. K. Cheung and L. M. Po, "Normalized partial distortion search algorithm for block motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 3, pp. 417–422, Apr. 2000.
[54] C.-H. Cheung and L.-M. Po, "A fast block motion estimation using progressive partial distortion search," in *Proc. Int. Symp. Intelligent Multimedia, Video Speech Process.*, May 2001, pp. 506–509.
[55] K. Lengwehasatit and A. Ortega, "Probabilistic partial-distance fast matching algorithms for motion estimation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 2, pp. 139–152, Feb. 2001.
[56] D. Quaglia and B. Montrucchio, "Sobol partial distortion algorithm for fast full search in block motion estimation," in *Proc. 6th Eurographics Workshop Multimedia*, Sep. 2001, pp. 77–84.
[57] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C : The Art of Scientific Computing*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 1993.
[58] J. N. Kim and T. S. 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, Aug. 1999.
[59] ——, "A fast full-search motion-estimation algorithm using representative pixels and adaptive matching scan," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, no. 7, pp. 1040–1048, Oct. 2000.
[60] J. N. Kim, S.-C. Byun, and B.-H. Ahn, "Fast full search motion estimation algorithm using various matching scans in video coding," *IEEE Trans. Syst., Man, Cybern. C*, vol. 31, no. 4, pp. 540–548, Nov. 2001.
[61] J. N. Kim, S. C. Byun, Y. H. Kim, and B. H. Ahn, "Fast full search motion estimation algorithm using early detection of impossible candidate vectors," *IEEE Trans. Signal Process.*, vol. 50, no. 9, pp. 2355–2365, Sep. 2002.
[62] R. C. Gonzales and R. E. Woods, *Digital Image Processing*. Reading, MA: Addison-Wesley, 1992.
[63] J. N. Kim and B. H. Ahn, "Lossless computational reduction of full search algorithm in motion estimation using appropriate matching unit from image localization," in *Proc. Int. Conf. Inform. Technol.: Coding and Computing*, Apr. 2001, pp. 447–451.
[64] H. G. Musmann, P. Pirsch, and H. J. Grallert, "Advances in picture coding," *Proc. IEEE*, vol. 73, no. 4, pp. 523–548, Apr. 1985.
[65] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge, MA/New York: MIT Press/ McGraw-Hill, 1990.

**Bartolomeo Montrucchio** (M'02) received the M.S. degree in electronic engineering and the Ph.D. degree in computer engineering from Politecnico di Torino, Torino, Italy, in 1998 and 2002, respectively.

He is currently an Assistant Professor of computer engineering with the Department of Computer Engineering, Politecnico di Torino. His current research interests include image analysis and synthesis techniques, motion estimation techniques, medical imaging and scientific visualization.

Prof. Montrucchio is a member of the IEEE Circuits and Systems Society and of the Eurographics Association.

**Davide Quaglia** (SM'00–M'04) received the M.S. and Ph.D. degrees from Politecnico di Torino, Torino, Italy, in 1999, and 2003, respectively, both in computer engineering.

He is currently with the Department of Computer Engineering, Politecnico di Torino, with a research grant supported by CERCOM, the Center for Wireless Multimedia Communications of Torino. His current research interests include video coding, robust delivery of multimedia signals over packet networks, motion estimation techniques, wireless networks, and signal processing for impaired-people applications.

Dr. Quaglia is a member of the IEEE Signal Processing and Engineering in Medicine and Biology Societies. In 2002, he was the chairman of the IEEE Student Branch of Politecnico di Torino.