Full search-equivalent pattern matching with Incremental Dissimilarity Approximations

Federico Tombari, Stefano Mattoccia, Luigi Di Stefano

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 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 skipping rapidly those candidates that cannot satisfy the matching criterion. The paper includes an experimental comparison between the proposed method and other full-search equivalent approaches known in literature, which proves the remarkable computational efficiency of our proposal.

I. 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 in 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 (PK)*, was recently proposed in literature. This method was shown to be much more efficient compared to the naive FS-approach as well as to the FFT. As regards 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 which deploys a succession of sufficient conditions, characterized by increasing effectiveness, for rapidly pruning those image

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

subwindows that cannot fulfill the matching criterion. The novel method, referred to as IDA (Incremental Dissimilarity Approximations) algorithm, is also compared to the FS, FFT, PK and SSDA algorithms. Experimental results concerning more than 6000 pattern matching instances prove that IDA significantly outperforms state-of-the-art approaches and can yield substantial speed-ups with respect to the FS.

The paper is structured as follows. Section II reviews previous work. Section III describes the IDA algorithm. Section IV proposes a variation of the basic IDA approach, called Hybrid IDA. Section V presents an experimental comparison between our methods and the other approaches considered throughout the paper. Conclusions are drawn in Section VI. Finally, in Appendix I we discuss the issue of generalizing our approach to an arbitrary metric.

II. Previous Work

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

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

where p is any positive real number [4].

Let now X be the template vector and $Y_1, \cdots, Y_N$ 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:

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

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.

A. 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 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.

B. Projection Kernels

The PK method [1] carries out fast full-search 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:

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

that 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] show also that PK is more effective with very small templates (i.e. of size 16×16 or 32×32).

C. Sequential Similarity Detection Algorithm

The SSDA method is a classical approach originally introduced to determine simple inequalities to speed-up SAD-based pattern matching. Let D be a threshold and X, Y_j 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, $\tilde{b}_j$, 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 not equivalent to the FS (i.e. non exhaustive). In particular, the choice of D determines a cost-performance trade-off: the higher D, the higher the mean number of calculations needed to evaluate the current candidate, and the higher the chance 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 increases along with b. Moreover, to obtain a more regular behavior the order of processed vector elements is randomly scrambled. However, it is practically unfeasible to determine a varying D which yield a FS-equivalent algorithm. Therefore, similarly to the other methods considered throughout the paper, we set D to a constant threshold higher than the global minimum: this turns SSDA into a FS-equivalent method.

III. INCREMENTAL DISSIMILARITY APPROXIMATIONS ALGORITHM

This section describes a novel signal domain method, referred to as *Incremental Dissimilarity Approximations* (IDA), aimed at speeding-up full-search 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 sub-vectors 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 sub-vectors (not necessarily with the same number of components) by defining a partition, P, of set $S = \{1, 2, \dots M\}$ into r disjoint sub-sets:

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

The minimum number of sub-vectors is 1, that is the vector is actually not partitioned into smaller sub-vectors, the maximum number is M, the vector partitioned into M one-dimensional disjoint sub-vectors. 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 sub-vectors associated with $S_t \in P$ as:

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

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

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

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

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

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

and summing up both members of the inequalities attains 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 which cannot represent a matching position. In fact, if the lower-bound of the dissimilarity function exceeds the threshold D that discriminates between matching and non-matching 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 sub-vectors 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

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

$$\sum_{t=1}^{r} \left| ||X||_{p,S_t} - ||Y_j||_{p,S_t} \right|^p \tag{16}$$

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 non-matching candidates.

Should condition (15) fail, the tightness of the lower-bounding function can be further increased by taking another sub-vectors 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 the r sub-vectors 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:

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

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

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

Should the last condition be not verified, the process ends up in computing the dissimilarity $||X - Y_j||^p_p$ by replacing $\left| ||X||_{p,S_l} - ||Y_j||_{p,S_l} \right|^p$ with $||X - Y_j||^p_{p,S_l}$ 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 non-exhaustive 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 $\left|||X||_{p,S_t} - ||Y_j||_{p,S_t}\right|^p$ can be computed much more rapidly, and independently from the sub-vectors cardinality, by calculating the partial norms using well-known fast incremental calculation schemes [5]–[7]. Consequently, since replacing differences of partial norms with the corresponding partial dissimilarities yields tighter bounding functions, the tighter is the bounding function the higher is its calculation time. 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 to partition 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 4 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 sub-vectors. In the 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, that 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.

IV. 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 of pruning 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 image size. Moreover, as it will be shown in Section V, although IDA turns out 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 to be not particularly effective, the matching process may be carried out using the faster between the FS and FFT approach, this choice made upon 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 in most cases a deterministic upper bound on response time.

Based on these considerations, we have devised the following variation to the basic IDA algorithm, referred to as *hybrid IDA* (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%, for small images, and 85%, for bigger images) hIDA carries out the matching process using the IDA algorithm, conversely it switches to the fastest between the FS and FFT algorithms. As it will shown in Section V, 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 between FS and FFT.

V. 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 Sec. II, 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 web site [15]) which refers to the case of the SSD function. With regards to the FFT-based algorithm, we used the very efficient implementation (*cvMatchTemplate* function) provided by *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 GB RAM running *Windows XP*.

Two different kinds of experiments were carried out. In *Experiment 1* we individually compare all the speed-up values yielded by the considered algorithms on an indoor sequence of 3 images acquired by means of a digital camera. This dataset is affected by real distortions since each image was taken at a slightly different pose with respect to that where the templates were extracted from. Instead, *Experiment 2* aims at evaluating the global performance of the algorithms on a large dataset of 120 images, with artificial noise at 5 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 4 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=1024)*
- S3) Images: 640×480; Templates: 64×64 *(M=4096)*
- S4) Images: 1280×960; Templates: 128×128 *(M=16384)*

This choice is suitable to both PK and FFT, since PK requires power of 2 dimensions for the template size and the FFT optimally fits into power of 2 image sizes. Since the proposed approach can be applied to any L_p-norm based dissimilarity measure, although of limited practical relevance, at the end of this section we also compare IDA to the FS in the case $p = 3$ for $S1$.

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

A. 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 parameter MMD (*Maximum Mean Difference*)

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

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% 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 dynamic 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 sub-vectors of adjacent elements, so that, as pointed

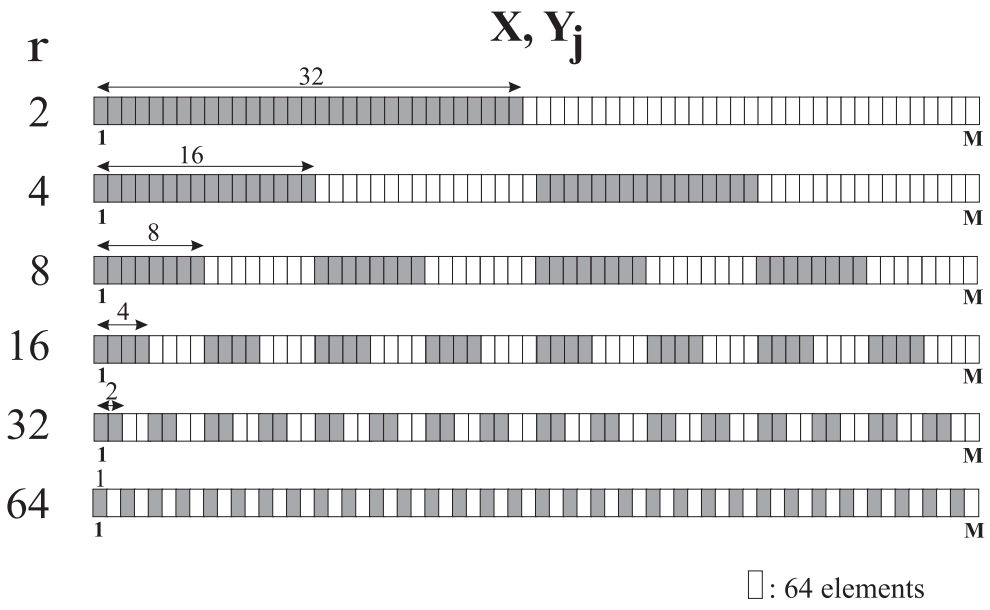

Fig. 1. The adopted partitioning schemes of vectors X and Y_j as a function of parameter r in the case $M = 64 \times 64$.

out in Section III, 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. $16, 32, 64, 128$ 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 show also the results yielded by IDA and hIDA using some given r values which 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 template side equal respectively to $\{16, 32, 64, 128\}$, as 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 $50\%, 50\%, 70\%, 85\%$ for template side equal respectively to $\{16, 32, 64, 128\}$. The prediction step analyses a subset of points obtained by selecting one point out of 20 along both the directions within the search area.

Fig. 2. Experiment 1: the 5 templates (top, left) and the 3 test images.

B. *Experiment 1*

In this experiment we first extracted 5 templates from an image, then we took 3 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 $S1$, $S2$, $S3$, $S4$. Hence, for each of the 4 scales we obtained 5 templates and 3 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. speed-ups) measured on the benchmarking platform.

Fig. 3 and Fig. 4 report the speed-ups yielded by IDA, hIDA, PK and FFT with respect to the

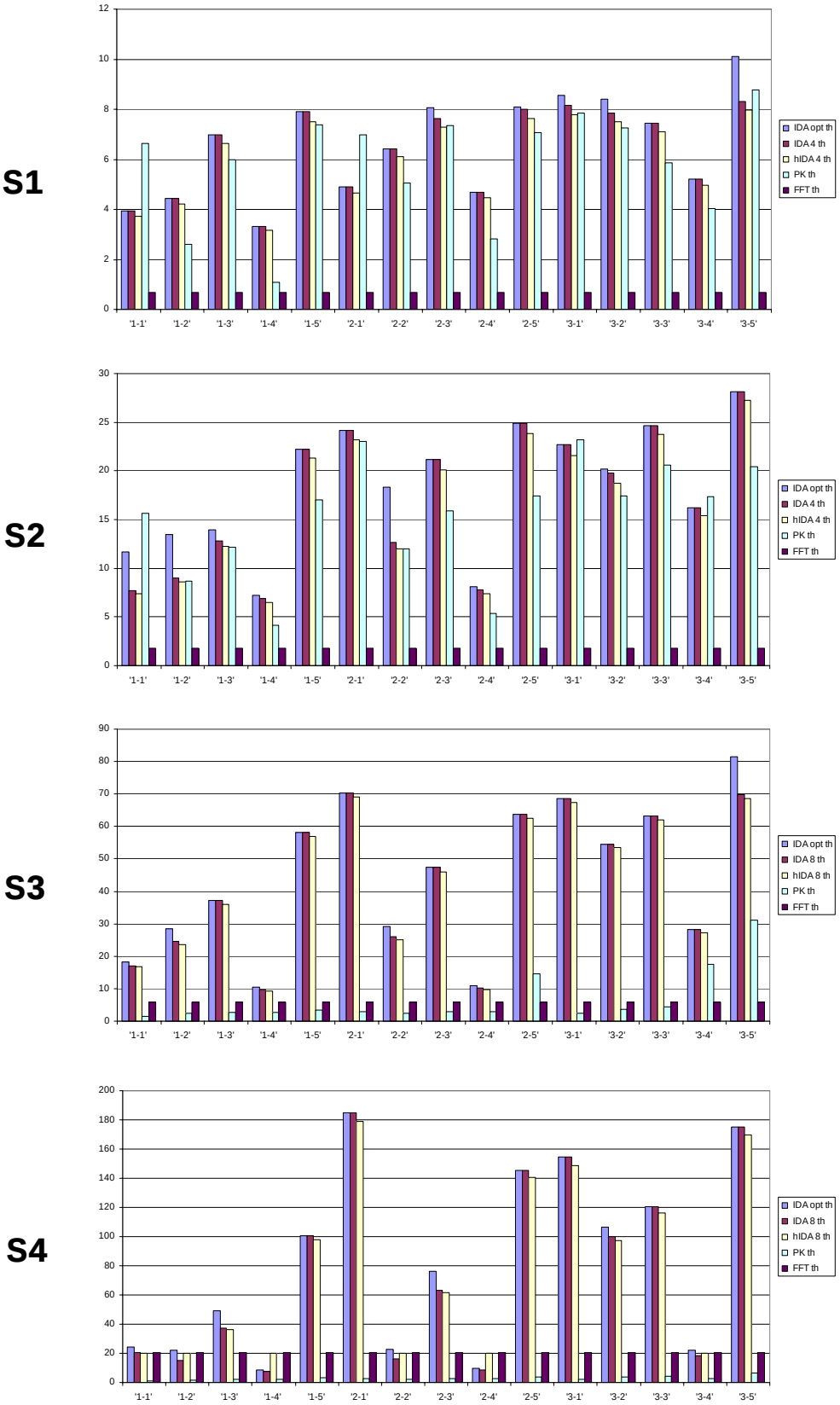

Fig. 3. Experiment 1: measured speed-ups in the SSD case, $D = th$.

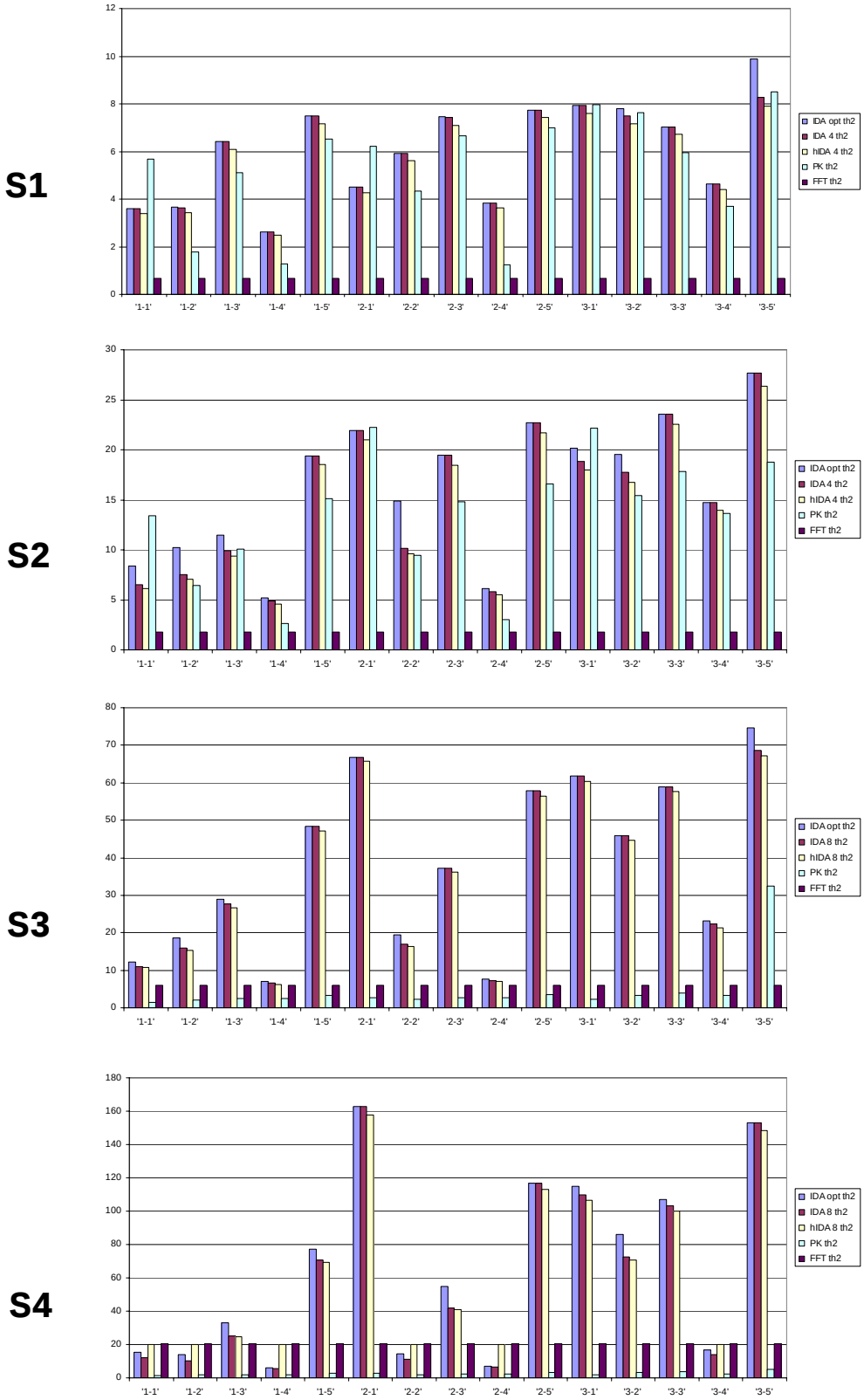

Fig. 4. Experiment 1: measured speed-ups in the SSD case, $D = th2$.

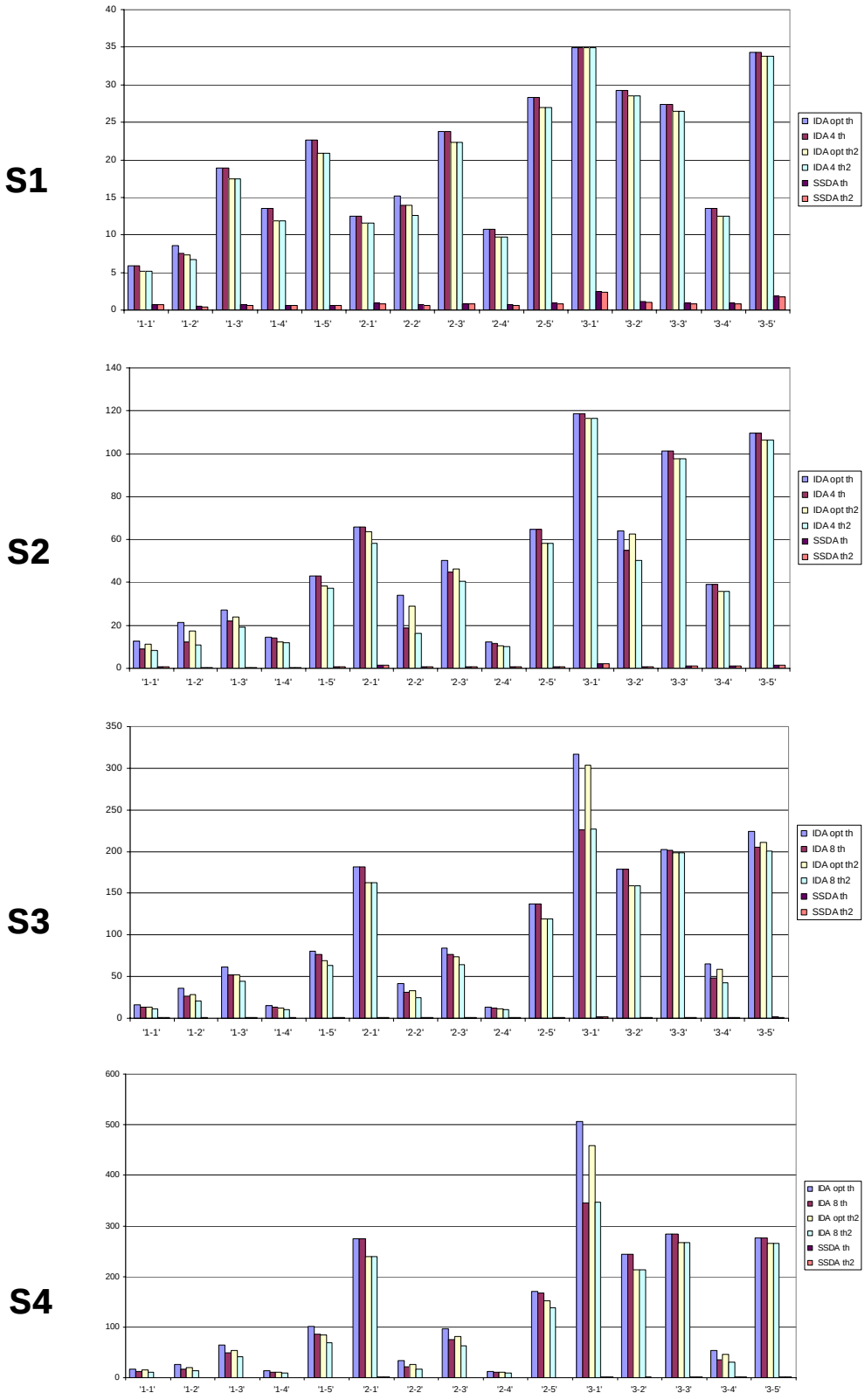

Fig. 5. Experiment 1: measured speed-ups in the SAD case.

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 speed-up (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 2 bars show the speed-up yielded respectively by the PK and FFT-based algorithm.

As far as Fig. 3 is concerned, the IDA algorithm, using the optimal r as well as 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 5 instances out of 45 (i.e. $1-1$, $2-1$ at $S1$ and $1-1$, $3-1$, $3-4$ at $S2$) while IDA r in only 6 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 (speed-up=20.5), IDA algorithms run notably faster in 9 instances out of 15 (reaching a maximum speed-up as high as 184.7 in instance $2-1$). As for hIDA, it is almost as fast as IDA in the former 9 instances and provides substantially the same speed-up 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 between IDA and the FFT. This is also demonstrated at $S1$, $S2$ 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 speed-up yielded by hIDA is 77.8, with a lowest speed-up equal to 20.0, which is very similar to the constant speed-up 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 affect notably the performance compared to the optimal choice, the speed-ups yielded by IDA r being generally very close to those of IDA *opt*.

As regards the PK algorithm, at $S1$ and $S2$ it turns out slower than IDA and hIDA in most instances, but always notably faster than the FFT algorithm. At $S3$ and $S4$ PK is significantly

outperformed by FFT in most instances, and turns out always 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 threshold value $th2$ is able to yield in the best cases speed-ups comparable to IDA and in the worst cases speed-ups similar to the FFT. Furthermore also in Fig. 4 it can be noticed that at $S3$ and $S4$ PK is always slower that IDA (30 out of 30) and in most instances slower that the FFT (29 out of 30).

Finally, Fig. 5 shows the speed-ups 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 speed-ups ranging from about 5 (worst case) up to more than 500 (best case). It is worth pointing out the ranges of the measured speed-ups 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 what means SSDA, the reported speed-ups are always dramatically lower than those yielded by IDA algorithms, with the algorithm being sometimes 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 II, 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 non-exhaustive scenario).

C. Experiment 2

Experiment 2 was aimed at assessing the performance of the examined algorithms on a larger dataset. This experiment includes a total of 120 images chosen between 3 databases: MIT [17], medical [18] and remote sensing [19]. The MIT database concerns mainly 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

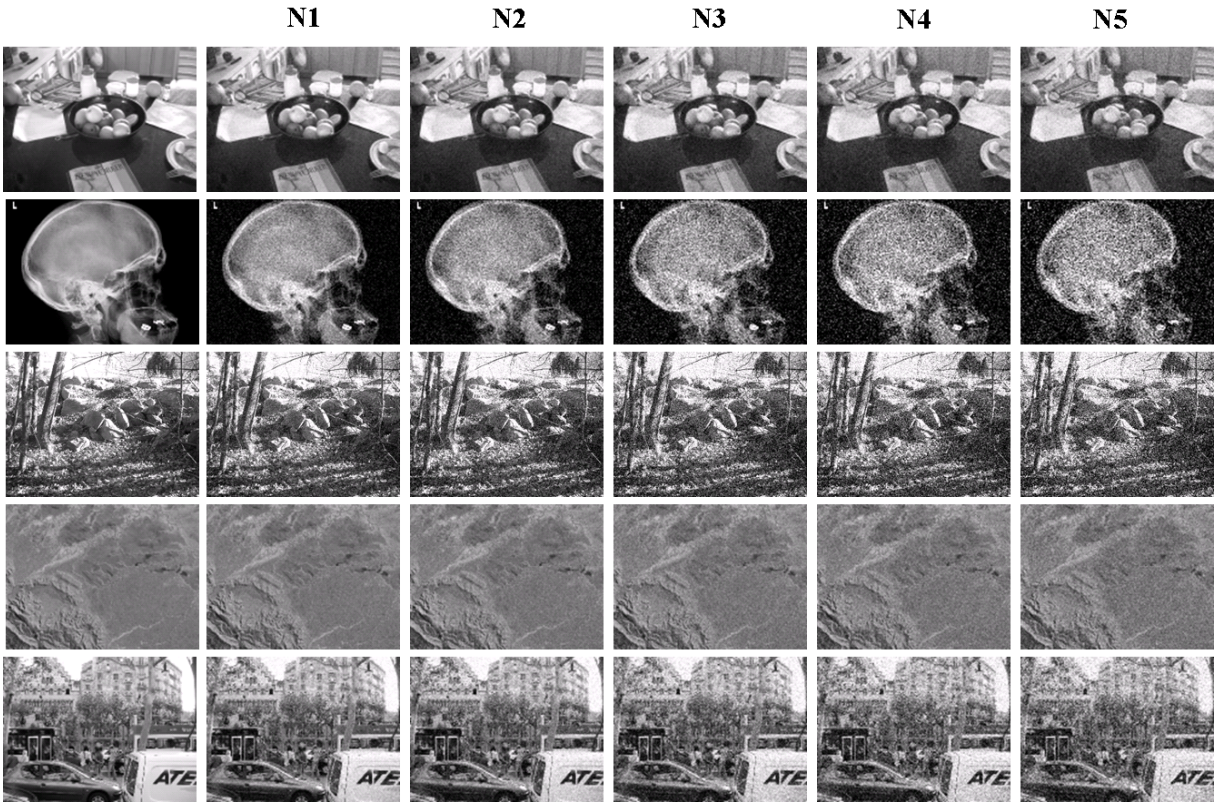

Fig. 6. 5 images of the dataset used in Experiment 2. Each row shows the noise-free image (leftmost) where templates were extracted from, together with the 5 corresponding noisy images.

satellite) images. All images have been subdivided into 4 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 showing a standard deviation of pixel intensities higher than a threshold (i.e. 45). Then, 5 different levels of i.i.d. zero-mean Gaussian noise, referred to as $N1, \cdots, N5$, were added to each image. The 5 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, 10.2 [3]. Hence, overall each algorithm was tested against 6000 pattern matching instances. Figure 6 shows 5 images of the dataset. For each of them, the 5 corresponding images with increasing (from left to right) noise levels are also shown.

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

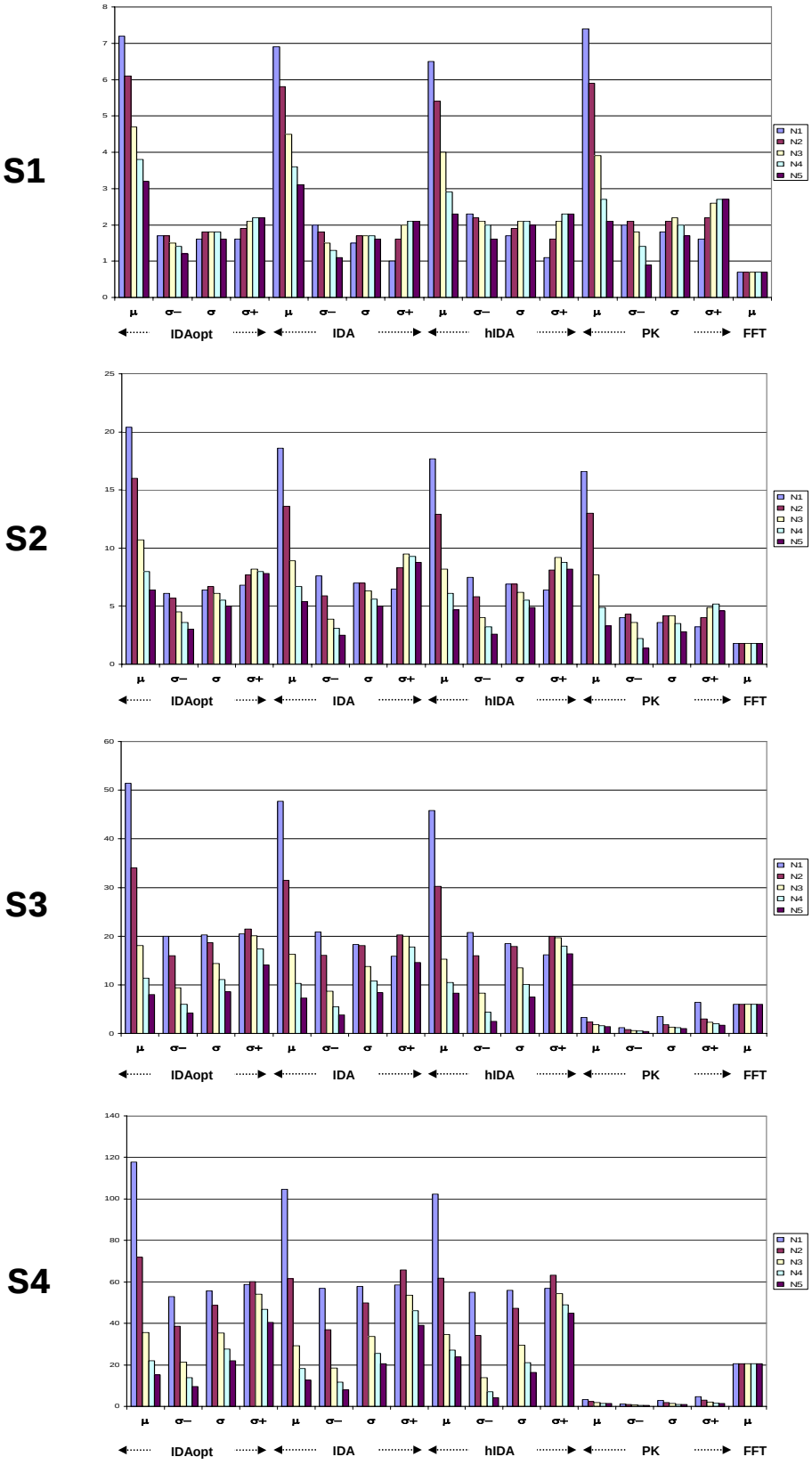

Fig. 7. Experiment 2: speed-ups yielded by the exhaustive techniques vs. FS algorithm at the 4 scales, SSD case ($p = 2$).

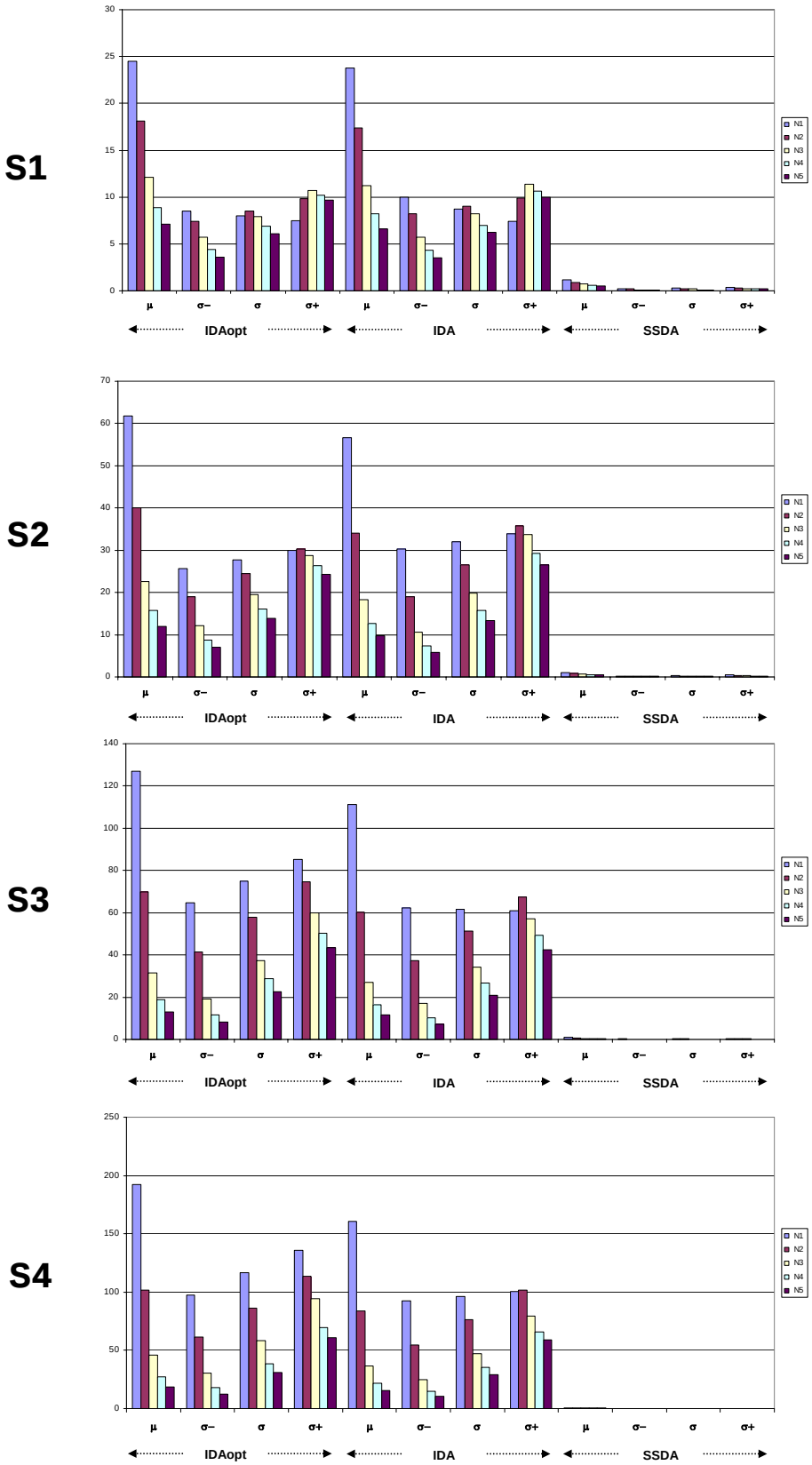

Fig. 8. Experiment 2: speed-ups yielded by the exhaustive techniques vs. FS algorithm at the 4 scales, SAD case ($p = 1$).

TABLE I

SPEED-UPS YIELDED BY IDA VS. 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

Due to the large size of the dataset, for each scale and noise level we provide a global indication (in terms of mean μ and standard deviation σ) of the measured speed-ups 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 yield, substantially, comparable speed-ups. 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 speed-ups 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, hIDA always yield much higher speed-ups than PK, which in turn is clearly outperformed also 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 speed-up, 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 $N1$,$N2$,$N3$, they also result much faster than FFT. However, at higher noise levels (e.g. $N5$ for IDA *opt*, $N4$ and $N5$ for IDA) the FFT turns out more effective. Nevertheless, it is worth observing that hIDA always outperforms PK and the FFT, resulting 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 dataset 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 speed-up is clearly asymmetric, with the right tail more pronounced (that is, σ_+ is sensibly greater than σ_-). Hence, the values which differ most from the mean occur for speed-ups above μ, while speed-ups lower than the mean show less dispersion with regards to μ.

For what concerns $p = 1$, Fig. 8 reports the speed-ups 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, yielding always substantial speed-ups, thus confirming the efficiency of the proposed approach on this larger dataset. Nevertheless, it is worth observing that the speed-ups 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 speed-up values are clearly asymmetric and right-tailed, this behavior getting more pronounced as the noise level increases.

D. Experiment with $p > 2$

We report here the results of an experiment addressing the case $p = 3$. In particular, Table I shows the mean and standard deviation of the speed-ups yielded by IDA *opt* and IDA 4 with regards to the FS algorithm on the dataset used in experiment 2 at $S1$. As it can be noted, also in this case both the considered algorithms run significantly faster than the FS approach.

VI. 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, comprising 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, performs overall 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 speed-ups (up to more than two orders of magnitude) with respect to the standard FS algorithm as well as to the SSDA algorithm.

VII. Acknowledgments

We wish to thank Prof. Massimo Ferri at Department of Mathematics, University of Bologna for his valuable suggestions concerning the material presented in Appendix I, Prof. Antonio Torralba and CSAIL at Massachusetts Institute of Technology for the use of MIT database, Prof. Rainer Koster and 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.

Appendix I
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

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

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

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

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. Let each of $d_t\left(\cdot\right), \;\; t = 1, \ldots r$ and $\tilde{d}\left(\cdot\right)$ be a metric, we define

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

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

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

being the smallest of such distances.

REFERENCES

[1] Y. Hel-Or and H. Hel-Or, "Real-time pattern matching using projection kernels," *IEEE Tran. Pattern Analysis and Machine Intelligence*, vol. 27, no. 9, pp. 1430–1445, 2005.

[2] D.I. Barnea and H.F. Silverman, "A class of algorithms for digital image registration," *IEEE Trans. on Computers*, vol. C-21, no. 2, pp. 179–186, 1972.

[3] F Tombari, S Mattoccia, and L Di Stefano, "Template matching based on the lp norm using sufficient conditions with incremental approximations," in *Proc. IEEE Int. Conf. on Advanced Video Surveillance Systems (AVSS 2006)*, Sydney, Australia, November 2006.

[4] H. L. Royden, *Real analysis*, p. 118, Prentice Hall, 3rd edition, 1988.

[5] F. Crow, "Summed-area tables for texture mapping," *Computer Graphics*, vol. 18, no. 3, pp. 207–212, 1984.

[6] M. Mc Donnel, "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," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[8] J.P. Lewis, "Fast template matching," in *Proc. Conf. on Vision Interface*, Quebec, Canada, May 1995, pp. 120–123.

[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*, Wiley, 2005.

[11] Changming 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`.

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

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

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

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