# Real-Time Pattern Matching Using Projection Kernels

Yacov Hel-Or, *Member*, *IEEE*, and Hagit Hel-Or, *Member*, *IEEE*

**Abstract**—A novel approach to pattern matching is presented in which time complexity is reduced by two orders of magnitude compared to traditional approaches. The suggested approach uses an efficient projection scheme which bounds the distance between a pattern and an image window using very few operations on average. The projection framework is combined with a rejection scheme which allows rapid rejection of image windows that are distant from the pattern. Experiments show that the approach is effective even under very noisy conditions. The approach described here can also be used in classification schemes where the projection values serve as input features that are informative and fast to extract.

**Index Terms**—Pattern matching, template matching, pattern detection, feature extraction, Walsh-Hadamard.

## 1 Introduction

Many applications in image processing and computer vision require finding a particular pattern in an image. This task is referred to as *pattern matching* and may appear in various forms. Some applications require detection of a set of patterns in a single image, for instance, when a pattern may appear under various transformations or when several distinct patterns are sought in the image. Other applications require finding a particular pattern in several images. The pattern is usually a 2D image fragment, much smaller than the image. In video applications, a pattern may also take the form of a 3D spatio-temporal fragment, representing a collection of 2D patterns (see Fig. 1).

The pattern matching task can be regarded as a degenerated classification problem where a nonpattern class is to be distinguished from a single point in the pattern class or where the probability distribution of the pattern class is Gaussian [25]. Nevertheless, this task is required in other applications beyond the scope of classification. In recently popular patch-based texture synthesis methods [8], [22], a massive search for specified patterns is applied for generating a new texture patch which is perceptually similar to an example texture. Due to the high complexity and time consuming requirements of this task, several approximated search methods were suggested [22], [27]. In other studies, pattern matching schemes are used in image denoising [39], image sharpening and resolution enhancement [13], [2], texture transfer [8], image compression [23], and image-based rendering [11].

Finding a given pattern in an image is typically performed by scanning the entire image and evaluating the similarity between the pattern and a local 2D window about each pixel. In this paper, we deal with Euclidean distance; however, our scheme is applicable to any distance measure that forms a norm. Although there are some arguments against the Euclidean distance as a similarity measure for images, it is still commonly used due to its simplicity and its favorable computational complexity. For a discussion on the Euclidean distance as a similarity metric, see [14], [10], [33], [1].

Assume a 2D $k \times k$ pattern, $P(x,y)$, is to be matched within an image $I(x,y)$ of size $n \times n$. For each pixel location $(x,y)$ in the image, the following distance is calculated:

$$d_E^2(I_{x,y}, P) = \sum_{\{i,j\}=0}^{k-1} (I(x+i, y+j) - P(i,j))^2, \qquad (1)$$

where $I_{xy}$ denotes a local $k \times k$ window of $I$ at coordinates $(x,y)$. The smaller the distance measure at a particular location, the more similar the $k \times k$ local window is to the pattern. If the distance is zero, the local window is identical to the pattern. In practice, however, windows whose distance are smaller than a predefined threshold are accepted as a match (to compensate for noise, digitization errors, small transformations, etc.). In principle, the distance should be calculated for every location in the image, hence, it must be applied $n^2$ times with $k^2$ multiplications and $2k^2$ additions at each step. Fortunately, this naive approach can be expedited using the FFT transform while exploiting the convolution theorem. This reduces the calculations to $36 \log n$ additions and $24 \log n$ multiplications for each pixel of the image. Table 1 summarizes the number of operations for each approach, including run times for search of different sized patterns in a $1K \times 1K$ image. Note that, in the naive approach, the operations may be calculated in integers, while, for the Fourier approach, calculations must be performed in float. Despite this, the Fourier approach is faster and, as the pattern size increases, the advantage of the Fourier approach becomes even more prominent. However, as can be seen, even using the Fourier approach, actual runtimes are still far from real time application requirements.

- *Y. Hel-Or is with the School of Computer Science, The Interdisciplinary Center, Herzeliya 46150, Israel. E-mail: toky@idc.ac.il.*
- *H. Hel-Or is with the Department of Computer Science, The University of Haifa, Haifa 30905, Israel. E-mail: hagit@cs.haifa.ac.il.*

*Manuscript received 23 Dec. 2003; revised 22 Oct. 2004; accepted 7 Dec. 2004; published online 14 July 2005.*
*Recommended for acceptance by J. Buhmann.*
*For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0446-1203.*

0162-8828/05/$20.00 © 2005 IEEE Published by the IEEE Computer Society

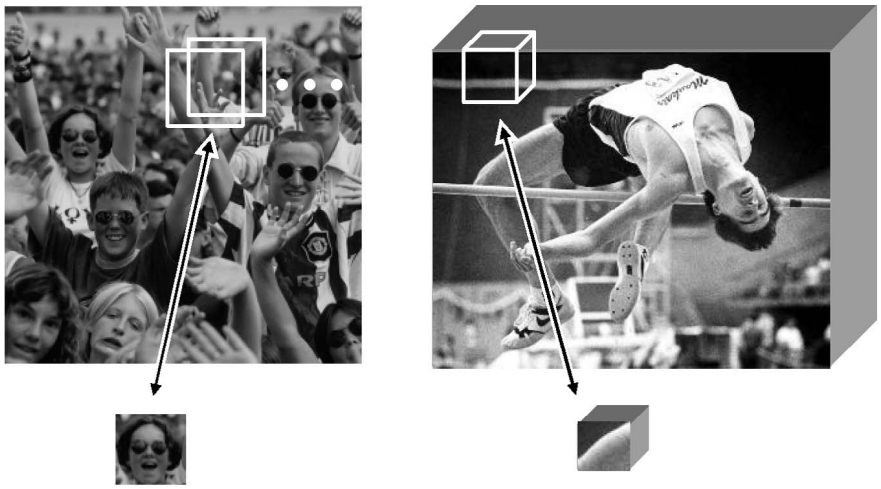


Fig. 1. Pattern matching in 2D and in 3D: For each pixel location, a local neighborhood is compared with the given pattern.

In this paper, we present a novel approach which reduces run-times by almost two orders of magnitude, as shown in Table 1. The approach is based on a projection scheme where lower bounds on the distance between a pattern and image windows are obtained using projections onto a set of kernels. The projection framework is combined with a rejection scheme which discards those windows whose distance bounds indicate that they do not match the pattern. A central point of this paper lies in the fact that a set of projection kernels are chosen such that they can be applied very fast. Therefore, tight lower bounds can be produced with very few operations, which, in turn, enable very fast rejection of a large portion of the image. The method was initially introduced in [17]. We expand this idea here, as well as introduce intensive experimental results and insights on the scheme.

The rejection approach has already been introduced (e.g., [31], [32]). It has been shown to be very effective in many applications, such as motion estimation, motion tracking, texture synthesis, classification, and more (e.g., [5], [26], [18], [3], [12], [38], [41]).

The projection approach coupled with rejection has been widely studied, especially in the context of classification. Such approaches aim to increase classification efficiency by choosing projection kernels that are optimal with respect to discrimination abilities [9], [3], [6], [19]. In this paper, a different scheme is suggested in which efficiency is achieved by choosing specific projection kernels that are very fast to apply. Thus, although more projections might be needed, the overall performance is enhanced. One major contribution of this paper is the novel method in which very fast projections can be performed.

TABLE 1
A Comparison between Existing Pattern Matching Approaches and the Proposed Approach

| | Naive | Fourier | New Approach |
|---|---|---|---|
| Average # operations per pixel | $+:\ 2k^2$ <br> $*:\ k^2$ | $+:\ 36\log n$ <br> $*:\ 24\log n$ | $+:\ 2\log k+\epsilon$ |
| Space | $n^2$ | $n^2$ | $2n^2\log k$ |
| Integer Arithmetics | Yes | No | Yes |
| Run time for $16\times 16$ | 1.33 Sec. | 3.5 Sec. | 54 Msec |
| Run time for $32\times 32$ | 4.86 Sec. | 3.5 Sec. | 78 Msec |
| Run time for $64\times 64$ | 31.30 Sec. | 3.5 Sec. | 125 Msec |

The idea of choosing projection kernels that are fast to apply was also suggested by [38], [28] in the context of classification. In [38], Viola and Jones use a set of projection kernels to produce a feature set for a classification system. The kernels are chosen such that they can be applied very rapidly using the integral image scheme [7]. This process is combined with a rejection phase where nonrelevant windows can be classified as such very efficiently. Viola and Jones's scheme is similar in spirit to the method suggested in this paper, however, in this work, we do not necessarily deal with classification problems but with a more general block matching approach. Classification techniques are impractical when many different patterns are sought or when the sought pattern is given online, e.g., in the cases of patch-based texture synthesis, block matching in motion estimation, etc. Additionally, the number of operations required to generate a projection kernel from an integral image is proportional to the order of sequency in the generated kernel (the number of sign changes along rows and columns of the projection kernel). Thus, taking into account a limited number of operations, the generated projection kernels are restricted to those with low sequency order. In some classification problems, however, such kernels can produce poor results as they may form noninformative feature inputs. In our method, this behavior is avoided since the projection kernels form a complete representation and include both low and high sequency kernels at various scales.

## 2 Bounding the Euclidean Distance Using Projections

Assume a $k\times k$ pattern is to be matched against a similarly sized window at a particular location in a given image. Referring to the pattern $\mathbf{p}$ and the window $\mathbf{w}$ as vectors in $\Re^{k^2}$,

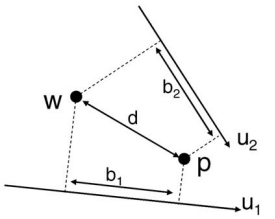



Fig. 2. Projection of $\mathbf{p}-\mathbf{w}$ onto projection vectors $\mathbf{u}_i$ produces lower bounds on the distance $\|\mathbf{d}\| = \|\mathbf{p}-\mathbf{w}\|$.

$\mathbf{d} = \mathbf{p} - \mathbf{w}$ is the difference vector between $\mathbf{p}$ and $\mathbf{w}$. Then, the Euclidean distance can be rewritten in vectorial form:

$$d_E(\mathbf{p},\mathbf{w}) = \|\mathbf{d}\| = \sqrt{\mathbf{d}^T\mathbf{d}}. \quad (2)$$

Now, assume that $\mathbf{p}$ and $\mathbf{w}$ are not given, but only the values of their projection onto a particular vector $\mathbf{u}$ (Fig. 2). Let

$$b = \mathbf{u}^T\mathbf{d} = \mathbf{u}^T\mathbf{p} - \mathbf{u}^T\mathbf{w}$$

be the projected distance value. Since the Euclidean distance is a norm, it follows from the Cauchy-Schwartz inequality that a lower bound on the actual Euclidean distance can be inferred from the projection values (see Appendix A):

$$d_E^2(\mathbf{p},\mathbf{w}) \geq b^2/\|\mathbf{u}\|^2. \quad (3)$$

If a collection of projection vectors are given $\mathbf{u}_1 \ldots \mathbf{u}_m$ along with the corresponding projected distance values $b_i = \mathbf{u}_i^T\mathbf{d}$, the lower bound on the distance can then be tightened (see Appendix B):

$$d_E^2(\mathbf{p},\mathbf{w}) \geq \mathbf{b}^T(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{b}, \quad (4)$$

where

$$\mathbf{U} = [\mathbf{u}_1 \ldots \mathbf{u}_m] \text{ and } \mathbf{b} = (b_1 \ldots b_m)^T \text{ so that } \mathbf{U}^T\mathbf{d} = \mathbf{b}.$$

Note that, if the projection vectors are orthonormal, the lower bound reduces to $\mathbf{b}^T\mathbf{b}$. As the number of projection vectors increases, the lower bound on the distance $d_E(\mathbf{p},\mathbf{w})$ becomes tighter. In the extreme case when the rank of $\mathbf{U}$ equals $k^2$, the lower bound reaches the actual Euclidean distance.

An iterative scheme for calculating the lower bound is also possible; given an additional projection vector $\mathbf{u}_{m+1}$ and projection value $b_{m+1}$, the previously computed lower bound can be updated without recalculating the inverse of the entire system $(\mathbf{U}^T\mathbf{U})^{-1}$. The iterative scheme is elaborated in Appendix C.

At first thought, it is unclear why a projection scheme should be used to calculate the distance between pattern $\mathbf{p}$ and all image windows $\mathbf{w}$, rather than computing the *exact* Euclidean distance directly. The answer lies in the appropriate selection of projection vectors. A large number of calculations can be spared if the vectors are chosen according to the following two necessary requirements:

- The projection vectors should be highly probable of being parallel to the vector $\mathbf{d} = \mathbf{p} - \mathbf{w}$.
- Projections of image windows onto the projection vectors should be fast to compute.

The first requirement implies that, on average, the first few projection vectors produce a tight lower bound on the pattern-window distance. This, in turn, will allow rapid rejection of image windows that are distant from the pattern. The second requirement arises from the fact that the projection calculations are performed many times for each window of the image. Thus, the complexity of calculating the projection plays an important role when choosing the appropriate projection vectors.

There are various sets of projection vectors that satisfy the above two requirements. We chose to demonstrate our approach using the Walsh-Hadamard basis vectors. It will be shown that these vectors capture a large portion of the pattern-window distance with very few projections on average. Additionally, a technique is presented here to calculate the projection values very efficiently. In fact, the presented technique is applicable to a large family of kernels of which the Walsh-Hadamard is a special case [4].

## 3 The Walsh-Hadamard Kernels

The Walsh-Hadamard transform has long been used for image representation under numerous applications. The elements of the (nonnormalized) basis vectors are orthogonal and contain only binary values ($\pm 1$). Thus, computation of the transform requires only integer additions and subtractions. The Walsh-Hadamard transform of an image window of size $k \times k$ (with $k$ a power of 2) is obtained by projecting the window onto $k^2$ Walsh-Hadamard basis vectors [40], [34], [21], [36]. In our case, it is required to project *each* $k \times k$ window of the $n \times n$ image onto the vectors. This results in a highly overcomplete image representation, with $k^2n^2$ projection values for the entire image; hence, the distinction between the Walsh-Hadamard transform of an image (producing $n^2$ projection values) and the *Windowed Walsh-Hadamard* transform.

The projection vectors associated with the 2D Walsh-Hadamard transform of order $k = 8$ are shown in Fig. 3. Each basis vector is of size $8 \times 8$, where white represents the value $+1$ and black represents the value $-1$. In Fig. 3, the basis vectors are displayed in order of increasing *sequency* (the number of sign changes along rows and columns of the basis vector). A diadic ordering of these vectors is shown by the overlayed arrows. This diadic ordering is induced by the algorithm discussed below and, although not exactly according to sequency, captures the increase in spatial frequency. The ordering of the basis vectors plays an important role with respect to the first requirement mentioned above, namely, that a high proportion of the window-pattern difference is captured by the first few projection vectors [20] (see Section 11 for further discussion). In terms of pattern matching, the expected lower bounds on distances between window and pattern are shown to be tight after very few projections. Fig. 4a displays this behavior by plotting the lower bound as a percentage of the total distance between image window and pattern versus the number of projection vectors used. It can be seen that the first 10 (of 256) projections capture over 70 percent of the distance. For comparison, Fig. 4b shows the same lower bound when using the standard basis for projection (delta functions), i.e., when calculating the Euclidean distance by accumulating squared differences of pixel

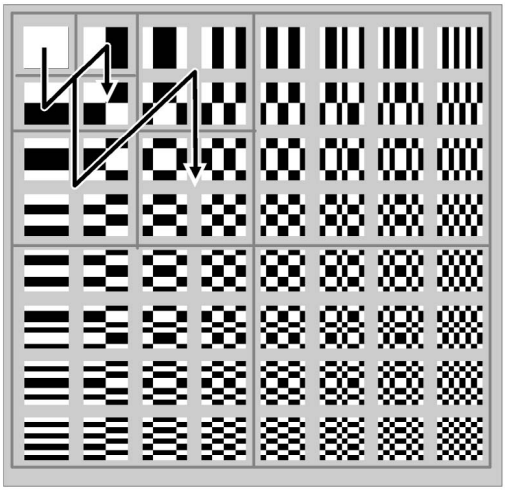


Fig. 3. The projection vectors of the Walsh-Hadamard transform of order $n = 8$ ordered with increasing spatial frequency. White represents the value $1$ and black represents the value $-1$. A diadic ordering of these basis vectors is shown by the overlayed arrows.

values. In this case, more than 180 projections were required to capture 70 percent of the distance.

As discussed above, a critical requirement of the projection vectors is the speed and efficiency of computation. An efficient method for calculating the projections of all image windows onto a sequence of Walsh-Hadamard vectors is introduced in the following section and is one of the key points of the pattern matching scheme presented in this paper.

## 4 The Projection Scheme

We introduce a novel scheme for pattern matching based on the projection of all image windows onto Walsh-Hadamard basis vectors. The suggested scheme uses a tree structure to implement these operations.

Consider first the 1D pattern matching case. Given a signal vector of length $n$ and a "pattern" vector of length $k$, the goal is to find appearances of the pattern in the signal. Toward this end, we project *each* window of the signal, of length $k$, onto a set of 1D Walsh-Hadamard basis vectors. These projections can be computed efficiently using the tree scheme depicted in Fig. 5.

There are $\log_2(k) + 1$ levels in the tree. Every node in the tree represents a vector of intermediate values used in the computation. The root node represents the original signal vector. The $i$th leaf represents a vector containing the projection values of all signal windows onto the $i$th Walsh-Hadamard basis vector. The values at each tree node are computed from the values of the father node. The symbols $+$ and $-$ represent the operations performed at a given tree node. A symbol $+$ $(-)$ on an edge connecting nodes at level $i$ and level $i + 1$ denotes the computation performed on the signal at level $i$ in which, for each $x$, the value of entry $x + \Delta$ is added (subtracted) to entry $x$ of the signal, where $\Delta = 2^i$. Thus, the two signals at level 1 are obtained by adding or subtracting consecutive entries in the signal of level 0, which is the original signal. The four signals at level 2 are obtained by adding/subtracting entries at distance 2 in the signals at level 1 and so on. Fig. 6 shows an example of computing the Walsh-Hadamard tree for a 1D signal when the pattern and window size is 4. Pseudocode for generating the Walsh-Hadamard tree is given in Appendix D.

The sequence of $+/-$ along a tree branch determines the projection vector onto which the image windows are projected. For example, the first branch shown in the tree of Fig. 5 calculates the DC component of each image window. The operations $+/-$ along the tree branches were designed to create projections onto Walsh-Hadamard vectors ordered in increasing diadic sequency (as shown in Fig. 5).

Computations within the tree are typically performed by descending from a node in the tree to its child. In the following, we list various types of projections that may be computed using the tree. The number of operations are given as well.

- **Projecting all windows onto a single projection vector.** This corresponds to evaluating *all* signal values at a single leaf of the tree. This is performed by descending the tree, top-down, from the root to the appropriate leaf. At each level, all entries at that

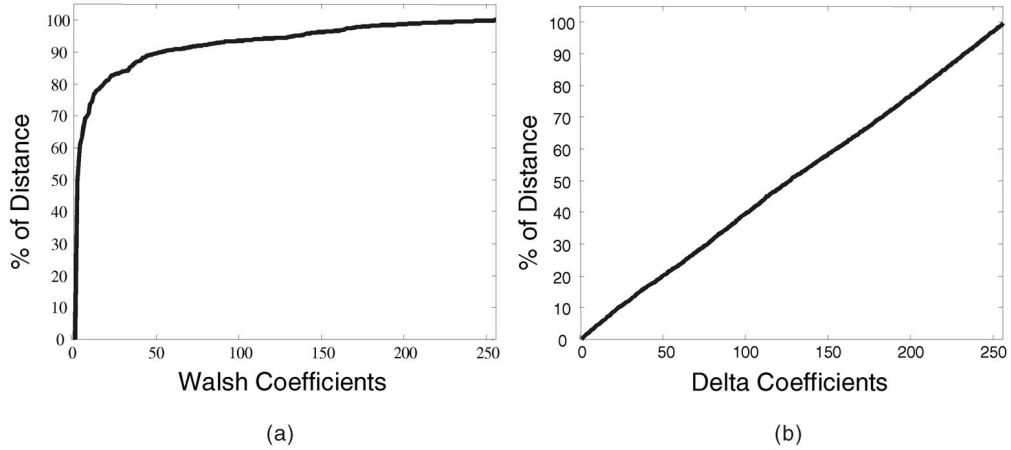



Fig. 4. The lower bound on the distance between image window and pattern as a percentage of the total distance versus the number of projection vectors used. The values displayed are averaged over 1,000 $16 \times 16$ pattern-window pairs chosen randomly from natural images. (a) Projections are onto Walsh-Hadamard basis vectors. (b) Projections are onto the standard basis vectors (i.e., delta functions).

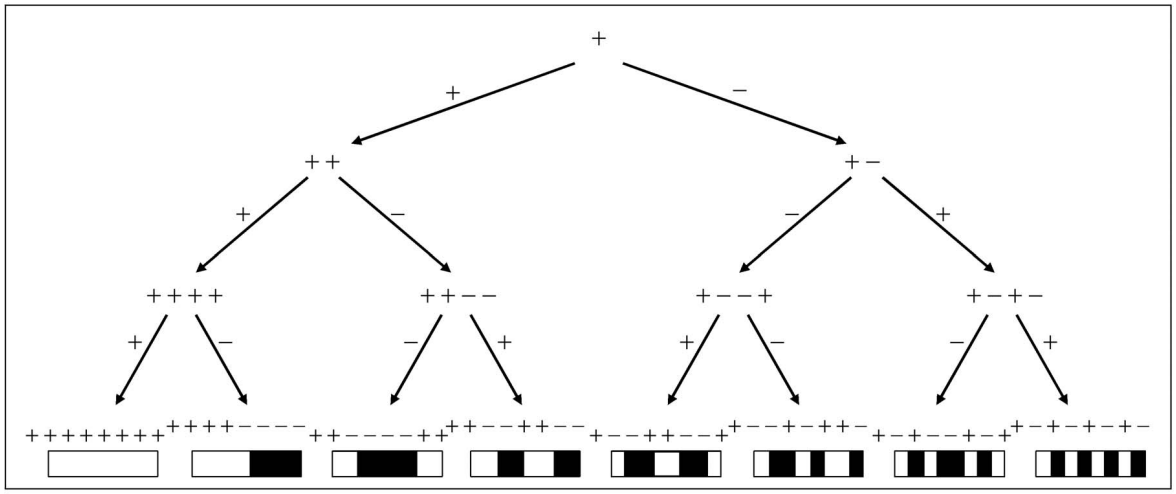


Fig. 5. The tree-scheme for computing projections of all windows of a 1D signal onto all Walsh-Hadamard kernels of order 8.

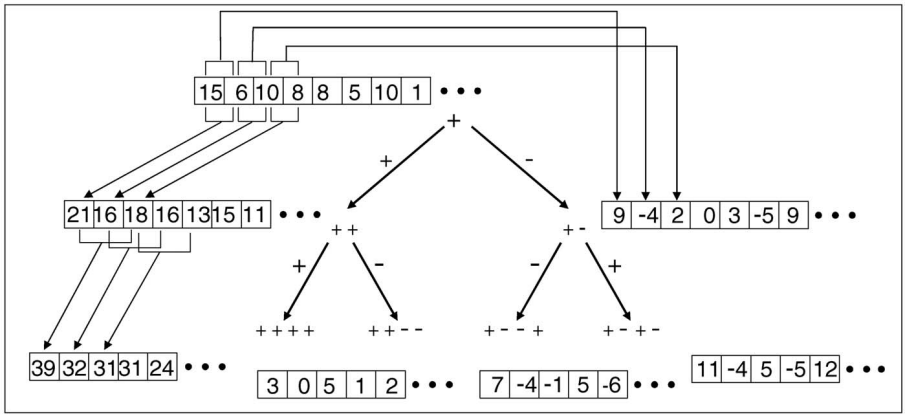



Fig. 6. An example of computing the Walsh-Hadamard tree for a 1D signal and windows of length 4.

node are calculated. Note that, due to the tree structure, every intermediate computation actually serves many windows. Descending from a node in the tree to its child requires a single operation per window, thus, at most $\log_2(k)$ additions/subtractions per signal window per projection are required.

- **Projecting all windows onto a set of consecutive projection vectors.** This corresponds to evaluating *all* entries in a consecutive set of leaf nodes of the tree. This extends the previously described type of projection. However, in the top-down descent of the tree, many nodes are common to several branches. For example, the first two projection vectors have $\log_2(k)$ nodes common to both branches. Thus, given the branch associated with the computation of the first projection vector, only one additional operation per entry is required for the second projection vector. In general, when computing the signal values at sequentially ordered leaf nodes, the Walsh-Hadamard tree is traversed in preorder. The projection of all windows onto the first $l$ projection vectors requires $m$ operations per signal window, where $m$ is the number of nodes preceding the $l$ leaf in the preorder tree traversal. Thus, projecting all windows onto *all* projection vectors requires only $2k-2$ operations per window, which is the number of edges in the full tree. Note that this requires an average of only two operations per window per projection vector, independent of the window size!
- **Projecting a single window onto a single projection vector.** This corresponds to evaluating a single entry in one of the leaves of the tree. To compute this value, a single branch of the tree must be traversed —the branch from the root to the leaf associated with the projection vector. However, since not all values of all the nodes in the branch are required for computation, the projection is calculated more efficiently by recursively ascending the tree bottom-up, from the leaf to the root node, and calculating only those entries that are missing yet are required for the computation. To compute an entry at level $i$, two entries are needed at level $i-1$, therefore, a total of at most $k-1$ operations (additions/subtractions) are required for this projection.

## 5 Using The Projection Scheme for Pattern Matching

The proposed pattern matching approach uses the Walsh-Hadamard tree structure and follows the projection scheme described above. As discussed in Section 2, a lower bound on the distance between each window and the pattern can be estimated from the projections. Thus, the complexity and runtime of the pattern matching process can be significantly

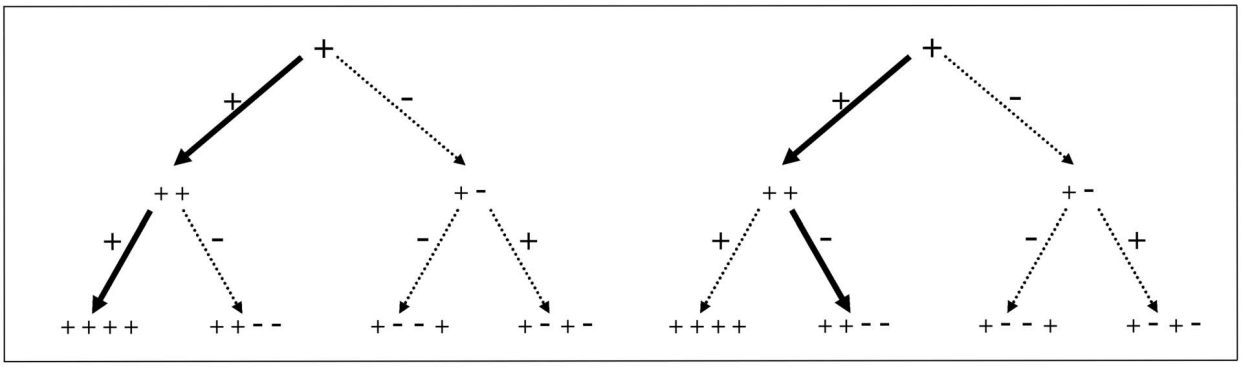



Fig. 7. Pattern matching scheme using the Walsh-Hadamard tree of order 4. (a) The initial step computes the first branch of the image associated with the first Walsh-Hadamard kernel (solid lines). (b) Computing the branch associated with the second kernel exploits the fact that several of the branch nodes have already been computed.

reduced by rejecting windows with lower bounds exceeding a given threshold value. Rejection of windows implies that many signal values at the tree leaves need not be computed.

In the pattern matching process, projections are performed onto vectors in the order determined by sequentially traversing the leaves. At all times, only a *single* branch of the Walsh-hadamard tree is maintained. This branch includes all nodes from the root to the current leaf. The algorithm proceeds as follows:

**Pattern Matching Algorithm**.

Assume the sought pattern $\mathbf{p}$ is projected (in a preprocessing stage) onto the set of $k$ Walsh-Hadamard basis vectors $\{\mathbf{u}_i\}_{i=1}^{k}$, resulting in $k$ values: $\hat{p}_i = \mathbf{u}_i^T\mathbf{p}$.

1. The first branch of the tree is computed, obtaining the projection of all signal windows $\{\mathbf{w}_x\}$ of length $k$ onto the first Walsh-Hadamard projection vector $\mathbf{u}_1$: $\hat{w}_{x,1} = \mathbf{u}_1^T\mathbf{w}_x$ (Fig. 7a).
2. This projection generates a lower bound on the true distance between each window $\mathbf{w}_x$ and the pattern $LB_1(x) = (\hat{w}_{x,1} - \hat{p}_1)^2$. All windows whose $LB$ value is greater than the given threshold are rejected.
3. Image windows that have not been rejected are projected onto the second vector $\mathbf{u}_2$: $\hat{w}_{x,2} = \mathbf{u}_2^T\mathbf{w}_x$. This is performed by replacing the maintained tree-branch associated with the first projection with the branch associated with the second projection (Fig. 7b). This produces updated lower bounds: $LB_2(x) = LB_1(x) + (\hat{w}_{x,2} - \hat{p}_2)^2$.
4. Steps 2 and 3 are repeated similarly for subsequent projection vectors for those windows that have not been rejected.
5. The process terminates after all $k$ kernels have been processed or until the number of nonrejected image windows reaches a predefined value.

During the matching process, it is possible to compute the projection values in two manners. The top-down approach calculates the projections of all windows. This method is preferable in the initial stages, when the number of rejected windows is small. However, at later stages, when the number of rejected windows is sufficiently large, it is preferable to compute the projection on individual windows, i.e., in a bottom-up manner (see Section 4). In practice, when only a very few windows remain, it is preferable to calculate the Euclidean distance directly for these windows. A rigorous analysis on the conditions for transitioning between the different forms of projections can be found in Section 10.

The complexity of the pattern matching process is calculated as follows: Initially (Step 1), $\log_2(k)$ operations per window are required to calculate the projection of all windows onto the first projection vector.[1] In the following stages of the process (repeated Steps 2-3), every projection requires only $l$ operations per window, where $l$ is the number of nodes that differ between the current branch and the previous branch of the tree. In practice, after very few projections, most of the signal windows are rejected and the bottom-up scheme is applied for the remaining windows. The number of operations per window beyond the first stage is shown experimentally to be negligible (Section 7).

The efficiency of computation is due to three main factors:

1. Using the recursive structure of the Walsh-Hadamard Tree, calculations applied to one window are exploited in neighboring windows.
2. Using the structure of the Walsh-Hadamard Tree, calculations applied to one projection vector are exploited in the subsequent vectors.
3. As discussed above, the first few Walsh-Hadamard vectors capture a large portion of the distance between pattern and window. Thus, in practice, after a few projections, the lower-bound on the distance is tight and enables most image windows to be rejected from further consideration.

In terms of memory demand, the proposed scheme requires more memory than the traditional approaches.

1. Note that the projection of all image windows onto the first vector can be performed directly using only two operations per window using a sliding sum. However, in the approach suggested here, intermediate computations are maintained that expedite computation of successive projections. Additionally, these intermediate computations allow flexibility and generalization of the process to other sets of projection kernels (see Section 12).

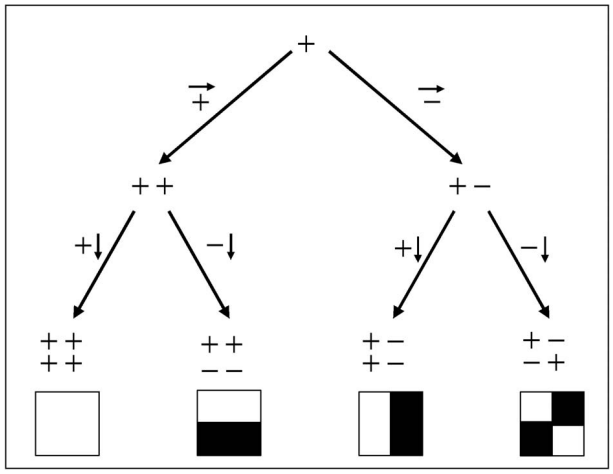


Fig. 8. The tree-scheme for computing projections of all 2D image windows onto all $2 \times 2$ Walsh-Hadamard kernels.

Traditional approaches, performing in-signal computations, require memory on the order of $n$ (signal size). The proposed approach maintains a branch of the Walsh-Hadamard tree at all times, requiring memory of size $n \log k$. However, considering the fact that floating memory is required for the naive approach and integer memory for the new approach and considering the typical scenario where $k$ (pattern size) is relatively small compared to $n$ (signal size), we find that this increase in memory is acceptable.

## 6 Pattern Matching in 2D

The pattern matching process described above for 1D signal easily extends to 2D images (as well as to higher dimensional data). Searching for a $k \times k$ pattern in a 2D image of size $n \times n$, each window of the image is projected onto a set of 2D Walsh-Hadamard kernels[2] using a Walsh-Hadamard tree. However, for the 2D case, the tree depth is $2\log_2(k)$ rather than $\log_2(k)$ and there are $k^2$ leaf nodes. Fig. 8 depicts such a tree for the Walsh-Hadamard transform of order $2 \times 2$. Each node represents an $n \times n$ image of intermediate values used in the computation. The root node represents the original $n \times n$ image. The $i$th leaf node represents an $n \times n$ image containing the projection values of all image windows onto the $i$th Walsh-Hadamard kernel. The operations performed at a given tree level are represented by the symbols along the edge. As in the 1D case, the operations performed are additions and subtractions of pixels at a distance $\Delta$ from each other, however, in the 2D case, a distinction is made between operations performed on the rows of the image and on the columns of the image. This is designated by the arrows in the symbols. Thus, the symbol $+\downarrow$ on an edge connecting nodes at level $i$ and level $i+1$ denotes the computation performed on the image at level $i$ in which, to every pixel $(x, y)$ of the image, the value of pixel $(x, y+\Delta)$ is added, where now $\Delta = 2^{\lfloor i/2 \rfloor}$. Descending from a node in the tree to its child image requires a single operation per pixel, thus, for every kernel, $2\log_2(k)$ operations (additions/subtraction) per pixel are required.

The order of projection kernels is crucial to the efficiency of the matching process. The projection used in the tree structure is that of diadically increasing frequency of the kernels (as shown in Fig. 3). The order of kernels is determined by the operations ($+$ or $-$) assigned to each edge. The diadically increasing order is obtained following the pseudocode given in Appendix D.

The Walsh-Hadamard tree can also be extended to deal with higher dimensional data (e.g., for spatio-temporal pattern matching in video sequences). The extension is straightforward and computations can be performed in a manner similar to the one and two-dimensional cases.

## 7 Experimental Results

The proposed scheme was tested on real images and patterns. The results show that the suggested approach reduces run-time by almost two orders of magnitude compared to the naive and FFT approaches.

To illustrate the performance of the suggested approach, we ran extensive experiments of Pattern Matching. As an example, Fig. 9 shows an original $256 \times 256$ image along with a $16 \times 16$ pattern that is to be matched. Figs. 10a, 10b, and 10c show the results of one run of the pattern matching scheme. According to the suggested method, the projection of all $65,536$ windows of the image onto the first Walsh-Hadamard kernel were calculated and all windows with projection values above a given threshold were rejected. In this experiment, the threshold was set to 168. After the first projection, only 602 candidate windows remained (Fig. 10a), i.e., only $0.92$ percent of the original windows. Following the second projection, only eight candidate windows remained (Fig. 10b) and, following the third projection, a single window containing the sought pattern remained (Fig. 10c).

The performance shown in this example is typical and is attained over many such examples. Fig. 11a shows the percentage of image windows remaining after each projection for images of size $256 \times 256$ and patterns of size $16 \times 16$. The results are the average over 1,000 image-pattern pairs (100 images, 10 patterns each). Using these results, an estimate of the average number of operations per pixel can be calculated. Fig. 11b displays the accumulated number of operations (additions/subtractions) versus the number of projections. The average number of operations per pixel for the whole process is $8.0085$, which is slightly over $2\log_2(16)$, as expected (see Section 6).

Runtime comparison was performed between the naive approach, the Fourier approach, and the suggested method using a $1K \times 1K$ image and patterns of size $16 \times 16$, $32 \times 32$, and $64 \times 64$. The experiments were performed on a PIII processor, 1.8 GHz. The average number of operations and run times are summarized in Table 1 in Section 1. It can be seen that the suggested approach is advantageous over the traditional approaches, especially for small pattern size. For larger pattern sizes, runtime increases, but still maintains a speed up of orders of magnitude over the two other methods.

2. The notion "kernels" in 2D replaces the notion "vectors" in 1D.

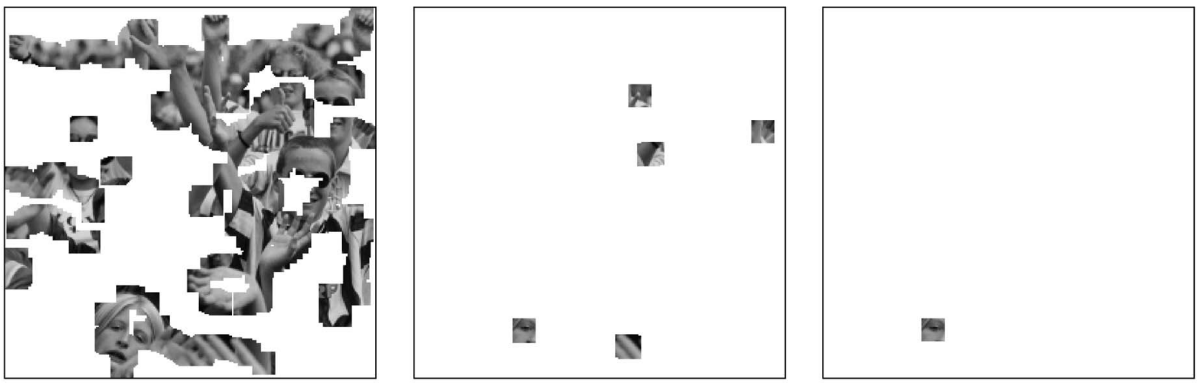
(a) (b)

Fig. 9. Inputs for the pattern matching example. (a) Original $256 \times 256$ image. (b) $16 \times 16$ pattern shown at large scale.

## 8 Additional Advantages

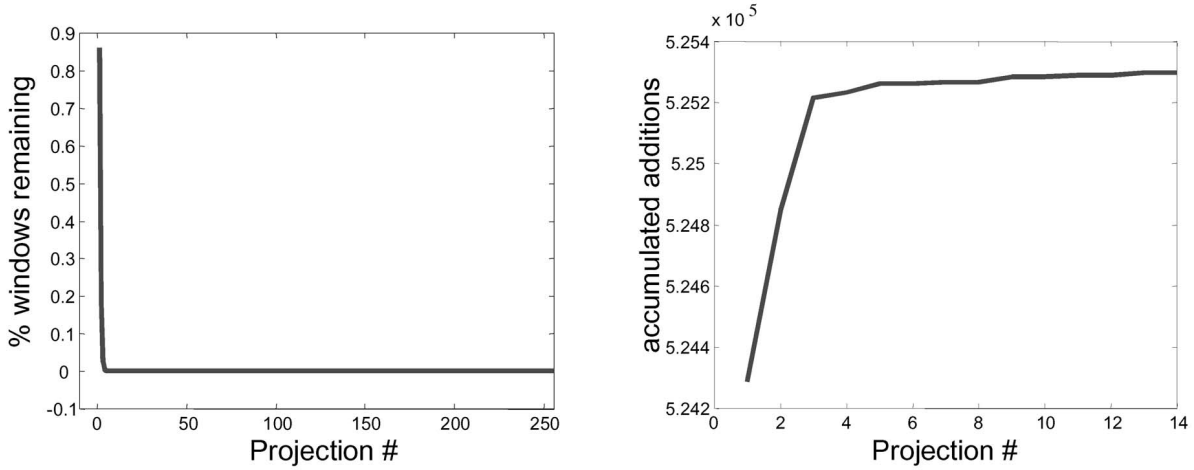
### 8.1 Varying Illumination

The Walsh-Hadamard tree has the additional advantage that it easily enables pattern matching that disregards the DC values. This can be used to compensate for illumination variations. The DC value of all windows is given by their projection onto the first Walsh-Hadamard kernel. In order to perform DC invariant pattern matching, the first branch of the Walsh-Hadamard tree may be skipped and the process initialized by computing the projections onto the second kernel. It is also possible to skip a number of tree branches if low frequency content of the pattern is irrelevant to the matching. The rest of the process continues as before. As an example, in Fig. 12a, an illumination gradient was added to an image. The illumination gradient produces a large variance in the appearance of a pattern in the image (Fig. 12c bottom). Using the original approach, the patterns required a very high threshold for detection and resulted in many false alarms as well. Performing DC invariant pattern



(a) (b) (c)

Fig. 10. Three steps in the pattern matching process. (a) Following the first projection, only 602 candidate windows remain ($0.92$ percent of the windows). (b) Following the second projection, only eight candidate windows remain. (c) After the third projection, a single window remains, corresponding to the correct pattern location.




(a) (b)

Fig. 11. (a) Percentage of image windows remaining after each projection. (b) Number of accumulated operations (additions/subtractions) versus the number of projections. The average number of operations per pixel is $8.0085$, which is slightly over $2log_2(16)$. Results are averaged over 1,000 image-pattern pairs. Images were of size $256 \times 256$ and patterns of size $16 \times 16$.

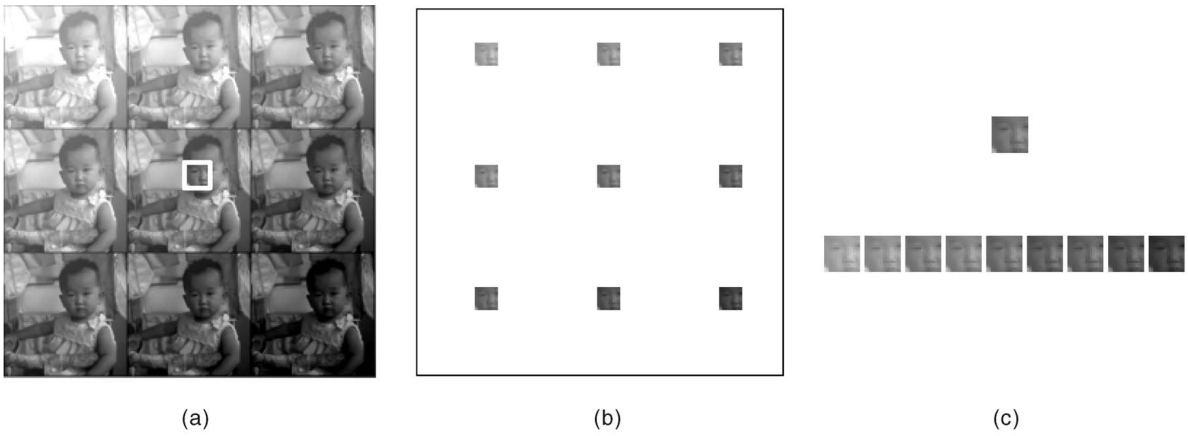



Fig. 12. (a) Original $256 \times 256$ image with illumination gradient. Pattern location is marked. (b) All nine patterns of varying illuminations were found. Four projections were required, using $8.0445$ operations per pixel. (c) The sought pattern (top) and examples of the pattern as appearing in the image (bottom).

matching, the pattern was detected using a very small threshold and required $8.0445$ operations per pixel.

### 8.2 Multiscale Pattern Matching

One of the characteristics of the Walsh-Hadamard tree is its inherent recursive nature. The subtrees from root to any level of the tree are themselves Walsh-Hadamard trees for smaller sized image windows. Within the Walsh-Hadamard tree containing $2log_2k + 1$ levels, developed for a window of size $k \times k$, the first $2log_2k - 2$ levels of the tree can be used to match patterns of size $\frac{k}{2} \times \frac{k}{2}$. Thus, the scheme presented in this paper can easily allow multiscale pattern matching with patterns scaled by powers of 2. This can be performed at almost no extra cost since the appropriate Walsh-Hadamard kernels for these scaled patterns are already given at the intermediate tree nodes. As an example, consider the recursive image of Fig. 13a. A pattern, shown in Fig. 13b, is sought within the image at different scales: $32 \times 32$, $16 \times 16$, and $8 \times 8$. A single Walsh-Hadamard tree is calculated. The leaves of the tree contain the projection values of all $32 \times 32$ windows of the image. The nodes at level $8$ contain the projections of all $16 \times 16$ image windows and level $6$ contains the projection values of all $8 \times 8$ windows. Multiscale matching is performed concurrently at all scales. Along a single Walsh-Hadamard tree branch, the nodes at levels $10$, $8$, and $6$ are considered and separate lower bounds are calculated accordingly. Note that, for every projection kernel at scale $8 \times 8$, four projection kernels at scale $16 \times 16$ and $16$ projections at scale $32 \times 32$ can be treated. Fig. 14 shows several steps in the multiscale matching process. Following the first projection, 314, 486, and 1,079 windows remain (0.48 percent, 0.74 percent, and 1.65 percent of all windows) at scales $32 \times 32$, $16 \times 16$, and $8 \times 8$, respectively. The $32 \times 32$ patterns required four projections, while the $16 \times 16$ and $8 \times 8$ required three projections each, at the appropriate scale.

## 9 Robustness

Performance of the pattern matching process, in terms of false alarms and miss detection rates, is dictated by the distance measure used, namely, the Euclidean distance. A discussion of the merits of the Euclidean distance is outside the scope of this paper, however, matching within a noisy environment may affect performance in terms of runtime. In the following, we show the runtime robustness of our method under nonexact pattern matching.

Appearances of a pattern in an image may vary due to noise, quantization, digitization, and transformation errors. In the following experiment, various levels of Gaussian noise were added to images. Figs. 15a and 15b show an original image and it's noisy version with Gaussian noise level of variance 35. Figs. 15c and 15d show an example of a pattern randomly selected from the original image and its associated appearance within the noisy image. Under these conditions, a higher threshold for rejection is required, which implies that fewer image windows are rejected after each projection and that the overall number of required projections increases. Fig. 16a displays this behavior where the percentage of image windows remaining after every projection is shown for various noise levels ranging from $0$ to $35$. Values are averaged over $100$ image-pattern pairs. To simulate a realistic scenario, only "interesting" patterns


Fig. 13. (a) Original $256 \times 256$ image with multiscale patterns. (b) A pattern to be sought within the image at scales: $32 \times 32$, $16 \times 16$, and $8 \times 8$.

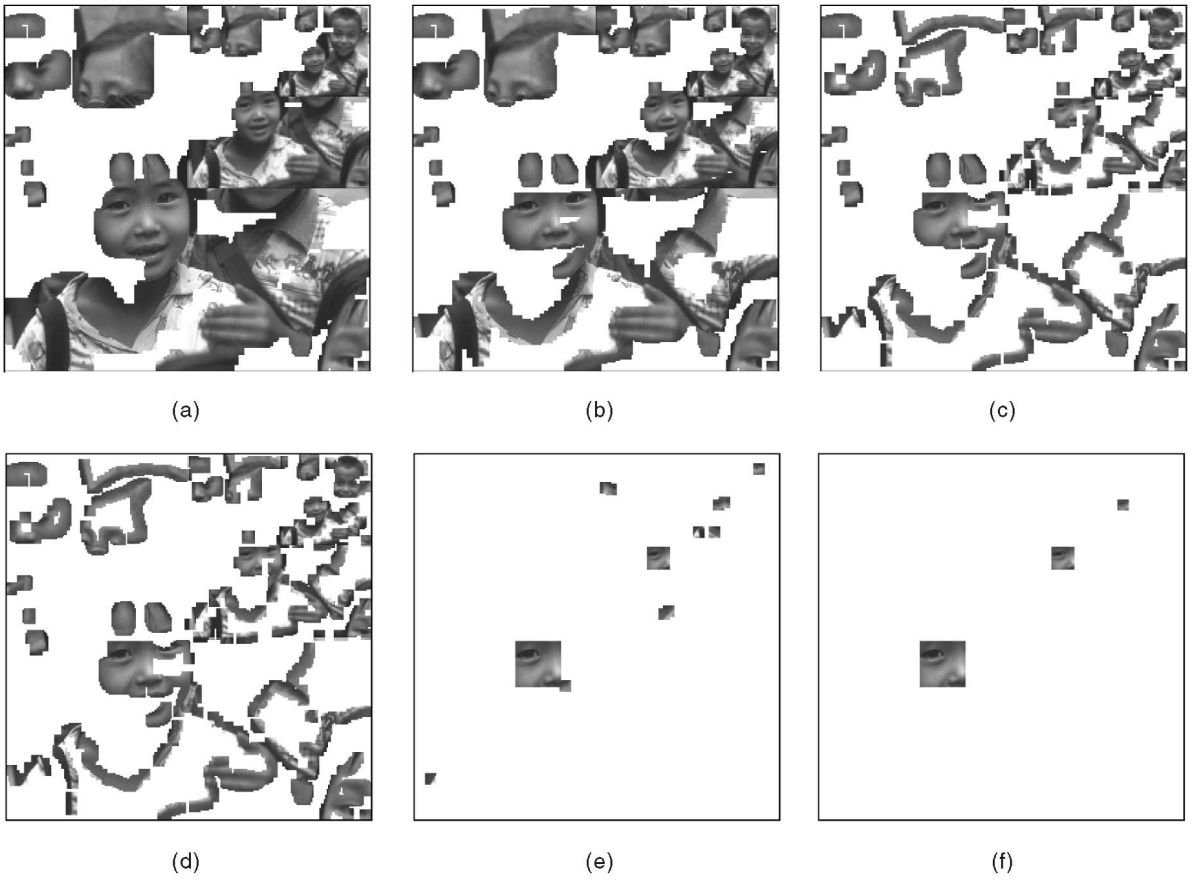



Fig. 14. Several steps in the multiscale pattern matching process. (a) Following the first projection, 314, 486, and 1,079 windows remain at scales $32 \times 32$, $16 \times 16$, and $8 \times 8$, respectively. (b)-(f) The scaled patterns required four, three, and three projections at the appropriate scale, respectively, however, a single Walsh-Hadamard tree was used and projection values for the various sized windows were found within common subtrees.

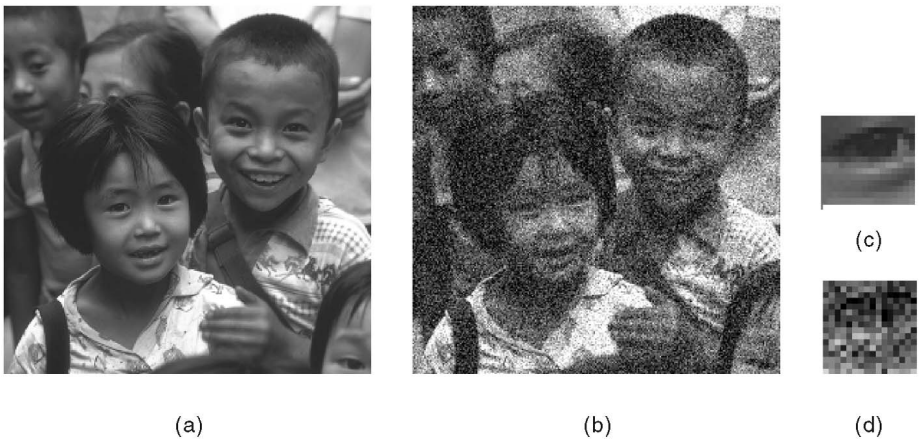



Fig. 15. (a) Original $256 \times 256$ image. (b) Very noisy version of original with Gaussian noise of variance 35 (see text). (c) Example of pattern (scaled). (d) The pattern as appearing in the noisy image (scaled).

were chosen randomly in the image (windows that responded strongly to the Harris Corner Detector [16]). The matching process was terminated when less than $0.25$ percent of the windows remained. The threshold used was determined by the noise variance and was set to be proportional to the noise level. As the noise level increases, fewer windows are rejected after each projection, however, the sharp decreasing profile for the various noise levels is similar. Although the number of required projections increases with noise level, efficiency of the pattern matching process is maintained, as shown in Fig. 16b. The figure shows the average number of operations per pixel required at every noise level. As can be seen, in this experimental scenario, there is an increase of $0.78$ percent in runtime for extremely noisy images, which is negligible. This is due to the fact that, although more projections onto kernels are required, these are performed on very few windows.

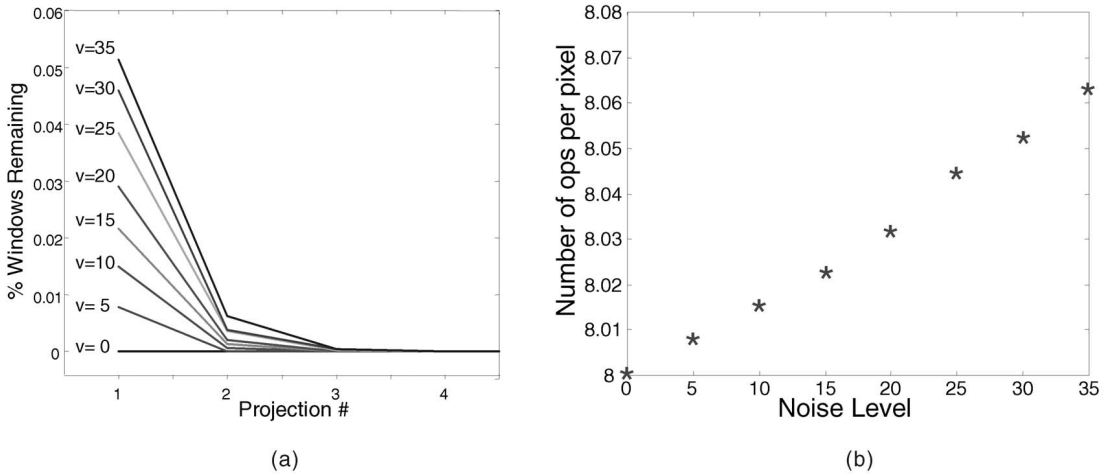



Fig. 16. For the image in Fig. 15a at various noise levels: (a) The percentage of image windows remaining after each projection. (b) The number of accumulated operations (additions/subtractions) versus the number of projections. The average number of operations per pixel at the highest noise level is $8.0623$. Data was collected over 100 pattern-image pairs.

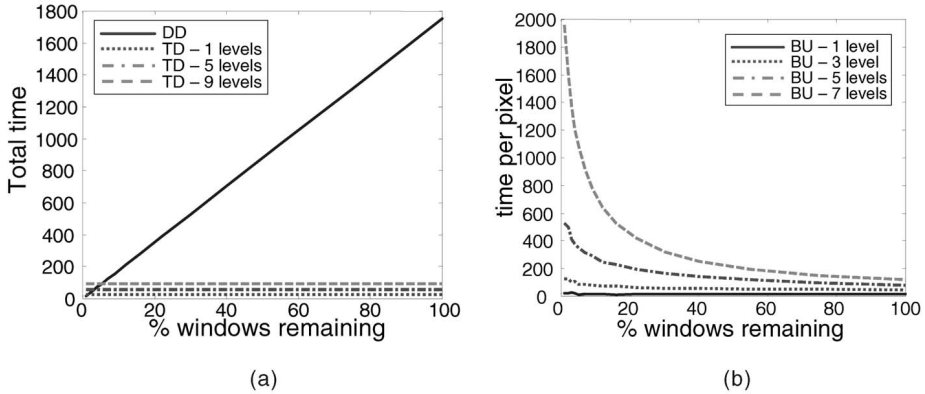



Fig. 17. (a) The total computation time for the DD and the TD approaches versus the percentage of remaining windows. The time required for TD is dependent on the number of tree levels that must be explored. (b) The average computation time per pixel for the BU approach versus the percentage of remaining of windows.

## 10 Computational Aspects

As described in Section 4, the pattern matching process involves three possible methods for evaluating the distance between image windows and pattern. The computational advantage of each method depends on the number of nonrejected windows remaining and on the position of the current projection kernel within the Walsh-Hadamard Tree.

The first method is the *direct distance* calculation (DD), which evaluates the distance between the pattern and a given window using the naive Euclidean distance. This method has the advantage that no further calculations are required since the distance is computed exactly. The DD approach is preferable when very few nonrejected windows remain in the image.

At the other end of the scale is the *top-down* approach, (TD) where a full branch of the Walsh-Hadamard tree is explored from the root down to the current leaf. In practice, the current branch needs to be explored only from a node level that has not been explored during previous projection calculations. The more levels that must be explored, the more expensive this step. Since projection values are calculated for all windows, this approach is preferable in the earlier stages of the process, when a large portion of the windows still remain to be matched. Fig. 17a shows the computational time (on a PIII processor, 1.8 GHz) for the DD and the TD approaches using a $1K \times 1K$ image as a function of the percentage of remaining windows. The time required for TD is given for various number of tree levels that must be explored. From the graph, it can be seen that, in the early stages of the process, when the percentage of remaining windows is greater than 4 percent, it is preferable to use the TD approach and, in subsequent stages, it might be preferable to use the DD approach, depending on the number of levels that must be explored.

The third approach is the *bottom-up* method (BU), which computes the distance between windows and a pattern by exploring only the necessary values within the tree for each nonrejected window. Using the BU method, the more windows that must be evaluated, the more efficient the computation becomes, since intermediate values are common to neighboring windows. Fig. 17b shows this behavior, where average time per pixel rapidly decreases as the percentage of remaining windows increases. Thus, the BU method is advantageous at intermediate stages when a large portion of the windows has already been rejected so that the

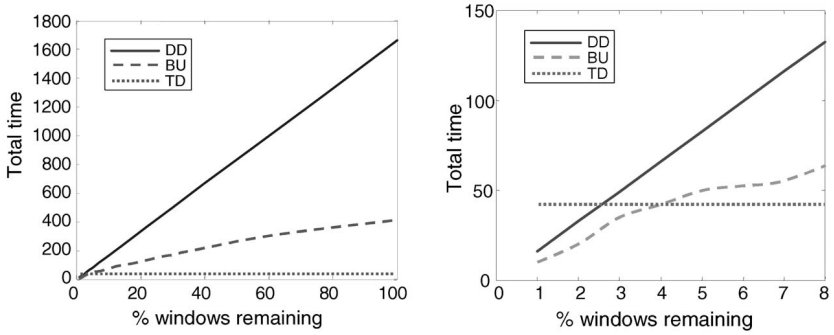



Fig. 18. (a) The total computation time for the TD, BU, and DD approaches versus the percentage of remaining windows when three levels of the branch must be explored. (b) Scaled version around the origin.

TD approach is not beneficial and yet more than a few windows still remain so that the DD approach is still expensive. Fig. 18 shows this behavior for the case where three levels of the branch must be explored. In this case, it is preferable to apply the TD method when the remaining percentage of windows is greater than 4 percent. When fewer than 4 percent remain, the BU is superior. Such a comparison can be designed for different number of levels. Table 2 summarizes these comparisons by stating which of the three methods should be used for a different number of levels when a $64 \times 64$ pattern is sought in a $1K \times 1K$ image. The table depicts the percentage of remaining windows below which a method should be used (assuming TD is always used initially with 100 percent of the windows). It is interesting to note that, when traversal of a few levels is required (up to four levels), it is preferable to use the TD method during the early stages and the BU method at later stages. If more than five levels are required, it is preferable to transfer directly from TD to DD. Only when five levels are required is it advantageous to use all three methods depending on the percentage of remaining windows.

TABLE 2
Pattern Matching Using Three Types of Distance Computations (TD, BU, and DD—See Text)

| # Levels | Use BU | Use DD |
|---|---|---|
| 1 | 10% | never |
| 2 | 7.5% | never |
| 3 | 5% | never |
| 4 | 1.8% | never |
| 5 | 1.2% | 0.6% |
| 6 | never | 1.0% |
| 7 | never | 1.2% |
| 8 | never | 1.4% |
| 9 | never | 1.6% |
| 10 | never | 1.8% |
| 11 | never | 2.0% |
| 12 | never | 2.2% |

*The preferable method is dependent on the percentage of remaining windows and on the number of tree levels that must be traversed to complete the computation. The percentage of remaining windows below which a method should be used is given for various numbers of levels (assuming TD is used initially with 100 percent of the windows). The values were determined experimentally for a $64 \times 64$ pattern sought in a $1k \times 1k$ image.*

## 11 Kernel Ordering

The advantage of rejection schemes as described in this paper relies strongly on the assumption that very few valid hypotheses remain after very few rejection steps. In the context of this paper, the projection kernels should have a high probability of being parallel to the difference between the pattern and image windows. This ensures that, after a few projections, the lower bound is sufficiently tight. The diadic ordering of kernels induced by the Walsh-Hadamard tree determines the increase in spatial frequency. This ordering of kernels is optimal when no a priori information is given on the pattern or on the images in which the pattern is sought. In this case, an inherent assumption is made that the pattern and images originate from *natural* images. This can be seen from the following derivations:

Let $\mathbf{p}$ and $\mathbf{w}$ be random variables representing a pattern and image window. If no a priori information is given, these variables are assumed to be associated with the probability distribution of natural images. Let $\mathbf{u}$ be a projection kernel. In the context of pattern matching, the optimal kernel to be used is that which maximizes:

$$\begin{aligned}\mathbf{u} &= argmax_{\mathbf{u}} E\{\|\mathbf{u}^T(\mathbf{p}-\mathbf{w})\|^2\} \\ &= argmax_{\mathbf{u}} E\{\mathbf{u}^T(\mathbf{p}-\mathbf{w})(\mathbf{p}-\mathbf{w})^T\mathbf{u}\},\end{aligned} \tag{5}$$

where $E$ is the expectation value over the distribution of $\mathbf{p}$ and $\mathbf{w}$. The solution of (5) is the eigenvector associated with the largest eigenvalue of the matrix $\mathbf{M}$

$$\mathbf{M} = E\{(\mathbf{p}-\mathbf{w})(\mathbf{p}-\mathbf{w})^T\}.$$

Since the distributions of $\mathbf{p}$ and $\mathbf{w}$ are independent, this can easily be shown to equal:

$$\begin{aligned}\mathbf{M} =& E\{(\mathbf{p}-E\{\mathbf{w}\})(\mathbf{p}-E\{\mathbf{w}\})^T\} \\ &+ E\{(\mathbf{w}-E\{\mathbf{w}\})(\mathbf{w}-E\{\mathbf{w}\})^T\}.\end{aligned} \tag{6}$$

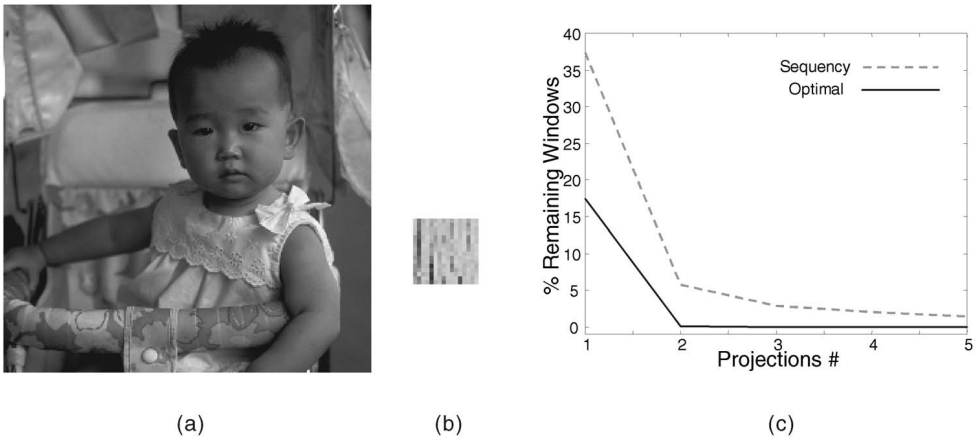



Fig. 19. (a) Original $256 \times 256$ "natural" image. (b) "Texture" pattern (scaled). (c) Percentage of remaining windows after each projection for the sequency order (dashed) and for the optimal order of kernels (solid).

When probability distribution of natural images is assumed for both $\mathbf{p}$ and $\mathbf{w}$, then $\mathbf{M}$ equals twice the covariance matrix of natural images. In this case, it is well-known that the eigenvectors of $\mathbf{M}$ associated with decreasing eigenvalues correspond to a sinusoidal basis ordered in increasing frequency [30], [29]. In terms of the Walsh-Hadamard kernels, this translates to the order of increasing sequency which is known to be optimal for array compactization of natural images [20].

However, if $\mathbf{p}$ is a specific pattern known in advance or is known to originate from a given probability distribution (e.g., a texture image), the first term in (6) may differ from the second term. Dependent on the distribution of $\mathbf{p}$, the eigenvalues of the first term may override those of the second term and bring about a different optimal order of the Walsh-Hadamard kernels. For example, consider the image and "texture" pattern in Figs. 19a and 19b. When searching for the "texture" pattern within the "natural" image, one can show that the sequency order of projection kernels does not perform well. Rather, an order that gives priority to vertically oriented and midfrequency ranged kernels is optimal in this case. This is shown in Fig. 19c, where the percentage of remaining windows after each projection is given for the sequency order and when using the optimal order of kernels for this pattern-image pair.

Reordering of the projection kernels is possible with the Walsh-Hadamard scheme by exchanging branches within the tree. A more generalized approach to reordering projection kernels and computing projections efficiently can be found in [4]. A detailed discussion and analysis of the ordering of kernels may be found there as well.

## 12 Discussion and Conclusion

This paper introduced a novel framework for pattern matching based on an efficient projection scheme. The proposed approach assumes the Euclidean distance is used for measuring similarity between pattern and image. Under this assumption, the suggested approach is advantageous over the existing traditional approaches first and foremost due to the reduction in time complexity of two orders of magnitude. The efficiency due to the projection scheme is further amplified by the fact that all computations of the transform coefficients can be performed using integer additions and subtractions.

The Walsh-Hadamard tree structure used in this paper provides additional advantages such as the ability to cope with varying illumination. The scheme also easily allows multiscale pattern matching, with patterns scaled by powers of two. This can be performed at almost no extra cost since the appropriate Walsh-Hadamard kernels for these scaled patterns are already given at the intermediate tree nodes. Extending this idea, intermediate nodes of the tree can be exploited as they represent projection values onto additional kernels. Optimizing search over *all* nodes of the tree can be performed, a process similar to a Best Basis search within Wavelet Packets [24].

The Walsh-Hadamard tree structure encompasses several other transform including the Haar and other wavelet packets. For example, the windowed Haar transform can be applied by traversing the first branch of the tree along with its immediate descendent nodes. In this paper, the Walsh-Hadamard kernels were chosen since there is then no need for normalization. This allows computations to remain as integer operations. The Haar kernels would require a different normalization factor for each kernel, requiring noninteger operations and an additional operation per pixel. However, the Haar kernels might be advantageous in some cases as they can be evaluated more efficiently in the bottom-up scheme (Section 10).

The technique explained in this paper was described in the pattern matching context, however, we emphasize that this is an example application. The method presented can be used to efficiently extract features from images. These can be directly used for classification applications, texture analysis, and image retrieval applications. The method can be further extended to deal with block-matching, where the window *most* similar to a given pattern is sought (rather than searching for *all* the matches—up to a given threshold). This can be implemented by continuously updating the threshold to equal the *actual* distance between the pattern and the window with the lowest bound found so far. Thus,

applications such as texture synthesis and block matching for motion estimation can take advantage of this technique.

Although the pattern matching approach described here was presented for the Euclidean norm, extensions to other norms are currently being developed. In principle, this can be done by applying the equivalence relation between norms such that the $L_2$ norm forms a lower and upper bound when multiplied by specific constants [15]. The method has already been extended to deal with normalized gray-scale correlation distance, which allows matching of patterns of any size by spatial masking and allows linear gray-scale deformations [37].

The proposed scheme requires more memory than the naive approaches. Naive approaches perform convolutions on the image, requiring the process to allocate and maintain memory size on the order of $n^2$ (the size of the image). The proposed approach maintains a branch of the Walsh-Hadamard tree at all times, requiring memory of size $2n^2 \log k$. However, considering the fact that floating memory is required for the naive approach and integer memory for the new approach and considering the typical scenario where $k$, the pattern size, is relatively small compared to the image size, we find that this increase in memory is acceptable. In a concurrent study [4], we use a novel technique that requires a memory size of only $2n^2$ and uses only two operations per pixel in order to calculate all window projections onto a kernel regardless of pattern (kernel) size and dimension. This new technique also allows more flexibility in the possible ordering of the projection kernels.

Finally, we emphasize that we do not aim at introducing a new classification technique or distance measure, rather, we assume a normed distance evaluation is used to determine whether a window is similar to the given pattern. Given that this distance measure is used, we showed a method that improves runtime and efficiency over other known techniques.

Software implementing the pattern matching scheme presented here is available and can be downloaded from: http://www.faculty.idc.ac.il/toky/Software/software.htm.

## Appendix A

## Distance Lower Bound for a Single Projection Vector

**Lemma 1.** *Given three vectors in $R^n$, $\mathbf{p}$, $\mathbf{w}$, and $\mathbf{u}$, the Euclidean distance between $\mathbf{p}$ and $\mathbf{w}$ satisfies the following:*

$$d_E(\mathbf{p},\mathbf{w}) \geq \|\mathbf{u}^T\mathbf{p} - \mathbf{u}^T\mathbf{w}\|/\|\mathbf{u}\|. \tag{7}$$

**Proof.** Define $\mathbf{d} = \mathbf{p} - \mathbf{w}$. Using Cauchy-Schwartz inequality for norms, it follows that:

$$\|\mathbf{u}\|\,\|\mathbf{d}\| \geq \|\mathbf{u}^T\mathbf{d}\|.$$

This implies:

$$d_E(\mathbf{p},\mathbf{w}) = \|\mathbf{p}-\mathbf{w}\| = \|\mathbf{d}\| \geq \frac{\|\mathbf{u}^T(\mathbf{p}-\mathbf{w})\|}{\|\mathbf{u}\|} = \frac{\|\mathbf{u}^T\mathbf{p} - \mathbf{u}^T\mathbf{w}\|}{\|\mathbf{u}\|}.$$

□

## Appendix B

## Distance Lower Bound for a Set of Projection Vectors

**Lemma 2.** *Given an unknown vector $\mathbf{d}$ in $R^n$ which satisfies the matrix equation:*

$$\mathbf{U}^T\mathbf{d} = \mathbf{b}, \tag{8}$$

*where $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_m]$ and $\mathbf{b} = (b_1 \cdots b_m)^T$ for some $m \leq n$. The Euclidean norm of $\mathbf{d}$ is bounded from below by:*

$$\|\mathbf{d}\|^2 \geq \mathbf{b}^T(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{b}.$$

**Proof.** Let the SVD decomposition of matrix $\mathbf{U}$ be: $\mathbf{U} = \mathbf{A}\mathbf{S}\mathbf{B}^T$, where $\mathbf{A}$ and $\mathbf{B}$ are $n \times m$ and $m \times m$ orthogonal matrices satisfying $\mathbf{A}^T\mathbf{A} = \mathbf{B}^T\mathbf{B} = I_m$ and $\mathbf{S}$ is an $m \times m$ diagonal matrix. The minimum norm solution for (8) is a vector $\hat{\mathbf{d}}$ satisfying [15]:

$$\hat{\mathbf{d}} = \mathbf{A}\mathbf{S}^{-1}\mathbf{B}^T\mathbf{b}.$$

This implies:

$$\hat{\mathbf{d}}^T\hat{\mathbf{d}} = \mathbf{b}^T\mathbf{B}\mathbf{S}^{-1}\mathbf{A}^T\mathbf{A}\mathbf{S}^{-1}\mathbf{B}^T\mathbf{b} = \mathbf{b}^T\mathbf{B}\mathbf{S}^{-2}\mathbf{B}^T\mathbf{b}.$$

Similarly, we have:

$$(\mathbf{U}^T\mathbf{U})^{-1} = (\mathbf{B}\mathbf{S}\mathbf{A}^T\mathbf{A}\mathbf{S}\mathbf{B}^T)^{-1} = (\mathbf{B}\mathbf{S}^2\mathbf{B}^T)^{-1} = \mathbf{B}\mathbf{S}^{-2}\mathbf{B}^T.$$

Since $\hat{\mathbf{d}}$ is the minimum norm solution, it follows that:

$$\|\mathbf{d}\|^2 \geq \|\hat{\mathbf{d}}\|^2 = \mathbf{b}^T\mathbf{B}\mathbf{S}^{-2}\mathbf{B}^T\mathbf{b} = \mathbf{b}^T(\mathbf{U}^T\mathbf{U})^{-1}\mathbf{b}.$$

□

## Appendix C

## Iterative Distance Lower Bound for a Set of Projection Vectors

The iterative scheme for calculating lower-bounds on the Euclidean distance is based on a sequence of projections onto vectors. At every step, a projection vector is applied to each image window and updated distance lower-bounds are calculated accordingly. The basic idea is to orthogonalize each projection vector in a manner similar to the Grahm-Schmidt process [35] so that only directions that are orthogonal to previously spanned directions are taken into account.

Assume that a set of $m$ *orthonormal* vectors $\{\mathbf{u}_1 \cdots \mathbf{u}_m\}$ have been applied so that

$$\mathbf{U}_m^T\mathbf{d} = \mathbf{b}_m,$$

where $\mathbf{U}_m = [\mathbf{u}_1 \cdots \mathbf{u}_m]$ and $\mathbf{b}_m$ is composed of $m$ projected differences $b_i = \mathbf{u}_i^T(\mathbf{p} - \mathbf{w})$. Since the $m$ projection vectors are orthonormal, the distance lower bound after $m$ projections can be calculated using (4):

$$LB_m^2(\mathbf{p},\mathbf{w}) = \mathbf{b}_m^T(\mathbf{U}_m^T\mathbf{U}_m)^{-1}\mathbf{b}_m = \mathbf{b}_m^T\mathbf{b}_m.$$

Given a new projection vector $\mathbf{v}$, along with its associated projected distance $\mathbf{v}^T\mathbf{d} = \alpha$, the component of $\mathbf{v}$ in the null space of $\mathbf{U}$ is calculated:

$$\mathbf{v}_N = \mathbf{v} - \mathbf{U}_m\mathbf{U}_m^T\mathbf{v}$$

so that $\mathbf{U}_m^T\mathbf{v}_N = 0$. Dividing $\mathbf{v}_N$ by its norm $\gamma = \|\mathbf{v}_N\|$ yields a normalized vector:

$$\hat{\mathbf{v}}_N = \frac{\mathbf{v}_N}{\gamma}.$$

Projecting $\mathbf{d}$ onto $\hat{\mathbf{v}}_N$ gives:

$$\hat{\mathbf{v}}_N^T\mathbf{d} = \frac{1}{\gamma}(\mathbf{v} - \mathbf{U}_m\mathbf{U}_m^T\mathbf{v})^T\mathbf{d} = \frac{1}{\gamma}(\alpha - \mathbf{v}^T\mathbf{U}_m\mathbf{b}_m) \doteq b_{m+1}.$$

Since $\hat{\mathbf{v}}_N$ is also orthogonal to $\mathbf{U}_m$, we can update matrix $\mathbf{U}_m$ to be: $\mathbf{U}_{m+1} = [\mathbf{U}_m \ \ \hat{\mathbf{v}}_N]$ and the vector $\mathbf{b}_m$ to be $\mathbf{b}_{m+1} = [\mathbf{b}_m^T \ \ b_{m+1}]^T$. Using (4) again, an updated lower-bound is calculated:

$$\begin{aligned} LB_{m+1}^2(\mathbf{p},\mathbf{w}) &= \mathbf{b}_{m+1}^T(\mathbf{U}_{m+1}^T\mathbf{U}_{m+1})^{-1}\mathbf{b}_{m+1} = \mathbf{b}_{m+1}^T\mathbf{b}_{m+1} \\ &= LB_m^2(\mathbf{p},\mathbf{w}) + \frac{1}{\gamma^2}(\alpha - \mathbf{v}^T\mathbf{U}_m\mathbf{b}_m)^2. \end{aligned}$$

Note that the term $\mathbf{v}^T\mathbf{U}_m$ is calculated only once at each projection step.

## APPENDIX D

## PSEUDOCODE FOR GENERATING THE WALSH-HADAMARD TREE

Following is pseudocode for generating the 1D Walsh-Hadamard tree described in Section 4. For convenience, nodes in the tree are indexed from 1 (root) to $2k-1$ along tree levels (breadth first order). Levels in tree are indexed from 0 (root) to $log_2 k - 1$, where $k$ is the window length.

```
node(1) = <original signal>

for nodeNum = 2 to 2k-1
  level = floor(log2(nodeNum))
  delta = 2^ level
  fatherNode = floor(nodeNum/2)
  op = even(nodeNum) * even(fatherNode)
  node(nodeNum) = node(fatherNode)
          + op * shift(node(fatherNode),delta)
end;
```

The function *even* returns $+1$ for even numbers and $-1$ for odd. To generate the 2D Walsh-Hadamard tree, replace fatherNode with grandfatherNode:

$$\text{grandfatherNode} = \text{floor}(\text{nodeNum}/4).$$

## ACKNOWLEDGMENTS

The authors would like to thank Yaron Ukrainitz for running the simulations and for his helpful suggestions for implementation.

## REFERENCES

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

[2] S. Baker and T. Kanade, "Hallucinating Faces," *Proc. Fourth Int'l Conf. Automatic Face and Gesture Recognition,* p. 83, Mar. 2000.

[3] S. Baker and S.K. Nayar, "Pattern Rejection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 544-549, 1996.

[4] G. Ben-Artzi, H. Hel-Or, and Y. Hel-Or, "Filtering with Gray-Code Kernels," *Proc. 17th Int'l Conf. Pattern Recognition,* pp. 556-559, Sept. 2004.

[5] J.L. Bentley, "Multidimensional Binary Search Trees Used for Associative Searching," *Comm. ACM,* vol. 18, no. 9, pp. 509-517, 1975.

[6] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery,* vol. 2, no. 2, pp. 121-167, 1998.

[7] F. Crow, "Summed-Area Tables for Texture Mapping," *Proc. SIGGRAPH,* vol. 18, no. 3, pp. 207-212, 1984.

[8] A. Efros and W.T. Freeman, "Image Quilting for Texture Synthesis and Transfer," *Proc. SIGGRAPH,* Aug. 2001.

[9] M. Elad, Y. Hel-Or, and R. Keshet, "Pattern Detection Using Maximal Rejection Classifier," *Proc. Int'l Workshop Visual Form,* pp. 28-30, May 2000.

[10] A.M. Eskicioglu and P.S. Fisher, "Image Quality Measures and Their Performance," *IEEE Trans. Comm.,* vol. 43, no. 12, pp. 2959-2965, 1995.

[11] A.W. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-Based Rendering Using Image-Based Priors," *Proc. Int'l Conf. Computer Vision,* 2003.

[12] F. Fleuret and D. Geman, "Graded Learning for Object Detection," *Proc. IEEE Workshop Statistical and Computational Theories of Vision,* pp. 544-549, 1999.

[13] 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.

[14] B. Girod, "Whats Wrong with Mean-Squared Error?" *Digital Images and Human Vision,* A.B. Watson, ed., chapter 15, pp. 207-220. MIT Press, 1993.

[15] G.H. Golub and C.F. Van Loan, *Matrix Computations.* Baltimore: John Hopkins Univ. Press, 1989.

[16] C.J. Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proc. Fourth Alvey Vision Conf.,* pp. 147-151, 1988.

[17] Y. Hel-Or and H. Hel-Or, "Real Time Pattern Matching Using Projection Kernels," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* pp. 1486-1493, Oct. 2003.

[18] X. Huo and J. Chen, "Building a Cascade Detector and Its Applications in Automatic Target Detection," *Applied Optics,* vol. 43, no. 2, pp. 293-303, 2004.

[19] D. Keren, M. Osadchy, and C. Gotsman, "Antifaces: A Novel, Fast Method for Image Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 23, no. 7, pp. 747-761, July 2001.

[20] H. Kitajima, "Energy Packing Efficiency of the Hadamard Transform," *IEEE Trans. Comm.,* pp. 1256-1258, 1976.

[21] M.H. Lee and M. Kaveh, "Fast Hadamard Transform Based on a Simple Matrix Factorization," *IEEE Trans. Acoustics, Speech, and Signal Processing,* vol. 34, no. 6, pp. 1666-1667, 1986.

[22] L. Liang, C. Liu, Y.Q. Xu, B. Guo, and H.Y. Shum, "Real-Time Texture Synthesis by Patch-Based Sampling," *ACM Trans. Graphics,* vol. 20, no. 3, pp. 127-150, July 2001.

[23] T. Luczak and W. Szpankowski, "A Suboptimal Lossy Data Compression Based on Approximate Pattern Matching," *IEEE Trans. Information Theory,* vol. 43, pp. 1439-1451, 1997.

[24] S. Mallat, *A Wavelet Tour of Signal Processing.* Academic Press, 1999.

[25] A.M. Mamlouk, J.T. Kim, E. Barth, and T. Martinetz, "One-Class Classification with Subgaussians," *DAGM 2003, Proc. 25th Pattern Recognition Symp.,* pp. 346-353, 2003.

[26] O.J. Murphy and S.M. Selkow, "The Efficiency of Using K-D Trees for Finding Nearest Neighbors in Discrete Space," *Information Processing Letters,* vol. 23, no. 4, pp. 215-218, 1986.

[27] A. Nealen and M. Alexa, "Fast and High Quality Overlap Repair for Patch-Based Texture Synthesis," *Proc. Computer Graphics Int'l,* 2004.

[28] C. Papageorgiou, M. Oren, and T. Poggio, "A General Framework for Object Detection," *Proc. Sixth Int'l Conf. Computer Vision,* pp. 555-562, Jan 1998.

[29] D.L. Ruderman, "Statistics of Natural Images," *Network: Computation in Neural Systems,* vol. 5, no. 4, pp. 517-548, 1994.

[30] D.L. Ruderman and W. Bialek, "Statistics of Natural Images: Scaling in the Woods," *Physical Rev. Letters,* vol. 73, no. 6, pp. 814-817, 1994.
[31] H. Samet, *Applications of Spatial Data Structures.* Reading, Mass.: Addison-Wesley, 1990.
[32] H. Samet, *The Design and Analysis of Spatial Data Structures.* Reading, Mass.: Addison-Wesley, 1990.
[33] S. Santini and R. Jain, "Similarity Measures," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 9, pp. 871-883, Sept. 1999.
[34] J.L. Shanks, "Computation of the Fast Walsh-Fourier Transform," *IEEE Trans. Computers,* vol. 18, pp. 457-459, 1969.
[35] G. Strang, *Linear Algebra and Its Applications.* Orlando, Fla.: Haecourt Brace Jovanovic, 1988.
[36] D. Sundararajan and M.O. Ahmad, "Fast Computation of the Discrete Walsh and Hadamard Transforms," *IEEE Trans. Information Processing,* vol. 7, no. 6, pp. 898-904, 1998.
[37] Y. Ukrainitz, Y. Hel-Or, and H. Hel-Or, "Real-Time Normalized Gray-Scale Correlation Using Projection Kernels," submitted to *Proc. Int'l Conf. Computer Vision,* 2005.
[38] P. Viola and M. Jones, "Robust Real-Time Object Detection," *Proc. Int'l Conf. Computer Vision Workshop Statistical and Computation Theories of Vision,* July 2001.
[39] T. Weissman, E. Ordentlich, G. Seroussi, S. Verdu, and M.J. Weinberger, "Universal Discrete Denoising: Known Channel," Technical Report HPL-2003-29, HP Labs, Feb. 2003.
[40] J. Kane, W.K. Pratt, and H.C. Andrews, "Hadamard Transform Image Coding," *Proc. IEEE,* vol. 57, no. 1, pp. 58-68, 1969.
[41] J. Wu, J.M. Rehg, and M.D. Mullin, "Learning a Rare Event Detection Cascade by Direct Feature Selection," *Advances in Neural Information Processing Systems 16,* S. Thrun, L. Saul, and B. Schölkopf, eds. Cambridge, Mass.: MIT Press, 2004.

**Yacov Hel-Or** received the BSc degree in physics and computer science from Bar-Ilan University, Israel, in 1985 and the PhD degree in computer science from the Hebrew University, Jerusalem, Israel, in 1993. During 1993-1994, he was a postdoctoral fellow in the Department of Applied Mathematics and Computer Science at The Weizmann Institute of Science, Rehovot, Israel. From 1994-1996, he was with the NASA Ames Research Center, Moffet Field, California, as a National Research Council associate. During 1996-1998, he was a researcher at Hewlett Packard Labs, Israel. He has been a faculty member at the Interdisciplinary Center, Israel, since 1998. His recent interests include computer vision, image processing, robotics, and computer graphics. He is a member of the IEEE.

**Hagit Hel-Or** received the PhD degree in computer science in 1994 from the Hebrew University, Jerusalem, Israel. She was a postdoctoral fellow for two years in the Vision Group in the Department of Psychology at Stanford University. She was with the Department of Mathematics and Computer Science at Bar-Ilan University, Ramat-Gan, Israel, for two years. Currently, she is a faculty member in the Department of Computer Science at Haifa University, Haifa, Israel. Her research area is in the field of imaging science and technologies and includes color vision, image processing, computational, and human vision. She is a member of the IEEE.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.