Performance Evaluation of Full Search Equivalent Pattern Matching Algorithms

Wanli Ouyang, *Member*, *IEEE*, Federico Tombari, *Member*, *IEEE*, Stefano Mattoccia, *Member*, *IEEE*, Luigi Di Stefano, *Member*, *IEEE*, and Wai-Kuen Cham, *Senior Member*, *IEEE*

Abstract—Pattern matching is widely used in signal processing, computer vision, and image and video processing. Full search equivalent algorithms accelerate the pattern matching process and, in the meantime, yield exactly the same result as the full search. This paper proposes an analysis and comparison of state-of-the-art algorithms for full search equivalent pattern matching. Our intention is that the data sets and tests used in our evaluation will be a benchmark for testing future pattern matching algorithms, and that the analysis concerning state-of-the-art algorithms could inspire new fast algorithms. We also propose extensions of the evaluated algorithms and show that they outperform the original formulations.

Index Terms—Pattern matching, template matching, fast algorithms, full search equivalent algorithm, performance evaluation.

1 Introduction

Pattern matching, also known as template matching, is the task of seeking a given pattern in a given image, as illustrated in Fig. 1. Pattern matching is widely used in signal processing, computer vision, and image and video processing. It has found applications in manufacturing for quality control [1], image-based rendering [2], image compression [3], object detection [4], superresolution [5], texture synthesis [6], block matching in motion estimation [7], [8], image denoising [9], [10], [11], road/path tracking [12], mouth tracking [13], image matching [14], and action recognition [15].

Suppose an $N_1 \times N_2$ pattern has to be sought in a given $J_1 \times J_2$ image of $J = J_1 J_2$ pixels, as shown in Fig. 1. The pattern will be compared to candidate windows of the same size in the image. The Full Search (FS) algorithm computes a similarity or dissimilarity measure between the *pattern* and all its equally sized *candidate windows* that can be extracted out of the image. We represent the pattern (template) as a length-N vector $\vec{X}_t$ and the candidate windows as $\vec{X}_w^{(j)}$, where subscripts $\cdot_t$ and $\cdot_w$ denote template and window, respectively, $j = 0, 1, \ldots, W-1$ and $N = N_1 N_2$. For example, if a 16×16 pattern is searched in a 256×256 image, we have $N = 256$, $J = 65{,}536$, and $W = (256 - 16 + 1)^2 = 58{,}081$.

There are several ways to compare two vectors in order to compute their similarity. These are represented by different matching measures that can be used for the comparison. In this paper, we consider a class of matching measures that find vast use in template matching applications, i.e., those derived from a distance measure based on the L_p norm. In particular, we denote as $d(\vec{X}_t, \vec{X}_w^{(j)})$ the distance between $\vec{X}_t$ and $\vec{X}_w^{(j)}$, which measures the dissimilarity between $\vec{X}_t$ and $\vec{X}_w^{(j)}$. The smaller is $d(\vec{X}_t, \vec{X}_w^{(j)})$, the more similar are $\vec{X}_t$ and $\vec{X}_w^{(j)}$. There are two different situations in pattern matching: 1) Detect all candidate windows having $d(\vec{X}_t, \vec{X}_w^{(j)}) < T$, for a given threshold T, and 2) find the window that leads to the minimum value of $d(\vec{X}_t, \vec{X}_w^{(j)})$ among all candidate windows. In this paper, we mainly consider situation 1, where $\vec{X}_w^{(j)}$ is said to match $\vec{X}_t$ when $d(\vec{X}_t, \vec{X}_w^{(j)}) < T$. As illustrated later, the algorithms for situation 1 can be easily modified to deal with situation 2.

The L_p norm of length-N vector $\vec{Z} = [z_0, z_1, \ldots, z_{N-1}]^T$ is defined as

$$\|\vec{Z}\|_p = (|z_0|^p + |z_1|^p + \cdots + |z_{N-1}|^p)^{1/p}. \quad (1)$$

Based on the L_p norm, the dissimilarity between $\vec{X}_t$ and $\vec{X}_w^{(j)}$ can be measured as $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p$. If $p = 1$, then the distance is the sum of absolute differences (SAD), while $p = 2$ yields the sum of squared differences (SSD).

There is research using other similarity or dissimilarity measures that make pattern matching robust to rotation, affine transformations, occlusion, and illumination variations. For example, the dissimilarity measure between the pattern descriptors and the candidate window descriptors is used in [16], [17], [18], [19], and [20], Hamming distance is used as the dissimilarity measure in [21], normalized cross correlation (NCC) is used as similarity measure in [22], [23], [24], [25], [26], and [27]. To limit the scope of the paper we have not included these measures and only consider algorithms that use SAD and SSD as a dissimilarity measure. As pointed out in [28], though there are arguments against SSD as a dissimilarity measure for images, it is still widely adopted due to its simplicity. Discussions concerning the use of SSD as a dissimilarity metric can be found in [29], [30], and [31].

- *W. Ouyang and W.-K. Cham are with the Chinese University of Hong Kong, China.*
- *F. Tombari, S. Mattoccia, and L. Di Stefano are with the University of Bologna, Bologna, Italy.*

Manuscript received 13 Sept. 2010; revised 31 Jan. 2011; accepted 12 Mar. 2011; published online 13 May 2011.
Recommended for acceptance by P. Felzenszwalb.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2010-09-0702.
Digital Object Identifier no. 10.1109/TPAMI.2011.106.

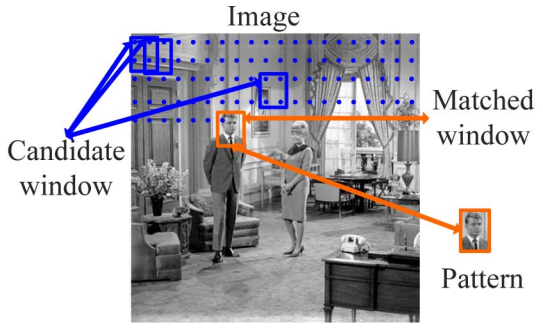


Fig. 1. Pattern matching in image "couple."

Since the FS algorithm is unacceptably slow in most applications, many faster approaches have been proposed in the literature [22], [25], [27], [28], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48]. Among these approaches, nonexhaustive algorithms yield computational savings by reducing the search space [38], [39], [40], [48] or by approximating patterns and windows using polynomials [41], [42], [43] or linear combination of simple features [44].

Conversely, exhaustive (or *FS-equivalent*) algorithms accelerate the pattern matching process and, at the same time, yield exactly the same result as FS. In the case of a dissimilarity-based search, a simple approach is known as Partial Distortion Elimination (PDE) [45] and its high efficiency consists of terminating the evaluation of the current dissimilarity measure as soon as it rises above the current minimum. Another approach suitable to dissimilarity-based searches consists of defining a rapidly computable lower bounding function of the adopted dissimilarity measure so as to quickly check one or more sufficient conditions to skip mismatched positions without carrying out the heavier computations required by the evaluation of the actual dissimilarity measure. Examples of such an approach include the algorithms in [28], [32], [33], [34], [35], [49], [46], [47], [50], [22], [26].

This paper is motivated by the intense research activity recently developed within this topic, but is not motivated by an exhaustive comparison of various proposals. The aim of the work is to compare and analyze state-of-the-art FS-equivalent algorithms for pattern matching in different conditions. For this aim, the paper selects the following five algorithms, which are recent and have been shown to yield notable speed-ups against the FS approach.

1. Alkhansari's Low Resolution Pruning (LRP) algorithm [32],
2. Tombari et al.'s Incremental Dissimilarity Approximation (IDA) algorithm [33],
3. Hel-Or and Hel-Or's projection-based algorithm (PWHT) using Walsh-Hadamard Transform (WHT) [28],
4. Ben-Artzi et al.'s projection-based algorithm using GCK (PGCK) [34], and
5. Ouyang and Cham's projection-based algorithm using fast WHT (FWHT) [35].

Table 1 summarizes the compared algorithms, together with the corresponding abbreviations. In addition to FS, the Fast Fourier Transform (FFT) approach is also used as a benchmark of comparison (in particular, we use the OpenCV implementation [51]).

TABLE 1
Abbreviations and References of Compared Algorithms

LRP	IDA	PWHT	PGCK	FWHT	FFT
[32]	[33]	[28]	[34]	[35]	[51]

The main contributions of this paper are as follows:

1. A unified framework for describing the algorithms evaluated in this paper. Under this framework, we give a theorem that relates LRP to the algorithms that use WHT basis vectors.
2. The computational analysis of IDA, PWHT, PGCK, and FWHT under the framework of pattern matching, which is not provided in the previous literature. Based on this analysis, we propose a new termination strategy for fast full-search equivalent pattern matching algorithms and two additional efficient formulations of LRP.
3. A quantitative performance evaluation of state-of-the-art FS-equivalent pattern matching algorithms based on a very large data set. This allows us to identify the best performing methods under a number of nuisances found in real-world applications. Moreover, the data sets, methodology, and results used in our evaluation provide a reference framework for testing future pattern matching algorithms.[1]

This paper is organized as follows: Section 2 introduces FS and FFT. Section 3 presents a unified framework that can represent all evaluated algorithms for further analysis. Based on this unified framework, the computational complexity of evaluated algorithms is analyzed in Section 4. Then, the algorithms are compared quantitatively using the data sets described in Section 5, which account for different sizes of images and patterns as well as for distortions caused by different types of noise. The testing environment and evaluation criterion will also be described in Section 5. Section 6 illustrates the experimental results. Finally, Section 7 presents a discussion and Section 8 draws conclusions.

2 FS and FFT Approach

2.1 Full Search Approach

With the FS algorithm, the distance between pattern $\vec{X}_t$ and each candidate window $\vec{X}_w^{(j)}$, i.e., $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p$ for $j = 0, \ldots, W-1$, is measured. If $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p < T$, then the window $\vec{X}_w^{(j)}$ is considered as a matching window; otherwise, it is considered as a mismatched window.

When SSD is used, FS requires about $2NW$ additions and NW multiplications. When SAD is used, FS requires about $2NW$ additions and NW absolute value operations.

2.2 Fast Fourier Transform

The FFT-based approach can be used only with the SSD. As pointed out in [33], the SSD function can be written as

1. Data sets and code to carry out the experiments will be made publicly available through a website.

TABLE 2
A Framework for Pattern Matching Using Lower Bound

Overall procedure:
Initially, set_{can} contains all candidate windows $\vec{X}_w^{(j)}$;
Step a (Rejection Step):
For $k{:=}1$ to N_{Maxk}:{
Step a.1: For $\vec{X}_w^{(j)}$ in set_{can} {Obtain $f_{low}(k,j)$.}
Step a.2: For $\vec{X}_w^{(j)}$ in set_{can}
{If $f_{low}(k,j) \geq T$, then remove $\vec{X}_w^{(j)}$ from set_{can}.}
Step a.3: $N_k = k$; if $Cond_{Ter}$ is true, then goto *Step b*.
}
Step b (FS-Step):
For each candidate window $\vec{X}_w^{(j)}$ in set_{can}:
{ Use $||\vec{X}_t - \vec{X}_w^{(j)}||_p^p$ to find out if $\vec{X}_w^{(j)}$ matches $\vec{X}_t$.}

$$\|\vec{X}_t - \vec{X}_w^{(j)}\|_2^2 = \|\vec{X}_t\|_2^2 + \|\vec{X}_w^{(j)}\|_2^2 - 2 \cdot <\vec{X}_t, \vec{X}_w^{(j)}>, \quad (2)$$

where $<\vec{X}_t, \vec{X}_w^{(j)}> = (\vec{X}_t)^T \vec{X}_w^{(j)}$ is the inner product between $\vec{X}_t$ and $\vec{X}_w^{(j)}$. FFT facilitates efficient computation of the inner product $<\vec{X}_t, \vec{X}_w^{(j)}>$ in (2).

As pointed out in [22], the FFT-based approach requires about $6Jlog_2J$ additions and $6Jlog_2J$ multiplications for computing the inner product. After that, W additions are required to obtain $2<\vec{X}_t, \vec{X}_w^{(j)}>$ from $<\vec{X}_t, \vec{X}_w^{(j)}>$ for $j = 0, 1, \ldots, W-1$. To compute the term $\|\vec{X}_w^{(j)}\|_2^2$ at each pixel location, J multiplications are required for squaring pixel values, $4W$ and $5W$ additions are required, respectively, by the box-filtering technique [52] and the integral image approach [53] for summing up the squared values $\|\vec{X}_w^{(j)}\|_2^2 = \sum_{x_{w,j,a} \in \vec{X}_w^{(j)}} (x_{w,j,a})^2$ for $j = 0, \ldots, W-1$. $\|\vec{X}_t\|_2^2$ can be obtained by N multiplications and $2N$ additions. Finally, $2W$ additions are needed for summing up the three terms in (2). In summary, the FFT-based approach requires at least $6Jlog_2J + 7W$ additions and $6Jlog_2J + J$ multiplications.

3 A Unified Framework for Pattern Matching Algorithms

In this section, we first introduce a unified framework for fast FS-equivalent pattern matching. The links and differences among the algorithms compared in this paper are then analyzed within this framework.

As highlighted in Table 2, the proposed framework consists of two steps:

- *Step a*. Mismatching candidate windows are eliminated from set_{can}. This step is called the rejection step.
- *Step b*. The remaining candidate windows in set_{can} undergo FS for finding out the matching windows. This step is called the FS-step.

In this framework, both FS and FFT directly evaluate the distance $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p$ for all candidate windows to find the matching windows. This corresponds to skipping the rejection step and starting from the FS-step in Table 2. The FFT approach is only different from FS in the computation of $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p$.

On the other hand, algorithms IDA, LRP, PWHT, PGCK, and FWHT start from the rejection step. The rejection step is a loop of k, where k increases from 1, and comprises three substeps. In *Step a.1*, a lower bounding function $f_{low}(k,j)$ is evaluated such that $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p \geq f_{low}(k,j)$. In *Step a.2*, if $f_{low}(k,j) \geq T$, then $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p \geq T$ and we can safely prune $\vec{X}_w^{(j)}$ from set_{can}, $f_{low}(k,j) \geq T$ being the rejection condition. In *Step a.3*, the loop of k terminates when a given termination condition, denoted as $Cond_{Ter}$ is satisfied. Throughout the rejection step, each candidate window undergoes checking of a succession of rejection conditions $f_{low}(k,j) \geq T$ in each loop of k until either it is pruned or the rejection step finishes. There are two conditions under which the rejection terminates: 1) k reaches a sufficient number, which is denoted as N_{Maxk}, and 2) the percentage of remaining candidate windows in iteration $k+1$, denoted as $Per_{can}^{(k+1)}$, is below a certain threshold, denoted as ϵ. The second termination condition, corresponding to $Cond_{Ter}$ in *Step a.3*, is a strategy used by Hel-Or and Hel-Or in [28] because it turns out to be faster to directly calculate the actual distance than evaluating the lower bound when the remaining candidates in set_{can} are very few. N_k records the actual number of loops of run in the rejection step.

TABLE 3
Symbols and Terms Used in the Paper

Symbol	Meaning
$A_{LRP}^{(k,h)}$	$A_{LRP}^{(k,h)} = h^{(k-1)} N_{can}^{(k)}$.
$B(N, N_k, W)$	The number of operations required by PWHT, PGCK or FWHT for transformation on images.
J	For $J_1 \times J_2$ image, we have $J = J_1 J_2$.
N	For $N_1 \times N_2$ pattern, we have $N = N_1 N_2$.
$N_{can}^{(FS)}$	The number of candidates in the FS-step.
$N_{can}^{(k)}$	The number of candidates at loop k.
N_k	The number of loops run in the rejection step.
N_{Maxk}	At most N_{Maxk} loops for the rejection step.
N_P	The number of partitions for IDA.
Set_{can}	The set of candidate windows.
T	Threshold for pattern matching.
U	Size of vector $\vec{Y}^{(U)} = V^{(U \times N)} \vec{X}$.
$V^{(U \times N)}$	The $U \times N$ transform matrix.
W	The number of candidate windows.
$\vec{X}_t$	The pattern/template represented by a vector.
$\vec{X}_w^{(j)}$	The jth candidate window represented by a vector.
$f_{low}(k,j)$	Lower bound function at loop k for jth window.
h	A small integer used by LRP.
i	Index of basis vector in transform matrix.
j	Index of candidate window.
k	kth loop for the rejection step.
$u(k)$	The number of basis vectors chosen for LRP, PWHT, GCK and FWHT at loop k.
$\otimes$	Kronecker product.
$\|\|\vec{z}\|\|_p$	L_p norm of vector $\vec{z}$.
$\|\|V\|\|_p$	Induced L_p norm of transform matrix V.

Table 3 shows the meaning for symbols in this paper.

The advantage of using $f_{low}(k,j)$ is that it is more efficient to compute $f_{low}(k,j)$ than to compute $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p$ and a small number of iteration k in the rejection step can eliminate a large number of mismatching windows.

The unified framework can be modified to find the window that has the minimum $\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p$ using the approaches proposed in [50], [28], and [27]. Taking the approach in [28] as an example, the threshold T can be adapted in each loop of k based on the minimum lower bound found in the kth loop.

The differences among the algorithms are summarized in Table 4. IDA, LRP, PWHT, PGCK, and FWHT are applicable for any dissimilarity measure based on the L_p-norm ($p \geq 1$), while FFT is based only on L_2-norm. FFT

TABLE 4
Differences for the Unified Framework in Table 2

Methods	FFT [51]	IDA [33]	LRP [32]	PKs [28], [34], [35]
Norm	L_2	L_p		
Steps	FS-step	Rejection step and FS-Step		
N_{Maxk}	N/A	N_P	$\log_h N$	N
f_{low}	N/A	Eqn(6)	Eqn(8)	

PWHT, PGCK, and FWHT share the column "PKs".

includes only the FS-step, while the others include both the rejection step and the FS-step. N_{Maxk} is not applicable for FFT; the IDA algorithm has $N_{Maxk} = N_P$; the LRP algorithm has $N_{Maxk} = \log_h N$; PWHT, PGCK, and FWHT have $N_{Maxk} = N$. The meaning of parameters N_P and h is given in Table 3 and will be illustrated later.

The FS-equivalent algorithms considered in the paper achieve high efficiency by using a lower bounding function, $f_{low}(k, j)$, that eliminates mismatching candidates early. Hence, the methods for estimating these lower bounds greatly affect the computational performance of FS-equivalent algorithms. In the following, we recall the lower bound estimation methods used by IDA, LRP, PWHT, PGCK, and FWHT.

3.1 The Lower Bounding Function for IDA

The IDA technique relies on partitioning the pattern vector, $\vec{X}_t$, and each candidate vector, $\vec{X}_w^{(j)}$, into a certain number of subvectors in order to determine a succession of pruning conditions characterized by increasing tightness and computational weight. Given an N-dimensional vector, IDA establishes a partition P of the vector into N_P disjoint subvectors (not necessarily with the same number of components). In particular, it defines a partition of set $\{1, 2, \ldots, N\}$ into N_P disjoint subsets $P = \{P_1, P_2, \ldots, P_{N_P}\}$, where $\cup_{m=1}^{N_P} P_m = \{1, 2, \ldots N\}$, $1 \le N_P \le N$, and N_P is a parameter.

Given partition P, IDA defines the *partial* L_p-norm of vector $\vec{X}_t$ and $\vec{X}_w^{(j)}$ limited to the subvector associated with $P_m \in P$ as

$$\|\vec{X}_t\|_{p,P_m} = \left(\sum_{n \in P_m} |x_{t,n}|^p\right)^{\frac{1}{p}}, \tag{3}$$

$$\|\vec{X}_w^{(j)}\|_{p,P_m} = \left(\sum_{n \in P_m} |x_{w,j,n}|^p\right)^{\frac{1}{p}}, \tag{4}$$

where $x_{t,n}$ is the nth element in pattern $\vec{X}_t$ and $x_{w,j,n}$ is the nth element in window $\vec{X}_w^{(j)}$. In addition, IDA defines the *partial* L_p-dissimilarity between $\vec{X}_t$ and $\vec{X}_w^{(j)}$ limited to the subvector associated with $P_m \in P$ as

$$\|\vec{X}_t - \vec{X}_w^{(j)}\|_{p,P_m}^p = \sum_{n \in P_m} |x_{t,n} - x_{w,j,n}|^p. \tag{5}$$

As proven in [33] using reverse triangle inequality, the lower bound used by IDA in iteration k is

$$\begin{aligned} f_{low}(k, j) = &\sum_{m=1}^{k-1} \|\vec{X}_t - \vec{X}_w^{(j)}\|_{p,P_m}^p \\ &+ \sum_{n=k}^{N_P} |\, \|\vec{X}_t\|_{p,P_n} - \|\vec{X}_w^{(j)}\|_{p,P_n}|^p, \end{aligned} \tag{6}$$

where the left term is the partial L_p-dissimilarity between $\vec{X}_t$ and $\vec{X}_w^{(j)}$ and the right term is an estimation of the remaining L_p-dissimilarity. Let us denote the modulo operation by %. When partitioning a set of size N into N_P subsets, we constrain that $N\%N_P = 0$ and the subsets P_n for $n = 1, \ldots, N_P$ have the same size, i.e., N/N_P. Thus, the candidate window is equally partitioned into N_P subwindows having the same size. The method presented in [33] does not imply this constraint, but this constraint normally makes IDA computationally more efficient and facilitates computational complexity analysis. Thus, the constraint is used in this paper as it is also the case of the experimental results reported in [33].

3.2 The Lower Bounding Function for LRP, PWHT, GCK, and FWHT

The transformation that projects a vector $\vec{X} \in \mathbb{R}^N$ onto a linear subspace spanned by $U(\le N)$ basis vectors $\vec{V}^{(0)}, \ldots \vec{V}^{(U-1)}$ can be represented as follows:

$$\vec{Y} = V^{(U \times N)}\vec{X} = [\vec{V}^{(0)} \ldots \vec{V}^{(U-1)}]^T \vec{X}, \tag{7}$$

where $\cdot^T$ is matrix transposition, vector $\vec{X}$ of length N is called the input window, vector $\vec{Y}$ of length U is called the projection value vector, the U elements in vector $\vec{Y}$ are called projection values, and $V^{(U \times N)}$ is a $U \times N$ matrix that contains U basis vectors $\vec{V}^{(i)}$ of length N for $i = 0, \ldots, U-1$. When $V^{(U \times N)}$ is a square matrix, i.e., $U = N$, we denote it as $V^{(N)}$. The above transformation is also called projection, e.g., in [28], [37], and [33], while basis vectors are called projection kernels in [28] and called filter kernels in [34]. Algorithms using transformation for pattern matching are called transform domain pattern matching algorithms, e.g., PWHT, PGCK, FWHT, and LRP.

3.2.1 LRP

LRP uses the following lower bounding function:

$$f_{low}(k, j) = \frac{\|V^{(u(k) \times N)}(\vec{X}_t - \vec{X}_w^{(j)})\|_p^p}{\|V^{(u(k) \times N)}\|_p^p}, \tag{8}$$

where $u(k)$ is the number of basis vectors chosen for transform domain pattern matching in the kth loop of the rejection step. The $\|V^{(u(k) \times N)}\|_p$ in (8) is the induced matrix p-norms is defined as follows:

$$\|V^{(u(k) \times N)}\|_p = \max_{\vec{Z} \ne 0} \frac{\|V^{(u(k) \times N)}\vec{Z}\|_p}{\|\vec{Z}\|_p}. \tag{9}$$

When not all elements in $V^{(u(k) \times N)}$ are zeros, it follows from (9) that

$$\|\vec{Z}\|_p \ge \frac{\|V^{(u(k) \times N)}\vec{Z}\|_p}{\|V^{(u(k) \times N)}\|_p}. \tag{10}$$

As pointed out in [32], the lower bound in (8) is derived from (10) by replacing the $\vec{Z}$ in (10) with $\vec{X}_t - \vec{X}_w^{(j)}$:

$$\|\vec{X}_t - \vec{X}_w^{(j)}\|_p^p \geq \frac{\|V^{(u(k)\times N)}(\vec{X}_t - \vec{X}_w^{(j)})\|_p^p}{\|V^{(u(k)\times N)}\|_p^p} = f_{low}(k,j). \quad (11)$$

The LRP algorithm is best explained using Kronecker product, which is denoted as $\otimes$. If A is a $U_1 \times Q_1$ matrix (a_{n_1,n_2}) and B is a $U_2 \times Q_2$ matrix (b_{m_1,m_2}), then $A \otimes B$ is the following $U_1U_2 \times Q_1Q_2$ matrix:

$$A \otimes B = \begin{bmatrix} a_{0,0}B & a_{0,1}B & \cdots & a_{0,Q_1-1}B \\ a_{1,0}B & a_{1,1}B & \cdots & a_{1,Q_1-1}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{U_1-1,0}B & a_{U_1-1,1}B & \cdots & a_{U_1-1,Q_1-1}B \end{bmatrix}. \quad (12)$$

Denote the $u(k) \times u(k)$ identity matrix as $I^{(u(k))}$. The 1D transform matrix for the LRP algorithm proposed in [32] is given by

$$V^{(u(k)\times N)} = I^{(u(k))} \otimes \mathbf{1}^{(1\times(N/u(k)))} = \begin{bmatrix} \mathbf{1}^{(1\times(N/u(k)))} & \mathbf{0}^{(1\times(N/u(k)))} & \cdots & \mathbf{0}^{(1\times(N/u(k)))} \\ \mathbf{0}^{(1\times(N/u(k)))} & \mathbf{1}^{(1\times(N/u(k)))} & \cdots & \mathbf{0}^{(1\times(N/u(k)))} \\ \cdots & \cdots & \cdots & \cdots \\ \mathbf{0}^{(1\times(N/u(k)))} & \mathbf{0}^{(1\times(N/u(k)))} & \cdots & \mathbf{1}^{(1\times(N/u(k)))} \end{bmatrix}, \quad (13)$$

where $\mathbf{0}^{(1\times(N/u(k)))}$ and $\mathbf{1}^{(1\times(N/u(k)))}$ are $1 \times (N/u(k))$ matrices with all elements equal to 0 and 1, respectively, $u(k) = h^{k-1}$ is the number of basis vectors used for evaluating the lower bound function in (8), and h can be any small integer number. The experiments in [32] use $h = 4$. As an example for the LRP matrix in (13), when $u(k) = 2$, $N = 4$, we have

$$V^{(2\times 4)} = I^{(2)} \otimes \mathbf{1}^{(1\times 2)} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \otimes [1\ 1] = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}. \quad (14)$$

As proven in [32], $\|V^{(u(k)\times N)}\|_p^p = (N/u(k))^{(p-1)/p}$ in (8) for the $V^{(u(k)\times N)}$ defined in (13). For candidate windows in the images, transformation using LRP is computed by summing pixel values in a rectangle.

3.2.2 PWHT, PGCK, and FWHT

The following inequality about the L_2 norm, proven in [28], is used for PWHT, PGCK, and FWHT:

$$\begin{aligned} \|\vec{X}_t - \vec{X}_w^{(j)}\|_2^2 &= \|\vec{D}\|_2^2 \\ &\geq (V^{(u(k)\times N)}\vec{D})^T (V^{(u(k)\times N)} V^{(u(k)\times N)^T})^{-1} (V^{(u(k)\times N)}\vec{D}) \quad (15) \\ &= f_{low}(k,j), \text{ where } \vec{D} = \vec{X}_t - \vec{X}_w^{(j)}. \end{aligned}$$

PWHT and FWHT use WHT as the transform matrix for pattern matching. The WHT transform matrix can be recursively defined as

$$M^{(N)} = M^{(2)} \otimes M^{(N/2)} = \begin{bmatrix} M^{(N/2)} & M^{(N/2)} \\ M^{(N/2)} & -M^{(N/2)} \end{bmatrix}, \quad (16)$$

where $M^{(1)} = 1$, $M^{(N)} = [\vec{M}^{(0)} \ldots \vec{M}^{(N-1)}]^T$ is an $N \times N$ matrix. As two examples for (16), we have

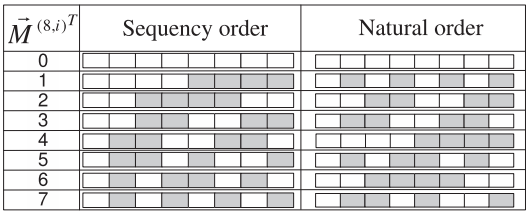


Fig. 2. WHT basis vectors in sequency order and natural order. White represents the value $+1$ and gray represents the value -1.

$$M^{(2)} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, M^{(4)} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}. \quad (17)$$

Since the elements in WHT basis vectors contain only 1 or -1, projection of input data onto WHT basis vectors requires only additions and subtractions. When $N = 8$, Fig. 2 shows the order-8 WHT basis vectors in natural order and sequency order. The WHT in (17) and (16) is in natural order. The natural order and the sequency order are different methods for ordering WHT basis vectors [54]. For sequency-ordered WHT, the spatial frequency extracted by the basis vector increases as the index i of basis vectors $\vec{M}^{(i)}$ increases. According to [28], PWHT is almost two orders of magnitude faster than FS and can deal with illumination effects and multiscale pattern matching.

The transform matrix for the GCK in [34] can be represented as follows:

$$V^{(N)} = M^{(N/R)} \otimes S^{(R)}, \quad (18)$$

where $M^{(N/R)}$ is the $N/R \times N/R$ WHT matrix and $S^{(R)}$ can be any $R \times R$ orthogonal matrix. The basis vectors in $\frac{1}{\sqrt{N}} M^{(u(k)\times N)}$ are orthonormal basis vectors. And we have $\|V^{(u(k)\times N)}\|_2^2 = 1$ if the basis vectors in $V^{(u(k)\times N)}$ are orthonormal. Thus, we have $\|M^{(u(k)\times N)}\|_2^2 = N$ for the WHT matrix in (16). Similarly, we have $\|V^{(u(k)\times N)}\|_2^2 = N/R$ for the GCK matrix in (18).

Considering the general lower bounding function defined in (8), we can select basis vectors of the $V^{u(k)\times N}$ in (8) from the LRP matrix in (13), from the WHT matrix $M^{(N)}$, or from the GCK matrix in (18). For example, if $N = 4$, $k = 2$, and $u(k) = k = 2$, then we can select the first two WHT basis vectors $\vec{M}^{(0)}$ and $\vec{M}^{(1)}$ in (17) for constructing the matrix $V^{(2\times 4)} = [\vec{M}^{(0)}\ \vec{M}^{(1)}]^T$ and use this matrix as the $V^{(u(k)\times N)}$ in (8).

3.3 Investigation on PWHT, PGCK, FWHT, and LRP

The $f_{low}(k,j)$ in (11) is used for LRP while the $f_{low}(k,j)$ in (15) is used for PWHT, PGCK, and FWHT. Let us consider the L_2 norm; if the basis vectors in $V^{(u(k)\times N)}$ in (15) are orthonormal, then we have $\|V^{(u(k)\times N)}\|_2 = 1$ in (11) and $(V^{(u(k)\times N)} V^{(u(k)\times N)^T})^{-1} = I^{(u(k))}$ in (15). Under this condition, both (11) and (15) yield the following relation:

$$\|\vec{X}_t - \vec{X}_w^{(j)}\|_2^2 \geq \|V^{(u(k)\times N)}(\vec{X}_t - \vec{X}_w^{(j)})\|_2^2 = f_{low}(k,j). \quad (19)$$

The inequality in (11) is more general than that in (15) because

- the inequality in (15) is only applicable for L_2 norm while the inequality in (11) is applicable for L_p norm for $p \geq 1$, and
- $V^{(u(k)\times N)}$ is required to have rank $u(k)$ for the inverse matrix in (15) while this is not required in (11).

The inequality in (15) proposed in [28] were used by PWHT, PGCK, and FWHT for the L_2 norm only. For other norms, the inequality proposed in [28] is as follows:

$$\|\vec{X}_t - \vec{X}_w^{(j)}\| \geq \frac{\|\vec{V}^{(i)^T}\vec{X}_t - \vec{V}^{(i)^T}\vec{X}_w^{(j)}\|}{\|\vec{V}^{(i)}\|}. \quad (20)$$

The inequality in (20) is a special case of that in (7) with $V^{(U\times N)}$ being a row vector in (7).

The following theorem describes the relationship between LRP and WHT:

Theorem 1. *When the $u = 2^n$ basis vectors in $V_{WHT}^{(u\times N)}$ are the first sequency-ordered WHT basis vectors, we have the LRP transform matrix*

$$V_{LRP}^{(u\times N)} = I^{(u)} \otimes \mathbf{1}^{(1\times(N/u))}, \quad (21)$$

such that: 1) The subspace spanned by the u basis vectors in $V_{WHT}^{(u\times N)}$ is equal to the subspace spanned by the u basis vectors in $V_{LRP}^{(u\times N)}$; 2) for any length-N input vector $\vec{X}$, if the basis vectors in $V_{LRP}^{(u\times N)}$ and $V_{WHT}^{(u\times N)}$ are normalized to have L_2 norm equal to 1, then $\|V_{WHT}^{(u\times N)}\vec{X}\|_2^2 = \|V_{LRP}^{(u\times N)}\vec{X}\|_2^2$; and 3) the transformation $V_{WHT}^{(u\times N)}\vec{X}$ requires at least $3u/2$ additions per pixel, while the transformation $V_{LRP}^{(u\times N)}\vec{X}$ requires three additions per pixel when computed on sliding windows.

The proof for Theorem 1 is provided in the Appendix, which can be found in the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPAMI.2011.106. The more the energy is packed, the more candidates can be eliminated by the rejection step. Theorem 1 states that the $u = 2^n$ sequency-ordered basis vectors in $V_{WHT}^{(u\times N)}$ and the u basis vectors in $V_{LRP}^{(u\times N)} = I^{(u)} \otimes \mathbf{1}^{(1\times(N/u))}$ are the same in subspace spanning and energy packing ability. However, the larger u is, the more computationally efficient the transformation $V_{LRP}^{(u\times N)}\vec{X}$ is than $V_{WHT}^{(u\times N)}\vec{X}$.

Recently, WHT have been used for motion estimation in [7], [8] based on L_1 norm, but the algorithms are not FS-equivalent. Actually, we can see from the analysis in (11) that the basis vectors selected from WHT and GCK are also applicable for FS-equivalent pattern matching using L_p norms as dissimilarity measure.

4 Computational Analysis of IDA, PWHT, PGCK, FWHT, and LRP

In this section, we analyze the computational complexity of IDA, PWHT, PGCK, FWHT, and LRP by counting the number of operations required by each algorithm. This analysis is summarized in Table 5. The terms related to the pattern $\vec{X}_t$ are computed once and for all at initialization time and hence not considered in the computational complexity since the associated complexity is negligible compared with the computation related to candidate windows on the image.

TABLE 5
Computation Required by the Algorithms with Symbols Illustrated in Table 3

Methods	Number of additions
FS	$2NW$
FFT	$6Jlog_2J + 7W$
IDA	$4W + 2N_PW + \sum_{k=2}^{N_P-1}[(\frac{2N}{N_P}+2)N_{can}^{(k)}] + \frac{2N}{N_P}N_{can}^{(FS)}$
LRP_{hier}	$\sum_k [2A_{LRP}^{(k,h)} + (h-1)J] + 2NN_{can}^{(FS)}$
LRP_{sld}	$2J + \sum_k (2A_{LRP}^{(k,h)} + 3W) + 2NN_{can}^{(FS)}$
LRP_{can}	$2J + \sum_k (2A_{LRP}^{(k,h)} + 3A_{LRP}^{(k,h)}) + 2NN_{can}^{(FS)}$
PKs	$B(N, N_k, W) + 2\sum_k N_{can}^{(k)} + 2NN_{can}^{(FS)}$
Methods	Number of power-p operations (LRP_{hier}, LRP_{sld} and LRP_{can} share "LRP")
FS	NW
FFT	$6Jlog_2J + J$
IDA	$J + N_PW + \frac{N}{N_P}(\sum_{k=2}^{N_P-1} N_{can}^{(k)}) + \frac{N}{N_P}N_{can}^{(FS)}$
LRP	$\sum_k A_{LRP}^{(k,h)} + NN_{can}^{(FS)}$
PKs	$\sum_k N_{can}^{(k)} + NN_{can}^{(FS)}$
Methods	Number of root-p operations
IDA	W
Methods	Number of comparison operations in Step a.2 (for IDA, PWHT, PGCK, FWHT and LRP)
	$\sum_k N_{can}^{(k)}$

PWHT, PGCK, and FWHT share the same row "PKs." In this table, $A_{LRP}^{(k,h)} = h^{(k-1)}N_{can}^{(k)}$.

To recall the notation already introduced, the $N_1 \times N_2$ pattern has $N = N_1N_2$ pixels, the $J_1 \times J_2$ image has $J = J_1J_2$ pixels and W candidate windows. At each iteration of k for $k = 1, 2, \ldots$ in the rejection step, some candidate windows are eliminated and the remaining ones will be considered in the next loop. We denote the number of candidates examined at iteration k in the rejection step as $N_{can}^{(k)}$ and the number of candidates examined in the FS-step as $N_{can}^{(FS)}$. Initially, $k = 1$, all candidates are examined, and we have $N_{can}^{(1)} = W$. Different algorithms have different values of N_{can}^{k}.

In the computational analysis, subtractions are given the same weight as additions. The computation of certain terms depends on the computation of the L_p norm in (1), which in turn is related to the computation of $|z|^p$ and $\sqrt[p]{z}$. The operations for computing $|z|^p$ and $\sqrt[p]{z}$ are here called power-p operation and root-p operation, respectively. When $p = 2$, power-p operation computes the square of z, i.e., z^2, and root-p operation computes the square root of z, i.e., $\sqrt{z}$. When $p = 1$, power-p operation computes the absolute value of z, i.e., $|z|$, and the root-p operation does nothing.

As a common step for algorithms IDA, PWHT, PGCK, FWHT, and LRP, *Step a.2* of Table 2 checks condition $f_{low}(k, j) \geq T$ for the $N_{can}^{(k)}$ candidates at iteration k, which requires $\sum_k N_{can}^{(k)}$ comparison operations. The computation required by *Step a.3* of Table 2 is negligible. The remaining steps in Table 2 that require analysis are *Step a.1* of the Rejection step, which computes the lower bounding

TABLE 6
Computation of the Lower Bounding Function Using IDA

Case 1: $k = 1$:
a.1.1: for each pixel $x_{w,a}$ in the image: {Obtain $\|x_{w,a}\|^p$}. ** J power-p operations are required for J pixels. **a.1.2**: for j=0 to $W-1$ { for m=1 to N_P {Obtain $\|\|\vec{X}_w^{(j)}\|\|_{p,P_m}$} }. ** $\|\|\vec{X}_w^{(j)}\|\|_{p,P_m} = \left(\sum_{n\in P_m} \|x_{w,j,n}\|^p\right)^{\frac{1}{p}}$ as defined in (4) is obtained by two steps: 1) sum up $\|x_{w,j,n}\|^p$ in the rectangle restricted by $\|\|\vec{X}_w^{(j)}\|\|_{p,P_m}$, where $\|x_{w,j,n}\|^p$ has been obtained as $\|x_{w,a}\|^p$ in the previous step; 2) obtain the pth root of the sum. $4W$ additions are required for obtaining the sum in the W rectangles using the box filtering technique [52]. W root-p operations are required for obtaining the pth root of the W sums. **a.1.3**: for j=0 to $W-1$ { $f_{low}(1,j) = \sum_{n=0}^{N_P-1} \|\|\vec{X}_t\|\|_{p,P_n} - \|\|\vec{X}_w^{(j)}\|\|_{p,P_n}\|^p$ }. ** $N_P W$ power-p operations, $2N_P W$ additions are required.
Case 2: $k > 1$:

a.1.4: for $N_{can}^{(k)}$ candidates $\vec{X}_w^{(j)}$ in set_{can}:
{Compute $\|\|\vec{X}_t - \vec{X}_w^{(j)}\|\|_{p,P_k}^p$.}
** $\|\|\vec{X}_t - \vec{X}_w^{(j)}\|\|_{p,P_k}^p$ as defined in (5) is the L_p norm of length-$\frac{N}{N_P}$ vector. $\frac{2N_{can}^{(k)}N}{N_P}$ additions and $\frac{N_{can}^{(k)}N}{N_P}$ power-p operations are required at iteration k.
a.1.5: for $N_{can}^{(k)}$ candidates $\vec{X}_w^{(j)}$ in set_{can}: {
$f_{low}(k,j) = f_{low}(k-1,j) + \|\vec{X}_t - \vec{X}_w^{(j)}\|_{p,P_k}^p - \left|\, \|\vec{X}_t\|_{p,P_k} - \|\vec{X}_w^{(j)}\|_{p,P_k}\right|^p$.
}
** $2N_{can}^{(k)}$ additions are required at iteration k for summing up the three terms that have been computed previously.

function $f_{low}(k,j)$, and the FS-step. In the following, we analyze the computational complexity of these two steps for each of the considered algorithms.

4.1 Computational Analysis for IDA

The computation of the lower bounding function for IDA is illustrated and analyzed in Table 6. When $k = 1$, which is *Case 1* of Table 6, the lower bounding function in (6) is given by

$$f_{low}(1,j) = \sum_{m=1}^{N_P} \left|\, \|\vec{X}_t\|_{p,P_m} - \|\vec{X}_w^{(j)}\|_{p,P_m}\right|^p. \tag{22}$$

When $k > 1$, i.e., *Case 2* of Table 6, the lower bounding function is

$$\begin{aligned} f_{low}(k,j) = f_{low}(k-1,j) &+ \|\vec{X}_t - \vec{X}_w^{(j)}\|_{p,P_k}^p \\ &- \left|\, \|\vec{X}_t\|_{p,P_k} - \|\vec{X}_w^{(j)}\|_{p,P_k}\right|^p. \end{aligned} \tag{23}$$

As a summary of Table 6, IDA requires $4W + 2N_P W + \sum_{k=2}^{N_P-1}[(\frac{2N}{N_P}+2)N_{can}^{(k)}]$ additions, $J + N_P W + \frac{N}{N_P}(\sum_{k=2}^{N_P-1} N_{can}^{(k)})$ power-p operations, and W root-p operations for obtaining the lower bounding function.

In the FS-step, IDA computes only the L_p norm dissimilarity for pixels in the last subwindow restricted by partition P_{N_P}. This subwindow contains $\frac{N}{N_P}$ pixels for each candidate window. Thus, the FS-step requires $\frac{2NN_{can}^{(FS)}}{N_P}$ additions and $\frac{NN_{can}^{(FS)}}{N_P}$ power-p operations.

TABLE 7
Computation of the Lower Bounding Function Using PWHT, GCK, and FWHT

a.1.1: for $N_{can}^{(k)}$ windows $\vec{X}_w^{(j)}$ in set_{can} { Obtain $\vec{V}^{(k-1)^T}\vec{X}_w^{(j)}$ }. ** $B(N, N_k, W)$ additions are required for obtaining $V^{(N_k\times N)}\vec{X}_w^{(j)}$ for all candidates, where $\vec{V}^{(k-1)^T}\vec{X}_w^{(j)}$ is one of its elements. **a.1.2**: for $N_{can}^{(k)}$ windows $\vec{X}_w^{(j)}$ in set_{can} { Obtain $\|\vec{V}^{(k-1)^T}\vec{X}_t - \vec{V}^{(k-1)^T}\vec{X}_w^{(j)}\|^p$ }. ** $N_{can}^{(k)}$ additions and $N_{can}^{(k)}$ power-p operations are required at iteration k. **a.1.3**: for $N_{can}^{(k)}$ windows $\vec{X}_w^{(j)}$ in set_{can} $\{f_{low}(k,j) = f_{low}(k-1,j) + \|\vec{V}^{(k-1)^T}\vec{X}_t - \vec{V}^{(k-1)^T}\vec{X}_w^{(j)}\|^p$ }. ** $N_{can}^{(k)}$ additions are required at iteration k for summing up the two terms that have been computed previously.

4.2 Computational Analysis for LRP

As pointed out in [32], LRP requires $\sum_k [2A_{LRP}^{(k,h)} + (h-1)J]$ additions and $\sum_k (A_{LRP}^{(k,h)})$ power-p operations for obtaining the lower bounding function, where $A_{LRP}^{k,h} = h^{(k-1)}N_{can}^{(k)}$. In the FS-step, LRP requires $2NN_{can}^{(FS)}$ additions and $NN_{can}^{(FS)}$ power-p operations.

The sliding transformation $V^{(u(k)\times N)}\vec{X}_w^{(j)}$ in

$$f_{low}(k,j) = \frac{\|V^{(u(k)\times N)}\vec{X}_t - V^{(u(k)\times N)}\vec{X}_w^{(j)}\|_p^p}{\|V^{(u(k)\times N)}\|_p^p}$$

is actually the sum of pixel values in rectangles when LRP is used for images. This transformation is computed by hierarchical summation on the entire image in [32], which requires $(h-1)J$ additions at each iteration k. This approach is denoted as LRP_{hier}.

4.3 Analysis for PWHT, PGCK, and FWHT

The lower bounding function for PWHT, PGCK, and FWHT is computed as

$$f_{low}(k,j) = f_{low}(k-1,j) + |\vec{V}^{(k-1)^T}\vec{X}_t - \vec{V}^{(k-1)^T}\vec{X}_w^{(j)}|^p, \tag{24}$$

where $k = 1, 2, \ldots, N_k$, $f_{low}(0,j) = 0$. N_k records the actual number of iterations run in the rejection step. The procedure and number of operations required to obtain the lower bounding function is reported in Table 7. Let $B(N, N_k, W)$ be the number of operations required by PWHT, PGCK, or FWHT to obtain $V^{(N_k\times N)}\vec{X}_w^{(j)}$ for W candidates. PWHT [28] has two different approaches to compute the transformation: top-down and bottom-up. We have $B(N, N_k, W) = WN_k \log N$ in the worst case and $B(N, N_k, W) = 2WN_k$ in the best case for top-down PWHT. For bottom-up PWHT, $B(N, N_k, W) = \sum_k NN_{can}^{(k)}$ in the worst case and $B(N, N_k, W) = \sum_k N_{can}^{(k)}$ in the best case. As for PGCK and FWHT, $B(N, N_k, W) = 2WN_k$ and $B(N, N_k, W) = 3WN_k/2$, respectively. As a summary of Table 7, PWHT, PGCK, and FWHT need $B(N, N_k, W) + 2\sum_k N_{can}^{(k)}$ additions and $\sum_k N_{can}^{(k)}$ power-p operations in order to calculate the lower bounding function.

In the FS-step, $2NN_{can}^{(FS)}$ additions and $NN_{can}^{(FS)}$ power-p operations are needed by PWHT, PGCK, and FWHT.

4.4 New Implementation of LRP

The hierarchical summation approach in [32] is a technique that is dependent on the parameter h. However, we can use the integral image method to compute the sum of pixel values in rectangles. The integral image method initially requires about $2J$ additions for constructing the integral image and then $3W$ additions for each iteration k for computing the sliding transformation over the image. This approach is denoted as LRP_{sld}. LRP_{sld} is advantageous over the hierarchical summation proposed in [32] because the computation required by LRP_{sld} is independent of h.

As another alternative way, we may compute the transformation for the $N_{can}^{(k)}$ candidates instead of computing it for the entire image, which would require $3h^{(k-1)}N_{can}^{(k)}$ additions for each k using the integral image method. This approach is denoted as LRP_{can}.

4.5 New Termination Condition Based on Computational Complexity Analysis

As illustrated before, the termination condition in [28] has been used by PWHT and PGCK for early termination of the rejection step, which corresponds to $Cond_{Ter}$ at *Step a.3* in Table 2. Let the computation required for the FS-step be C_{FS} and the computation required for iteration $k+1$ in the rejection step be $C_{Rej}^{(k+1)}$. The idea behind the strategy in [28] is that the rejection step should be terminated when it is more efficient to directly calculate the actual distance of the remaining candidates in the FS-step than to continue the computation in the rejection step, i.e., when $C_{FS} < C_{Rej}^{(k+1)}$. The strategy proposed in [28] terminates the rejection step when the percentage of remaining candidate windows checked in iteration $k+1$, i.e., $Per_{can}^{(k+1)}$, is below a certain threshold ϵ. Hence, in the absence of a computational analysis, setting a threshold ϵ is intuitively a simplified version of the comparison $C_{FS} < C_{Rej}^{(k+1)}$. However, the computational complexity given in Table 5 allows us to devise a novel and principled approach to the early termination strategy.

The new termination strategy based on computational analysis consists of terminating the rejection step if $Cond_{Ter,1}$ is true or $Cond_{Ter,2}$ is true. $Cond_{Ter,1}$ is true when the computation required for the iteration $k+1$ of the rejection step is greater than the computation required for the FS-step. $Cond_{Ter,2}$ is true when the computation required by iteration k of the rejection step is greater than the computation it saves. This strategy is easily applicable to all the algorithms evaluated in this paper. In the following experiments, this new termination strategy is used for PGCK and FWHT unless specified otherwise. In Section 6.7, we will compare the original strategy in [28] with the proposed strategy.

5 Performance Evaluation

5.1 Data Set

In order to evaluate the performance of the compared algorithms, 14 data sets containing different sizes of patterns and images are used. As shown in Table 8, the data sets are denoted as $In_1 - Tn_2$ for $n_1 = 1, 2, 3, 4, 5$, $n_2 = 1, \ldots, 4$, where n_1 corresponds to image size and n_2 corresponds to pattern size. Large n_1 or n_2 corresponds to large size. For example, data sets $I2 - T1$ and $I2 - T2$ have the same image size 320×240 but different pattern sizes.

TABLE 8
Data Sets Used in the Experiments

Dataset	Image size	J	Pattern size	N
$I1 - T1$	160×120	19200	16×16	256
$I2 - T1$	320×240	76800	16×16	256
$I2 - T2$	320×240	76800	32×32	1024
$I3 - T1$	640×480	307200	16×16	256
$I3 - T2$	640×480	307200	32×32	1024
$I3 - T3$	640×480	307200	64×64	4096
$I4 - T1$	1280×960	1228800	16×16	256
$I4 - T2$	1280×960	1228800	32×32	1024
$I4 - T3$	1280×960	1228800	64×64	4096
$I4 - T4$	1280×960	1228800	128×128	16384
$I5 - T1$	2560×1920	4915200	16×16	256
$I5 - T2$	2560×1920	4915200	32×32	1024
$I5 - T3$	2560×1920	4915200	64×64	4096
$I5 - T4$	2560×1920	4915200	128×128	16384

J and N correspond to the number of pixels in image and pattern, respectively.

The experiments include a total of 150 grayscale images chosen among three databases: MIT [55], medical [56], and remote sensing [57]. 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. The 150 images have been subdivided into five groups of 30 images, each group being characterized by a size of images in $I1 - T1$, $I2 - T1$, $I3 - T1$, $I4 - T1$, $I5 - T1$, (i.e., 160×120, 320×240, 640×480, $1{,}280 \times 960$, $2{,}560 \times 1{,}920$). 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 data set. Data sets having the same image size share the same images but have different patterns in both size and location. For example, data sets $I2 - T1$ and $I2 - T2$ share the same 30 images having size 320×240 but have different patterns in size and location. So, each data set contains 300 image-pattern pairs. Since we have 14 data sets, there are 4,200 image-pattern pairs in all. This data set originated from that in [33] and is extended in this paper to include more sizes of patterns.

5.2 Evaluation Criterion

In the experiments, both SSD and SAD are used as the dissimilarity measure between pattern and candidate window. If the SSD or SAD between any candidate window in the image and the pattern is below the threshold, the candidate window is regarded as matching the pattern. Given a pattern having N pixels, the SSD threshold is set as follows:

$$T_{SSD} = 1.1 \cdot SSD_{min} + N, \tag{25}$$

where SSD_{min} is the SSD between the pattern and the best matching window. Similarly, we have the following for the SAD threshold:

$$T_{SAD} = 1.1 \cdot SAD_{min} + N, \tag{26}$$

TABLE 9
Parameters Used in the Experiments

LRP	$h = 4$
IDA	If $N \leq 1024$ in Table 8, $N_p = 4$; If $N > 1024$ in Table 8, $N_p = 8$.
PWHT/PGCK/FWHT	$N_{Maxk} = 96$

where SAD_{min} is the SAD between the pattern and the best matching window.

We developed the code for IDA and FWHT since we are authors of these algorithms. The code for Hel-Or and Hel-Or's PWHT [28] is from the authors' website [58], with some small modifications introduced by us to deal with large patterns. The code for PGCK [34] is based on the code provided by Hel-Or and Hel-Or, which was previously used for motion estimation in [8] and was modified by us for the pattern matching task. Finally, we wrote the code for LRP according to the algorithm described in [32]. All of the algorithms are written in C, compiled with VC 6.0, and run on windows XP systems as single-thread tasks. Our implementation checks that all algorithms find the same matched windows and same distance as FS in all experiments for assuring correctness results are measured as speed-ups over the FS algorithm in terms of execution time. As an example, the speed-up of IDA over FS in execution time is measured as the execution time required by FS divided by that required by IDA. The parameter N_p for IDA are the same as in [33]. The parameter h for LRP is 4, which is the same as in [32]. The parameters used in the experiments are summarized in Table 9. The parameter N_{Maxk} is set as 50 for PWHT in [28], while it is set as 96 in our experiments because we find that setting $N_{Maxk} = 96$ makes PWHT faster than setting $N_{Maxk} = 50$ in most cases, especially when the pattern size is large and the noise level is high. Since all the evaluated algorithms find the same matching windows as the FS, the only concern is computational efficiency, which is assessed here in terms of execution time. In particular, as listed in Table 10, we have measured the execution time on three Intel CPUs and one AMD CPU with different memory sizes. The memory requirement for all algorithms is smaller than 1 GB in our implementation. If not otherwise specified, the reported speed-ups in execution time are averages of the speed-ups measured in each of the four environments. As an example, the speed-up of IDA in execution time is the average of the speed-ups of IDA over FS measured in the four environments.

In Sections 4.2 and 4.4, three different implementations of the LRP algorithm were introduced. LRP_{hier} is the original implementation that uses the hierarchical summation. LRP_{sld} computes the transformation in a sliding manner on the entire image using the integral image, while LRP_{can} computes the transformation for $N_{can}^{(k)}$ candidate windows at loop k. These three implementations were evaluated in the experiments.

TABLE 10
Hardware Environment Used in the Experiments

Environment	CPU	Memory size
$Env1$	Intel core 2 (6400) 2.13GHz	3G
$Env2$	Intel Pentium 4 2.8GHz	1G
$Env3$	Intel Xeon 3GHz	2G
$Env4$	AMD Athlon 64 X2 2.21GHz	3G

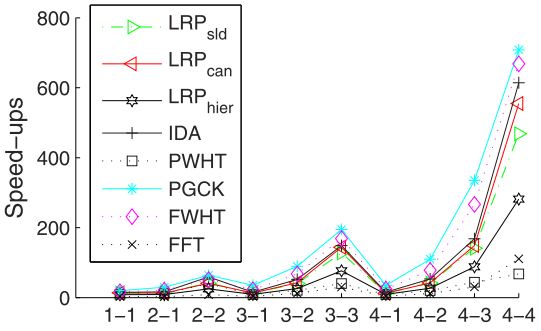


Fig. 3. Speed-ups in execution time for images without noise when SSD is used. The X-axis corresponds to data sets $I1 - T1, I2 - T1, \ldots, I4 - T4$ in Table 8.

6 Experimental Results

6.1 Experiment on Images without Noise

In this experiment, we evaluate algorithms on the data sets described in Table 8, which correspond to different sizes of images and patterns.

The speed-ups in execution time yielded by the evaluated algorithms using SSD as the dissimilarity measure are shown in Fig. 3. In this experiment, the algorithms can be ordered from fastest to slowest as follows:

1. PGCK,
2. FWHT, IDA, and LRP,
3. PWHT,
4. FFT.

PGCK is faster than FWHT because very few (less than 4) basis vectors are computed in this experiment. The experimental results in [35] also show that FWHT is slower than PGCK when the number of basis vectors used is less than 4. With the exception of data set $I4 - T4$, where FFT is faster than PWHT, FFT is always slower than the other evaluated algorithms.

The speed-ups in execution time using SAD are shown in Fig. 4. PGCK and IDA are the fastest in this experiment, with IDA faster than PGCK in data sets $I1 - T1$, $I2 - T1$, $I2 - T2$, $I3 - T2$, $I3 - T3$, and $I4 - T4$. Ordering of the other algorithms is similar to that attained using SSD (Fig. 3).

6.2 Experiment on Images with Gaussian Noise

In this experiment, four different levels of iid zero-mean Gaussian noise are added to each image of the data sets described in Table 8. The four Gaussian noise levels having

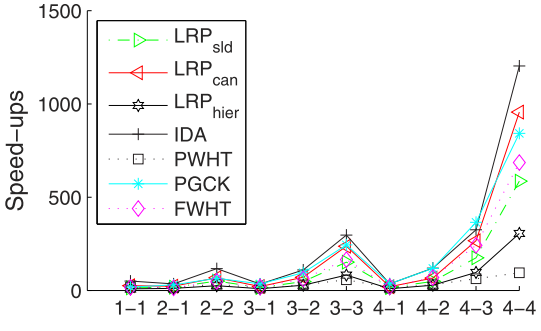


Fig. 4. Speed-ups in execution time for images without noise when SAD is used. The X-axis corresponds to data sets $I1 - T1, I2 - T1, \ldots, I4 - T4$ in Table 8.

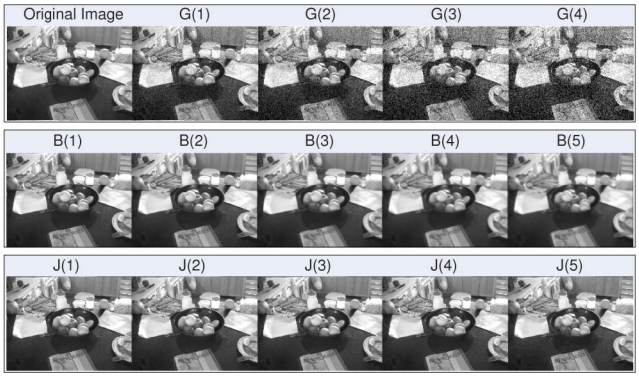


Fig. 5. An image from the data set and its distorted images. First row: The original image and its images with Gaussian noise levels $G(1)$ to $G(4)$; second row: images with blurring levels $B(1)$ to $B(5)$; third row: images with JPEG compression quality levels $J(1)$ to $J(5)$.

variances 325, 650, 1,300, and 2,600[2] for 8-bit pixel value are referred to as $G(1)$, $G(2)$, $G(3)$, and $G(4)$, respectively. Fig. 5 shows an image from the data set in Table 8 and its distorted images with noise levels $G(1)$ to $G(4)$, where the 640×480 distorted images have PSNR 23.23, 20.4, 17.7, and 16.1 when compared with the original image.

The speed-ups in execution time using SSD for this experiment are shown in Fig. 6. Each subfigure in Fig. 6 corresponds to a data set in Table 8. In Fig. 6, the top row corresponds to the smallest image size and bottom row corresponds to the largest image size; the leftmost column corresponds to the smallest pattern size and the rightmost column corresponds to the largest pattern size. This is similar for Figs. 7, 8, 9, 10, 11.

As shown in Fig. 6, the speed-up of FFT over FS is independent of the noise level, while the speed-ups of the other fast algorithms decrease as the noise level increases from $G(1)$ to $G(4)$, so that FFT turns out to be in most cases the fastest method when the noise level is very high, i.e., $G(4)$ in Fig. 6. At the lower noise levels, LRP_{sld} is, in most cases, the fastest algorithm, and quite close to the fastest in other cases. FWHT turns out to be faster than PGCK. This is because the actual number of computed bases N_k is greater than five in this experiment, where FWHT is faster than PGCK in computing the transformation. PGCK is faster than PWHT in most cases because PGCK is more efficient than PWHT in computing the transformation. FWHT is faster than LRP when noise is low and pattern size is small, especially when image size is large. When the number of required basis vectors is large, LRP outperforms FWHT in computing the transformation, which is proven in Theorem 1. IDA performs well when pattern-size is 16×16 and has similar performance as PWHT in other cases.

6.3 Experiment for Blurred Images

In this experiment, five different levels of Gaussian low pass filter are used for blurring each image of the data sets described in Table 8. The five blurring levels, which are referred to as $B(1)$, $B(2)$, $B(3)$, $B(4)$, and $B(5)$, correspond to Gaussian low pass filter having standard deviation $\sigma = 0.2, 0.9, 1.6, 2.3, 3$. Fig. 5 shows an image from the data set in Table 8 distorted with blurring levels $B(1)$ to $B(5)$, where the distorted 640×480 images have PSNR 27.79, 27.18, 25.36, 24.14, and 23.49, respectively, when compared with the original image. In practice, blur is introduced by changes of camera focus or by application of simple denoising techniques.

The speed-ups in execution time using SSD for this experiment are shown in Fig. 7. FFT, FWHT, and LRP_{sld} are the three fastest algorithms. FFT is the fastest for image size 160×120 with blurring levels $B(2)$ to $B(5)$, and, for image sizes 320×240 and 640×480 with blurring levels $B(4)$ and $B(5)$, FWHT is the fastest for data sets $I4 - T1$ and $I5 - T2$ with blurring levels $B(1)$ to $B(2)$, and, for data set $I5 - T1$ with blurring levels $B(1)$ to $B(3)$, LRP_{sld} is the fastest in other cases. FWHT is faster than PGCK; PGCK is faster than PWHT. IDA is faster than PGCK for pattern size 16×16 and has similar speed-up as PGCK in other cases.

6.4 Experiment for JPEG Compressed Images

In this experiment, five different JPEG compression quality levels are used for encoding the original image of the data sets described in Table 8. The JPEG compression quality levels, which are referred to as $J(1)$, $J(2)$, $J(3)$, $J(4)$, and $J(5)$, correspond to quality measure $Q_{JPG} = 90, 70, 50, 30, 10$, respectively

The speed-ups in execution time using SSD are shown in Fig. 8. The performance of FFT algorithm is independent of compression quality, while the speed-up of other fast algorithms decreases as the image quality decreases from $J(1)$ to $J(5)$. We can see that almost all the algorithms outperform the FFT algorithm, with an exception that PWHT is outperformed by FFT in data sets $I4 - T4$ and $I5 - T4$. PGCK and FWHT are the fastest or close to the fastest in most cases for pattern size 16×16 and 32×32. LRP_{can} and FWHT are the fastest in most cases for pattern size 64×64 and 128×128.

6.5 Experimental Results When SAD Is Used

For the experiments in this section, speed-ups in execution time are evaluated using SAD as the dissimilarity measure. The perturbations correspond to those introduced in Sections 6.2-6.4. Since the FFT approach cannot be applied when SAD is used, it does not appear in the figures.

Fig. 9 shows the results for images with Gaussian noise. IDA is the fastest for pattern size 16×16, while LRP_{can} and LRP_{sld} are the fastest and have similar performance in other cases.

The experimental results for low pass filter blurred images in Fig. 10 show that the algorithms from the fastest to the slowest for pattern size 16×16 can be ordered as:

1. IDA,
2. LRP_{sld},
3. LRP_{can},
4. LRP_{hier},
5. PGCK and FWHT,
6. PWHT.

The order for other pattern sizes is

1. LRP_{sld} or LRP_{can},
2. LRP_{hier},
3. IDA,
4. PWHT, PGCK, and FWHT,

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

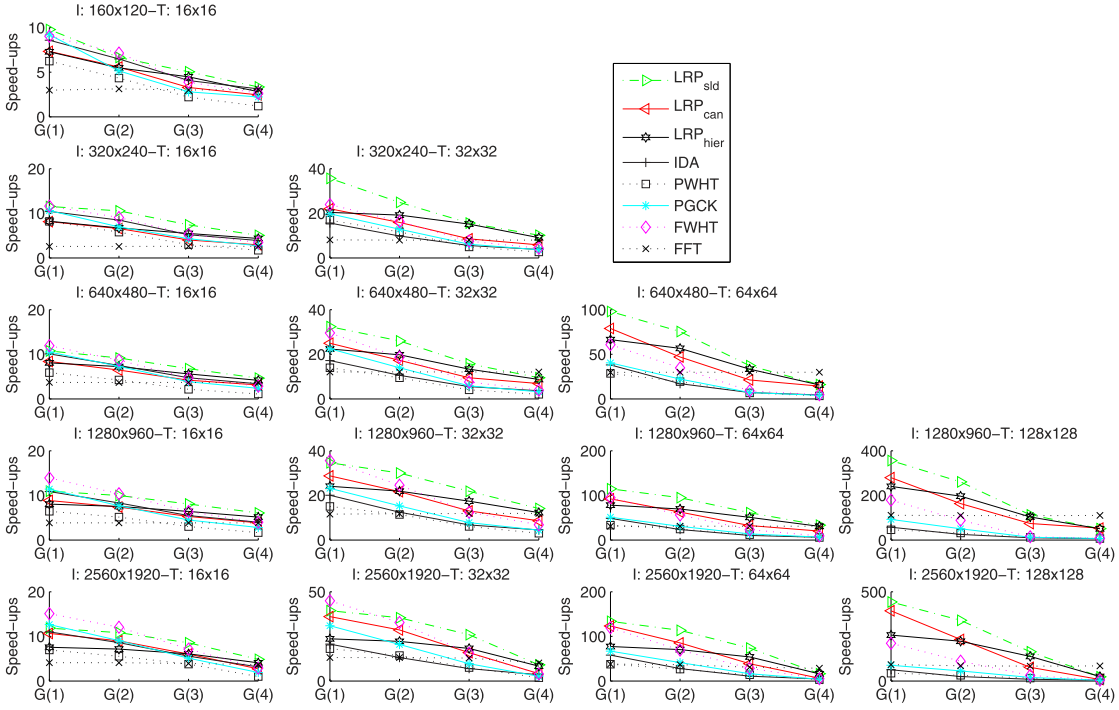


Fig. 6. Speed-ups in execution time for images with Gaussian noise when SSD is used. "I: 160×120-T: 16×16" corresponds to the 16×16 pattern sought in the 160×120 image denoted as $I1 - T1$ in Table 8.

where LRP_{can} is the fastest for data sets $I3 - T3$, $I4 - T2$, $I4 - T3$, and $I4 - T4$ with noise levels $B(1)$ and $B(2)$, while LRP_{sld} is the fastest for other cases.

The experimental results for JPEG compressed images in Fig. 10 show that IDA is the fastest for pattern size 16×16. IDA is also the fastest for quality levels $J(1)$ and $J(2)$ in all data sets except that it is outperformed by LRP_{can} for $J(2)$ in $I3 - T3$ and $I4 - T4$. Otherwise, when pattern size is larger than 16×16 and noise level is high, LRP_{can} is the fastest in most cases.

In summary, IDA is the fastest in execution time when pattern size is 16×16, while LRP_{sld} is the fastest in larger pattern sizes for data sets with Gaussian noise and Gaussian low pass filter. LRP_{can} is the fastest for large pattern size and high noise level, while IDA is the fastest in other cases for JPEG compressed images. The comparative performance

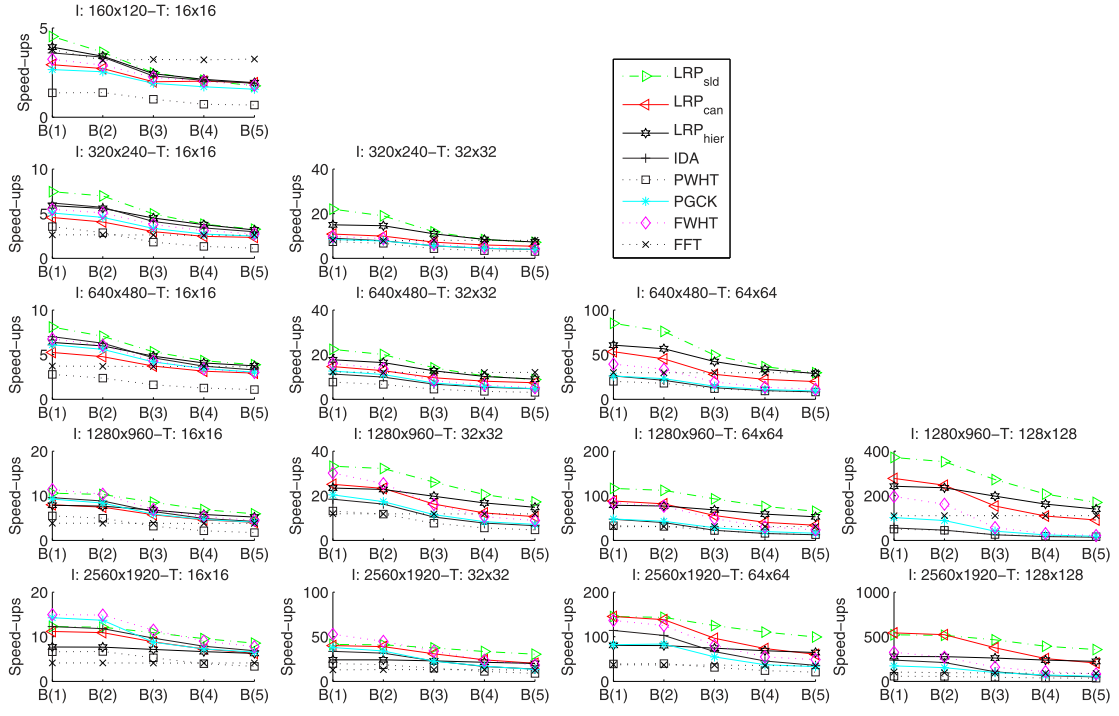


Fig. 7. Speed-ups in execution time for blurred images when SSD is used.

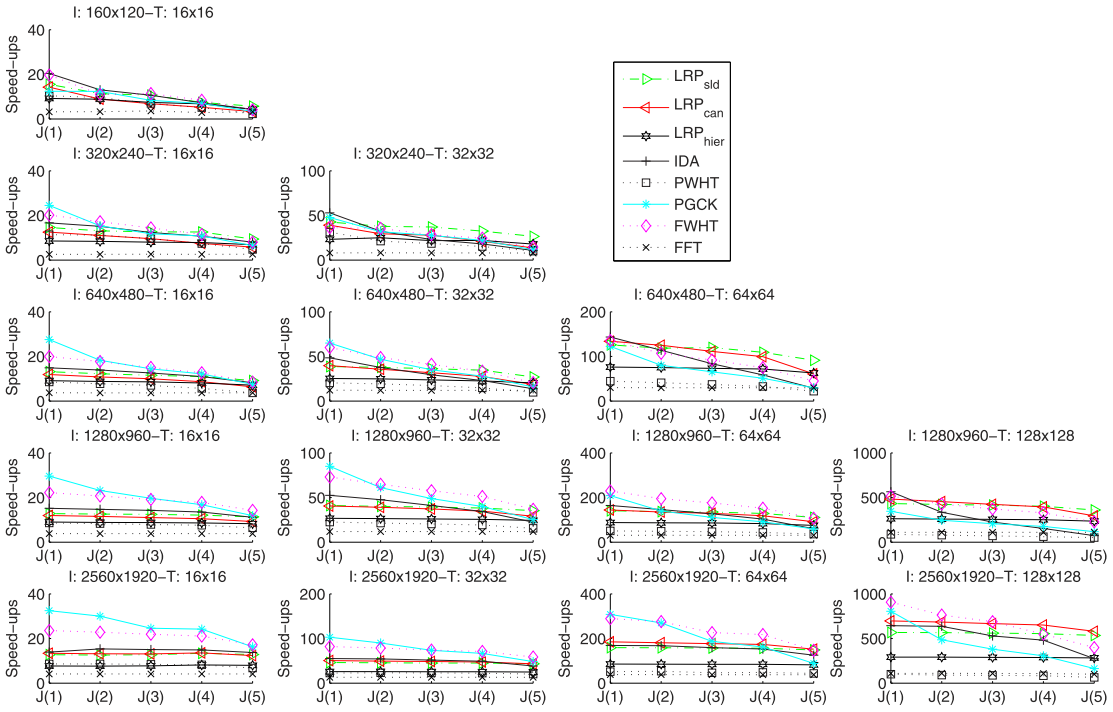


Fig. 8. Speed-ups in execution time for JPEG compressed images when SSD is used.

of LRP_{sld}, LRP_{can}, and IDA using SAD is similar to that using SSD for the three noise types. PWHT, PGCK, and FWHT are inefficient in the experiments when SAD is used and perform better when SSD is used.

6.6 Analysis of the Experimental Results

The experimental results show some common properties of the evaluated FS-equivalent algorithms. 1) With pattern size fixed, the speed-ups over FS achieved by the fast algorithms increase by less than 2 as image size J increases by 4, e.g., from $I2-T1$ to $I3-T1$ and from $I3-T1$ to $I4-T1$. 2) With image size fixed, the speed-ups increase by about 2 to 4 as pattern size N increases by 4, e.g., from $I2-T1$ to $I2-T2$. 3) The speed-up for IDA, PWHT, PGCK, FWHT, and LRP decreases as the distortion level increases, e.g., from $G(1)$ to $G(4)$, because the difficulty in efficiently rejecting mismatching windows for these algorithms increases.

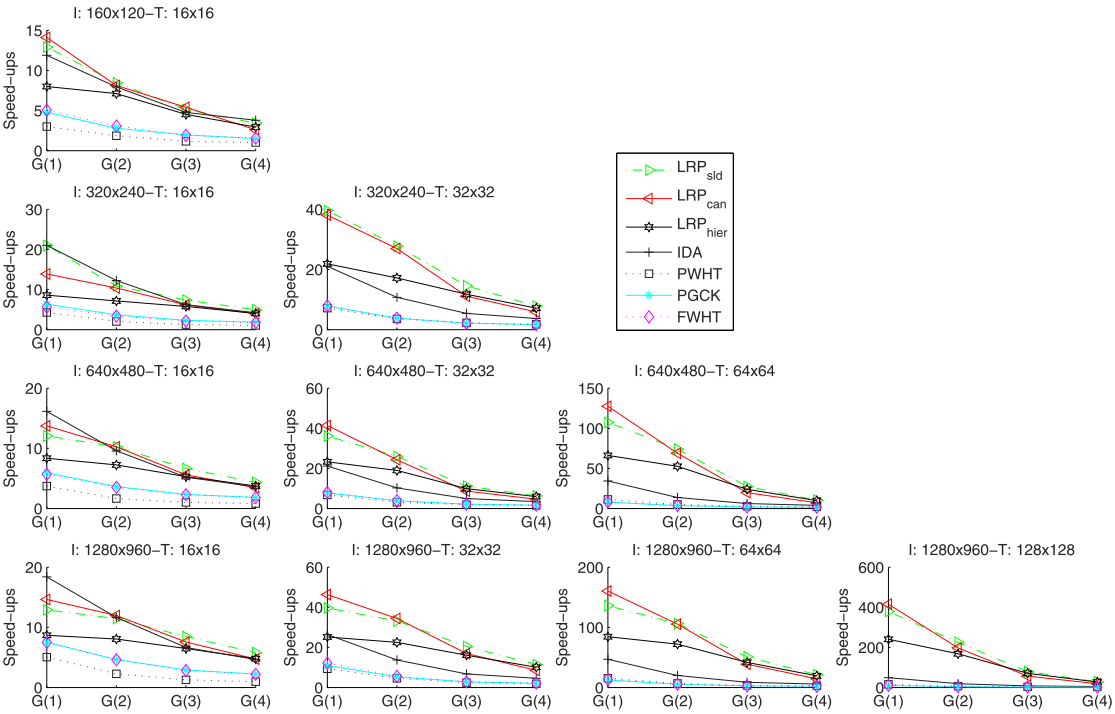


Fig. 9. Speed-ups in execution time for images with Gaussian noise when SAD is used.

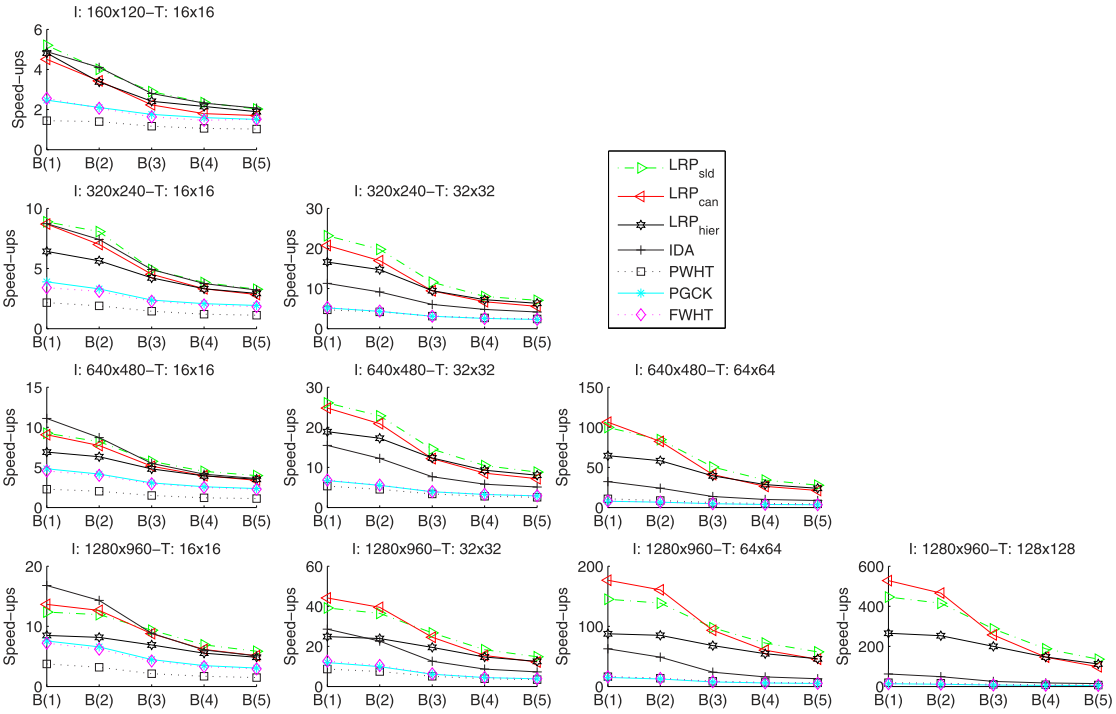


Fig. 10. Speed-ups in execution time for blurred images when SAD is used.

PGCK is faster than LRP and FWHT for data sets without noise in Figs. 3 and 4 and for data sets having low noise and small pattern size in Figs. 8 and 11. LRP and FWHT are faster than PGCK for most cases in Figs. 6, 7, and 8. Analysis of these results is as follows: The computational efficiency of FS-equivalent pattern matching algorithms is dependent on two factors: 1) the rejection power of lower bounds, which is dependent on the tightness of the lower bound function, and 2) the cost of computing lower bounds. As a comparison of LRP and WHT: 1) WHT can obtain a tighter lower bound and thus can reject more mismatched candidates compared to LRP when the number of bases u is not a power of 2; and 2) LRP is more and more efficient than WHT in computing transformation as u is larger and larger according to Theorem 1. The experiment in Fig. 12 is used as a example

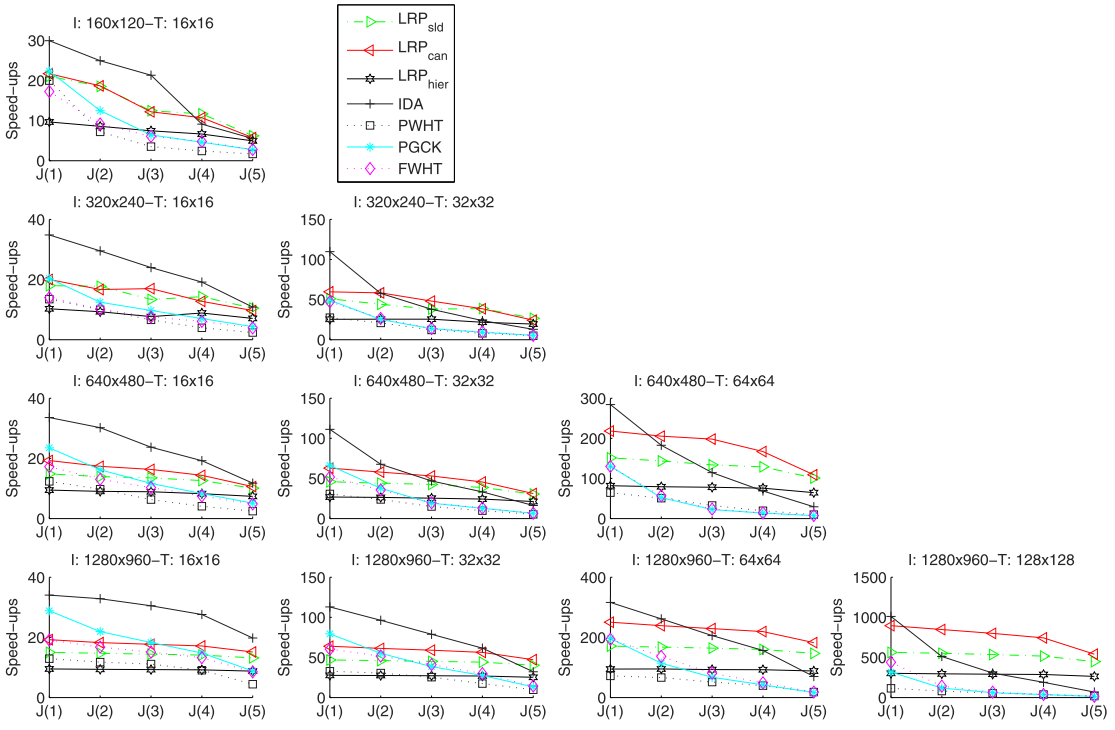


Fig. 11. Speed-ups in execution time for JPEG compressed images when SAD is used.

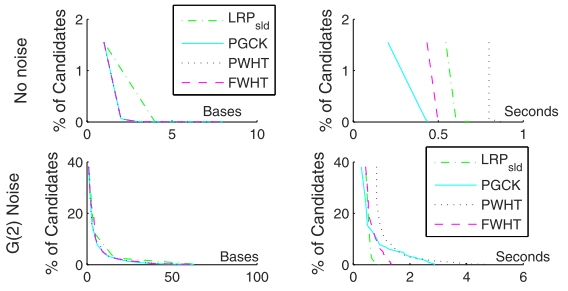


Fig. 12. The percentage of remaining candidates as a function of the number of bases (left column) and execution time (right column), when no noise is added (upper row) and with Gaussian noise $G(2)$ (bottom row). The experiment is done for data set $I2-T2$ in Table 8 on the PC $Env1$ in Table 10.

to compare LRP and other algorithms using WHT. Fig. 12 shows the results on data set $I2-T2$, i.e., image size 320×240 and pattern size 32×32, with no noise and Gaussian noise $G(2)$. The execution time in Fig. 12 is measured on the $Env1$ in Table 10. When the noise level is low, e.g., the no noise case, upper left subfigure in Fig. 12, small u (less than 4) is sufficient for rejecting a large amount of candidates. When u is small, LRP will not obviously outperform WHT in computing the lower bound, while the tightness of the lower bound becomes the main factor for computational efficiency. As a result, WHT algorithms such as PWHT, PGCK, and FWHT outperform LRP when the noise level is low in Figs. 3, 4, 8, and 11, considering that the lower bound for WHT is tighter than LRP when $1<u<4$. PGCK is more efficient than FWHT and PWHT when $u<4$ in Figs. 3, 4, 8, 11, and 12. On the other hand, as the noise level increases, the rejection step requires more bases to reject mismatched candidates. Taking the experiment in Fig. 12 as an example, to have 2 percent remaining candidates, only $u=1$ basis is required in the no noise case, while about $u=18$ bases are required for $G(2)$ noise. As u and the noise level increase, the rejection ability for different algorithms is close, while the time required for the transformation varies and becomes the main factor that influences the computational efficiency. As illustrated in Theorem 1, the larger u is, the more efficient LRP is compared with PWHT, PGCK, and FWHT in computing the transformation. Thus, LRP performs increasingly better compared with PWHT, PGCK, and FWHT as the noise level becomes larger from $G(1)$ to $G(4)$ in Fig. 6, from $B(1)$ to $B(5)$ in Fig. 7, or from $J(1)$ to $J(5)$ in Fig. 8. Similarly, FWHT is faster than PGCK and PWHT in pattern matching when the number of computed bases is large because FWHT computes the transformation faster.

LRP_{sld} performs better than LRP_{can} as the noise becomes larger from $G(1)$ to $G(4)$, from $B(1)$ to $B(5)$, or from $J(1)$ to $J(5)$. At loop k of the rejection step, LRP_{sld} and LRP_{can} require $3W$ and $3h^{(k-1)}N_{can}^{(k)}$ additions, respectively, for computing transformation. If the noise increases, then $N_{can}^{(k)}$ at loop k increases because the difficulty of rejection step in rejecting candidate windows increases. As $N_{can}^{(k)}$ increases, the $3h^{(k-1)}N_{can}^{(k)}$ additions required by LRP_{can} increase while the $3W$ additions required by LRP_{sld} keeps unchanged. Hence, LRP_{sld} is increasingly more efficient than LRP_{can} in computing the transformation and in computing the pattern matching. Assume $h=4$, $k=2$, and an 16×16 pattern is searched in a 256×256 image; we have $N=256$, $J=65{,}536$, and $N_{can}^{(1)}=W=(256-16+1)^2=58{,}081$. Assume $N_{can}^{(2)}=10$ in the first example, then LRP_{sld} requires $3W=174{,}243$ additions and LRP_{can} requires $3h^{(k-1)}N_{can}^{(k)}=3\times 4^{(2-1)}\times 10=120$ additions in computing transformation at loop $k=2$. Thus, LRP_{can} is more efficient than LRP_{sld} in computing transformation for this example. In this situation, the rejection step is so efficient that $N_{can}^{(1)}-N_{can}^{(2)}=58{,}071$ mismatched candidates are eliminated at the first loop of k. Situations similar to this example usually happen when noise level is small. In the second example, assume $N_{can}^{(2)}$ increases from 10 in the first example to $11W/12$ in this example, then LRP_{sld} requires $3W$ additions and LRP_{can} requires $3\times 4^{(2-1)}\times 11W/12=11W$ additions for computing transformation in loop $k=2$. Thus, LRP_{sld} is more efficient than LRP_{can} in computing transformation for this example. In this situation, the rejection step is so inefficient that only $N_{can}^{(1)}-N_{can}^{(2)}=W/12$ candidates are eliminated at the first loop of k. Situations similar to this example usually happen when larger noises are added.

LRP_{sld} outperforms LRP_{hier} in all experiments. LRP_{hier} computes transformation hierarchically and accesses memory in a jumping manner. LRP_{sld} computes transformation in sliding window manner and accesses memory continuously. Therefore, LRP_{sld} has less cache miss and is faster than LRP_{hier}.

6.7 Termination Strategy Comparison

In this experiment, we compare the proposed termination strategy with the strategy in [28]. PGCK and FWHT are used as the testing algorithms. SSD is used as the dissimilarity measure. The images with JPEG compression quality levels $J(1)$ to $J(5)$ as introduced in Section 6.4 for the data sets in Table 8 are used as the testing data sets. The speed-ups are average of the speed-ups in the environments introduced in Table 10. The experimental results in Fig. 13 show that PGCK/FWHT using the proposed termination strategy is faster than PGCK/FWHT using the strategy in [28] when pattern size is larger than 16×16.

7 Discussions

7.1 Summary of the Evaluation Results

First, we consider execution time as the criterion when SSD is used as the dissimilarity measure.

- For data set without noise, PGCK is the fastest.
- For data sets with Gaussian noise and blur, FWHT is the fastest when pattern is small, image size is large, and distortion level is low, FFT is the fastest when distortion level is high, and LRP_{sld} is the fastest in other cases.
- For data sets with JPEG compression, PGCK and FWHT are the fastest when pattern size is small or distortion level is small, while LRP_{can} is the fastest in other cases.

Second, we consider execution time as the criterion when SAD is used as the dissimilarity measure.

- For a data set with Gaussian noise and Gaussian low pass filter, IDA is the fastest in execution time when pattern size is 16×16, while LRP_{sld} is the fastest in larger pattern sizes.

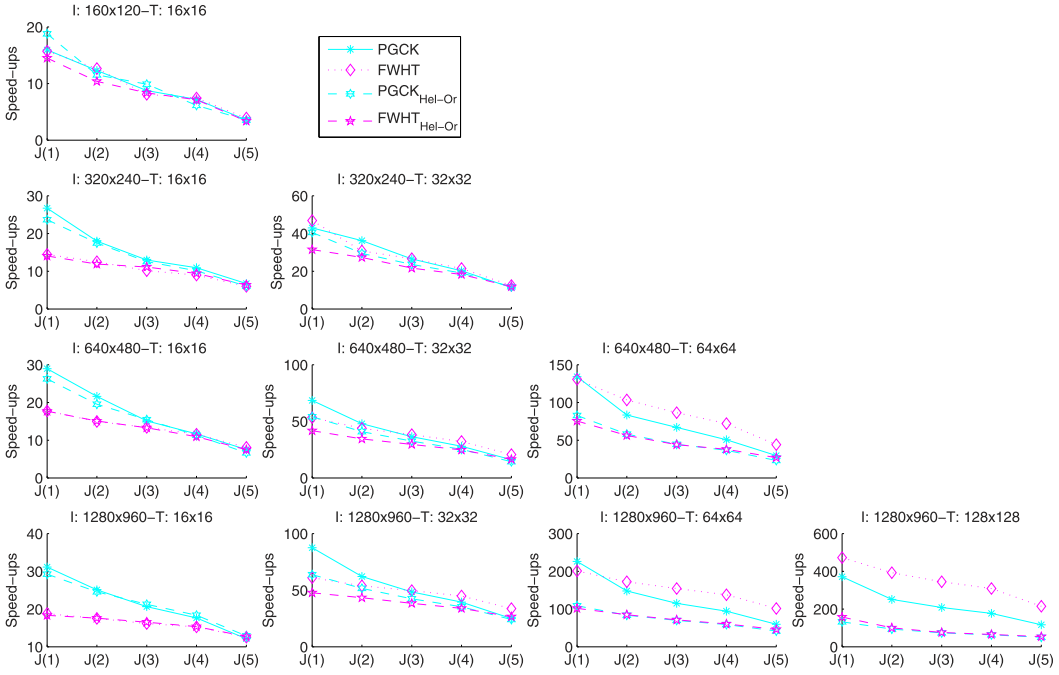


Fig. 13. Comparison of termination strategies for JPEG compressed images using SSD. PGCK and FWHT: Results using the proposed strategy; PGCK_{Hel-Or} and FWHT_{Hel-Or}: results using Hel-Or and Hel-Or's strategy in [28].

- For JPEG compressed images, LRP_{can} is the fastest for large pattern size and high distortion level, while IDA is the fastest in other cases.

Table 11 summarizes the aforementioned results.

7.2 Miscellaneous Properties of the Evaluated Algorithms

Although this paper mainly focuses on the computational efficiency of the compared algorithms, we list the miscellaneous properties of these algorithms so that suitable choice of algorithm can be made for specific applications.

1. PWHT, PGCK, and FWHT can be made DC invariant, while FS, IDA, FFT, and LRP currently cannot. DC variation is a specific light change. This property of PWHT is used in wide baseline image matching [14].
2. LRP, PWHT, PGCK, and FWHT can help deal with multiscale pattern matching when the sizes of candidate windows are the integer multiples of the size of the pattern or vice versa, while FS, IDA, and FFT currently cannot. PWHT and LRP can compute transformation for windows having different scales at the same time when the pattern is scaled by powers of 2. As shown by the experimental result in [28], when an 8×8 pattern is sought within a image at scales 8×8, 16×16, 32×32 simultaneously, PWHT can compute the transform for window sizes 8×8, 16×16, 32×32 on the image simultaneously and the remaining window is less than 2 percent after the first projection. This property is utilized in [7].
3. Sometimes the pattern to be matched may not be rectangular; the method proposed by Ben-Yehuda et al. in [37] helps PWHT, PGCK, and FWHT to deal with this problem by segmenting the pattern into multiple dyadic components. The irregularity of pattern will have no influence on FS, will have small influence on IDA, and will have much influence on algorithms FFT, LRP, PWHT, PGCK, and FWHT that require input data size to be $a \cdot 2^b$ for $a = 1, 2, \ldots, b = 1, 2, \ldots$.

8 Conclusion

In this paper, we have presented execution time evaluation and computational complexity analysis of recent FS-equivalent algorithms. The algorithms have been compared considering different sizes of images and patterns in the presence of Gaussian noise, image blur, and JPEG compression. Our experimental results clearly show how the fastest algorithm is different under different conditions. Nonetheless, experimental evidence also suggests that, overall, LRP may be considered the best performing algorithm, for it turns out to be the fastest in most cases. This nicely agrees with Theorem 1, which provides a theory to analyze why

TABLE 11
Best Overall Algorithms for Measured Execution Times for Different Disturbance Factors and Matching Measures

	No Disturbance	Gaussian Noise	Blur	JPEG compression
SSD	PGCK	LRP / FWHT / FFT	LRP / FWHT / FFT	LRP / PGCK / FWHT
SAD	PGCK / IDA	LRP / IDA	LRP / IDA	LRP / IDA

LRP is faster than other recent approaches that use WHT for transform domain pattern matching in these cases. Throughout the paper, we have discussed the motivations underpinning the relative merits and limits of the considered algorithms under the different working conditions so as to possibly inspire the development of new methods in the active research field of fast full search equivalent pattern matching.

ACKNOWLEDGMENTS

The authors wish to thank Professor Yacov Hel-Or and Professor Hagit Hel-Or for providing their code implementing GCK and WHT, 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, and the anonymous reviewers for many constructive suggestions.

REFERENCES

[1] M.S. Aksoy, O. Torkul, and I.H. Cedimoglu, "An Industrial Visual Inspection System that Uses Inductive Learning," *J. Intelligent Manufacturing,* vol. 15, no. 4, pp. 569-574, 2004.

[2] A.W. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-Based Rendering Using Image-Based Priors," *Proc. IEEE Ninth Int'l Conf. Computer Vision,* vol. 2, pp. 1176-1183, 2003.

[3] T. Luczak and W. Szpankowski, "A Suboptimal Lossy Data Compression Based on Approximate Pattern Matching," *IEEE Trans. Information Theory,* vol. 43, no. 5, pp. 1439-1451, Sept. 1997.

[4] R.M. Dufour, E.L. Miller, and N.P. Galatsanos, "Template Matching Based Object Recognition with Unknown Geometric Parameters," *IEEE Trans. Image Processing,* vol. 11, no. 12, pp. 1385-1396, Dec. 2002.

[5] W.T. Freeman, T.R. Jones, and E.C. Pasztor, "Example-Based Super-Resolution," *IEEE Computer Graphics and Applications,* vol. 22, no. 2, pp. 56-65, Mar./Apr. 2002.

[6] A. Efros and T. Leung, "Texture Synthesis by Non-Parametric Sampling," *Proc. IEEE Seventh Int'l Conf. Computer Vision,* pp. 1033-1038, Sept. 1999.

[7] C.M. Mak, C.K. Fong, and W.K. Cham, "Fast Motion Estimation for H.264/AVC in Walsh Hadamard Domain," *IEEE Trans. Circuits Systems for Video Technology,* vol. 18, no. 6, pp. 735-745, June 2008.

[8] Y. Moshe and H. Hel-Or, "Video Block Motion Estimation Based on Gray-Code Kernels," *IEEE Trans. Image Processing,* vol. 18, no. 10, pp. 2243-2254, Oct. 2009.

[9] A. Buades, B. Coll, and J.-M. Morel, "A Non-Local Algorithm for Image Denoising," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 2, pp. 60-65, June 2005.

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image Denoising by Sparse 3D Transform-Domain Collaborative Filtering," *IEEE Trans. Image Processing,* vol. 16, no. 8, pp. 2080-2095, Aug. 2007.

[11] R. Zhang, W. Ouyang, and W.K. Cham, "Image Deblocking Using Dual Adaptive Fir Wiener Filter in the DCT Transform Domain," *Proc. IEEE Int'l Conf. Acoustics, Speech, and Signal Processing,* pp. 1181-1184, Apr. 2009.

[12] Y. Alon, A. Ferencz, and A. Shashua, "Off-Road Path Following Using Region Classification and Geometric Projection Constraints," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 689-696, June 2006.

[13] Y. Shina, J.S. Jua, and E.Y. Kim, "Welfare Interface Implementation Using Multiple Facial Features Tracking for the Disabled People," *Pattern Recognition Letters,* vol. 29, no. 13, pp. 1784-1796, Oct. 2008.

[14] Q. Wang and S. You, "Real-Time Image Matching Based on Multiple View Kernel Projection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2007.

[15] X. Wu, "Template-Based Action Recognition: Classifying Hockey Players Movement," MS thesis, Univ. of British Columbia, 2005.

[16] D.G. Lowe, "Distinctive Image Features from Scale-Invarian Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[17] K. Mikolajczyk and C. Schmid, "A Performance Evaluation of Local Descriptors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 27, no. 10, pp. 1615-1630, Oct. 2005.

[18] H. Bay, T. Tuytelaars, and L.J.V. Gool, "Surf: Speeded up Robust Features," *Proc. European Conf. Computer Vision,* vol. 1, pp. 404-417, 2006.

[19] V. Lepetit and P. Fua, "Keypoint Recognition Using Randomized Trees," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 9, pp. 1465-1479, Sept. 2006.

[20] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid, "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study," *Int'l J. Computer Vision,* vol. 73, no. 2, pp. 213-238, 2007.

[21] O. Pele and M. Werman, "Robust Real-Time Pattern Matching Using Bayesian Sequential Hypothesis Testing," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 30, no. 8, pp. 1427-1443, Aug. 2008.

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

[23] L. Di Stefano and S. Mattoccia, "Fast Template Matching Using Bounded Partial Correlation," *J. Machine Vision and Applications,* vol. 13, pp. 213-221, 2003.

[24] L. Di Stefano and S. Mattoccia, "A Sufficient Condition Based the Cauchyschwarz Inequality for Efficient Template Matching," *Proc. Int'l Conf. Image Processing,* vol. 1, pp. 269-272, Sept. 2003.

[25] S. Mattoccia, F. Tombari, and L. Di Stefano, "Fast Full-Search Equivalent Template Matching by Enhanced Bounded Correlation," *IEEE Trans. Image Processing,* vol. 17, no. 4, pp. 528-538, Apr. 2008.

[26] W.H. Pan, S.D. Wei, and S.H. Lai, "Efficient NCC-Based Image Matching in Walsh-Hadamard Domain," *Proc. 10th European Conf. Computer Vision: Part III,* 2008.

[27] S.-D. Wei and S.-H. Lai, "Fast Template Matching Based on Normalized Cross Correlation with Adaptive Multilevel Winner Update," *IEEE Trans. Image Processing,* vol. 17, no. 11, pp. 2227-2235, Nov. 2008.

[28] 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.

[29] B. Girod, *What's Wrong with Mean-Squared Error?* Chapter 15, MIT Press, 1993.

[30] S. Santini and R. Jain, "Similarity Measures," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 9, pp. 871-883, Sept. 1999.

[31] A.J. Ahumada, "Computational Image Quality Metrics: A Review," *Proc. Int'l Symp. Soc. Information Display,* vol. 24, pp. 305-308, 1998.

[32] M.G. Alkhansari, "A Fast Globally Optimal Algorithm for Template Matching Using Low-Resolution Pruning," *IEEE Trans. Image Processing,* vol. 10, no. 4, pp. 526-533, Apr. 2001.

[33] F. Tombari, S. Mattoccia, and L. Di Stefano, "Full Search-Equivalent Pattern Matching with Incremental Dissimilarity Approximations," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 1, pp. 129-141, Jan. 2009.

[34] G. Ben-Artz, H. Hel-Or, and Y. Hel-Or, "The Gray-Code Filter Kernels," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 3, pp. 382-393, Mar. 2007.

[35] W. Ouyang and W.K. Cham, "Fast Algorithm for Walsh Hadamard Transform on Sliding Windows," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, no. 1, pp. 165-171, Jan. 2010.

[36] G.J. VanderBrug and A. Rosenfeld, "Two-Stage Template Matching," *IEEE Trans. Computers,* vol. 26, no. 4, pp. 384-393, Apr. 1977.

[37] M. Ben-Yehuda, L. Cadany, and H. Hel-Or, "Irregular Pattern Matching Using Projections," *Proc. 12th Int'l Conf. Image Processing,* vol. 2, pp. 834-837, 2005.

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

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

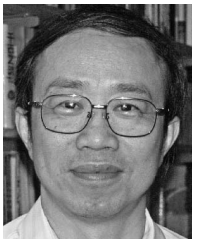
[40] W. Krattenthaler, K. Mayer, and M. Zeiler, "Point Correlation: A Reduced-Cost Template Matching Technique," *Proc. IEEE First Int'l Conf. Image Processing,* vol. 1, pp. 208-212, 1994.
[41] K. Briechle and U.D. Hanebeck, "Template Matching Using Fast Normalized Cross Correlation," *Proc. SPIE Optical Pattern Recognition XII,* pp. 95-102, 2001.
[42] P.S. Heckbert, "Filtering by Repeated Integration," *Proc. ACM SIGGRAPH '86,* pp. 315-321. 1986.
[43] P. Simard, L. Bottou, P. Haffner, and Y. Le Cun, "Boxlets: A Fast Convolution Algorithm for Signal Processing and Neural Networks," *Proc. Conf. Advances in Neural Information Processing Systems II,* vol. 11, pp. 571-577, 1999.
[44] F. Tang, R. Crabb, and H. Tao, "Representing Images Using Nonorthogonal Haar-Like Bases," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 12, pp. 2120-2134, Dec. 2007.
[45] C.D. Bei and R.M. Gray, "An Improvement of the Minimum Distortion Encoding Algorithm for Vector Quantization," *IEEE Trans. Comm.,* vol. 33, no. 10, pp. 1132-1133, Oct. 1985.
[46] W. Li and E. Salari, "Successive Elimination Algorithm for Motion Estimation," *IEEE Trans. Image Processing,* vol. 4, no. 1, pp. 105-107, Jan. 1995.
[47] H.S. Wang and R.M. Mersereau, "Fast Algorithms for the Estimation of Motion Vectors," *IEEE Trans. Image Processing,* vol. 8, no. 3, pp. 435-438, Mar. 1999.
[48] O. Pele and M. Werman, "Accelerating Pattern Matching or How Much Can You Slide?" *Proc. Eighth Asian Conf. Computer Vision,* 2007.
[49] W. Ouyang, R. Zhang, and W.K. Cham, "Fast Pattern Matching Using Orthogonal Haar Transform," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* 2010.
[50] H. Schweitzer, R. Deng, and R.F. Anderson, "A Dual Bound Algorithm for Very Fast and Exact Template-Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 33, no. 3, pp. 459-470, Mar. 2011.
[51] http://sourceforge .net/projects/opencvlibrary, 2011.
[52] M.J. McDonnell, "Box-Filtering Techniques," *Computer Graphics Image Processing,* vol. 17, pp. 65-70, 1981.
[53] P. Viola and M. Jones, "Robust Real-Time Face Detection," *Int'l J. Computer Vision,* vol. 57, no. 2, pp. 137-154, 2004.
[54] Y.A. Geadah and M.J.G. Corinthios, "Natural, Dyadic, and Sequency Order Algorithms and Processors for the Walsh-Hadamard Transform," *IEEE Trans. Computers,* vol. 26, no. 5, pp. 435-442, May 1977.
[55] A. Torralba, http://people.csail.mit.edu/torralba/images, 2011.
[56] R Koster, www.data-compression.info/corpora/lukascorpus, 2011.
[57] NASA, http://zulu.ssc.nasa.gov/mrsid, 2011.
[58] Y. Hel-Or and H. Hel-Or, www.faculty.idc.ac.il/toky/software/software.htm, 2011.

Wanli Ouyang received the BS degree in computer science from Xiangtan University, Hunan, China, in 2003. He received the MS degree in computer science from the College of Computer Science and Technology, Beijing University of Technology, Beijing, China. He received the PhD degree in 2011 and is now a postdoctoral fellow in the Department of Electronic Engineering, The Chinese University of Hong Kong. His research interests include image processing, computer vision, and pattern recognition. He is a member of the IEEE.

Federico Tombari received the BEng, MEng, and PhD degrees from the University of Bologna in 2003, 2005, and 2009, respectively. Currently, he is a postdoctoral researcher in Computer Vision Lab, DEIS (Department of Electronics, Computer Science, and Systems), University of Bologna. His research interests include computer vision and pattern recognition; in particular they include stereo vision and 3D reconstruction, object recognition, algorithms for video-surveillance. He has coauthored more than 40 refereed papers on international conferences and journals and he is a member of the IEEE and IAPR-GIRPR.

Stefano Mattoccia received the Msc degree in electronic engineering and the PhD degree in computer science from the University of Bologna in 1997 and 2002, respectively, where he is currently an assistant professor on the Faculty of Engineering, Department of Computer Science and Systems. His research interests include computer vision, especially 3D vision and matching, and embedded computer architectures for computer vision. In these fields he is the author of more than 50 refereed articles and two patents. He is a member of the IEEE, the IEEE Computer Society, IAPR, and the Interest Group on 3D Rendering, Processing, and Communications of the IEEE MMTC.

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. In 2009, he joined the Board of Directors of Datalogic SpA as an independent director. His research interests include computer vision, image processing, and computer architecture. He is the author of more than 100 papers and five patents. He is a member of the IEEE, the IEEE Computer Society, and the IAPR-IC.



Wai-Kuen Cham received the graduation degree in electronics from The Chinese University of Hong Kong in 1979. He received the MSc and PhD degrees from Loughborough University of Technology, United Kingdom, in 1980 and 1983, respectively. From June 1984 to April 1985, he was a senior engineer in Datacraft Hong Kong Limited and a lecturer in the Department of Electronic Engineering, Hong Kong Polytechnic (now The Polytechnic University of Hong Kong). Since May 1985, he has been with the Department of Electronic Engineering, The Chinese University of Hong Kong. He is a senior member of the IEEE.