Pattern Recognition Letters 32 (2011) 2119–2127

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Adaptive Low Resolution Pruning for fast Full Search-equivalent pattern matching

Federico Tombari [a,*], Wanli Ouyang [b], Luigi Di Stefano [a], Wai-Kuen Cham [b]

[a] DEIS/ARCES, University of Bologna, Bologna, Italy
[b] Dept. of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China

ARTICLE INFO

Article history:
Received 18 October 2010
Available online 24 August 2011
Communicated by S. Todorovic

Keywords:
Pattern matching
Template matching
Low Resolution Pruning
Full Search-equivalent

ABSTRACT

Several recent proposals have shown the feasibility of significantly speeding-up pattern matching by means of Full Search-equivalent techniques, i.e. without approximating the outcome of the search with respect to a brute force investigation. These techniques are generally heavily based on efficient incremental calculation schemes aimed at avoiding unnecessary computations. In a very recent and extensive experimental evaluation, Low Resolution Pruning turned out to be in most cases the best performing approach. In this paper we propose a computational analysis of several incremental techniques specifically designed to enhance the efficiency of LRP. In addition, we propose a novel LRP algorithm aimed at minimizing the theoretical number of operations by adaptively exploiting different incremental approaches. We demonstrate the effectiveness of our proposal by means of experimental evaluation on a large dataset.

1. Introduction

Pattern matching is a computationally expensive technique aimed at locating the instances of a pre-defined *pattern*, or *template*, within an image. Pattern matching is widely adopted for tasks such as industrial inspection (quality control, defect detection) and fiducial-based pick-and-place, though it has also been used for image compression, face detection, action recognition. The standard technique for pattern matching, here recalled as Full-Search (FS), is based on the *sliding window* approach: i.e. the pattern is compared with all the equally-sized subsets contained in the image (also referred to as *image candidates*) by means of a similarity or dissimilarity score. Typical dissimilarity measures employed for this task are those derived from the *p-norm*, such as the *Sum of Absolute Differences* (SAD) and the *Sum of Squared Differences* (SSD). With these measures, when dissimilarity turns out to be below a pre-defined matching threshold, an instance of the pattern is located. The use of the sliding window approach guarantees that all below-threshold instances are located. Nevertheless, this leads to a computationally intensive process, so that many proposals in literature attempt to decrease its computational burden. In such a framework, several techniques have been proposed that try to speed-up the FS algorithm by means of an approximated search (Rosenfeld and Vanderburg, 1977, Vanderburg and Rosenfeld, 1977; Barnea and Silverman, 1972).

More interestingly, other proposals aim at speeding-up pattern matching without deteriorating the outcome of the search, that is, while achieving exactly the same results as the FS approach: we will recall these approaches as *exhaustive* or *FS-equivalent*. Recently this research topic has been particularly active, with many novel proposals attaining dramatic speed-ups with respect to the FS. The state of the art in the field is currently represented by the Low Resolution Pruning (LRP) algorithm (Gharavi-Alkhansari, 2001), the Incremental Dissimilarity Approximations (IDA) algorithm (Tombari et al., 2009), and by approaches based on the Walsh–Hadamard Transform such as Projection Kernels (PK) (Hel-Or and Hel-Or, 2005) and Grey-Code Kernels (Ben-Artz et al., 2007; Ouyang and Cham, 2009). Even more recently, a novel approach based on the Haar Transform has been proposed (Ouyang et al., 2010). Such wealth of new proposals has inspired a comprehensive comparative analysis based on a theoretical investigation of their commonalities and differences as well as on an experimental evaluation carried out on a large image dataset characterized by different nuisance factors and with different hardware platforms (Ouyang et al., in press). The results reported in (Ouyang et al., in press) show clearly, that, overall, LRP turns out to be the best performing algorithm. More precisely, LRP and GCK are the most efficient methods with the SSD measure under different nuisances (Gaussian noise, blurring, JPEG compression), while LRP and IDA are the most efficient ones under the same nuisances and with the SAD measure.

Given its prominence among *FS-equivalent* fast pattern matching methods, in this paper we investigate on how to further improve the LRP algorithm. In particular, LRP requires the computation of the image candidates at different resolution levels, this step being performed efficiently by means of incremental techniques that exploit recursive schemes. However, we have carried out a computational

* Corresponding author.
E-mail address: federico.tombari@unibo.it (F. Tombari).

0167-8655/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.patrec.2011.07.030

analysis of LRP showing that, given the high efficiency of the overall algorithm, in most cases the above step accounts for a significant fraction of the total number of operations, the only exception being those applications where a large number of patterns needs to be compared to the same target image. Hence, a specific investigation on the most efficient strategies for optimizing the computation of image candidates at different resolutions holds the potential to further improve the performance of the LRP algorithm.

This paper includes a twofold contribution. Firstly, we propose a computational analysis of LRP which allows us to demonstrate that the hierarchical incremental scheme for attaining the image candidates originally proposed in (Gharavi-Alkhansari, 2001) represents a non-optimal solution and, accordingly, to highlight more efficient methods suited to LRP. Secondly, based on this analysis, we propose a novel LRP algorithm aimed at minimizing the theoretical number of operations by adaptively exploiting different incremental approaches. By means of an extensive quantitative evaluation performed on a vast dataset, we show significant computational benefits in terms of measured execution times brought in by the proposed approach.

2. Derivation of the method

2.1. The LRP algorithm

Let $X = [x_1, x_2, \ldots, x_N]$ be a N-sized vector representing the pattern and $Y_1, \ldots, Y_K$ be the K N-sized vectors representing the image candidate vectors against which X must be matched, each candidate extracted from a different squared *sliding window* on the image. Also, let J be the length of image vector (i.e. the total number of image pixel). Fig. 1 helps to better understand the notation used throughout the paper and how image candidates are defined.

The first step of LRP is a *Rejection step* carried out over $T + 1$ levels of resolution, starting from level T (lowest resolution) up to level 0 (full resolution). As illustrated in Fig. 2, at each level t, the length of each image candidate vector Y_j and of the pattern vector X is reduced by a factor M by computing each new element as the sum of the M neighboring elements at the finer resolution. Then, each candidate undergoes checking the following pruning condition:

$$\delta_p\left(X^t, Y_j^t\right) > D^t \tag{1}$$

with X^t, Y_j^t being, respectively, the pattern and current candidate at resolution t, and δ_p representing the dissimilarity measure induced from the *p-norm* (SSD with $p = 2$, SAD with $p = 1$):

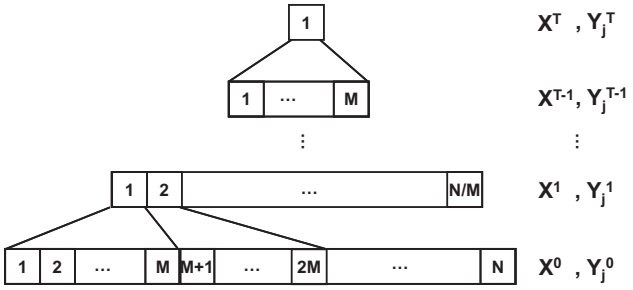

Fig. 2. In LRP, each image candidate Y_j^t and the pattern X^t are transformed to their lower-resolution level by summing up M-sized subsets of neighboring elements. The number reported in each block is the index of the corresponding vector element.

$$\delta_p(X, Y_j) = \|X - Y_j\|_p^p = \sum_{i=1}^{N} |x_i - y_{j,i}|^p. \tag{2}$$

Finally, D^t is obtained from a matching threshold computed, e.g. by selecting a minimum allowed distance δ_p^{min} as:

$$D^t = \|A\|_{p,t}^p \cdot \delta_p^{min} \tag{3}$$

with $\|A\|_{p,t}$ defined as Gharavi-Alkhansari (2001):

$$\|A\|_{p,t} = M^{\frac{t\cdot(p-1)}{2p}}. \tag{4}$$

If Eq. (1) holds, then Y_j is removed from the list of possible pattern instances for the next levels. Once the pruning conditions are tested at all levels, during the second step (*FS step*) of the algorithm a FS process is carried out over the remaining candidates by checking:

$$\delta_p(X, Y_j) < \delta_p^{min}. \tag{5}$$

All candidates for which (5) holds represent a pattern instance in the image.

2.2. Computational analysis

Table 1 reports the computational analysis of the LRP algorithm in terms of different operations (i.e. additions, multiplications (if $p = 2$), absolute values (if $p = 1$), branches) for the two steps of the LRP algorithm. Term Ω represents the cost to compute the image candidates at different resolutions as shown in Fig. 2 (whilst the pattern at different resolutions is computed and stored once

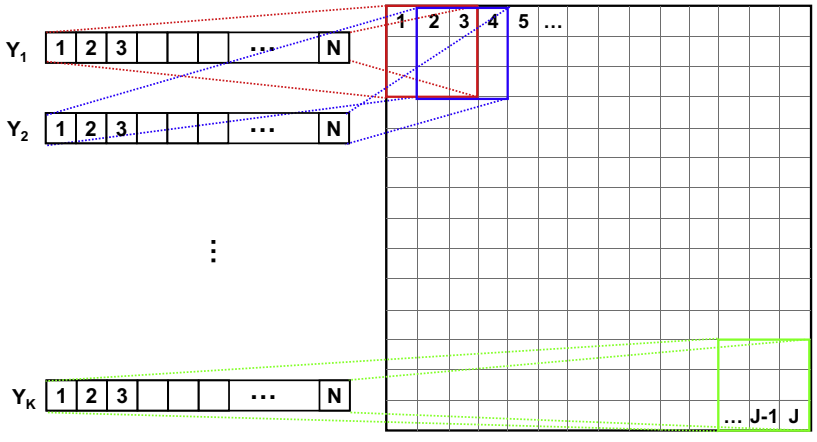

Fig. 1. Notation used throughout the paper. K image candidates, each of length N, are extracted from different squared *sliding window* out of the image, the latter having size J pixels. The number reported in each block is the index of the corresponding vector element.

Table 1
Numbers of operations required by LRP in the Rejection and FS steps.

	Rejection step	FS step
Additions	$\Omega + \sum_{t=1}^{T} 2M^{t-1}L_t$	$2NL_0$
Mult. ($p = 2$)/abs. values ($p = 1$)	$\sum_{t=1}^{T} M^{t-1}L_t$	NL_0
Branches	$\sum_{t=1}^{T} L_t$	L_0

at initialization or offline) and accounts only for additions. The remaining terms in the *Rejection step* are those needed to compute the *p-norm* for the remaining candidates at each resolution (additions and multiplications/abs. values) and to test the condition in (1) (branch instructions): hence they depend on the current number of remaining candidates at each level t, denoted as L_t. As far as the *FS step* is concerned, additions and multiplications/abs. values are needed to compute the *p-norm* at full resolution and branches to test (5): both depend on the remaining candidates at the FS stage, denoted as L_0.

Table 2 allows for a detailed analysis of the computational cost associated with the calculation of the image candidates at different resolutions (term Ω). In particular, the Table reports the number of additions required by the adoption of different incremental schemes. It is worth pointing out here that, by carefully comparing the two Tables, it can be argued that the number of operations required by the calculation of the image candidates is comparable to that of the whole LRP algorithm. As for the different methods available to compute term Ω, the original method proposed in (Gharavi-Alkhansari, 2001), here recalled as *Hierarchical*, requires a number of operation that depends on M. Two state-of-the-art alternatives are represented by the Box-Filtering (BF) (Mc Donnel, 1981) and Summed Area Table (SAT) techniques (Crow, 1984). As for BF, it exploits a double recurrence along rows and columns to compute sums of rectangular windows, with a cost of 4 operations per candidate at each level.

For what concerns the SAT technique, we have devised two different strategies. Both strategies require a fixed computational load represented by the computation of the Summed Area Table (accounting for $2J$ additions). Then, the former, dubbed SAT_{sld}, computes all low resolution candidates in a *sliding window* fashion, i.e. exploiting row and column recurrences similar to the BF technique. The latter, instead, computes low resolution candidates only when needed, i.e. at each level only for those candidates that have not been previously pruned from the list. This strategy, dubbed SAT_{can}, cannot exploit row and column recurrence since it computes candidates sparsely over the image area, but it holds the potential for a decreased computational burden if at certain levels the majority of candidates has been already pruned.

Based on a theoretical analysis it can be demonstrated that SAT_{sld} is more advantageous compared to the BF and Hierarchical approaches. In fact, unlike Hierarchical, its complexity does not depend on M (M is typically 4 but can also be much higher). The computational advantage of SAT_{sld} over Hierarchical will also be demonstrated in the experimental section. As for comparison to BF, since typically in pattern matching $N \ll J \approx K$, it follows that SAT_{sld} complexity can be approximated to $K(3T + 2)$, which compares favorably to that of BF, i.e. $4TK$. Instead, we cannot establish theoretically which method is more advantageous between SAT_{sld} and SAT_{can}, since the complexity of the latter is data-dependent.

Table 2
Additions required by different incremental techniques within the LRP framework.

Method	Ω
Hierarchical	$(M-1)TJ$
BF	$4TK$
SAT_{sld}	$2J + 3TK$
SAT_{can}	$2J + \sum_{t=1}^{T} 3M^{t-1}L_t$
Proposed	$2J + \sum_{t=1}^{T} 3\min\{M^{t-1}L_t, K\}$

2.3. Proposed algorithm

In the previous subsection we have proposed a computational analysis highlighting how incremental schemes based on SAT are more suited than other approaches to the computation of the low-resolution candidates of the LRP algorithm. In particular, we have outlined two different strategies based on SAT to compute low-resolution candidates, i.e. SAT_{sld} and SAT_{can}, with the complexity of the former being smaller with respect to other approaches, and that of the latter being data-dependent. Even though the efficiency of method SAT_{can} over other approaches cannot be demonstrated theoretically by computational analysis due to its inherent data-dependency, experimental evidence shows that, overall, its performance is comparable to that of SAT_{sld}, although the benefits of one SAT approach over the other can notably vary with the image and pattern sizes as well as with the kind of nuisance factors present in the data (see Section 3).

In this subsection we propose to exploit the previous computational analysis to devise a novel LRP method able to adaptively select the best approach between SAT_{sld} and SAT_{can} at each resolution level based on the current data. The main idea of the novel algorithm is to minimize the theoretical number of required operations by jointly exploiting both incremental schemes, so as to run as fast as the more efficient among the two approaches at each resolution level. Thus, this adaptive approach holds the potential to run much faster than both individual methods when more than two resolution levels are analyzed.

More precisely, at each level t, we want to run either SAT_{sld} or SAT_{can} depending on the number of required operations reported in Table 2. Hence, by looking at the computational costs in Table 2, at each level t the following condition is tested in order to select the algorithm requiring less additions to compute the image candidates at that level:

$$3M^{t-1}L_t < 3K, \tag{6}$$

which, in turn, that can be rewritten as:

$$L_t < \frac{K}{M^{t-1}}, \tag{7}$$

where the right hand term is constant for each pattern matching instance. If (7) holds, then SAT_{can} is used to compute the current image candidates, otherwise SAT_{sld} is the chosen method. Obviously, both techniques shares the same, unique Summed Area Table computed once for the image.

An additional improvement that we propose is aimed at rendering the SAT_{sld} approach more efficient when the number of candidates reduces to a small number. In this case, sketched as a toy example in Fig. 3, the sliding window approach does not need to take into account all pixels over the image area, but instead an effective image area can be determined as the smallest rectangular bounding box (depicted in red in figure) including all remaining image candidates (depicted in green in figure). By updating, at each level, this bounding box (which at the beginning is the same as the image area) we can avoid computing useless low resolution image candidates when SAT_{sld} algorithm is employed due to Eq. (7).

In a recent work Mattoccia et al. (2009), an algorithm has been proposed aimed at improving the efficiency of the LRP algorithm by means of an additional, initial coarse-to-fine search aimed at finding good initialization candidates. It is important to note here that our paper and Mattoccia et al. (2009) deals with different as-

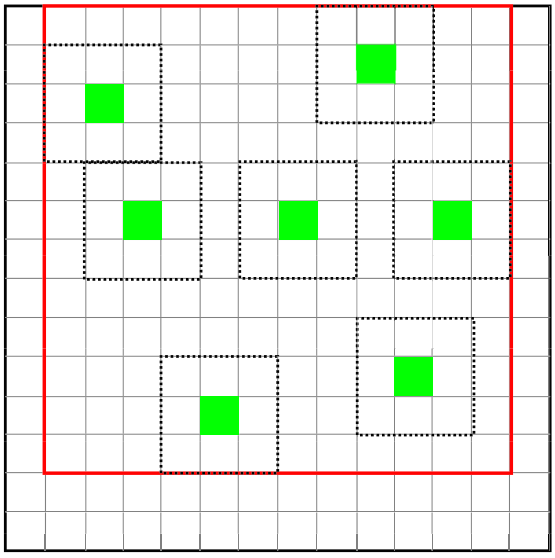

Fig. 3. With SAT_{sld}, computations need to be performed only on the effective area determined by the bounding box (in red) including all remaining candidates in the list (depicted in green). More specifically, the rectangles with dashed line describe the remaining windows and the green boxes describe the center of these windows. $N = 9$ in this example. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

pects of the LRP algorithm. In particular, we assume the rejection threshold to be provided and focus on an efficient way of obtaining the image candidates at different resolutions. Conversely, Mattoccia et al. (2009) used the standard Hierarchical approach to obtain the image candidates at different resolutions and investigates on efficient ways for achieving the rejection threshold. Thus, our proposal and the methods described in (Mattoccia et al., 2009) can be naturally combined to further improve the speed-up.

3. Experimental results

In this section we report on a large set of experiments aimed at assessing the computational benefits brought in by the proposed novel version of the LRP algorithm. In particular, we compare here the proposed adaptive approach for the computation of image candidates (hereinafter referred to as LRP_{NEW}) to the use of two fixed strategies based respectively on SAT_{sld} and SAT_{can} (we will denote these methods as, respectively LRP_{sld} and LRP_{can}). In addition, we evaluate the LRP algorithm relying on the Hierarchical strategy as originally proposed in (Gharavi-Alkhansari, 2001), this strategy referred to as LRP_{hier}. We also consider in the evaluation the deployment of the FFT through the correlation theorem, which is a conventional exhaustive approach to speed-up SSD-based pattern matching.

We test the performance of the evaluated algorithms on a vast image database consisting of 10 datasets containing different sizes of patterns and images. As shown in Table 3, the datasets are denoted as $Scalen_1$-n_2 for $n_1 = 1,2,3,4$, $n_2 = 1,\ldots,n_1$, where n_1 corresponds to image size and n_2 corresponds to pattern size. For example, datasets *Scale*2-1 and *Scale*2-2 have the same image size 320×240 but different pattern sizes.

Each of the 10 datasets has been obtained starting from a total of 120 grayscale images chosen among three databases: MIT,[1] medical,[2] and remote sensing.[3] 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 (radiographies) and remote sensing (Landsat satellite) images. The 120 images have been subdivided into four groups of 30 images, each group characterized by a size of images in *Scale*1-1, *Scale*2-1, *Scale*3-1, *Scale*4-1, (i.e. 160×120, 320×240, 640×480, 1280×960). For each image, 10 patterns were randomly selected among those showing a standard deviation of pixel intensities higher than a threshold (i.e. 45) for each dataset. Datasets having the same image size share the same images but have different patterns in both size and location. For example, datasets *Scale*2-1 and *Scale*2-2 share the same 30 images having size 320×240 but have different patterns in size and location. Thus, each dataset contains 300 image-pattern pairs. Since tests are performed over 10 datasets, there are overall 3000 image-pattern pairs.

Table 3
Datasets used in the experiments.

Dataset	Image size	J	Pattern size	N
*Scale*1-1	160×120	19,200	16×16	256
*Scale*2-1	320×240	76,800	16×16	256
*Scale*2-2	320×240	76,800	32×32	1024
*Scale*3-1	640×480	307,200	16×16	256
*Scale*3-2	640×480	307,200	32×32	1024
*Scale*3-3	640×480	307,200	64×64	4096
*Scale*4-1	1280×960	1,228,800	16×16	256
*Scale*4-2	1280×960	1,228,800	32×32	1024
*Scale*4-3	1280×960	1,228,800	64×64	4096
*Scale*4-4	1280×960	1,228,800	128×128	16,384

For all 10 datasets, we add different synthetic disturbance factors in order to evaluate the algorithms under different nuisances typically present in pattern matching scenarios. In particular, we compare the algorithms under 3 different kinds of distortions: Gaussian noise, Blur, JPEG compression. As for Gaussian noise, 4 different levels of i.i.d. zero-mean Gaussian noise are added to each image of the datasets described in Table 3, these 4 noise levels having variances 325, 650, 1300 and 2600[4] and referred to in the following as $G(1)$, $G(2)$, $G(3)$ and $G(4)$, respectively. As for Blur, 5 different levels of Gaussian low-pass filter are used for blurring each image of the datasets in Table 3. The 5 blurring levels, referred to as $B(1)$, B (2), $B(3)$, $B(4)$ and $B(5)$, correspond to Gaussian filters having standard deviation $\sigma = 0.2, 0.9, 1.6, 2.3$ and 3, respectively. As for distortion induced by JPEG compression, 5 different JPEG compression quality levels are used with each image of the datasets in Table 3. The JPEG compression quality levels, referred to as $J(1)$, $J(2)$, $J(3)$, $J(4)$ and $J(5)$, correspond to quality measures $Q_{JPG} = 90, 70, 50, 30$ and 10, respectively, with higher Q_{JPG} meaning higher quality (less image degradation due to compression). Fig. 4 shows an image from the dataset in Table 3 and its distorted versions for the 3 different kinds of nuisances.

The experimental comparison is performed by using, as matching measure, both the SSD and the SAD. The matching threshold is chosen as follows. Given a pattern having N pixels, the SSD threshold, T_{SSD}, and SAD threshold, T_{SAD}, are given by:

$$\begin{aligned} T_{SSD} &= 1.1 \cdot SSD_{min} + N, \\ T_{SAD} &= 1.1 \cdot SAD_{min} + N, \end{aligned} \tag{8}$$

where SSD_{min} and SAD_{min} represent, respectively, the SSD and SAD between the pattern and the best matching window.

The parameter M for LRP is chosen to be 4, which is the same used in the experiments reported in (Gharavi-Alkhansari, 2001). Since all the evaluated algorithms find the same matched windows

[1] http://people.csail.mit.edu/torralba/images.
[2] www.data-compression.info/Corpora/LukasCorpus.
[3] http://zulu.ssc.nasa.gov/mrsid.

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

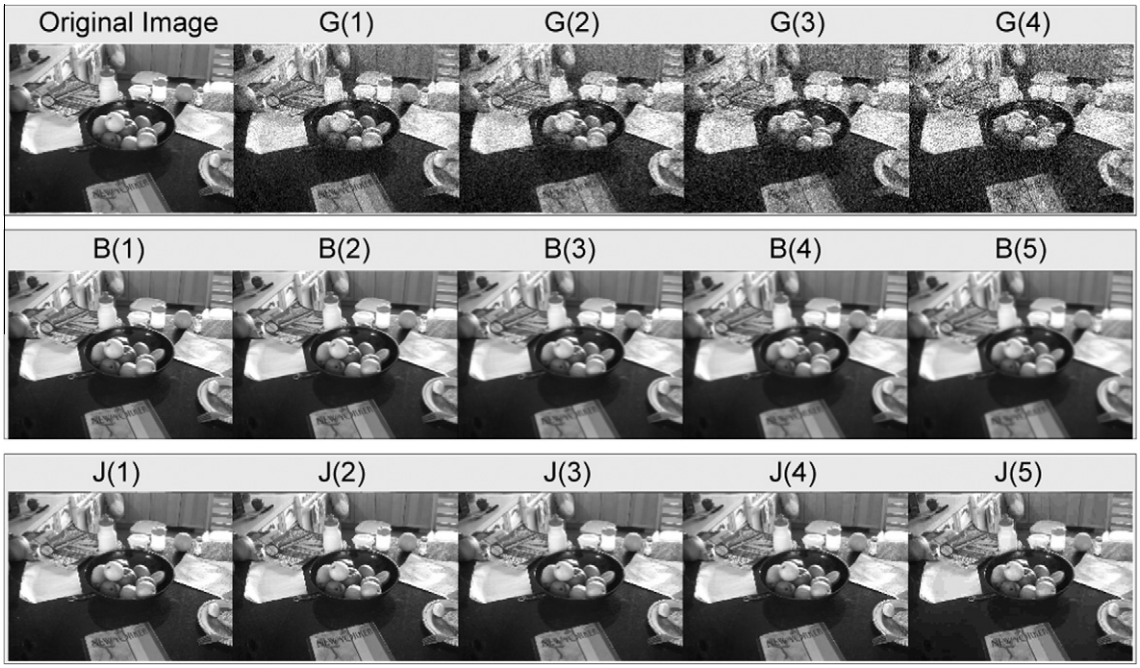

Fig. 4. An image from the dataset and its distorted versions. 1st row: the original image and its images with Gaussian noise levels $G(1)$ to $G(4)$; 2nd row: images with blurring levels $B(1)$ to $B(5)$; 3rd row: images with JPEG compression quality levels $J(1)$ to $J(5)$.

as the FS, the only concern is computational efficiency, which is assessed here based on actual execution time measurements. In particular, results are given in terms of speed-ups with respect to the FS algorithm calculated as ratios of measured execution times. All algorithms are written in C, while experiments are performed on 3 different Window platforms, 2 Intel CPUs (Xeon and Core2) and 1 AMD CPU (Athlon 64 X2) with different memory sizes. The reported speed-ups in terms of execution times are averages of the speed-ups measured on each of the 3 platforms.

3.1. Experimental results with the SSD measure

This section presents the experimental results obtained using the SSD as dissimilarity measure. Fig. 5, left shows the speed-ups yielded by the 4 LRP algorithms and the FFT with respect to the FS approach when no noise is added to the images of the considered datasets. In addition, Figs. 6–8 show the speed-ups in presence of distortions induced by, respectively, Gaussian noise, Blur and JPEG compression.

The common structure of Figs. 6–8 is as follows: a figure contains 10 sub-figures that correspond to the 10 datasets in Table 3; sub-figures in the same column, e.g. *Scale*3-3, *Scale*4-3, have the same pattern size and sub-figures in the same row have the same image size, e.g. *Scale*2-1, *Scale*2-2; in each sub-figure, the *X*-axis corresponds to the same datasets having distortion from low to high and the *Y*-axis to the speed-up over FS; each algorithm has the same line color and type in all figures.

By analyzing the reported results, it is clear that the proposed algorithm, LRP_{NEW}, outperforms both other LRP approaches (LRP_{can},LRP_{sld}) when no noise is added on the images (Fig. 5, left). In turn, these 3 LRP algorithms outperform the LRP_{hier} approach over all image and pattern sizes. Furthermore, the computational efficiency of all LRP algorithms is always higher than that of the conventional FFT-based method, LRP_{can} running faster than LRP_{sld} on the considered dataset.

Conversely, when Gaussian noise or Blur are added to the images (Figs. 6 and 7), LRP_{sld} turns out to be more efficient than LRP_{can} on the whole dataset. Nevertheless, the proposed approach, LRP_{NEW}, is able to estimate properly the computational load of both approaches yielding overall improved performance with regards to both previous techniques: speed-ups given by LRP_{NEW} are always at least as high as those given by LRP_{sld}, being in most cases significantly higher, especially with larger image and pattern sizes. As for LRP_{hier}, according to the theoretical analysis in the previous sec-

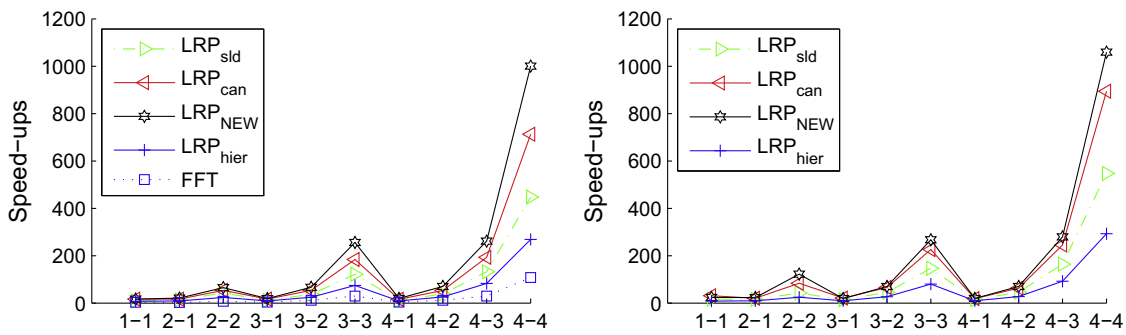

Fig. 5. Speed-ups with no noise for the SSD measure (left) and the SAD measure (right).

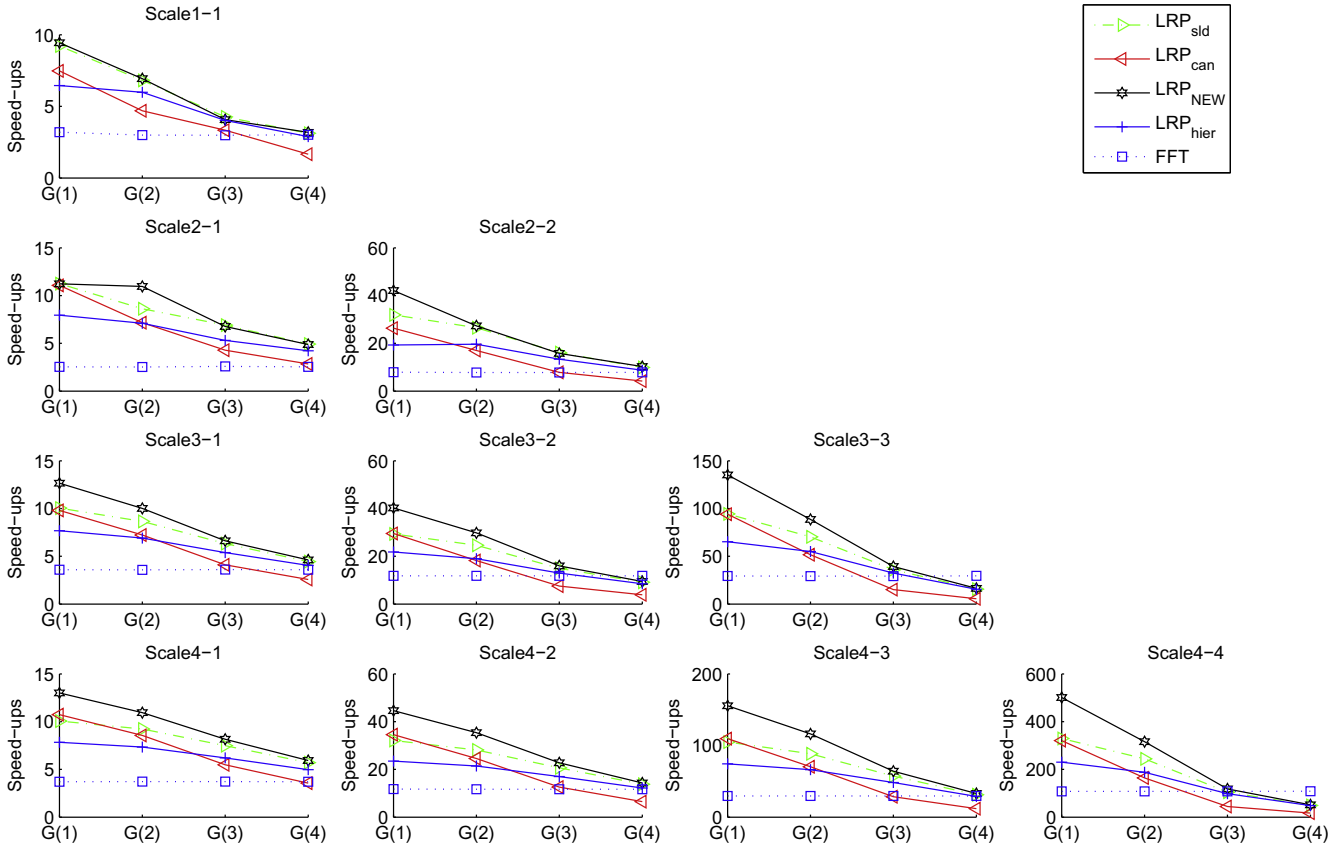

Fig. 6. Gaussian noise, SSD.

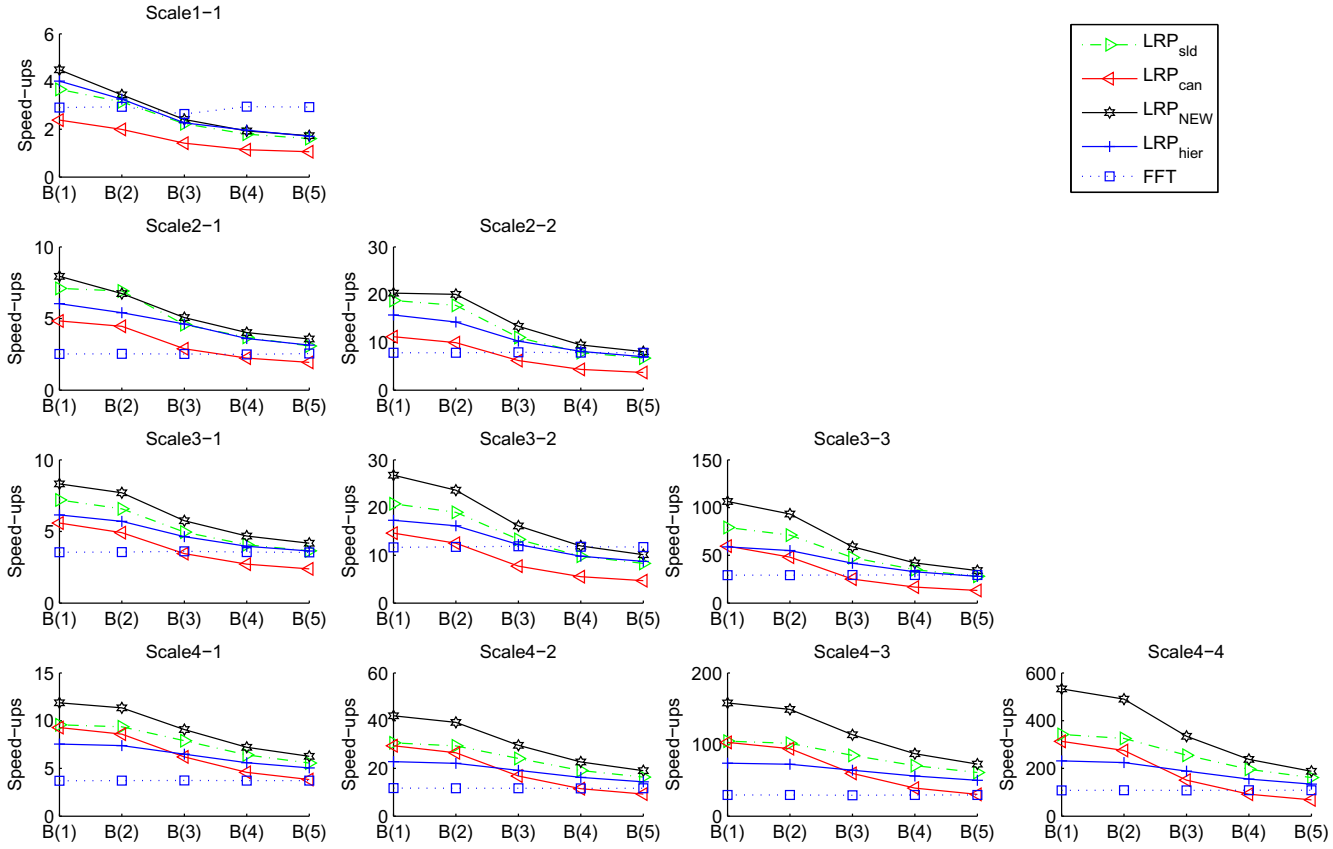

Fig. 7. Blur, SSD.

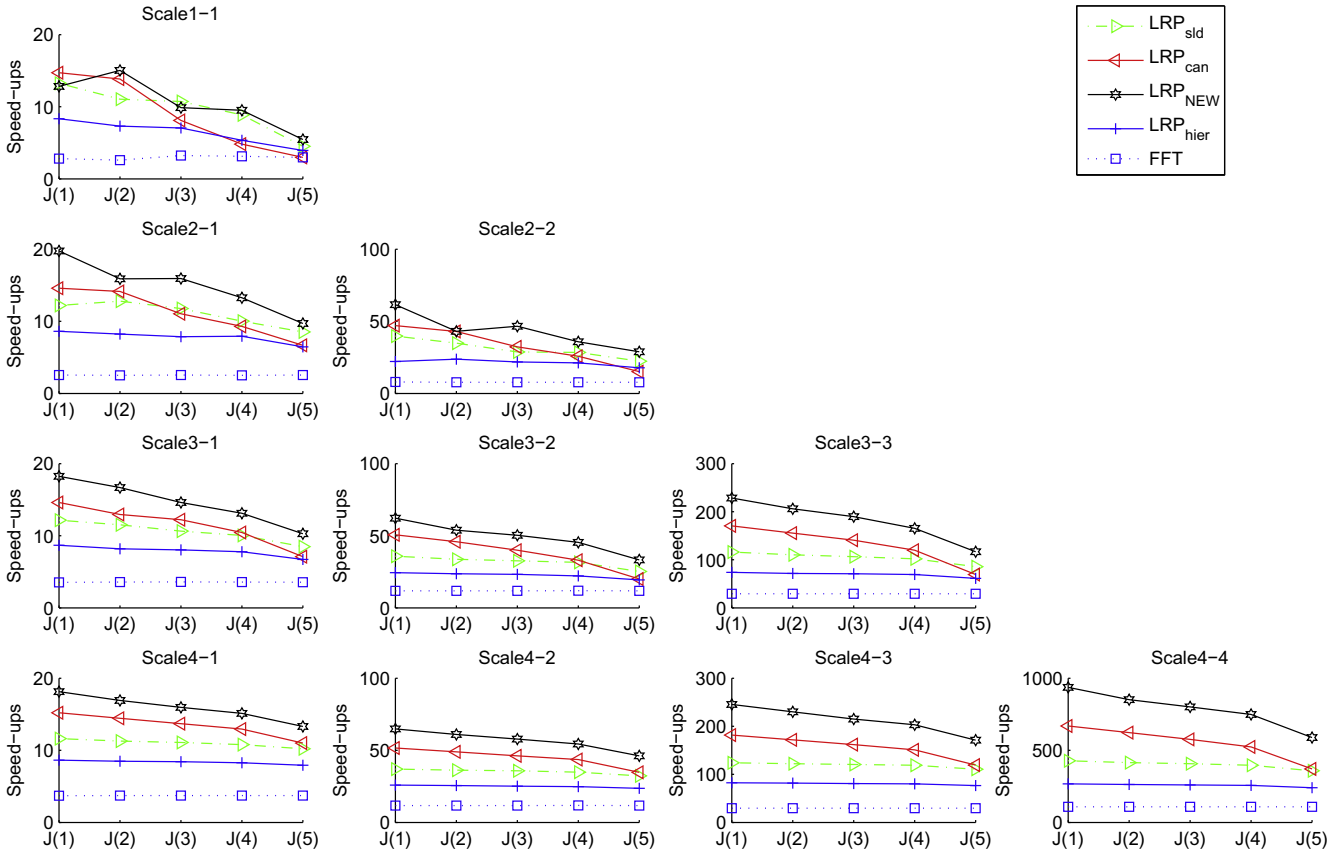

Fig. 8. JPEG compression, SSD.

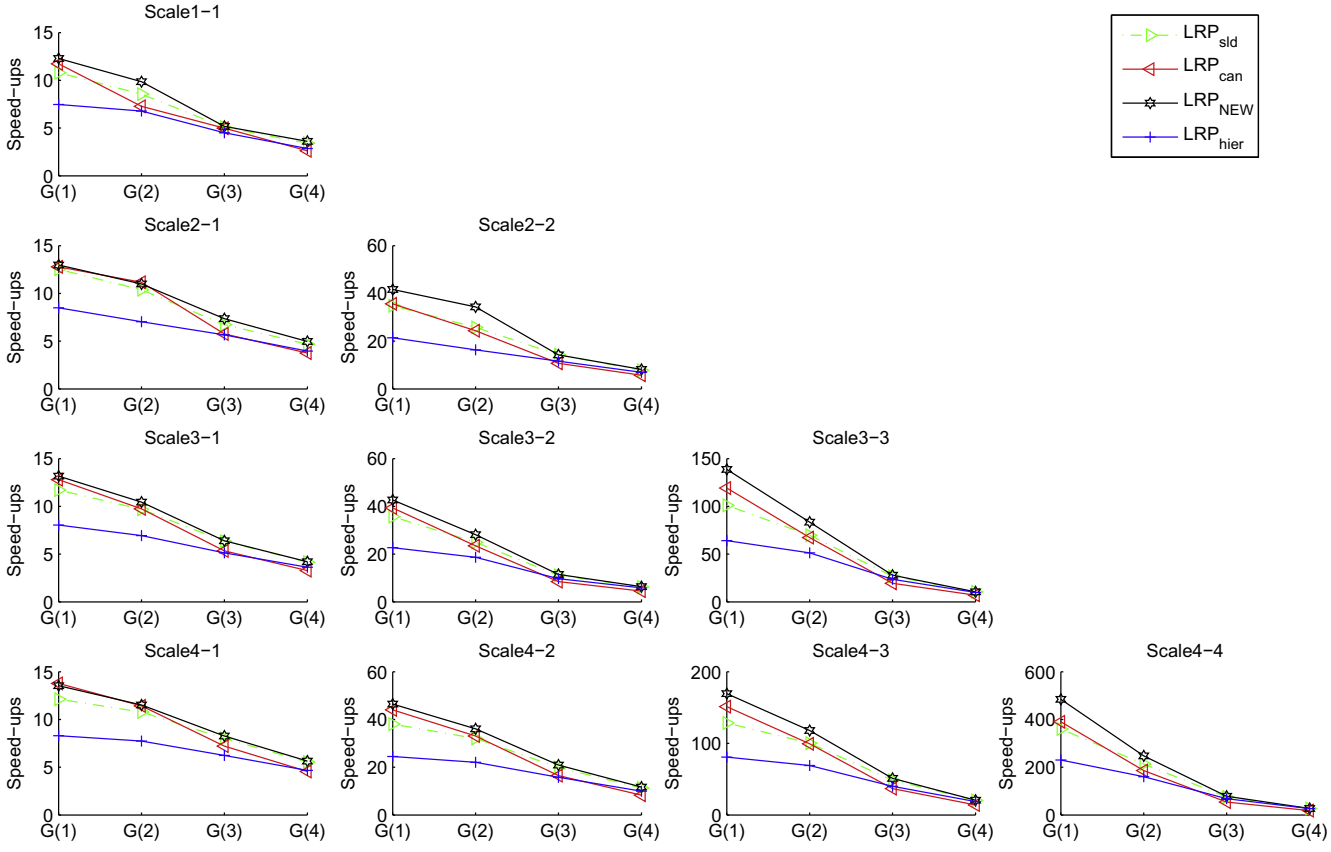

Fig. 9. Gaussian noise, SAD.

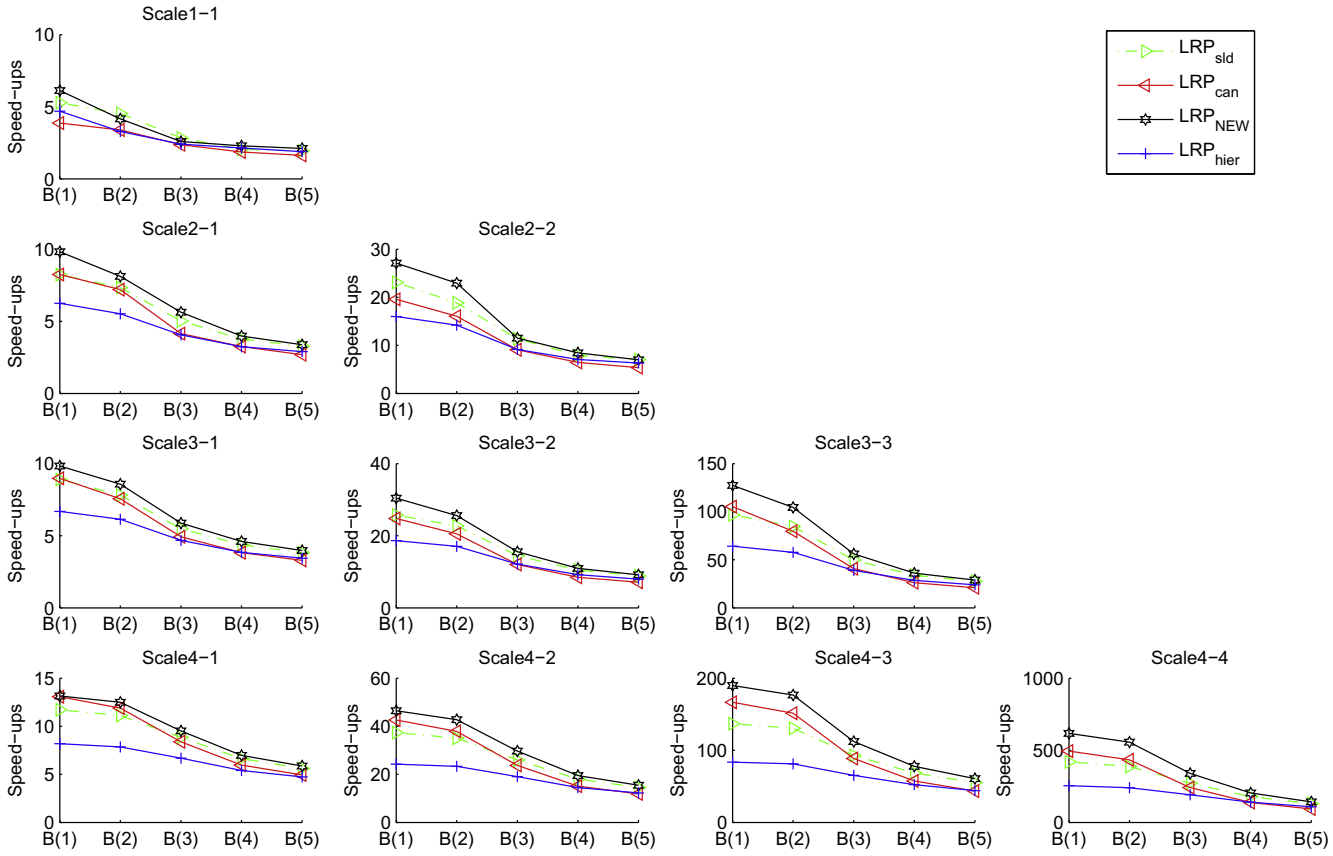

Fig. 10. Blur, SAD.

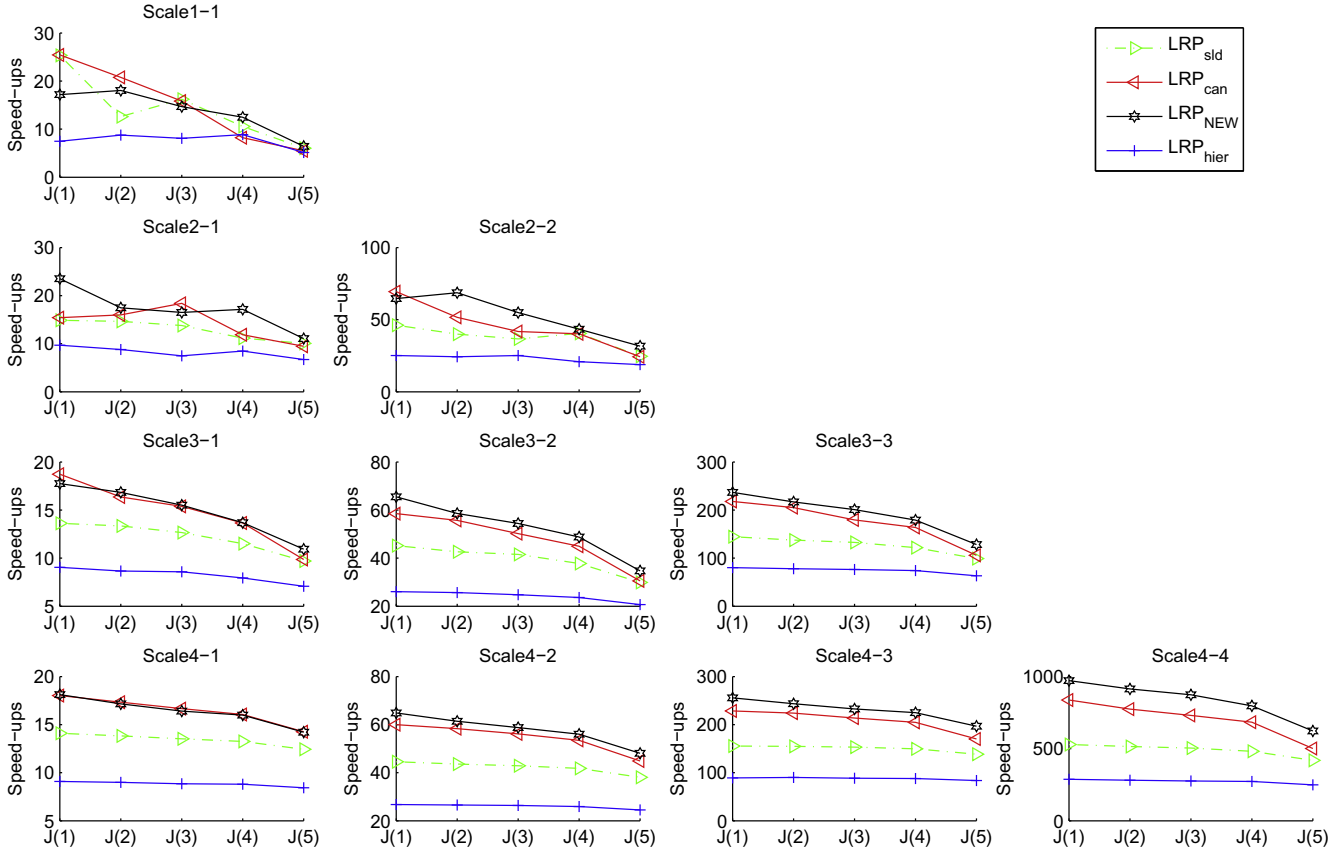

Fig. 11. JPEG compression, SAD.

tion, it is always slower than LRP_{sld}, while it tends to be faster than LRP_{can} at higher noise levels. As regards the FFT, a part from *Scale*1–1 with Blur distortion, it is never faster than the proposed approach, running in most cases notably slower.

Finally, in the case of JPEG distortion (Fig. 8) the computational benefits brought in by the proposed approach are even more relevant, with LRP_{NEW} outperforming much more significantly all 3 LRP approaches as well as the FFT over the whole dataset and at all levels of distortion. Mean speed-ups with respect to the FS algorithm ranges from one order of magnitude (small patterns and images) up to almost three orders of magnitude (large image and pattern sizes). In turn, LRP_{sld} runs slower than LRP_{can}, while LRP_{hier} is the slowest between the LRP approaches and faster than the FFT.

3.2. Experimental results with the SAD measure

This section presents the experimental results obtained using the SAD as dissimilarity measure. Since the FFT cannot be deployed with the SAD, only the 4 LRP algorithms are evaluated here. Fig. 5, right shows the speed-ups of the 4 LRP approaches with respect to the FS algorithm when no noise is added to the images of the considered datasets. Besides, Figs. 9–11 refer to the measured speed-ups in presence of distortions induced by, respectively, Gaussian noise, Blur and JPEG compression.

Analogously to the SSD case, when no noise is added to the images (Fig. 5, right), LRP_{NEW} is the best algorithm on the whole dataset. In particular, it yields improved performance compared to LRP_{can} and LRP_{sld}, the former running faster than the latter. In turn, LRP_{hier} is the slowest among the 4 LRP approaches.

As for Gaussian noise and Blur, the proposed approach yields at least comparable results to the faster between LRP_{can} and LRP_{sld}, in many cases attaining higher speed-ups. As for LRP_{hier}, it is still the slowest among the 4 LRP algorithms, the difference in efficiency being more significant with increasing pattern and image sizes.

Finally, in case of distortions due to JPEG compression, the proposed approach is not the fastest one only in one case out of 10 (i.e. *Scale*1-1), while in all other cases it yields at least equivalent performances compared to LRP_{can} and LRP_{sld}, in most cases outperforming both methods. As for LRP_{hier}, it runs again notably slower than the other 3 LRP algorithms.

4. Conclusions

We have proposed a novel adaptive strategy to compute image candidates at the different resolution levels that allows for improving significantly the computational efficiency of the Low Resolution Pruning by Gharavi-Alkhansari (2001). The proposed approach was shown to outperform other LRP strategies as well as the FFT and FS algorithms in experiments using different measures (SAD, SSD), different kinds of nuisances and different pattern and image sizes. Since, according to a recent comprehensive evaluation (Ouyang et al., in press), LRP represents the state-of-the art for exhaustive pattern matching algorithms, the novel method proposed in this paper significantly advances the state-of-the-art in the field and holds the potential to set the standard for exhaustive pattern matching in the future.

References

Rosenfeld, A., Vanderburg, G., 1977. Coarse-fine template matching. IEEE Trans. Systems Man Cybernet. 7, 104–107.

Vanderburg, G., Rosenfeld, A., 1977. Two-stage template matching. IEEE Trans. Image Process. 26, 384–393.

Barnea, D., Silverman, H., 1972. A class of algorithms for digital image registration. IEEE Trans. Comput. C-21 (2), 179–186.

Gharavi-Alkhansari, M., 2001. A fast globally optimal algorithm for template matching using low-resolution pruning. IEEE Trans. Image Process. 10 (4), 526–533.

Tombari, F., Mattoccia, S., Stefano, L.D., 2009. Full search-equivalent pattern matching with incremental dissimilarity approximations. IEEE Trans. Pattern Anal. Machine Intell. 31 (1), 129–141.

Hel-Or, Y., Hel-Or, H., 2005. Real time pattern matching using projection kernels. IEEE Trans. Pattern Anal. Machine Intell. 27 (9), 1430–1445.

Ben-Artz, G., Hel-Or, H., Hel-Or, Y., 2007. The gray-code filter kernels. IEEE Trans. Pattern Anal. Machine Intell. 29 (3), 382–393.

Ouyang, W., Cham, W., 2009. Fast algorithm for walsh hadamard transform on sliding windows. IEEE Trans. Pattern Anal. Machine Intell. 32 (1), 165–171.

Ouyang, W., Zhang, R., Cham, W., 2010. Fast pattern matching using orthogonal haar transform. In: Proc. CVPR 2010.

Ouyang, W., Tombari, F., Mattoccia, S., Di Stefano, L., Cham, W., in press. Performance evaluation of full search equivalent pattern matching algorithms, IEEE Trans. Pattern Anal. Machine Intell. <http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.106>.

Mc Donnel, M., 1981. Box-filtering techniques. Comput. Graphics Image Process. 17, 65–70.

Crow, F., 1984. Summed-area tables for texture mapping. Comput. Graphics 18 (3), 207–212.

Mattoccia, S., Tombari, F., Di Stefano, L., 2009. Enhanced low-resolution pruning for fast full-search template matching. In: Proc. Advanced Concepts for Intelligent Vision Systems (ACIVS 2009).