Real Time Pattern Matching Using Projection Kernels

Yacov Hel-Or

School of Computer Science
The Interdisciplinary Center
Herzeliya, Israel
toky@idc.ac.il

Hagit Hel-Or

Dept of Computer Science
University of Haifa
Haifa, 31905, Israel
hagit@cs.haifa.ac.il

September 3, 2002

Abstract

A novel approach to pattern matching is presented, which reduces time complexity by two orders of magnitude compared to naive approaches enabling real-time performance. The suggested approach uses an efficient projection scheme which allows the distance between a pattern and a window to be bounded efficiently from below. The projection scheme is combined with a rejection framework which efficiently determines the windows that do not match the pattern. Thus, using relatively few projections, a large number of windows are rejected from further computations. Experiments show that the approach is efficient even under very noisy conditions.

Keywords: Pattern Matching, Template Matching, Walsh-Hadamard, Feature Detection, Rejection Scheme.

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 typically 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.

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 video pattern matching, the similarity is calculated for each pixel in the 3D spatio-temporal stack, where the distance is calculated between two 3D data cubes (See Figure 1).

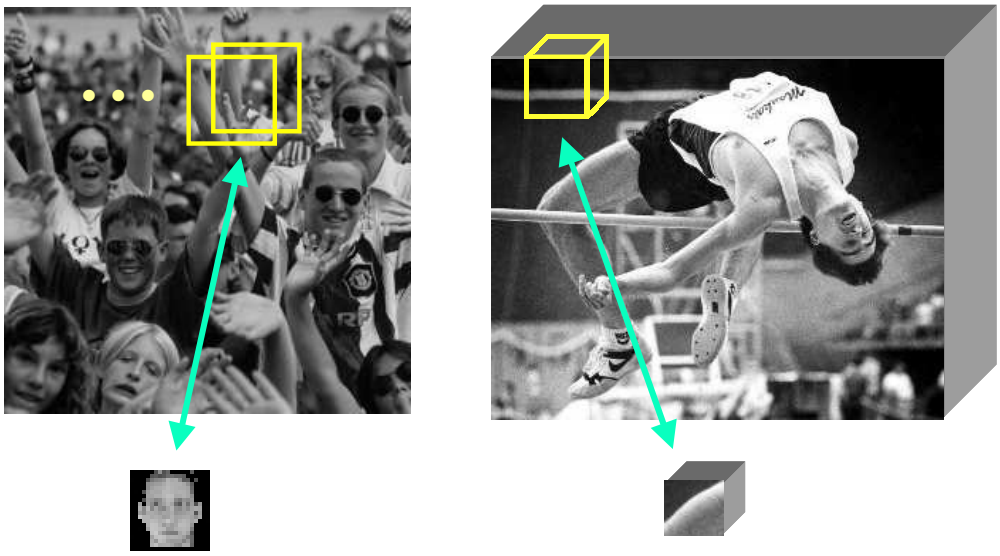


Figure 1: Pattern matching in 2D and in 3D: For each pixel location, a local neighborhood is compared with the given pattern.

There are various measures of similarity that may be used in this scheme and it is arguable as to which measure is preferable (see [19, 17, 21, 2] for reviews and discussions). One of the more common approaches is to assume that the similarity measure is a norm distance, specifically, the Euclidean distance. Given two images $I_1(x, y)$ and $I_2(x, y)$, the Euclidean distance between them is evaluated by:

$$d_E(I_1, I_2) = \|I_1(x, y) - I_2(x, y)\| = \sqrt{(\sum_{x,y}[I_1(x, y) - I_2(x, y)]^2)}$$

There is a strong argument against the Euclidean distance as an image similarity metric in that it does not take into account the 2D topology of an image and it is not in accord with human perception [10, 8, 18, 24]. However there is a great advantage in using this measure since it allows the naive calculations to be greatly reduced by introducing a convolution scheme. This significantly reduces complexity and runtime. Another reason for the popularity of the Euclidean distance, is the ability to reduce the representation of linear spaces using PCA approaches. This is very helpful when a *set* of patterns are sought in an image. In such cases, it is possible to efficiently represent the pattern space using a small number of pattern basis vectors, so that the complexity of the search process can be reduced significantly (see e.g. [3]). In this paper, we deal with Euclidean distance, however, our scheme is applicable to any distance measure that forms a norm. The Euclidean distance is a particular case in this scheme.

2 The Euclidean Distance and its Complexity

Assuming a given 2D $k \times k$ pattern $P(x, y)$ is to be matched within an image $I(x, y)$ whose dimensions are $n \times n$. In principle, for each pixel location (x, y) in the image, the following distance

measure is calculated:

$$d_E^2(I,P)_{x,y} = \sum_{\{i,j\}=0}^{k-1} (I(x+i,y+j) - P(i,j))^2 \tag{2.1}$$

The smaller the distance measure at a particular location, the more similar the $k \times k$ local window is to the given pattern. If the distance is zero, the local window is identical to the pattern. In principle, this distance should be calculated for each pixel 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 and exploiting the convolution theorem. The Euclidean distance defined above can be re-written as:

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

The first term over all pixel locations, can be calculated by applying a convolution between the squared image $I^2(x,y)$ and a $k \times k$ mask of ones. The second term is fixed for each (x,y) and can be calculated only once. The third term can be calculated by applying a convolution between the $k \times k$ pattern and the image $I(x,y)$. Since convolution can be applied rapidly using FFT, it can be shown that the number of operations that are required for calculating the distance measure is $36 \log n$ additions and $24 \log n$ multiplications, for each pixel of the image. Table 1 summarizes the number of operations for each approach, in the case where a pattern of 32×32 pixels is sought in a $1K \times 1K$ image. Run times are calculated based on average values of the appropriate operations for the machines listed, and do not take into account time required for memory allocation, memory access and overhead [22]. 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 performs better, and as the pattern size increases, the Fourier approach becomes more advantageous. Actual run times for this pattern matching case are given in the Results Section (Section 6).

The run times obtained for patterns and images of reasonable size, are still vastly long for most real time applications. To improve run times, heuristics are introduced such as reducing the search space by assuming a priori probability on the location of the pattern within the image. Another common approach uses multiresolution schemes to quickly search for the pattern in low resolution versions of the image and then estimate the location of the pattern at higher resolutions [1, 6, 15].

	Naive	Fourier	Walsh-Hadamard
Average # operations per pixel	$+:\ 2k^2$ $*:\ k^2$	$+:\ 36 \log n$ $*:\ 24 \log n$	$+:\ 2 \log k + \epsilon$
Space	n^2	n^2	$2n^2 \log k$
Integer Arithmetics	Yes	No	Yes
Run time for PII, 450 MHz	20.3 Sec	5.3 Sec	23.5 Msec

Table 1: A comparison between pattern matching approaches for a 32×32 pattern sought in a $1K \times 1K$ image.

In this paper, a new approach is suggested which reduces complexity without compromising the exact results. The approach is shown to be highly efficient in terms of run time, reducing the complexity by two orders of magnitude. Theoretical run time for this approach is given in Table 1. Run time is based on machine operations only, as described above.

3 Euclidean Distance in Sub-Space

Assume a $k \times k$ pattern $\mathbf{p}$ is to be matched against a similar sized window $\mathbf{w}$ at a particular location in a given image. Referring to the pattern $\mathbf{p}$ and the window $\mathbf{w}$ as vectors in $\Re^{k^2}$, the Euclidean distance can be re-written in vectorial form:

$$d_E^2(\mathbf{p}, \mathbf{w}) = \|\mathbf{p} - \mathbf{w}\|^2 \tag{3.2}$$

Now, assume that $\mathbf{p}$ and $\mathbf{w}$ are not given, but only the values of their projection onto a particular unit vector $\mathbf{u}$: $\mathbf{u}^T\mathbf{p}$ and $\mathbf{u}^T\mathbf{w}$, where $\|\mathbf{u}\| = 1$. 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):

$$d_E(\mathbf{p}, \mathbf{w}) \geq d_E(\mathbf{u}^T\mathbf{p}, \mathbf{u}^T\mathbf{w}) \tag{3.3}$$

If an additional projection vector $\mathbf{v}$ is given along with its projection values: $\mathbf{v}^T\mathbf{p}$ and $\mathbf{v}^T\mathbf{w}$, it is possible to tighten the lower bound on the distance (Figure 2). The calculations follow the Grahm-Schmidt process for orthogonalization. For the new projection vector $\mathbf{v}$, its component along the direction orthogonal to to the previous projection vector $\mathbf{u}$ is calculated. Accordingly,

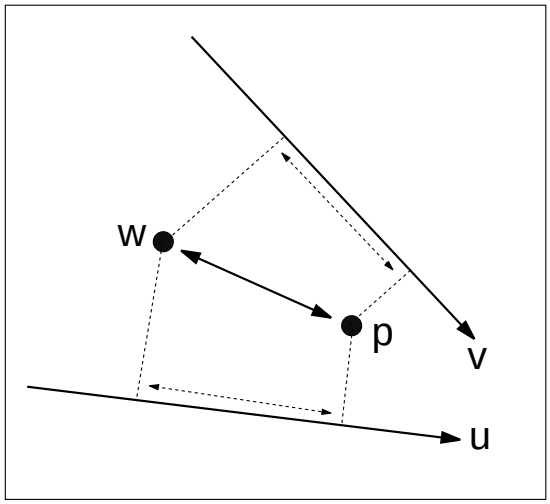


Figure 2: Projection of $\mathbf{p} - \mathbf{w}$ onto projection vectors produces lower bounds on the distance $\|\mathbf{p} - \mathbf{w}\|$.

the projection values of $\mathbf{p}$ and $\mathbf{w}$ along the orthogonal direction are inferred. Defining the previous bound:

$$\beta = d_E(\mathbf{u}^T\mathbf{p}, \mathbf{u}^T\mathbf{w})$$

and the distance between the projection values onto the new vector $\mathbf{v}$:

$$\alpha = d_E(\mathbf{v}^T\mathbf{p}, \mathbf{v}^T\mathbf{w})$$

the updated bound is (see Appendix B):

$$d_E^2(\mathbf{p}, \mathbf{w}) \geq \beta^2 + \frac{1}{\|\mathbf{v} - \mathbf{u}\mathbf{u}^T\mathbf{v}\|^2}\left(\alpha - \beta\mathbf{v}^T\mathbf{u}\right)^2$$

Note, that when $\mathbf{v}$ and $\mathbf{u}$ are parallel, the bound does not change. On the other hand, when $\mathbf{u}$ and $\mathbf{v}$ are orthogonal, the new bound is higher and thus tighter:

$$d_E^2(\mathbf{p}, \mathbf{w}) \geq \beta^2 + \frac{\alpha^2}{\|\mathbf{v}\|^2}$$

This scheme can be generalized for any number of projection vectors that are given sequentially;

Define the distance vector $\mathbf{d} = \mathbf{w} - \mathbf{p}$, and assume that the projection values of $\mathbf{d}$ onto k orthonormal projection vectors are given:

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

where $U = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots \mathbf{u}_k]$ is a matrix composed of the k orthonormal projection vectors, and $\mathbf{b}$ are the projection values. It is straightforward to verify that the lower-bound on the distance is:

$$d_E^2(\mathbf{p}, \mathbf{w}) = \mathbf{d}^T\mathbf{d} \geq \mathbf{b}^T\mathbf{b}$$

Now, if a new projection vector $\mathbf{v}$ is given where $\mathbf{v}^T\mathbf{d} = \alpha$, the new lower-bound will be (See Appendix B for proof):

$$\mathbf{d}^T\mathbf{d} \geq \mathbf{b}^T\mathbf{b} + \frac{1}{\gamma^2}(\alpha - \mathbf{v}^T U\mathbf{b})^2$$

where $\gamma = \|\mathbf{v} - UU^T\mathbf{v}\|$

Note, that 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 where $r = k^2$ and the projection vectors are linearly independent, the lower bound reaches the actual Euclidean distance. In addition, it can be seen from the algorithm that if the projection vectors $\mathbf{u}_i$ are mutually orthonormal, the lower bound is a direct sum of the projected difference:

$$LB = \sum_i \left(\mathbf{u}_i^T\mathbf{p} - \mathbf{u}_i^T\mathbf{w}\right)^2$$

4 Finding Efficient Projection Vectors

According to the discussion in the previous section, it is clear that any set of linearly independent projection vectors, provides a lower-bound on the Euclidean distance between a pattern $\mathbf{p}$ and an

image window $\mathbf{w}$. At first thought, however, it is unclear why the pattern $\mathbf{p}$ and all image windows $\mathbf{w}$ should be projected in order to calculate the distance $\|\mathbf{d}\| = \|\mathbf{p} - \mathbf{w}\|$ when the *exact* Euclidean distance can be computed directly for each window $\mathbf{d}^T\mathbf{d}$. The answer lies in the appropriate selection of the projection vectors. A large saving in calculations can be gained if the projection vectors are chosen carefully according to the following two necessary requirements:

- The projection vectors should have a high probability 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 should capture a large proportion of the distance between the pattern and the image window. If this requirement is met, a tight lower bound will be quickly obtained, and this, in turn, will allow rapid rejection of image windows that are distant from the pattern. The second requirement arises from the fact that in the pattern matching scenario discussed here, the projection calculations are performed many times (for each window of the image). As such, the efficiency of the projection calculations plays an important role when choosing the appropriate projection vectors.

The idea of choosing projection vectors that are fast to apply is not new [25, 12, 7]. In [25] Viola uses a set of projection vectors to produce a feature set, which serves as input to a face detection classifier. In Viola's framework, a set of vectors are chosen such that they can be applied very rapidly using an integral image scheme [5]. This process also includes a rejection phase where non-relevant windows can be classified as such very efficiently. Viola's scheme is similar in spirit to the method suggested in this paper, however in this work, we do not deal necessarily with classification problems. Additionally, in Viola's work, the projection vectors are restricted to those applicable in the integral image scheme. In some classification problems, however, such vectors can produce poor results as they may form non-informative feature inputs. In our suggested method, this behavior is avoided since the projection vectors form a complete representation of the input.

According to the above requirements, the Walsh-Hadamard basis vector is a good candidate for the projection vectors. It will be shown below that these vectors capture a large portion of the image energy with very few projections. Additionally, a scheme is suggested to calculate the projection values very efficiently, over all $k \times k$ windows in an image.

4.1 The Walsh-Hadamard Basis Vector

The Walsh-Hadamard transform has long been used for image representation under numerous applications. The elements of the basis vectors take only binary values (± 1). Thus, computation of the transform requires only integer additions and subtractions. The Walsh-Hadamard transform of an image window of size $k \times k$ ($k = 2^m$) is obtained by projecting the image onto k^2 basis vectors (projection vectors) [26, 20, 14, 23]. In our case, it is required to project *each* $k \times k$ window of the image onto the basis vectors. This results in a highly over-complete image representation, with k^2n^2 projection values for the entire image.

The basis vectors associated with the 2D Walsh-Hadamard transform of order $n = 2^3 = 8$ are shown in Figure 3. Each basis vector is of size 8×8 where black represents the value $+1$, and

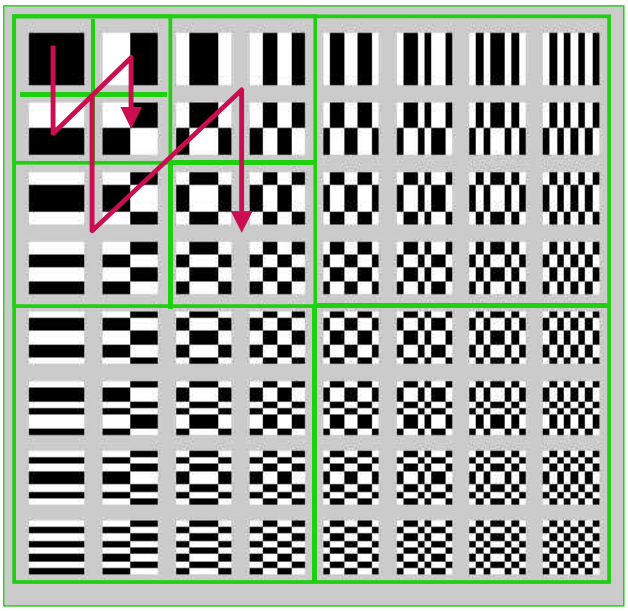


Figure 3: The projection vectors of the Walsh-Hadamard transform of order $n = 8$ ordered with increasing spatial frequency. Black represents the value 1, and white represents the value -1. A diadic ordering of these basis vectors is shown by the overlaying arrows.

white represents the value -1. In Figure 3, the basis vectors are displayed in order of increasing spatial frequency (the notion of frequency when discussing Walsh-Hadamard transforms is often denoted *sequency* and refers to the number of sign changes along rows and columns of the basis vector). A diadic ordering of these vectors is shown by the overlaying arrows. The 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 energy is captured by the first few basis vectors [13]. In terms of pattern matching, the lower bounds on distances between window and pattern, are shown to be tight after very few projections. Figure 4a displays this behavior by plotting the lower bound as a percentage of the total distance between image and pattern, versus the number of projection vectors used. The values displayed are averaged over 100 pattern-window pairs chosen randomly from natural images. It can be seen that the first 10 (out of 256) kernels capture over 70% of the distance. For comparison, Figure 4b shows the same lower bound when using the standard basis for projection (delta functions), i.e. when calculating the Euclidean distance in the spatial domain by accumulating pixel difference values.

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 projection vectors is introduced. The efficiency of computation is due to three main factors:

1. We take advantage of the fact that calculations applied to one window, can be exploited in the computation of the neighboring windows. This idea has been exploited in other windowing schemes such as the integral image [25], median filtering [9], and convolution using FFT [16].

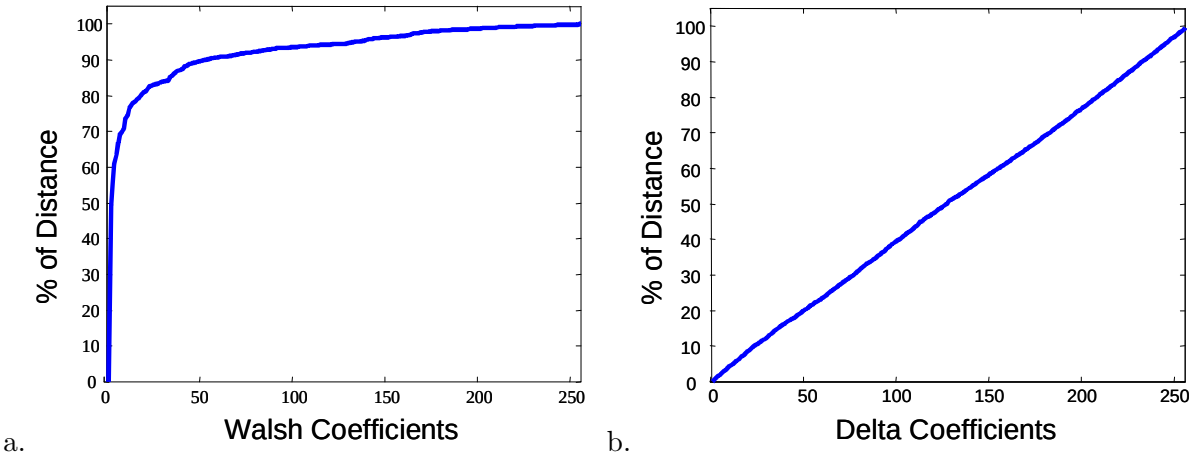


Figure 4: The lower bound on the distance between image window and pattern, as a percentage of the total distance, versus the number of projection kernels used. The values displayed are averaged over 100 $16x16$ pattern-window pairs chosen randomly from natural images. a) Projections are onto Walsh-Hadamard basis vectors. b) Projections are onto the standard basis vectors (i.e. Euclidean distance in the spatial domain).

2. We take advantage of the recursive structure of the Walsh-Hadamard basis vectors, where projections can be applied in a cascading manner. Thus, calculations applied to one basis vector, can be exploited in the computation of the next basis vectors.

3. Finally, we exploit the characteristic of the Walsh-Hadamard projection vectors discussed above, in which the first few vectors capture a large proportion of the distance energy between the pattern and a 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.

5 The Walsh-Hadamard Transform for Pattern Matching

5.1 The Walsh-Hadamard Tree Structure

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. Towards this end, we project each window of the signal, of length k, onto a set of 1D Walsh-Hadamard basis vectors. The 1D Walsh-Hadamard Transform of order k, projects a single window of the signal onto k basis vectors. In our case, however, we need the projection values for each window of the signal. These projections can be computed efficiently using the tree scheme depicted in Figure 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.

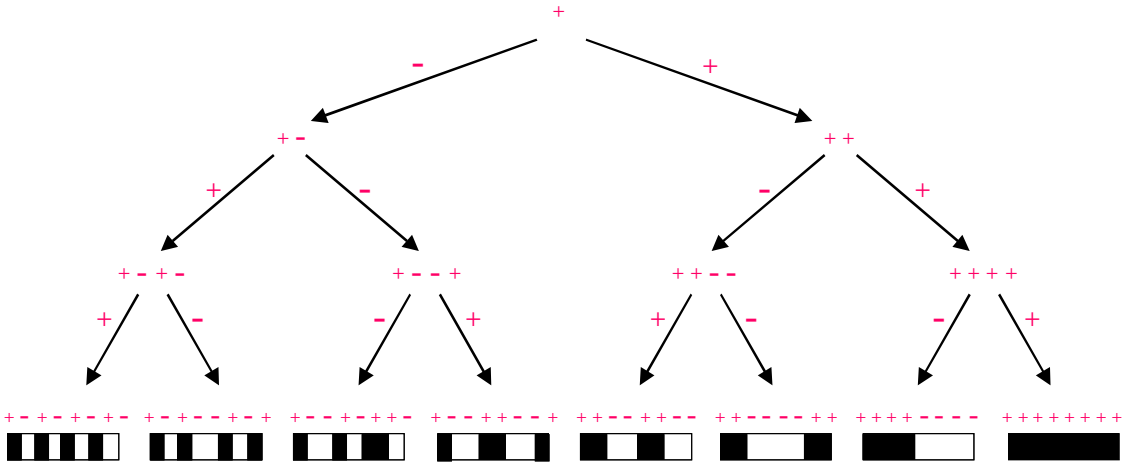


Figure 5: The tree-scheme for computing projections of all windows of a 1D image vector onto all Walsh-Hadamard kernels of order 8.

The symbols + and − represent the operations performed at the given tree level. 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 to every entry j of the signal, the value of entry $j+\Delta$ is added (subtracted), where $\Delta = 2^i$. Thus, the 2 signals at level 1 are obtained by adding or subtracting consecutive entries in the signal of level 0 which is the original signal. The 4 signals at level 2 are obtained by adding/subtracting entries at distance 2 in the signals at level 1, and so on. Figure 6 shows an example of the Walsh-Hadamard tree for a 1D signal when the pattern and window size are 4.

The Walsh-Hadamard tree is used to compute projections of signal windows onto Walsh-Hadamard basis vectors. Computations within the tree are typically performed by descending from a node

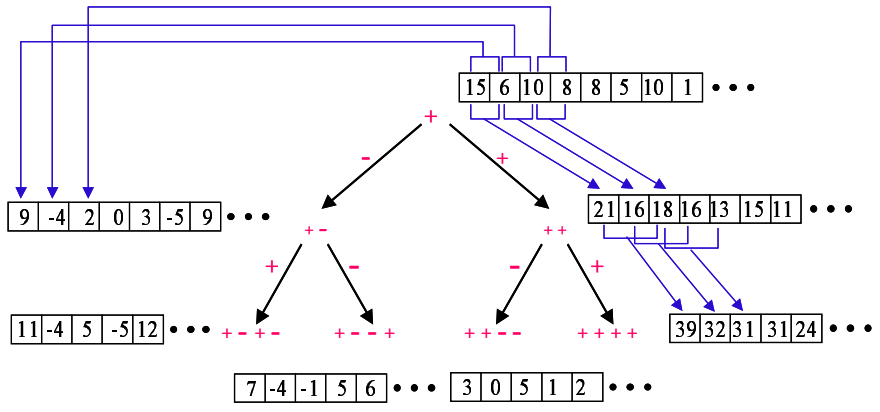


Figure 6: The tree-scheme for computing projections of all windows of a 1D image vector onto all Walsh-Hadamard kernels of order 8.

in the tree to its child. This requires a single operation per pixel: addition or subtraction of two values in the parent node. The following lists various 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 down to the appropriate leaf. At each level, all entries at that node are calculated. Note, that due to the tree structure, every intermediate computation actually serves many windows. Thus, computation is efficient and a total of only $\log_2(k)$ operations per signal entry 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. Thus projection can be calculated as described above for each projection vector. However in the top-down descent of the tree, many nodes are common to several branches. For example, the first 2 projection vectors, have $\log_2(k)$ nodes in common to both paths. Thus, given the branch of signals 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 the leaf nodes of the tree in right to left order, the Walsh-Hadamard tree is traversed in pre-order. The projection of all windows onto the first l projection vectors requires m operations per pixel, where m is the number of nodes preceding the l leaf in the pre-order tree traversal. Thus, projecting all windows onto **all** projection vectors requires only $2k - 2$ operations per pixel (about 2 operations per pixel per projection vector!).
- **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 required for the computation. To compute an entry at level i, two entries are needed at level $i - 1$, therefore, a total of $k - 1$ operations (additions/subtractions) are required for this projection.

5.2 Using The Walsh-Hadamard Tree Structure for Pattern Matching

The pattern matching approach we propose, uses the Walsh-Hadamard tree structure and follows the projection schemes described above. In general, all signal windows must be projected onto all Walsh-Hadamard basis vectors. However, as described in Section 3, a lower bound on the distance between each window and the pattern can be estimated from the projections. Thus, complexity and run time of the pattern matching process can be significantly reduced by rejecting windows with lower bounds exceeding a given threshold value. The rejection of windows implies that many signal values at the tree leaves need not be computed.

In the pattern matching process, the projections are performed on basis vectors in the order determined by traversing the leaves in right to left order. At each step in the process, only a **single**

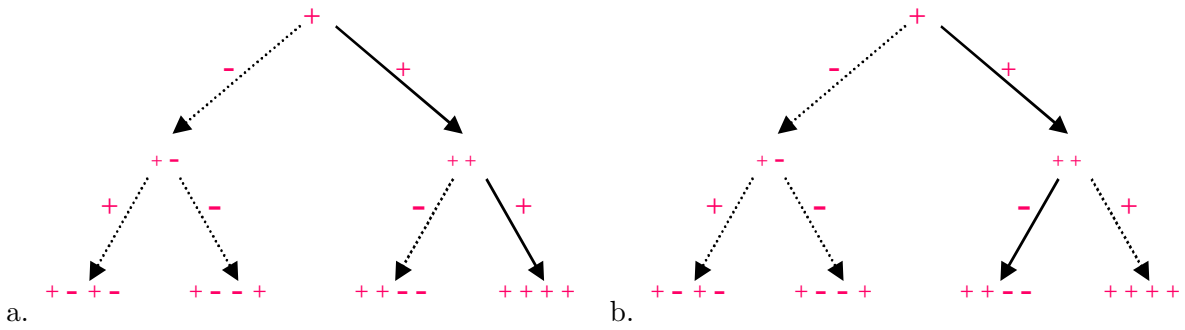


Figure 7: Pattern Matching Scheme using the Walsh-Hadamard tree of order 4. a) Initial step computes the first branch of the image, associated with the first Walsh-Hadamard kernel (solid lines). b) Computing the branch associated with the second kernel, exploits the fact that several of the branch nodes have already been computed.

branch of the Walsh-hadamard tree is maintained. This branch includes all the nodes from the root down to the current leaf. Therefore, only the signals associated with these nodes must be stored, and the memory required for this process is $n(\log_2(k)+1)$.

Assume the sought pattern $\mathbf{p}$ is projected onto the set of k Walsh-Hadamard basis vectors $\{\mathbf{u}_i\}$, resulting in k values: $\hat{p}_i = \mathbf{p}^T\mathbf{u}_i$, for $i = 1..k$. The pattern matching process then proceeds as follows:

1. The first branch of the tree is computed, obtaining the projection of all signal windows $\{\mathbf{w}_i\}$ onto the first Walsh-Hadamard projection vector $\mathbf{u}_1$: $\hat{w}_i^1 = \mathbf{w}_i^T\mathbf{u}_1$ (Figure 7a).
2. This projection sets a lower bound on the true distance between each window $\mathbf{w}_i$ and the pattern: $LB_i = (\hat{w}_i^1 - \hat{p}_1)^2$. According to the lower bound values, any window whose LB value is greater than the given threshold can be rejected.
3. The windows of the image that have not been rejected, are projected onto the second projection vector $\mathbf{u}_2$: $\hat{w}_i^2 = \mathbf{w}_i^T\mathbf{u}_2$. This is performed by replacing the maintained tree-branch associated with the first leaf node with the branch associated with the second leaf node (Figure 7b). This produces updated lower bounds: $LB_i = LB_i + (\hat{w}_i^2 - \hat{p}_2)^2$.
4. Steps 2 and 3 are repeated for the subsequent projection vector and only for those image windows that have not been rejected.
5. The process terminates after all k kernels have been processed or until the number of non-rejected 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 the signal windows. This method is preferable in the initial stages (typically only the first stage), when the number of rejected windows is small. However, in advanced stages, when the number of rejected windows is sufficiently large, it is preferable to compute the projection on a pixel by pixel basis, i.e. in a bottom-up manner (see Section 5.1). In

practice, when only a very few image windows remain, it is preferable to calculate the Euclidean distance directly for these windows (using the standard basis).

The complexity of the pattern matching process may be calculated as follows: Initially (Step 1), $\log_2(k)$ operations per pixel are required to calculate the projection of all windows onto the first projection vector. In the following stages of the process (repeated Steps 2-3), every projection requires only $2^l - 1$ operations per pixel where l is the number of nodes that differ between the current branch and the previous branch of the tree. However, these $2^l - 1$ operations are required **only** for those signal values associated with the non-rejected windows. The number of computations performed during the process is evaluated and discussed in Section 5.3 and Section 6.

5.3 The Walsh-Hadamard Transform for Pattern Matching in 2D

The pattern matching process described above for 1D signal, can be extended to 2D images. Assume that all appearances of a pattern of size $k \times k$ are sought in a 2D image of size $n \times n$. Towards this end, each $k \times k$ window of the image is projected onto a set of 2D Walsh-Hadamard kernels. A Walsh-Hadamard tree similar to that described in Section 5.1 is used here. 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. Figure 8 depicts such a tree for the Walsh-Hadamard transform of order 2×2. 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 $\overrightarrow{+}, \overrightarrow{-}$, $+\downarrow$ and $-\downarrow$. As in the 1D case, the operations performed are additions and subtractions of pixels at a distance Δ 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 appearing in the symbols. Thus, the symbol $+\downarrow$ on an edge connecting nodes at level i and level $i+1$ denotes

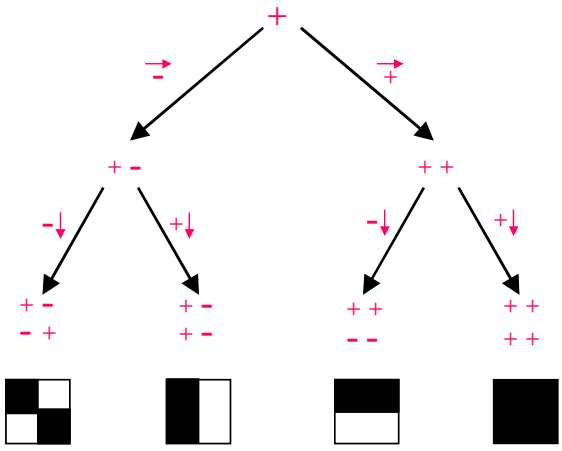


Figure 8: The tree-scheme for computing projections of all windows of a 2D image onto all Walsh-Hadamard kernels of order 2×2.

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. There are $2\log_2(k) + 1$ levels in the tree, thus, for every leaf (kernel), $2\log_2(k)$ operations (additions/subtraction) per pixel are required. As in the 1D case, some nodes are common to many kernel projection computations, and thus, significant computation reduction is achieved.

The pattern matching process in 2D follows the same steps as described above for the 1D case. In the 2D case, $2\log_2(k) + 1$ images of size $n \times n$ are maintained at all times, corresponding to a single branch of the 2D Walsh-Hadamard tree. At each step of the process, image windows are projected onto a new projection kernel by updating the maintained branch to represent the branch associated with the new kernel. As in the 1D case, the projections of all image windows onto the i-th kernel can be computed from the branch of images associated with the computation of the $(i-1)$-th kernel. The number of additional operations required per pixel is dependent on the number of tree nodes that differ between the branch associated with the i-th kernel and the branch associated with the $(i-1)$-th kernel. (The number of differentiating nodes is easily computed since it equals the most significant bit that differs between the binary representation of the indices of the two kernels). Thus in 2D, although projecting all image windows onto a single kernel requires $2\log_2(k)$ operations per pixel, projecting all image windows onto **all** kernels requires only $2(k^2-1)$ operations per pixel. However, during the 2D pattern matching process, not even all these $2(k^2-1)$ operations per pixel are required, since many windows are rejected during the process. As in the 1D scheme, the projection values define a lower bound on the true distance between pattern and window. During the pattern matching process, windows with a lower bound exceeding a given threshold are rejected.

The Walsh-Hadamard tree that produces projections in order of diadically increasing frequency (as shown in Figure 8) is defined by the operations (+ or -) assigned to each edge. The rule defining these operations is as follows: Let $S = (+ - + - - + -+)$ be a seed sequence, and $P = \{S\}^*$ an arbitrary number of repetitions of S. At each level of the tree, scanning the operations applied at each node from right to left, yields a sequence equal to an initial string of P. From this rule it is straightforward to calculate the operation that is to be applied at any given tree level for any given branch.

In terms of complexity, the process initially requires (Step 1) $2\log_2(k)$ operations per pixel to calculate the projection of all windows onto the first projection vector. In the following stages of the process (repeated Steps 2-3), every projection requires only $2^l - 1$ operations per pixel where l is the number of nodes that differ between the current branch and the previous branch of the tree. However, these $2^l - 1$ operations are required **only** for those signal values associated with the non-rejected windows. Thus, to increase efficiently, the order of projection kernels is very important. The order of projection should be that which produces tight lower bounds on the distances between pattern and windows, using as few projections as possible. The order chosen is that of diadically increasing frequency of the kernels (Figure 3). It will be shown in Section 6 that this ordering indeed captures most of the image energy, i.e. produces tight lower bounds on the distances with very few projections. Thus the number of operations per pixel beyond the first step becomes negligible.

The number of computations actually performed during the pattern matching process is evaluated and discussed in Section 5.3 and Section 6.

Figure 9: Inputs for the pattern matching example. a. Original 256×256 image. b. 16×16 pattern. c. The pattern shown at large scale.

6 Experimental Results

The proposed pattern matching scheme has been tested on real images and patterns. As an example, Figure 9a shows an original image of size 256×256 and Figures 9b-c show the 16×16 pattern that is to be matched. Figures 10a-c show the results of the pattern matching scheme. Following the suggested method, the projection of all 65536 windows of the image onto the first Walsh-Hadamard kernel were calculated. All windows with projection values above a given threshold were rejected (the threshold value allowed up to 0.7 pixel difference on average). After the first projection, only 602 candidate windows remained (Figure 10a), i.e. only 0.92% of the original windows. Following the second projection only 8 candidate windows remained (Figure 10b) and following the third

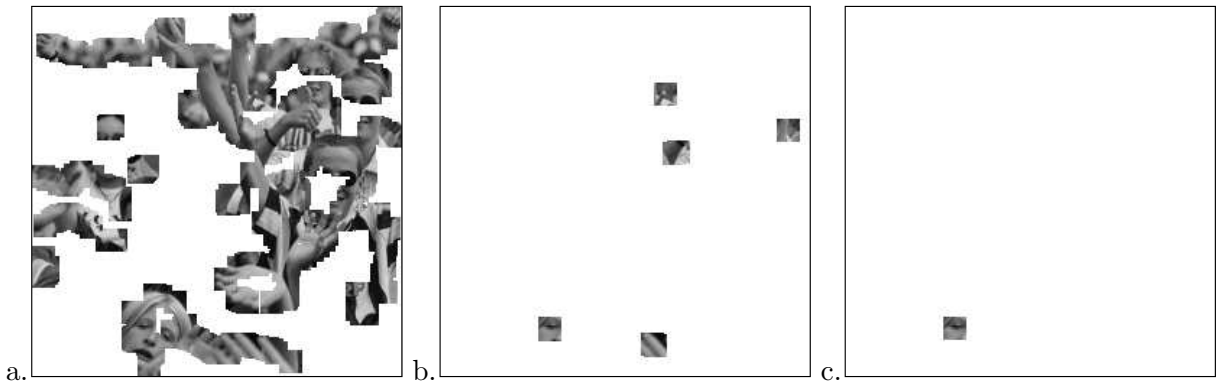


Figure 10: Three steps in the pattern matching process. a. Following the first projection only 602 candidate windows remain (0.92% of the windows). b. Following the second projection only 8 candidate windows remain. c. After the third projection a single window remains corresponding to the correct pattern location.

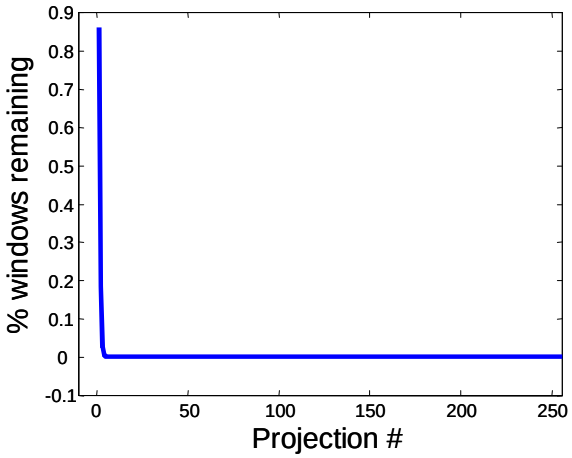


Figure 11: The percentage of image windows remaining after each projection for images of size 256×256 and patterns of size 16×16. The results are the average over 100 image-pattern pairs.

projection a single window with the correct pattern remained (Figure 10c).

The performance shown in this example is typical and is attained over many such examples. Figure 11 shows the percentage of image windows remaining after each projection for images of size 256×256 and patterns of size 16×16. The results are the average over 100 image-pattern pairs. Using these results, an estimate of the average number of operations required per pixel can be calculated. Figure 12 displays the accumulated number of operations (additions/subtractions) verses the number of projections. The average number of operations per pixel for the whole process is 8.0154 which is slightly over $2\log_2(16)$ as expected (see Section 5.3).

Run time comparison was performed between the Naive approach, the Fourier approach, and the

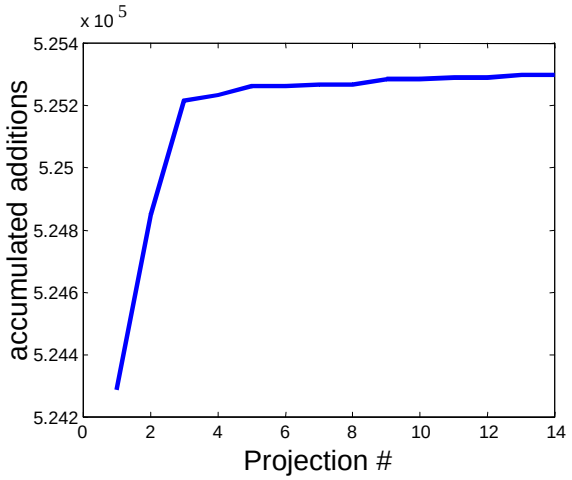


Figure 12: The number of accumulated operations (additions/subtractions) verses the number of projections averaged over 100 image-pattern pairs. Images were of size 256×256 and patterns of size 16×16. The average number of operations per pixel is 8.0154 which is slightly over $2log_2(16)$.

suggested method for the above experiment using patterns of size 16×16 and 32×32. The experiments were performed on a PIII processor, 1.8 GHz. The average number of operations and run times are summarized in Table 2.

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

Table 2: A comparison between existing pattern matching approaches and the proposed approach.

6.1 Pattern matching with noise

The ideal situation depicted in the example shown above is not typical. Usually the pattern is allowed some variation in appearance due to noise, quantization, digitization and transformation errors. In the following experiment, images contain noisy versions of the pattern. Figure 13a shows the original image which contains several embedded copies of a given pattern. Figure 13b shows a noisy version of the image with noise level of 40 (i.e. each pixel value in the image was increased/decreased by at most 40 gray values). Figure 13c shows that all 10 noisy patterns were detected even under these very noisy conditions. The pattern matching scheme under these conditions require a larger threshold for rejection (allowing the acceptable distance between pattern and image to increase). This implies that fewer image windows are rejected after each projection

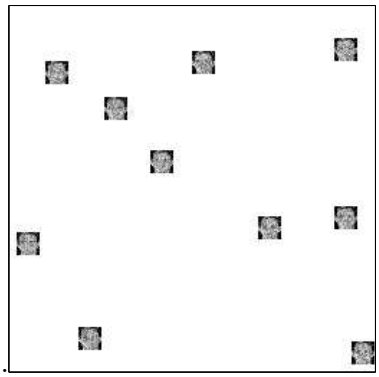
a.  b.  c. 

Figure 13: a. Original 256×256 image containing several repeated patterns of size 16×16 positioned randomly in the image. b. Very noisy version of original with noise level set to 40 (see text). c. All noisy patterns were detected.

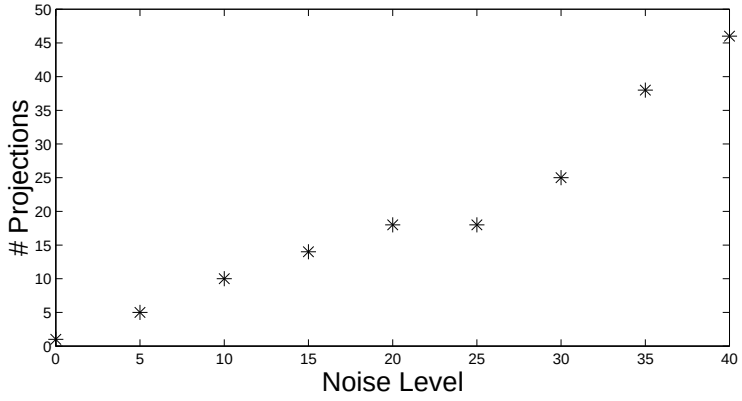


Figure 14: The minimum number of projections required to find all noisy patterns in an image as a function of the noise level.

and that the overall number of required projections increases with the threshold. Figure 14 shows this behavior over a range of noise levels. The minimum number of projections required to find all noisy patterns in a given image is shown as a function of the noise level. In all cases, the threshold used is the minimum value that produces no miss-detections. The threshold value and the number of projections required increase as the noise level increases.

Figure 15 shows the percentage of image windows remaining after each projection, for images of varying noise levels. The images are of size 256×256 and the pattern of size 16×16. As the noise level increases, fewer windows are rejected after each projection, however the decreasing profile for the various thresholds is similar (Figure 15 inset shows a scaled version of the values for the first few projections). Figure 16 shows the number of accumulated operations (additions/subtractions) performed following each projection for various noise levels (Figure 13). The average number of

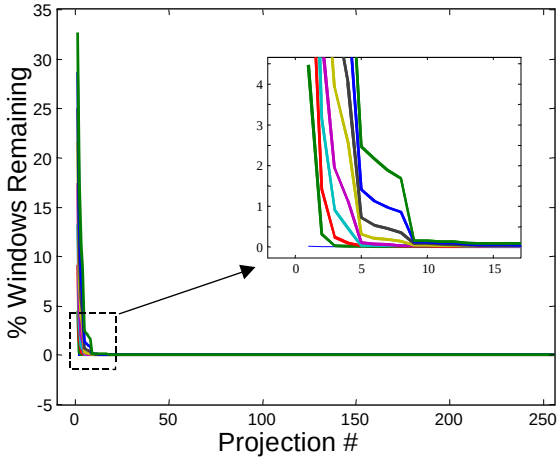


Figure 15: The percentage of image windows remaining after each projection for the image shown in Figure 13a at various noise levels. The image is of size 256×256 and the pattern of size 16×16.

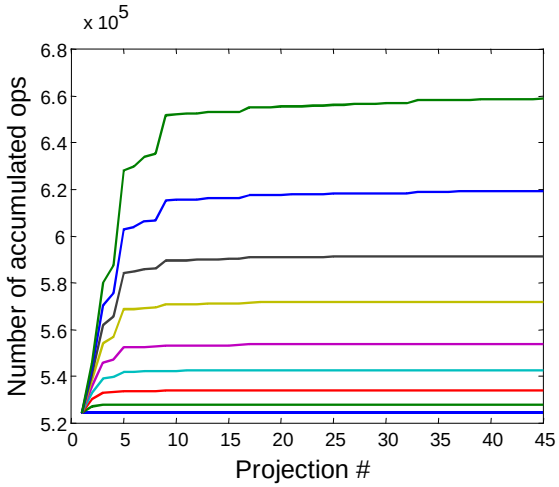


Figure 16: The number of accumulated operations (additions/subtractions) verses the number of projections for the image shown in Figure 13a at various noise levels. Image is of size 256×256 and pattern of size 16×16. The average number of operations per pixel at the highest noise level is 10.0623.

operations per pixel at the maximal noise level is 10.0623 which is only slightly higher than the value obtained for non-noisy images of the same size (Section 6 and Figure 12).

7 Computational Aspects

The pattern matching process involves three types of techniques for evaluating the distance between image windows and pattern. The computational advantage of each technique, depends on the number of non-rejected 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 image windows remain in the pattern matching process. 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. Using this approach, a projection value is calculated for *all windows* in the image providing lower bounds on the actual distances. However, many calculations can be spared since intermediate values in the computation are given from neighboring windows and from previous projections. Thus the current branch needs to be explored only from a node level that has not been explored during the previous projection calculations. The more levels that have to be explored the more expensive this approach. Since projection values are calculated for all windows, this approach is preferable in the earlier stages of the matching process, when a large portion of the windows still remain to be matched. Figure 17 (left) shows the computational time for the DD and the TD approaches, v.s. the number of remaining windows (in percentage). 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 when the percentage of remaining windows is greater than 5% it is preferable to use the TD

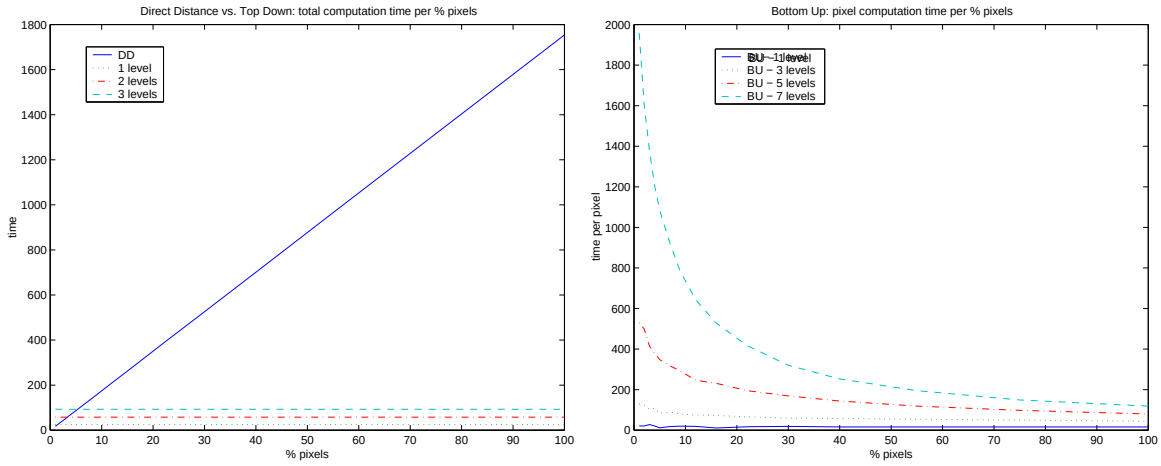


Figure 17: Left: The computation time for the DD and the TD approaches, v.s. the percentage of remaining of windows. The time required for TD is given for various numbers of tree levels that must be explored. Right: The average computation time per pixel for the BU approach v.s. the percentage of remaining of windows.

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 pattern by exploring only the necessary values within the tree, for each non-rejected window. Values are calculated recursively in a bottom-up manner from tree leaf up to the tree node whose values have already been computed (Section 5). Here too, exploiting previously explored values existing in the current branch, can greatly reduce computations. Using the BU method, the more windows that are evaluated, the greater the saving in computations, since intermediate values are common to neighboring windows. Figure 17 (right) shows this behavior, where average time per pixel rapidly declines as the number of remaining

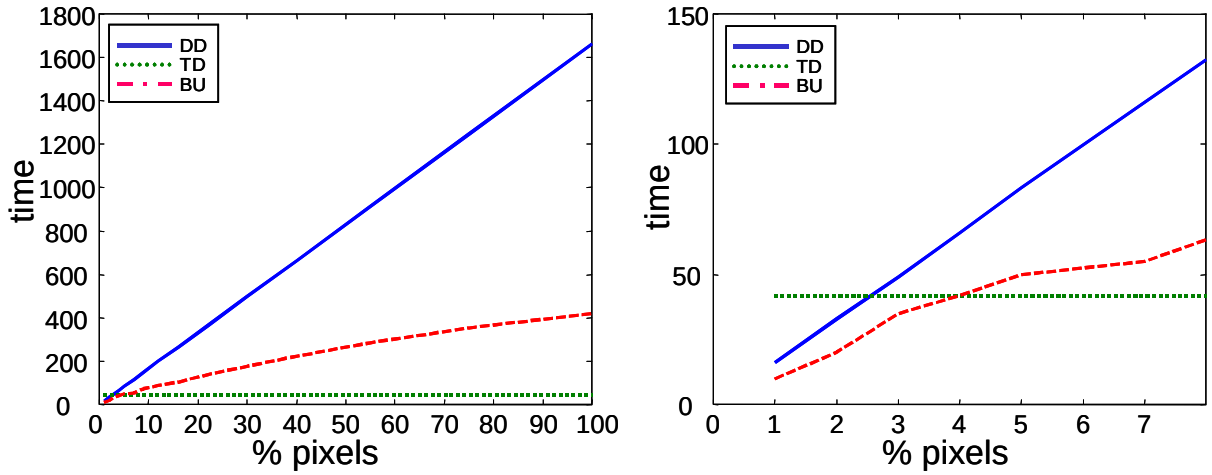


Figure 18: The computation time for the TD, BU and DD approaches, v.s. the percentage of remaining of windows when 3 levels of the branch must be explored. Scaled version shown on right.

windows (in percentage) increases. Thus, the BU method is beneficial at intermediate stages, when a large portion of the windows has already been rejected (so that TD approach is not beneficial), and yet more than a few windows still remain so that the DD approach is expensive. Figure 18 shows this behavior for the case where 3 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%. In a narrow range, between 4% and 2.5% remaining windows, it is preferable to use the BU method, and when less than 2.5% windows remain, the DD is superior. Such a comparison can be designed for different cases depending on the number of levels. Table 3 summarizes which of the 3 methods should be used in each case, where a pattern of size 64×64 is sought in a $1k \times 1k$ image. The table depicts the percentage of remaining windows at which a method should be used (assuming TD is used initially with 100% of the windows). It is interesting to note that when traversal of few levels are required (up to 4 levels), it is preferable to use the TB method during the early stages and the BU method at later stages. If more than 5 levels are required, it is preferable to transfer directly from TB to DD. Only when 5 levels are required, it is advantageous to use all three methods depending on the remaining percentages of windows.

# Levels	Use BU	Use DD
1	10%	never
2	7.5%	never
3	5%	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%

Table 3: Pattern matching using 3 types of distance computations (TD, BU and DD - see text). 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 at which a method should be used is given for various number of levels (assuming TD is used initially with 100% of the windows). The values were determined experimentally where a pattern of size 64×64 was sought in a $1k \times 1k$ image.

8 Discussion

The pattern matching scheme proposed in this paper assumes the Euclidean distance is used for measuring similarity between pattern and image. Under this assumption the proposed scheme is advantageous over the existing naive schemes first and foremost due to the reduction in time complexity of 2 orders of magnitude (as shown in Table 2). The reduction in complexity is due to a number of factors:

- The projection of an image window onto the Walsh-Hadamard basis kernels is characterized by a fast decay of the energy content in the coefficients [13].
- The transform coefficients (projection values) can be computed efficiently and incrementally using the tree structure described in Section 5.
- The incremental computation and the concentration of energy in the first few transform coefficients, induces a fast rejection scheme where many image windows are rejected in early stages of the process.
- Independent of the rejection scheme, is the fact that all computations of the transform coefficients are performed using integer additions and subtractions which is much faster than floating operations in most hardware [22].

In addition to the advantage in time complexity, the proposed scheme has the following additional advantages:

- It easily allows disregarding of the DC values from the pattern and all image windows. This can be used to partially compensate for illumination variations. The DC value of all windows are in fact the values of the projection onto the first Walsh-Hadamard kernel. Thus the first branch of the Walsh-Hadamard tree (Section 5) may be skipped and the process initialized by computing the projections onto the second kernel.
- The scheme easily allows incorporating a multi-scale pattern matching, where a pattern at several scales is sought in the image. Under the Walsh-Hadamard tree scheme, matching patterns whose size are smaller than the original pattern by powers of 2, can be applied almost for free, since the appropriate Walsh-Hadamard kernels for these patterns are already given at the intermediate tree nodes.
- The projection values of neighboring image windows tend to have similar and their difference can be bounded. This bound is relatively tight at the first few projection kernels. This fact can be exploited in a multi-scale scheme, where only a fraction of the projection values are calculated, and bounds (above and below) can be calculated for the missing values. The process may be repeated recursively to obtain a pattern matching scheme with improved run time.

The proposed technique as presented in this paper, has several limitations which are intended to be resolved in future extension:

- The Pattern Matching approach described in this paper, works only for the Euclidean norm. Future work will extend this approach to other norms. In principle, this can be done by applying the equivalence relation between norms, such that the L_2 norm forms a lower and upper bounds when multiplied by specific constants [11].
- At this stage, the suggested approach supports matching patterns whose size equals a power of 2. Future work will extend the approach to images of any size using the general size Walsh-Hadamard basis [26].

- Run times of the suggested algorithm can be further improved by exploiting the special structure of the Walsh-Hadamard basis vectors. In a current study we use a novel technique that uses only 2 operations per pixel in order to calculate all window projections onto a kernel regardless of pattern (kernel) size [4].

Finally, we should note the fact that 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, this increase in memory, is acceptable.

9 Conclusion

A novel approach to pattern matching is presented, which reduces time complexity by 2 orders of magnitude compared to the traditional approaches. The suggested approach uses an efficient projection algorithm which allows the distance between a pattern and a window to be measured in the Walsh-Hadamard domain rather than the spatial domain. Rather than scanning the image windows and computing all projection values for each image, the suggested approach scans the projection vectors and for each vector projects all image windows. This allows the efficient projection algorithm to be combined with a rejection scheme. Thus, using relatively few projections, a large number of windows are rejected from further computations. Complexity of computation is reduced by using the Walsh-Hadamard tree structure which exploits computations performed for a given projection vector, in the computations performed for the following vector. Additionally, projection computations performed on one window are exploited in the projection computations of the neighboring windows. Experiments show that the approach is efficient even under very noisy conditions. This approach can be implemented to any dimensional signal domain.

Acknowledgments

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

Appendix

A Distance Lower Bound for a Single Projection Vector

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

$$d_E(\mathbf{p}, \mathbf{w}) \geq d_E(\mathbf{u}^T\mathbf{p}, \mathbf{u}^T\mathbf{w}) \tag{1.4}$$

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

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

Since $\|u\| = 1$, this implies:

$$\|\mathbf{p} - \mathbf{w}\| \geq \|\mathbf{u}^T(\mathbf{p} - \mathbf{w})\| = \|\mathbf{u}^T\mathbf{p} - \mathbf{u}^T\mathbf{w}\|$$

Since the distance $d_E(x, y)$ is the Euclidean norm of $(x - y)$ we have:

$$d_E(\mathbf{p}, \mathbf{w}) \geq d_E(\mathbf{u}^T\mathbf{p}, \mathbf{u}^T\mathbf{w}) \qquad \square$$

B Distance Lower Bound for a Set of Projection Vectors

Lemma 2 *Given is the following linear system with the unknown vector $\mathbf{d}$:*

$$\begin{pmatrix} U^T \\ \mathbf{v}^T \end{pmatrix} \mathbf{d} = \begin{pmatrix} \mathbf{b} \\ \alpha \end{pmatrix} \tag{2.5}$$

where U is an $n \times k$ orthonormal matrix, such that $U^TU = I_k$, $\mathbf{v}$ and $\mathbf{d}$ are n dimensional vectors, $\mathbf{b}$ is a k dimensional vector, and α is a scalar. The norm-2 of $\mathbf{d}$ is bounded from below:

$$\mathbf{d}^T\mathbf{d} \geq \mathbf{b}^T\mathbf{b} + \frac{1}{\gamma^2}(\alpha - \mathbf{v}^TU\mathbf{b})^2$$

where $\gamma = \|\mathbf{v} - UU^T\mathbf{v}\|$

Proof: Let's $Range(U)$ and $Null(U)$ be the column space and the null space of U, respectively. The projection of vector $\mathbf{v}$ onto $Null(U)$ is:

$$\mathbf{v}_N = \mathbf{v} - UU^T\mathbf{v} \tag{2.6}$$

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

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

Using Equation 2.6 it is easily verified that:

$$\begin{pmatrix} U & \mathbf{v} \end{pmatrix} = \begin{pmatrix} U & \hat{\mathbf{v}}_N \end{pmatrix} \begin{pmatrix} I & U^T\mathbf{v} \\ 0 & \gamma \end{pmatrix}$$

Note however that $[U \quad \hat{\mathbf{v}}_N]$ is orthonormal, i.e. $[U \quad \hat{\mathbf{v}}_N]^T \, [U \quad \hat{\mathbf{v}}_N] = I$. Rewriting Equation 2.5 using the above relation yields:

$$\begin{pmatrix} U & \mathbf{v} \end{pmatrix}^T \mathbf{d} = \begin{pmatrix} I & 0 \\ \mathbf{v}^T U & \gamma \end{pmatrix} \begin{pmatrix} U^T \\ \hat{\mathbf{v}}_N^T \end{pmatrix} \mathbf{d} = \begin{pmatrix} \mathbf{b} \\ \alpha \end{pmatrix}$$

However, since:

$$\frac{1}{\gamma} \begin{pmatrix} \gamma I & 0 \\ -\mathbf{v}^T U & 1 \end{pmatrix} \begin{pmatrix} I & 0 \\ \mathbf{v}^T U & \gamma \end{pmatrix} = I$$

It is straightforward to see that:

$$\begin{pmatrix} U^T \\ \hat{\mathbf{v}}_N^T \end{pmatrix} \mathbf{d} = \frac{1}{\gamma} \begin{pmatrix} \gamma I & 0 \\ -\mathbf{v}^T U & 1 \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \alpha \end{pmatrix} \tag{2.7}$$

Since $[U \quad \hat{\mathbf{v}}_N]$ is orthonormal, the minimum norm-2 solution for $\mathbf{d}$ gives [11]:

$$\tilde{\mathbf{d}} = \frac{1}{\gamma} \begin{pmatrix} U & \hat{\mathbf{v}}_N \end{pmatrix} \begin{pmatrix} \gamma I & 0 \\ -\mathbf{v}^T U & 1 \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ \alpha \end{pmatrix} = \begin{pmatrix} U & \hat{\mathbf{v}}_N \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ (\alpha - \mathbf{v}^T U \mathbf{b})/\gamma \end{pmatrix}$$

and as such yields:

$$\mathbf{d}^T\mathbf{d} \geq \tilde{\mathbf{d}}^T\tilde{\mathbf{d}} = \begin{pmatrix} \mathbf{b}^T & (\alpha - \mathbf{v}^T U \mathbf{b})/\gamma \end{pmatrix} \begin{pmatrix} U^T \\ \hat{\mathbf{v}}_N^T \end{pmatrix} \begin{pmatrix} U & \hat{\mathbf{v}}_N \end{pmatrix} \begin{pmatrix} \mathbf{b} \\ (\alpha - \mathbf{v}^T U \mathbf{b})/\gamma \end{pmatrix}$$

However, since $[U \; \hat{\mathbf{v}}_N]$ is orthonormal it gives:

$$\mathbf{d}^T\mathbf{d} \geq \mathbf{b}^T\mathbf{b} + \frac{1}{\gamma^2}(\alpha - \mathbf{v}^T U \mathbf{b})^2 \qquad \square$$

References

[1] E.H. Adelson, C.H. Anderson, J.R. Bergen, P.J. Burt, and J.M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984.

[2] A. J. Ahumada. Computational image quality metrics: A review. In *Society for Information Display International Symposium*, volume 24, pages 305–308, 1998.

[3] S. Baker, S.K. Nayar, and H. Murase. Parametric feature detection. *International Journal of Computer Vision*, 27:27–50, 1998.

[4] G. Ben-Artzi, H. Hel-Or, and Y. Hel-Or. Very fast image pattern matching. *In preperation.*

[5] F. Crow. Summed-area tables for texture mapping. *SIGGRAPH*, 18(3):207–212, 1984.

[6] J. Crowley and A. Sanderson. Multiple resolution representation and probabilistic matching of 2-d grey scale shape. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:113–121, 1987.

[7] M. Elad, Y. Hel-Or, and R. Keshet. Pattern detection using maximal rejection classifier. In *IWVF4 - 4th International Workshop on Visual Form*, pages 28–30, Capri, Italy, May 2000.

[8] A.M. Eskicioglu and P.S. Fisher. Image quality measures and their performance. *IEEE Transactions on Communication*, 43(12):2959–2965, 1995.

[9] Y. Gil and M. Werman. Computing 2d min, max and median filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15:504–507, 1993.

[10] B. Girod. Whats wrong with mean-squared error? In A.B. Watson, editor, *Digital Images and Human Vision*, chapter 15, pages 207–220. MIT press, 1993.

[11] G.H. Golub and C.F. Van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, MD, 1989.

[12] D. Keren, M. Osadchy, and C. Gotsman. Antifaces: A novel, fast method for image detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(7):747–761, 2001.

[13] H. Kitajima. Energy packing efficiency of the hadamard transform. *IEEE Transactions on Communication*, pages 1256–1258, 1976.

[14] M.H. Lee and M. Kaveh. Fast hadamard transform based on a simple matrix factorization. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 34(6):1666–1667, 1986.

[15] W.J. MacLean and J.K.T. Tsotsos. Fast pattern recognition using gradient-descent search in an image pyramid. In *Proceedings of the 15 International Conference on Pattern Recognition*, pages 877–881, 2000.

[16] B. Porat. *A Course in Digital Signal Processing.* John Wiley and Sons, NY, 1979.

[17] J. Puzicha, Y. Rubner, C. Tomasi, and J.M. Buhmann. Empirical evaluation of dissimilarity measures for color and texture. In *IEEE International Conference on Computer Vision*, pages 1165–1172, 1999.

[18] R. Russel and P. Sinha. Perceptually-based comparison of image similarity metrics. Technical Report AI Memo 2001-14, MIT Artificial Intelligence Laboratory, 2001.

[19] S. Santini and R. Jain. Similarity measures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):871–883, 1999.

[20] J.L. Shanks. Computation of the fast walsh-fourier transform. *IEEE Transactions on Computers*, C18:457–459, 1969.

[21] D.A. Silverstein and J.E. Farrell. Quantifying perceptual image quality. In *The Society for Imaging Science and Technology, Image processing quality and capture*, 1998.

[22] Carl Staelin. Personal communication, 2002.

[23] D. Sundararajan and M.O. Ahmad. Fast computation of the discrete walsh and hadamard transforms. *IEEE Transactions on Image Processing*, 7(6):898–904, 1998.

[24] P.C. Teo and D.J. Heeger. Perceptual image distortion. In *Proceedings of the First IEEE International Conference on Image Processing*, volume 2, pages 982–986, 1994.

[25] P. Viola and M. Jones. Robust real-time object detection. In *ICCV Workshop on Statistical and Computation Theories of Vision*, Vancouver Canada, July 2001.

[26] J. Kane W.K. Pratt and H.C. Andrews. Hadamard transform image coding. *Proceedings IEEE*, 57(1):58–68, 1969.