Full-Search-Equivalent Pattern Matching with Incremental Dissimilarity Approximations

Federico Tombari, *Student Member*, *IEEE*, Stefano Mattoccia, *Member*, *IEEE*, and Luigi Di Stefano, *Member*, *IEEE*

Abstract—This paper proposes a novel method for fast pattern matching based on dissimilarity functions derived from the L_p norm, such as the *Sum of Squared Differences* (SSD) and the *Sum of Absolute Differences* (SAD). The proposed method is a full-search equivalent, i.e., it yields the same results as the *Full Search* (FS) algorithm. In order to pursue computational savings, the method deploys a succession of increasingly tighter lower bounds of the adopted L_p norm-based dissimilarity function. Such bounding functions allow for establishing a hierarchy of pruning conditions aimed at rapidly skipping those candidates that cannot satisfy the matching criterion. The paper includes an experimental comparison between the proposed method and other FS-equivalent approaches known in the literature, which proves the remarkable computational efficiency of our proposal.

Index Terms—Pattern matching, IDA, SSD, SAD, efficient, full-search equivalent.

1 Introduction

Pattern matching aims at locating the instances of a given template into a reference set. This task occurs in numerous image analysis applications and consists of determining the regions of the reference image that are *similar* to the template according to a given criterion and discarding those that are *dissimilar*. The *Full Search* (FS) pattern-matching algorithm relies on calculating, at each position of the reference image, a function measuring the degree of similarity or dissimilarity between the template and the portion of the image currently under examination, referred to as *image subwindow*. Once the chosen function is computed for all *subwindows*, a threshold is usually adopted so as to classify between matching and mismatching patterns. L_p norm-based dissimilarity functions are widely used in pattern-matching applications involving images of the same modality, as thoroughly discussed in [1]. The most popular L_p norm-based dissimilarity functions are the *Sum of Squared Differences* (SSD) and the *Sum of Absolute Differences* (SAD). For what concerns the SSD, though the typical alternative to the naive FS algorithm is represented by the FFT-based approach, a novel fast FS-equivalent method [1], referred to here as *Projection Kernels (PKs)*, was recently proposed in the literature. This method was shown to be much more efficient compared to the naive FS-approach, as well as to the FFT. With regard to the SAD, a well-known classical approach is the *Sequential Similarity Detection Algorithm (SSDA)* [2].

In this paper, we propose[1] a novel FS-equivalent method that deploys a succession of sufficient conditions, characterized by increasing effectiveness, for rapidly pruning those image subwindows that cannot fulfill the matching criterion. The novel method, referred to as the Incremental Dissimilarity Approximations (IDAs) algorithm, is also compared to the FS, FFT, PK, and SSDA algorithms. Experimental results concerning more than 6,000 pattern-matching instances prove that IDA significantly outperforms state-of-the-art approaches and can yield substantial speedups with respect to the FS.

The paper is structured as follows: Section 2 reviews previous work. Section 3 describes the IDA algorithm. Section 4 proposes a variation of the basic IDA approach, called Hybrid IDA (hIDA). Section 5 presents an experimental comparison between our methods and the other approaches considered throughout the paper. Conclusions are drawn in Section 6. Finally, in the Appendix, we discuss the issue of generalizing our approach to an arbitrary metric.

2 Previous Work

The L_p norm of an M-dimensional vector $X = [x_1, \cdots, x_M]^T$ is defined as

$$\|X\|_p = \left(\sum_{i=1}^{M} |x_i|^p \right)^{\frac{1}{p}}, \tag{1}$$

where p is any positive real number [4].

Now, let X be the template vector and $Y_1, \cdots, Y_N$ be the N candidates (corresponding to the image subwindows) against whom X must be matched, each candidate having the same cardinality as the template vector (i.e., $Y_j = [y_{j,1}, \cdots, y_{j,M}]^T$).

The generic function based on the L_p norm measuring the dissimilarity between X and Y_j can be written as

• The authors are with the Dipartimento di Elettronica, Informatica e Sistemistica (DEIS), Advanced Research Center on Electronic Systems (ARCES), University of Bologna, Viale Risorgimento, 2, 40136 Bologna, Italy.
E-mail: {federico.tombari, stefano.mattoccia, luigi.distefano}@unibo.it.

Manuscript received 21 Feb. 2007; revised 26 Sept. 2007; accepted 16 Jan. 2008; published online 13 Feb. 2008.
Recommended for acceptance by W. Förstner.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0121-0207.
Digital Object Identifier no. 10.1109/TPAMI.2008.46.

1. Preliminary results of this research work appeared in [3].

$$\|X - Y_j\|_p^p = \sum_{i=1}^{M} \left|x_i - y_{j,i}\right|^p. \tag{2}$$

If $p = 1$, then (2) coincides with the SAD function, while $p = 2$ yields the SSD function.

We will now briefly review the FFT-based, PK, and SSDA approaches for fast FS-equivalent pattern matching with L_p norm-based dissimilarity functions.

2.1 Fast Fourier Transform

A common approach for speeding up the FS pattern-matching process based on the SSD function relies on the *Fast Fourier Transform* (FFT). The SSD function can be written as

$$\|X - Y_j\|_2^2 = \|X\|_2^2 + \|Y_j\|_2^2 - 2 \cdot \Theta(X, Y_j), \tag{3}$$

where

$$\Theta(X, Y_j) = \sum_{i=1}^{M} x_i \cdot y_{j,i} \tag{4}$$

represents the *dot product* between X and Y_j. In order to achieve computational savings, the FFT approach calculates Θ in the frequency domain according to the correlation theorem. As it would be inefficient to compute $\|Y_j\|_2^2$ in the frequency domain, this term is usually calculated directly by means of efficient incremental techniques [5], [6], [7], as described in [8], while $\|X\|_2^2$ is computed once and for all at initialization time.

Compared to the FS algorithm, the FFT-based approach is more efficient when the template size is large enough compared to the image size. The FFT-based approach cannot be adopted for SAD-based pattern matching since the correlation theorem does not apply to the L_1 norm case.

2.2 Projection Kernels

The PK method [1] carries out a fast FS-equivalent SSD-based pattern matching in the signal domain. With PK, each basis vector U of the *Walsh-Hadamard* transform is used as a projection vector. Then, projecting the template vector X and the candidate vector Y_j onto each of these projection vectors yields a *projected distance* B_j:

$$B_j = U^T X - U^T Y_j, \tag{5}$$

which can be used to determine a lower bound of the SSD function:

$$\|X - Y_j\|_2^2 \geq \frac{B_j^2}{\|U\|_2^2}. \tag{6}$$

Therefore, if D is the threshold that discriminates between matching and mismatching candidates, it is possible to establish the condition

$$D < \frac{B_j^2}{\|U\|_2^2}, \tag{7}$$

which allows for safely pruning Y_j from the list of candidates. Furthermore, the lower bound can be tightened by using a collection of projection vectors along with the corresponding projected distances. Hence, an iterative algorithm is proposed in [1]: At each step, the lower bound is tightened so as to increase the effectiveness of the current pruning condition for those candidates that were not pruned by the previous one.

According to [1], PK is almost two orders of magnitude faster than the FS and FFT-based approaches, but it is more demanding in terms of memory requirements. It is worth pointing out that the size of the template is constrained to be a power of 2. The experimental results reported in [1] also show that PK is more effective with very small templates (i.e., size 16×16 or 32×32).

2.3 Sequential Similarity Detection Algorithm

The SSDA [2] method is a classical approach originally introduced to determine simple inequalities to speedup the SAD-based pattern matching. Let D be a threshold and let X, Y_j be the template-candidate pair under evaluation. During the computation of the SAD function, at each new element pair $x_b, y_{j,b}$, condition

$$\sum_{i=1}^{b} \left|x_i - y_{j,i}\right| > D \tag{8}$$

is tested. As soon as (8) is satisfied, the evaluation process is terminated and the value of the last vector index, $\tilde{b}_j$, is recorded. Once this is done for all candidates, the best matching candidates correspond to those having high $\tilde{b}_j$. Typically, D is much lower than the global minimum and SSDA turns out to not be equivalent to the FS (i.e., nonexhaustive). In particular, the choice of D determines a cost-performance trade-off: The higher D is, the higher the mean number of calculations needed to evaluate the current candidate and the higher the chance that the resulting matching candidates will coincide with those yielded by FS. In order to better deal with this issue, D is not kept constant, but it increases along with b. Moreover, to obtain a more regular behavior, the order of the processed vector elements is randomly scrambled. However, it is practically unfeasible to determine a varying D that yields an FS-equivalent algorithm. Therefore, similarly to the other methods considered throughout the paper, in our experiments, we set D to a constant threshold higher than the global minimum: This turns SSDA into an FS-equivalent method.

3 Incremental Dissimilarity Approximations Algorithm

This section describes a novel signal domain method, referred to as *Incremental Dissimilarity Approximation* (IDA), aimed at speeding up the FS-equivalent pattern matching based on the L_p norm. IDA relies on partitioning the template vector, X, and each candidate vector, Y_j, into a certain number of subvectors in order to determine a succession of pruning conditions characterized by increasing tightness and computational weight.

Given an M-dimensional vector, we establish a partition of the vector into r disjoint subvectors (not necessarily with the same number of components) by defining a partition, P, of set $S = \{1, 2, \ldots M\}$ into r disjoint subsets:

$$\begin{cases} P = \{S_1, S_2 \ldots S_r\}, r \in S \\ \bigcup_{u=1}^{r} S_u = S \\ S_u \cap S_v = \phi, \forall u \neq v, u, v \in \{1, 2, \ldots r\}. \end{cases}$$

The minimum number of subvectors is 1, that is, the vector is actually not partitioned into smaller subvectors, the maximum number is M, the vector partitioned into M one-dimensional disjoint subvectors. Details concerning an efficient implementation of such partitioning will be discussed later.

Given P, we define the *partial* L_p norm of vectors X, Y_j restrained to the subvectors associated with $S_t \in P$ as

$$\|X\|_{p,S_t} = \left(\sum_{i \in S_t} |x_i|^p\right)^{\frac{1}{p}}, \tag{9}$$

$$\|Y_j\|_{p,S_t} = \left(\sum_{i \in S_t} |y_{j,i}|^p\right)^{\frac{1}{p}}, \tag{10}$$

and the *partial* L_p-dissimilarity between X and Y_j restrained to the subvectors associated with $S_t \in P$ as

$$\|X - Y_j\|_{p,S_t}^p = \sum_{i \in S_t} |x_i - y_{j,i}|^p. \tag{11}$$

Then, by virtue of the *triangular inequality* applied on corresponding subvectors, we establish the following r inequalities:

$$\|X - Y_j\|_{p,S_t}^p \geq \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p, \quad t = 1, \ldots r, \tag{12}$$

and summing up both members of the inequalities attain a *lower bound* of the function measuring the dissimilarity between X and Y_j:

$$\|X - Y_j\|_p^p \geq \sum_{t=1}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p. \tag{13}$$

This inequality provides a sufficient condition that allows for pruning those candidates that cannot represent a matching position. In fact, if the lower bound of the dissimilarity function exceeds the threshold D that discriminates between matching and nonmatching candidates

$$\sum_{t=1}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p > D, \tag{14}$$

then, from (13) and (14), Y_j cannot be a matching pattern.

If (14) does not hold, rather than computing from scratch the term $\|X - Y_j\|_p^p$, we can obtain another pruning condition based on a tighter lower bound by considering a subvectors pair and replacing in the left-hand term of (14) the difference between the *partial* norms with the corresponding *partial* L_p-dissimilarity:

$$\|X - Y_j\|_{p,S_i}^p + \sum_{t=1, t \neq i}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p > D. \tag{15}$$

Since the following relation holds as a consequence of the triangular inequality

$$\begin{aligned} \|X - Y_j\|_p^p &\geq \|X - Y_j\|_{p,S_i}^p + \sum_{t=1, t \neq i}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p \\ &\geq \sum_{t=1}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p, \end{aligned} \tag{16}$$

the lower bound appearing at the left-hand side of (15) is tighter compared to that of (14) and, hence, the associated pruning condition is potentially more effective in skipping nonmatching candidates.

Should condition (15) fail, the tightness of the lower bounding function can be further increased by taking another subvectors pair and, again, replacing the difference between the *partial* norms with the corresponding *partial* L_p-dissimilarity. This process can be iteratively applied to all of the r subvectors pairs resulting from P so as to determine up to r sufficient conditions that can be sequentially checked when matching each candidate vector Y_j. These r conditions are based on the following succession of increasingly tighter lower bounds:

$$\begin{aligned} &\sum_{t=1}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p \\ &\leq \|X - Y_j\|_{p,S_i}^p + \sum_{t=1, t \neq i}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p \\ &\leq \|X - Y_j\|_{p,S_i}^p + \|X - Y_j\|_{p,S_k}^p \\ &+ \sum_{t=1, t \neq i,k}^{r} \left|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}\right|^p \leq \cdots \\ &\cdots \leq \sum_{t=1, t \neq l}^{r} \|X - Y_j\|_{p,S_t}^p + \left|\|X\|_{p,S_l} - \|Y_j\|_{p,S_l}\right|^p. \end{aligned} \tag{17}$$

Hence, throughout the matching process, each vector Y_j undergoes checking of a succession of sufficient conditions, starting from (14), until either it is pruned or the following last condition is reached:

$$\sum_{t=1, t \neq l}^{r} \|X - Y_j\|_{p,S_t}^p + \left|\|X\|_{p,S_l} - \|Y_j\|_{p,S_l}\right|^p > D. \tag{18}$$

Should the last condition not be verified, the process ends up in computing the dissimilarity $\|X - Y_j\|_p^p$ by replacing $|\|X\|_{p,S_l} - \|Y_j\|_{p,S_l}|^p$ with $\|X - Y_j\|_{p,S_l}^p$ in the left-hand term of (18). Then, Y_j is classified as a valid pattern if

$$\|X - Y_j\|_p^p < D. \tag{19}$$

The proposed method can be straightforwardly modified to deal with the task of locating the most similar instance of a template within a reference image (template matching). In such a case, the term D is not a constant but represents the best similarity score *found so far*. As proposed in [3], D might be conveniently and rapidly initialized by selecting an initial guess for the best matching candidate through a fast nonexhaustive algorithm [9], [10]. Then, for each candidate Y_j, the algorithm is the same as described previously, the only difference being that, if condition (19) holds, then $\|X - Y_j\|_p^p$ is assigned to D (i.e., the best score *found so far* is updated).

The key point of the IDA algorithm is that it achieves computational savings since, compared to $\|X - Y_j\|_{p,S_t}^p$, the term $|\|X\|_{p,S_t} - \|Y_j\|_{p,S_t}|^p$ can be computed much more rapidly and independently from the subvectors cardinality by calculating the partial norms using well-known fast incremental calculation schemes [5], [6], [7]. Consequently, since replacing the differences of partial norms with the corresponding partial dissimilarities yields tighter bounding functions, the tighter the bounding function is, the higher its calculation time is. Therefore, as described in this section, IDA establishes a succession of increasingly tighter bounding functions, with the next computationally more demanding function calculated only when required, i.e., when a candidate has not been pruned by the previous condition.

In order to efficiently compute the partial norms by means of incremental calculation schemes, the adopted partitioning scheme for X and Y_j must follow certain rules of regularity [11]. In particular, we propose partitioning X and Y_j according to a splitting of template and image subwindows into r rectangular regions and calculate the partial norms by the one-pass *box-filtering* method proposed in [6]. In our implementation, a box-filtering function fills in an array of partial norms by computing the norm of each rectangular region of given dimensions belonging to the reference image. As described in [6], this is done by exploiting a double recursion on the rows and columns of the reference image, which requires only four elementary operations per image point, independently of the sizes of the rectangular region. Hence, to obtain the required partial norms, we need to run as many box filters as the number of differently sized regions corresponding to subvectors. In a particular case of r equally sized regions, a single box filter is needed by IDA. This results in a memory footprint on the order of N, which compares favorably with the PK technique, which requires a memory footprint on the order of $N \log M$ [1].

It is worth pointing out that the idea of partitioning the vectors in order to deploy tighter bounding functions has been already proposed in other fields such as motion estimation [12], [13] and vector quantization [14]. Nevertheless, our approach differs from these proposals since it incorporates the idea of successively refining the bounding functions by means of the partial dissimilarity concept and it is not based on a multiresolution scheme.

4 Hybrid IDA Algorithm

The main drawback of techniques such as IDA, PK, and SSDA is data dependency that results in unpredictable response times. In fact, the computational efficiency of these techniques relies on the ability to prune mismatching candidates by means of the adopted sufficient conditions, which in turn depends on the data. Conversely, with the FS approach, response time depends only on the image and pattern sizes and, with the FFT approach, only on the image size. Moreover, as will be shown in Section 5, although IDA turns out to be generally faster than the FS and FFT approaches, in some cases, it happens to be slower than the FFT approach.

We observed that the overall behavior of the IDA algorithm can be predicted with a high degree of reliability by evaluating the pruning efficiency of its sufficient conditions on a small subset of points uniformly distributed over the image. This task requires a fixed and small computation time and it is particularly meaningful when images have high spatial similarity within large neighborhoods, as occurs in most cases. Hence, in those pattern-matching instances where IDA is predicted not to be particularly effective, the matching process may be carried out using the faster one between the FS and FFT approach; this choice is made upon the image and template sizes. Such an approach requires a small overhead with respect to the basic IDA algorithm and, as, generally, the prediction turns out to be correct, it guarantees a deterministic upper bound on the response time in most cases.

Based on these considerations, we have devised the following variation to the basic IDA algorithm, referred to as hIDA. Given a fixed and small subset of points uniformly distributed over the image, hIDA evaluates the percentage of points within this subset where the first sufficient condition (i.e., (14)) succeeds in pruning the corresponding candidate. In case this percentage is higher than a certain threshold (typically between 50 percent, for small images, and 85 percent, for bigger images), hIDA carries out the matching process using the IDA algorithm; conversely, it switches to the faster between the FS and FFT algorithms. As will be shown in Section 5, thanks to the computational efficiency and reliability of the prediction step, in most cases, hIDA guarantees that, in problem instances favorable to the IDA approach, the performance is substantially equivalent to that of the basic IDA algorithm, while, in those few cases less favorable to IDA, the performance is substantially equivalent to that of the faster one between the FS and FFT.

5 Experimental Results

This section is aimed at assessing the performance of IDA and hIDA by comparing them with the FS algorithm as well as with the fast exhaustive algorithms presented in Section 2, i.e., the PK, FFT-based, and SSDA algorithms. The IDA, hIDA, FS, and SSDA algorithms were implemented in *C*. As for PK, we compiled and ran the original authors' *C* code (available at their website [15]), which refers to the case of the SSD function. With regard to the FFT-based algorithm, we used the very efficient implementation (*cvMatchTemplate* function) provided by the *OpenCV* library [16]. Hence, we compared IDA and hIDA to the FS, PK, and FFT algorithms in the case of the SSD function $(p = 2)$ and then IDA to the FS and SSDA algorithms in the case of the SAD function $(p = 1)$.[2] The benchmarking platform was an *AMD Athlon* processor with 3 Gbytes of RAM running *Windows XP*.

Two different kinds of experiments were carried out. In *Experiment 1*, we individually compare all of the speedup values yielded by the considered algorithms on an indoor sequence of three images acquired by means of a digital camera. This data set is affected by real distortions since each image was taken at a slightly different pose with respect to that from which the templates were extracted. Instead, *Experiment 2* aims at evaluating the global performance of the algorithms on a large data set of 120 images, with artificial noise at five different levels added on each image. In this latter experiment, results are shown by means of statistical indicators.

In order to evaluate the performance of the algorithms with different image dimensions, for both experiments, four different scales, $S1, \cdots, S4$, of patterns and images have been used:

- S1. Images: 160×120; Templates: 16×16 $(M = 256)$.
- S2. Images: 320×240; Templates: 32×32 $(M = 1{,}024)$.
- S3. Images: 640×480; Templates: 64×64 $(M = 4{,}096)$.

2. *hIDA has not been considered in the case $p = 1$ since it deploys the FFT.*

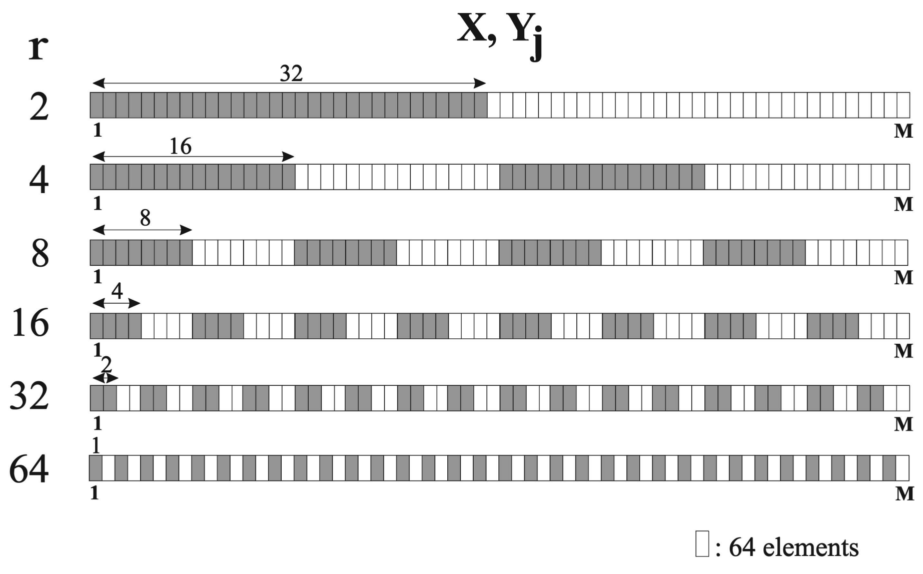

Fig. 1. The adopted partitioning schemes of vectors X and Y_j as a function of parameter r in the case $M = 64 \times 64$.

- S4. Images: 1,280 × 960; Templates: 128 × 128 ($M = 16{,}384$).

This choice is suitable to both PK and FFT since PK requires power of two dimensions for the template size and the FFT optimally fits into power of two image sizes. Since the proposed approach can be applied to any L_p norm-based dissimilarity measure, although it is of limited practical relevance, at the end of this section, we also compare IDA to the FS in the case $p = 3$ for $S1$.

5.1 Parameters of the Algorithms

For what concerns Experiment 1, for each pattern-matching instance, the threshold D was set to two different values, referred to as th and $th2$. The value th is chosen to be very close to the global minimum, i.e., to the value $\|X - Y_W\|_p^p$, where Y_W is the best matching image subwindow.

In the case $p = 2$, since the authors' code of the PK algorithm requires the parameter *Maximum Mean Difference* (MMD),

$$MMD = \sqrt{\frac{SSD_{min}}{M}} \tag{20}$$

so that the threshold D is computed as

$$D = MMD^2 \cdot M, \tag{21}$$

we set th as

$$th = \left\lceil \sqrt{\frac{\|X - Y_W\|_2^2}{M}} \right\rceil. \tag{22}$$

The second value, $th2$, was chosen to be less selective than th, i.e., 10 percent higher:

$$th2 = th \cdot 1.10. \tag{23}$$

In the case $p = 1$, the threshold values th and $th2$ were set as follows:

$$th = \|X - Y_W\|_1^1 + 1, \tag{24}$$

$$th2 = th \cdot 1.05, \tag{25}$$

with $th2$ tighter than in (23) in order to compensate for the reduced dynamics of the dissimilarity function.

Instead, in Experiment 2, only the case $D = th$ was considered.

The parameters of the algorithms were kept constant throughout all experiments. In particular, for what means the PK algorithm, the number of Walsh-Hadamard kernels was set to the default value suggested by the authors in their code. As for IDA and hIDA, we partitioned templates and image subwindows into r equally sized subvectors of adjacent elements so that, as pointed out in Section 3, only a single incremental calculation process is required by the algorithms. With such a choice, r is the only parameter of the IDA algorithm. In order to further limit the degrees of freedom of the adopted partitioning scheme, we constrained r to be a power of 2 ranging from 2 up to the template side (i.e., 16, 32, 64, and 128 in both experiments), as described graphically in Fig. 1 in the case of a 64 × 64 pixels template. In both experiments, the results yielded by the IDA algorithm with the choice of parameter r yielding the best performance are referred to as *IDA opt*. We also show the results yielded by the IDA and hIDA using some given r values that can be regarded as generally good default choices for the considered template sizes. In particular, parameter r was set to {4, 4, 8, 8} for the template side equal, respectively, to {16, 32, 64, 128}, since in most cases the IDA approach is more efficient if a higher r is used with bigger templates. For what concerns the hIDA algorithm, it requires the setting of an additional parameter, i.e., the threshold on the percentage of candidates pruned within the prediction step that determines whether the search process is carried out using IDA or the fastest between the FS and the FFT. This parameter was set to 50 percent, 50 percent, 70 percent, and 85 percent for the template side equal, respectively, to {16, 32, 64, 128}. The prediction step analyzes a subset of points obtained by selecting one point out of 20 along both of the directions within the search area.

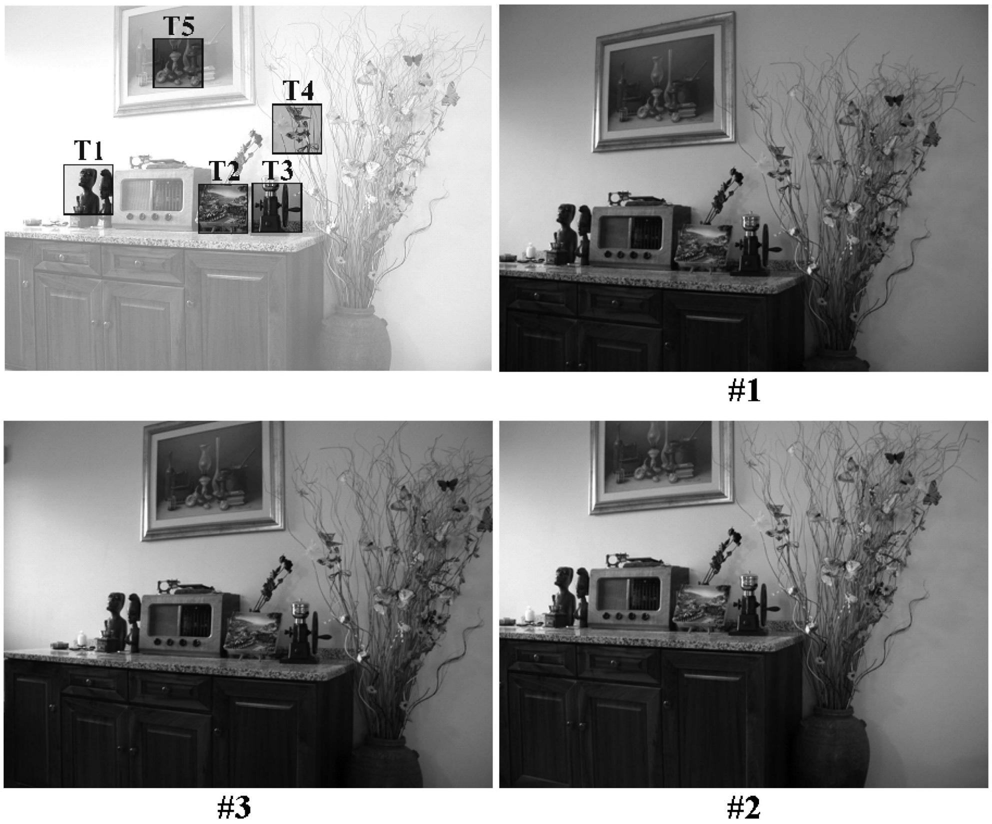

Fig. 2. Experiment 1: The five templates (top, left) and the three test images.

5.2 Experiment 1

In this experiment, we first extracted five templates from an image and, then, we took three other shots of the same scene from slightly different positions (templates and images are shown in Fig. 2). All templates and images were scaled according to scales $S1$, $S2$, $S3$, and $S4$. Hence, for each of the four scales, we obtained five templates and three images, resulting overall in 60 pattern-matching instances. Hereinafter, each instance will be denoted by the pair *test image number-template number* (e.g., the pair $1-2$ denotes template 2 matched into test image 1).

Experimental results are given out as ratios of execution times (i.e., speedups) measured on the benchmarking platform.

Figs. 3 and 4 report the speedups yielded by IDA, hIDA, PK, and FFT with respect to the FS SSD-based algorithm setting, respectively, $D = th$ and $D = th2$. For each pattern-matching instance of each scale, the first two bars concern IDA: The leftmost regards the value of parameter r providing the highest speedup (i.e., *IDA opt*), the other, tagged as *IDA* r, $r \in \{4, 8\}$, the default value of r. Then, the third bar, tagged as *hIDA* r, $r \in \{4, 8\}$, regards hIDA, with r set to the same default value as IDA. Finally, the last two bars show the speedup yielded, respectively, by the PK and FFT-based algorithm.

As far as Fig. 3 is concerned, the IDA algorithm, using the optimal r and the default r, turns out to be very effective in most instances of $S1$, $S2$, and $S3$. As a matter of fact, with these scales, IDA *opt* and IDA r are both always much faster than the FFT-based algorithm. Furthermore, IDA *opt* does not outperform PK in only five instances out of 45 (i.e., $1-1$, $2-1$ at $S1$ and $1-1$, $3-1$, $3-4$ at $S2$) while IDA r does not outperform PK in only six instances out of 45 (the previous 5 plus $3-5$ at $S1$).

For what concerns $S4$, though the computational efficiency of the FFT algorithm is very high due to the image and template sizes ($\text{speedup} = 20.5$), IDA algorithms run notably faster in nine instances out of 15 (reaching a maximum speedup as high as 184.7 in instance $2-1$). As for hIDA, it is almost as fast as IDA in the former nine instances and provides substantially the same speedup as the FFT-based algorithm in the remaining 6. Hence, at this scale, the effectiveness of the prediction step is clearly shown since hIDA allows for deploying the template-matching algorithm more suited to the data by correctly selecting the faster one between IDA and the FFT. This is also demonstrated in $S1$, $S2$, and $S3$, where IDA clearly outperforms the FFT and hIDA provides substantially the same computational savings as IDA. It is also interesting to note that, at $S4$, the average speedup yielded by hIDA is 77.8, with a lowest speedup equal to 20.0, which is very similar to the constant speedup yielded by the FFT. As a result of these considerations, it turns out that IDA is particularly suited to small size images, while hIDA provides the best overall performance.

Moreover, for what concerns a comparison between IDA *opt* and IDA r, it can be noticed that the choice of a default r in most instances does not notably affect the performance compared to the optimal choice, the speedups yielded by IDA r being generally very close to those of IDA *opt*.

With regard to the PK algorithm, at $S1$ and $S2$, it turns out slower than IDA and hIDA in most instances but always

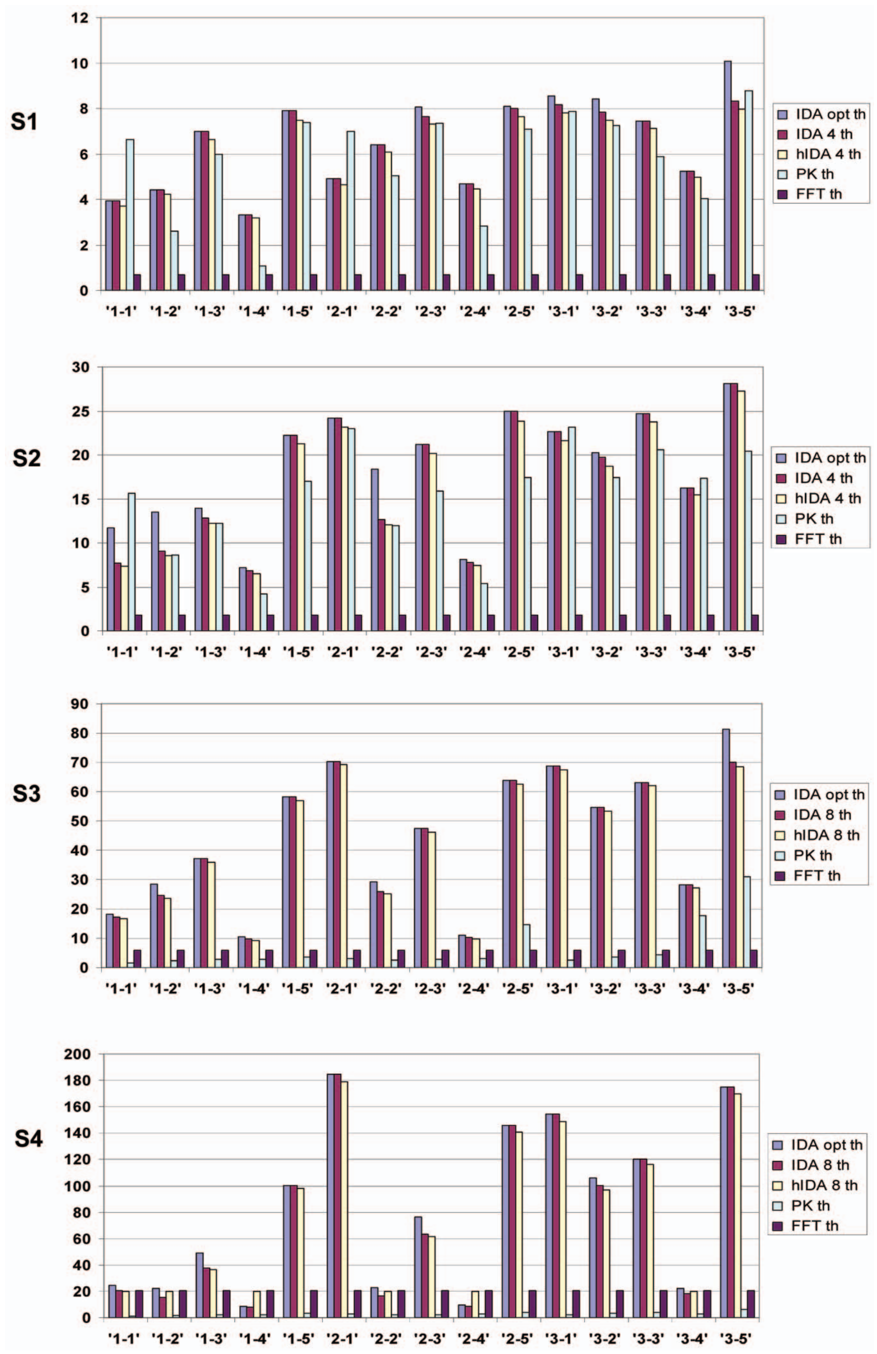

Fig. 3. Experiment 1: Measured speedups in the SSD case, $D = th$.

notably faster than the FFT algorithm. At $S3$ and $S4$, PK is significantly outperformed by FFT in most instances and turns out to always be slower than IDA and hIDA.

The results reported in Fig. 4 substantially confirm the outcomes of the previous comparative analysis. Focusing our attention on $S4$, the hIDA algorithm also with the threshold value $th2$ is able to yield, in the best cases, speedups comparable to the IDA and, in the worst cases, speedups similar to the FFT. Furthermore, also in Fig. 4, it can be noticed that, at $S3$ and $S4$, PK is always slower than IDA (30 out of 30) and, in most instances, slower than the FFT (29 out of 30).

Finally, Fig. 5 shows the speedups yielded by IDA and SSDA with respect to the FS SAD-based algorithm (i.e., $p = 1$) with $D = th$ and $D = th2$. For each instance of this experiment, the first and third bars refer to IDA with the optimal value of parameter r (respectively, for $D = th$ and $D = th2$), the second and fourth bars to IDA with the default choice of r (respectively, for $D = th$ and $D = th2$). The last two bars refer to SSDA, respectively, for $D = th$ and $D = th2$. The figure shows that IDA is always much faster than the FS algorithm, with speedups ranging from about 5 (worst case) up to more than 500 (best case). It is worth pointing out the ranges of the measured speedups with the less favorable parameter settings (i.e., default r and less selective threshold $D = th2$, fourth bar of each instance): from 5.2 to 34.9 at $S1$, from 8.4 to 116.6 at $S2$, from 10.4 to 226.7 at $S3$, and from 9.0 to 346.4 at $S4$. For

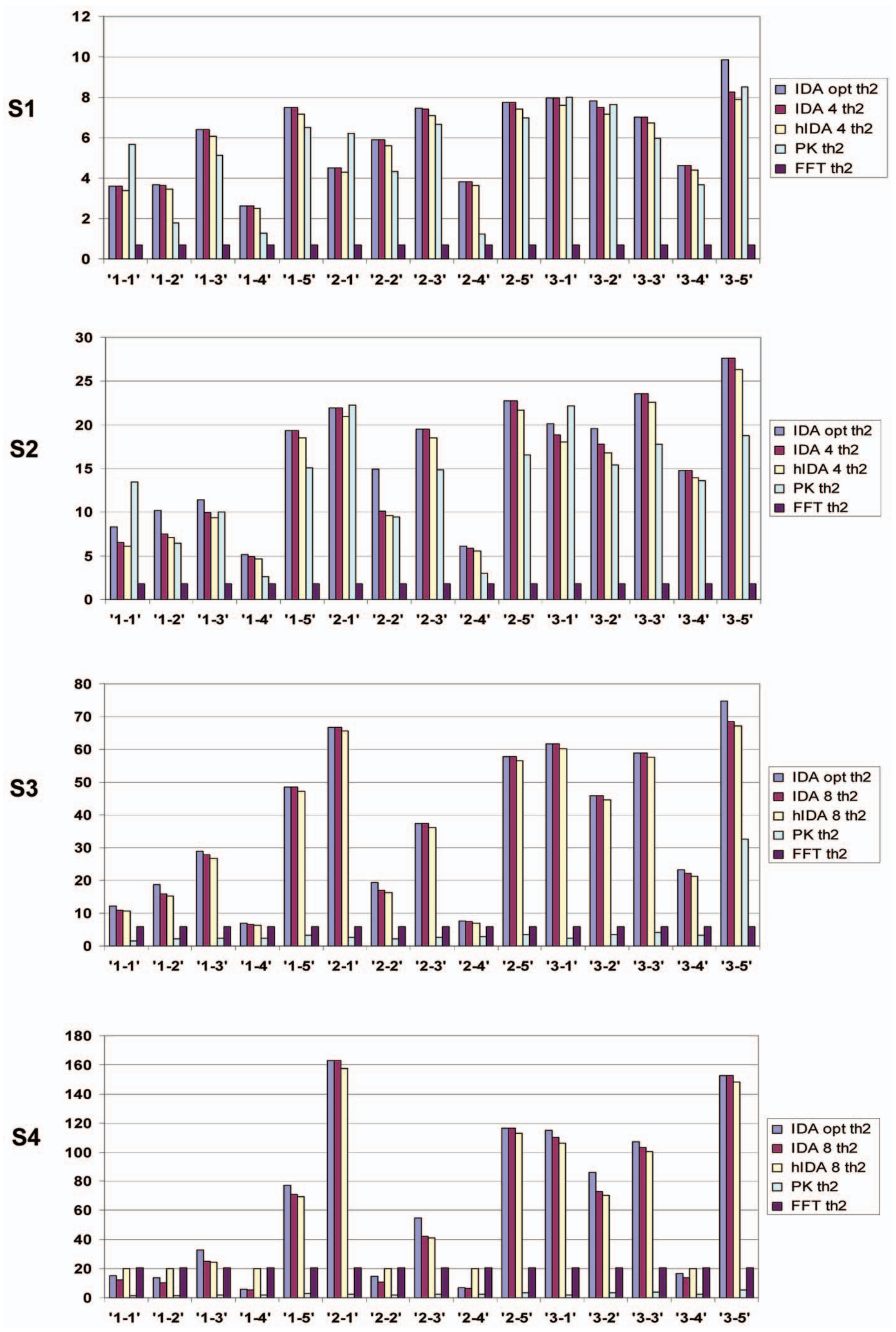

Fig. 4. Experiment 1: Measured speedups in the SSD case, $D = th2$.

what means SSDA, the reported speedups are always dramatically lower than those yielded by IDA algorithms, with the algorithm sometimes being even slower than the FS. This has to be ascribed to the significant number of test operations (as high as M) performed by SSDA, which slow down the method, particularly at large scales. Furthermore, this is also due to the fact that, as explained in Section 2, in order to guarantee the exhaustiveness of the search, the pruning threshold for SSDA must be set to a constant value higher than the global minimum (i.e., th or $th2$), while this algorithm was originally conceived to perform best with a varying D much lower than the global minimum (i.e., in a nonexhaustive scenario).

5.3 Experiment 2

Experiment 2 was aimed at assessing the performance of the examined algorithms on a larger data set. This experiment includes a total of 120 images chosen between three databases: MIT [17], medical [18], and remote sensing [19]. The MIT database is mainly concerned with indoor, urban, and natural environments, plus some object categories such as cars and fruits. The two other databases are composed, respectively, of medical (radiographs) and remote sensing (Landsat satellite) images. All images have been subdivided into four groups of 30 images, each group being characterized by a different scale and with scales being the same as in Experiment 1 (i.e., $S1, \cdots, S4$). For each image, 10 templates were randomly selected among those

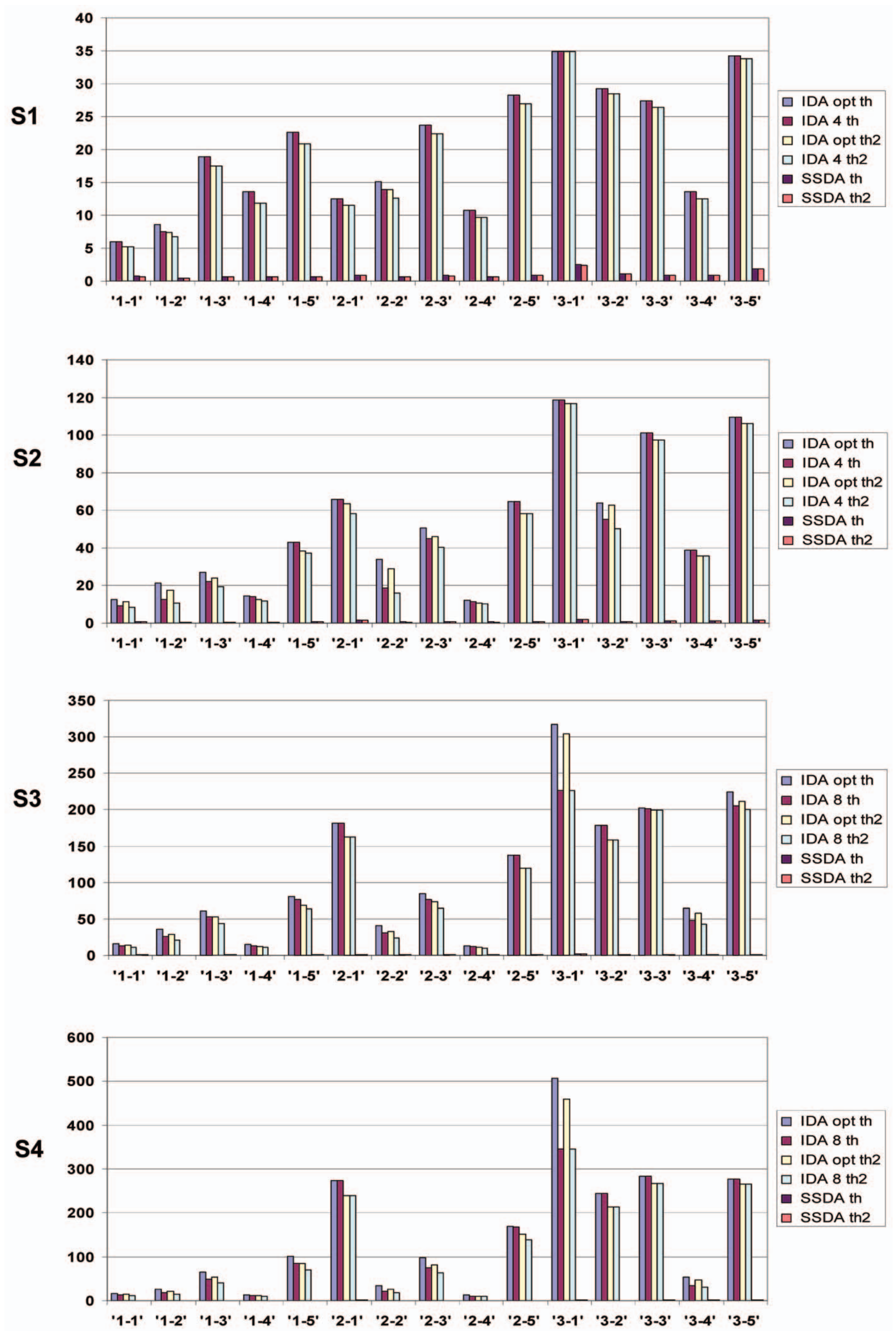

Fig. 5. Experiment 1: Measured speedups in the SAD case.

showing a standard deviation of pixel intensities higher than a threshold (i.e., 45). Then, five different levels of independent and identically distributed zero-mean Gaussian noise, referred to as $N1, \cdots, N5$, were added to each image. The five noise levels range from very low noise to very high noise, the variances of the Gaussian distribution being, respectively, 1.3, 2.6, 5.1, 7.7, and 10.2.[3] Hence, overall, each algorithm was tested against 6,000 pattern-matching instances. Fig. 6 shows five images of the data set. For each of them, the five corresponding images with increasing (from left to right) noise levels are also shown.

3. Corresponding to 0.005, 0.01, 0.02, 0.03, and 0.04 on normalized pixel intensities ranging within [0, 1].

Due to the large size of the data set, for each scale and noise level, we provide a global indication (in terms of mean μ and standard deviation σ) of the measured speedups with respect to the FS algorithm on the same benchmark platform as in Experiment 1. Moreover, in order to better assess the behavior of the algorithms, we show two additional descriptors that allow for measuring the asymmetry of the distribution. These descriptors, referred to as σ_- and σ_+, represent the square root of the mean square error with respect to μ of the population, respectively, below and above μ.

Fig. 7 reports, for $p = 2$ and $D = th$, the performance of IDA *opt*, IDA, hIDA, PK, and FFT. The figure shows that, at $S1$ and $S2$, IDA, hIDA, and PK substantially yield

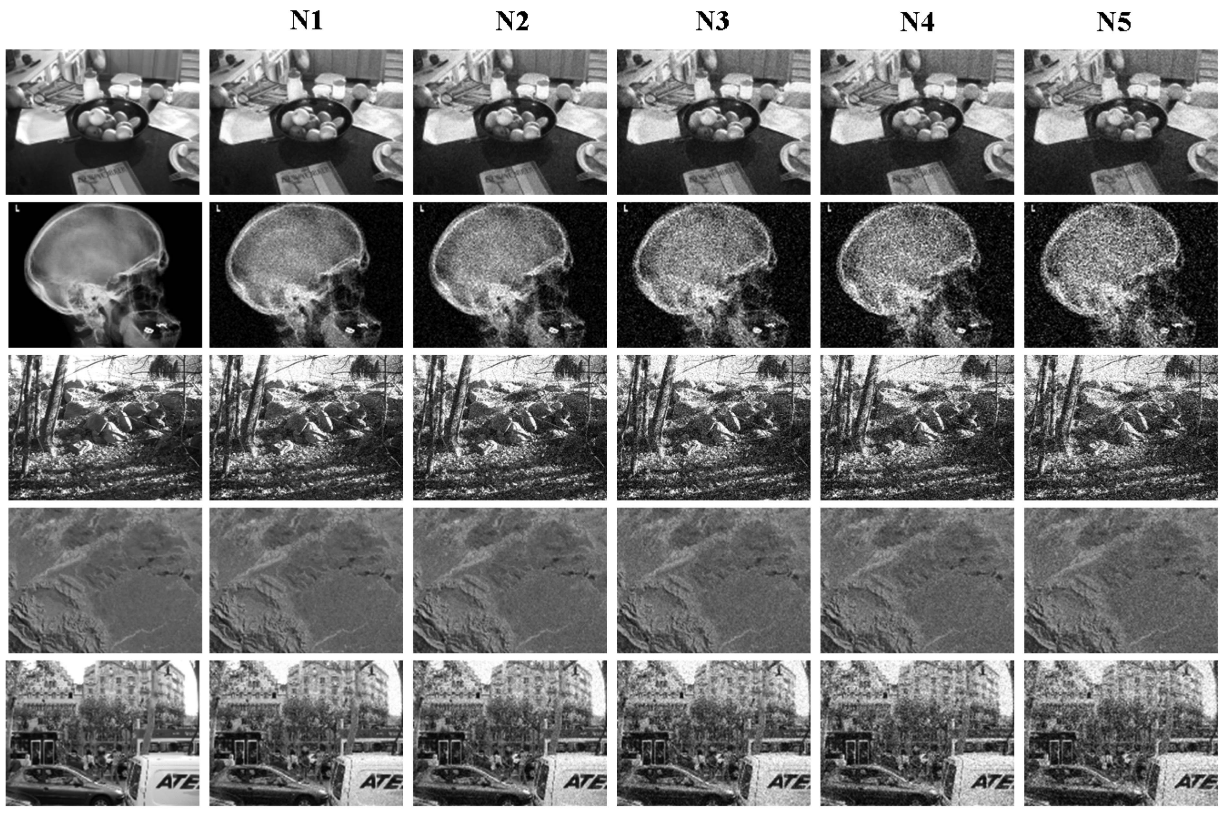

Fig. 6. Five images of the data set used in Experiment 2. Each row shows the noise-free image (leftmost) from which templates were extracted, together with the five corresponding noisy images.

comparable speedups. The algorithms are always notably faster than the FFT, although their efficiency decreases significantly with increasing noise. Nevertheless, on average, IDA *opt*, IDA, and hIDA turn out to be more robust to noise than PK. Furthermore, it is worth pointing out that, at $S2$, IDA *opt* and IDA always provide mean speedups higher than PK. Moreover, in both scales, IDA *opt* and IDA are always slightly more efficient than hIDA. For what concerns $S3$, IDA *opt*, IDA, and hIDA always yield much higher speedups than PK, which in turn is also clearly outperformed by the FFT. Conversely, our algorithms always perform better than the FFT. However, though all data dependent algorithms are significantly affected by noise, our algorithms, according to the larger dynamic of the mean speedup, show a more substantial decrease of the computational efficiency with increasing noise. The comparison between our algorithms indicates that hIDA tend to perform slightly better at higher noise levels. As for $S4$, IDA *opt* and IDA always dramatically outperform PK and, at noise levels $N1$, $N2$, and $N3$, they are much faster than FFT. However, at higher noise levels (e.g., $N5$ for IDA *opt* and $N4$ and $N5$ for IDA), the FFT turns out to be more effective. Nevertheless, it is worth observing that hIDA always outperforms PK and the FFT, resulting in the best choice for large images.

Overall, the results of Experiment 2 confirm the trend inferable from Experiment 1: At $S1$, IDA and PK perform best; at $S2$, IDA is the best choice; at $S3$, IDA and hIDA are comparable and yield the best results; at $S4$, hIDA is the best performing algorithm.

The standard deviation, σ, reported in Fig. 7 confirms, on this larger data set, the notable data dependency of IDA *opt*, IDA, hIDA, and PK highlighted in Experiment 1. Although, for these algorithms, σ is significantly high at high noise levels, it is worth observing that, in all such cases, the distribution of the speedup is clearly asymmetric, with the right tail more pronounced (that is, σ_+ is sensibly greater than σ_-). Hence, the values that differ most from the mean occur for speedups above μ, while speedups lower than the mean show less dispersion with regard to μ.

For what concerns $p = 1$, Fig. 8 reports the speedups yielded by IDA *opt*, IDA, and SSDA with respect to the FS SAD-based algorithm with $D = th$. Similarly to Experiment 1, at each scale, IDA *opt* and IDA dramatically outperform SSDA and FS, always yielding substantial speedups, thus confirming the efficiency of the proposed approach on this larger data set. Nevertheless, it is worth observing that the speedups are significantly affected by noise. The figure also confirms the significant data dependency of IDA *opt*, IDA, and SSDA. Similarly to $p = 2$, the distributions of the speedup values are clearly asymmetric and right tailed; this behavior gets more pronounced as the noise level increases.

5.4 Experiment with $p > 2$

We report here the results of an experiment addressing the case $p = 3$. In particular, Table 1 shows the mean and standard deviation of the speedups yielded by IDA *opt* and IDA 4 with regard to the FS algorithm on the data set used

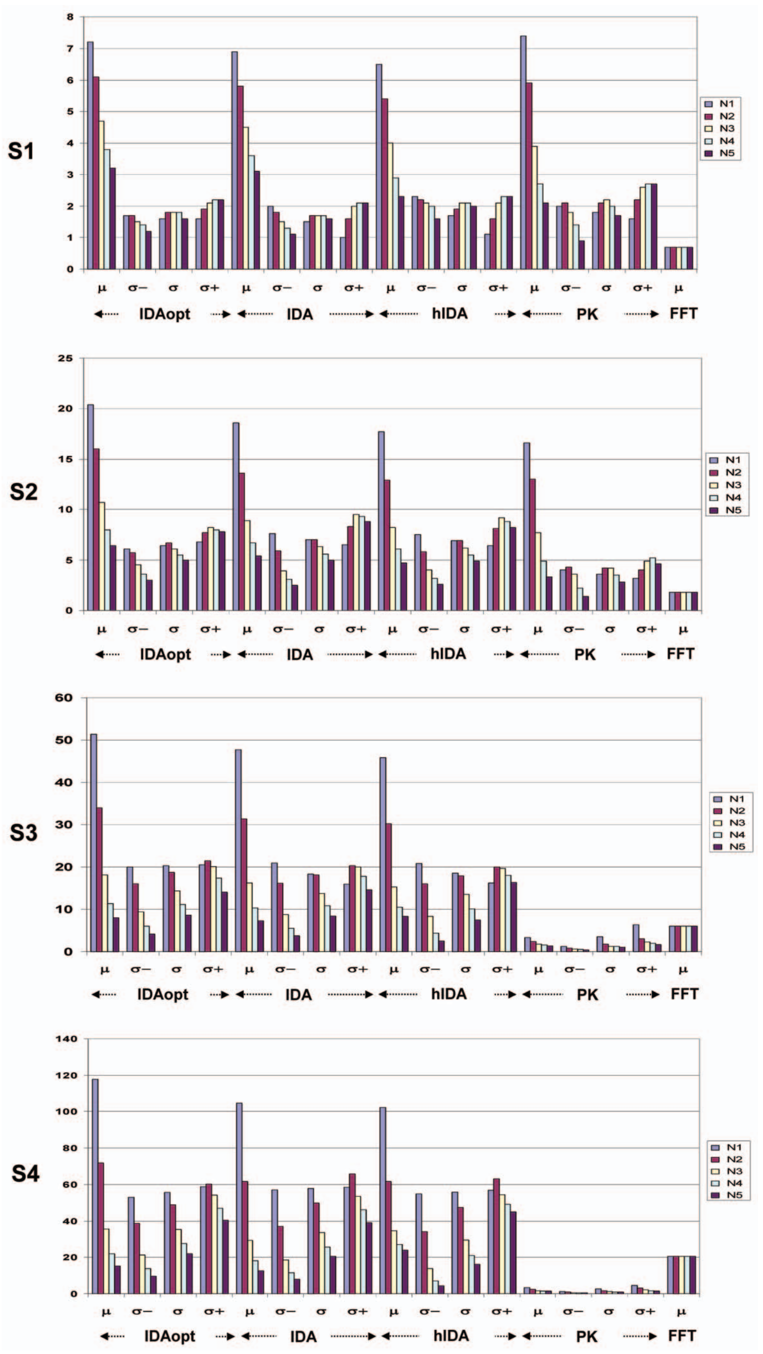

Fig. 7. Experiment 2: Speedups yielded by the exhaustive techniques versus the FS algorithm at the four scales, SSD case $(p = 2)$.

in Experiment 2 at $S1$. As can be noted, also in this case, both the considered algorithms run significantly faster than the FS approach.

6 Conclusions

We have proposed a novel method for fast FS-equivalent pattern matching with L_p norm-based dissimilarity functions. The method relies on increasingly tighter sufficient conditions capable of pruning many candidates at a small computational cost. The proposed experiments, comprised of thousands of pattern-matching instances, showed that, for what concerns the SSD function, with images and templates of small and medium size, the proposed algorithm, referred to as IDA, overall performs better than the FFT and PK algorithms, which are state-of-the-art FS-equivalent pattern-matching approaches. As image and template sizes grow, though IDA gets increasingly faster than the FS and PK, in some cases it happens to be slower than the FFT. Hence, we have proposed a further approach, referred to as hIDA, which provides the best overall performance in such cases. Finally, the experimental results with the SAD function show that IDA yields substantial speedups (up to more than two orders of magnitude) with respect to the standard FS algorithm and to the SSDA algorithm.

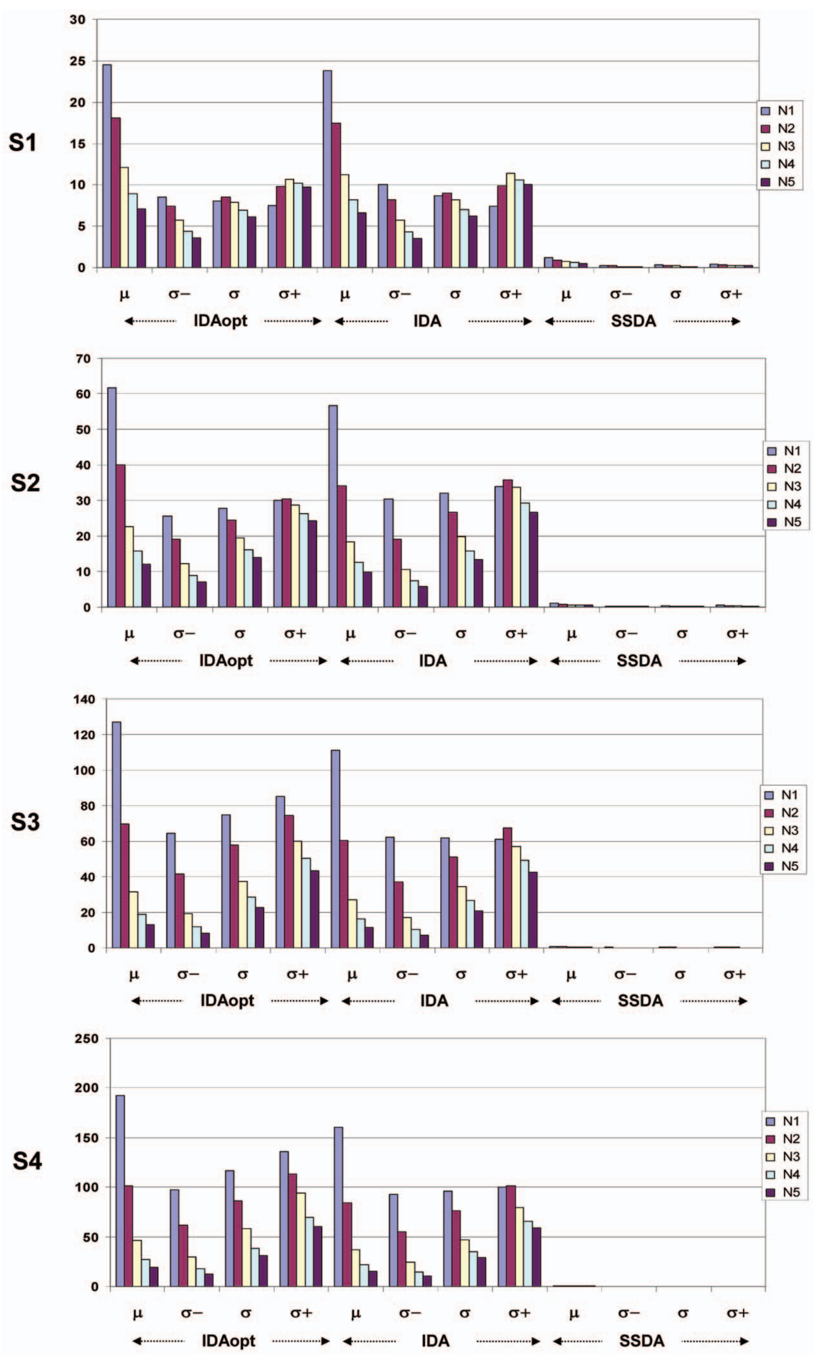

Fig. 8. Experiment 2: Speedups yielded by the exhaustive techniques versus FS algorithm at the four scales, SAD case $(p = 1)$.

Appendix

TABLE 1
Speedups Yielded by IDA versus FS in the Case $p = 3$, Measured on the Images of Experiment 2 at $S1$

S	N	IDA opt		IDA 4	
		μ	σ	μ	σ
S1	N1	5.2	0.8	5.0	0.8
	N2	4.6	0.9	4.4	0.9
	N3	3.9	0.9	3.7	1.0
	N4	3.4	0.9	3.2	0.9
	N5	3.0	0.9	2.9	0.9

On the Generalization of the Proposed Method

An interesting issue raised by a reviewer concerned whether the proposed approach could be generalized to an arbitrary metric. According to the notation adopted throughout the paper, we indicate the arbitrary metric used to evaluate dissimilarities as $d(X, Y_j)$ and the corresponding *partial* distances induced by P as $d(X, Y_j)_{S_t}$. Though the triangular inequality can still be applied to subvectors

$$d(X, Y_j)_{S_t} \geq \left| d(X, 0)_{S_t} - d(Y_j, 0)_{S_t} \right|, \quad t = 1, \ldots r, \tag{26}$$

summation of both members now yields

$$\sum_{t=1}^{r} d(X,Y_j)_{S_t} \geq \sum_{t=1}^{r} \left| d(X,0)_{S_t} - d(Y_j,0)_{S_t} \right|. \tag{27}$$

Therefore, a sufficient condition for the right-hand side of (27) to be a lower bound of $d(X,Y_j)$ is

$$d(X,Y_j) \geq \sum_{t=1}^{r} d(X,Y_j)_{S_t}. \tag{28}$$

Unfortunately, the above inequality does not hold for an arbitrary metric (e.g., for the L_p-distance when $p > 1$).

Interestingly, though perhaps of rather limited practical relevance, it is possible to define at least one class of metrics that allows for the generalization of our method. Letting each of $d_t(\cdot)$, $t = 1, \dots r$, and $\tilde{d}(\cdot)$ be a metric, we define

$$\bar{d}(X,Y_j) = \left(\sum_{t=1}^{r} d_t(X,Y_j)_{S_t} \right) + \tilde{d}(X,Y_j), \tag{29}$$

with distances between subvectors denoted according to the usual notation. It is straightforward to prove that function $\bar{d}(X,Y_j)$ is a metric and that (29) defines a class of distances satisfying sufficiently condition (28), with

$$\bar{d}_0(X,Y_j) = \sum_{t=1}^{r} d_t(X,Y_j)_{S_t} \tag{30}$$

being the smallest of such distances.

ACKNOWLEDGMENTS

The authors wish to thank Professor Massimo Ferri in the Department of Mathematics at the University of Bologna for his valuable suggestions concerning the material presented in the Appendix, Professor Antonio Torralba and CSAIL at the Massachusetts Institute of Technology for the use of the MIT database, Professor Rainer Koster and the Institute for Clinical Radiology and Nuclear Medicine of the Lukas Hospital Neuss for the use of the medical image database, and NASA for the use of the remote sensing image database.

REFERENCES

[1] Y. Hel-Or and H. Hel-Or, "Real-Time Pattern Matching Using Projection Kernels," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 9, pp. 1430-1445, Sept. 2005.

[2] D.I. Barnea and H.F. Silverman, "A Class of Algorithms for Digital Image Registration," *IEEE Trans. Computers,* vol. 21, no. 2, pp. 179-186, Feb. 1972.

[3] F. Tombari, S. Mattoccia, and L. Di Stefano, "Template Matching Based on the lp Norm Using Sufficient Conditions with Incremental Approximations," *Proc. IEEE Int'l Conf. Advanced Video Surveillance Systems,* Nov. 2006.

[4] H.L. Royden, *Real Analysis,* third ed. Prentice Hall, 1988.

[5] F. Crow, "Summed-Area Tables for Texture Mapping," *Computer Graphics,* vol. 18, no. 3, pp. 207-212, 1984.

[6] M. McDonnel, "Box-Filtering Techniques," *Computer Graphics and Image Processing,* vol. 17, pp. 65-70, 1981.

[7] P. Viola and M.J. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.

[8] J.P. Lewis, "Fast Template Matching," *Proc. Conf. Vision Interface,* pp. 120-123, May 1995.

[9] B. Zitová and J. Flusser, "Image Registration Methods: A Survey," *Image and Vision Computing,* vol. 21, no. 11, pp. 977-1000, 2003.

[10] A. Goshtasby, *2-D and 3-D Image Registration for Medical, Remote Sensing and Industrial Applications.* John Wiley & Sons, 2005.

[11] C. Sun, "Moving Average Algorithms for Diamond, Hexagon, and General Polygonal Shaped Window Operations," *Pattern Recognition Letters,* vol. 27, no. 6, pp. 556-566, 2006.

[12] C.H. Lee and L.H. Chen, "A Fast Motion Estimation Algorithm Based on the Block Sum Pyramid," *IEEE Trans. Image Processing,* vol. 6, no. 11, pp. 1587-1591, 1997.

[13] X.Q. Gao, C.J. Duanmu, and C.R. Zou, "A Multilevel Successive Elimination Algorithm for Block Matching Motion Estimation," *IEEE Trans. Image Processing,* vol. 9, no. 3, pp. 501-504, 2000.

[14] Z. Pan, K. Kotani, and T. Ohmi, "Fast Encoding Method for Vector Quantization Using Modified l_2-Norm Pyramid," *IEEE Signal Processing Letters,* vol. 12, no. 9, pp. 609-612, 2005.

[15] www.faculty.idc.ac.il/toky/Software/software.htm, 2008.

[16] http://sourceforge.net/projects/opencvlibrary, 2008.

[17] http://people.csail.mit.edu/torralba/images, 2008.

[18] www.data-compression.info/Corpora/LukasCorpus, 2008.

[19] http://zulu.ssc.nasa.gov/mrsid, 2008.

Federico Tombari received the BEng and MEng degrees from the University of Bologna, Italy, in 2003 and 2005, respectively. In 2006, he joined the Dipartimento di Elettronica, Informatica e Sistemistica (DEIS) at the University of Bologna as a PhD student and the Advanced Research Center on Electronic Systems (ARCES) at the University of Bologna as a collaborator. His research interests concern computer vision and pattern recognition. He is a student member of the IEEE and a member of the IAPR-IC.

Stefano Mattoccia received the MS degree in electronic engineering and the PhD degree in computer science engineering from the University of Bologna, Italy, in 1997 and 2002, respectively. He joined the Dipartimento di Elettronica Informatica e Sistemistica (DEIS) and the Advanced Research Center on Electronic Systems for Information and Communication Technologies (ARCES) at the University of Bologna, where he is currently a research associate on the Faculty of Engineering. His research interests include computer vision, image processing, and computer architectures. He is the author of more than 30 refereed papers and two patents. He is a member of the IEEE and the IAPR.

Luigi Di Stefano received the degree in electronic engineering from the University of Bologna, Italy, in 1989 and the PhD degree in electronic engineering and computer science from the Department of Electronics, Computer Science and Systems (DEIS) at the University of Bologna in 1994. In 1995, he spent six months at Trinity College Dublin as a postdoctoral fellow. He is currently an associate professor at DEIS. He also joined the Advanced Research Centre on Electronic Systems "Ercole De Castro" (ARCES), a research center instituted at the University of Bologna in 2001. His research interests include computer vision, image processing, and computer architecture. He is the author of more than 70 papers and five patents. He is a member of the IEEE and the IAPR-IC.