Real Time Pattern Matching Using Projection Kernels

Yacov Hel-Or

School of Computer Science
The Interdisciplinary Center
Herzeliya, Israel

Hagit Hel-Or

Dept of Computer Science
University of Haifa
Haifa, Israel

Abstract

A novel approach to pattern matching is presented, which reduces time complexity 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. 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.

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.

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 strong 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 [5].

Assume a 2D $k \times k$ pattern $P(i,j)$ is to be matched within an image $I(i,j)$ of size $n \times n$. For each $k \times k$ image window $I_{(x,y)} = \{I(i,j) \mid x+k > i \geq x \ , \ y+k > j \geq y\}$, the following distance is calculated:

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

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, 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 Fourier approach becomes even more advantageous. However, as can be seen, actual run times are still far from real time application requirements.

In this paper, we present a novel approach which reduces run times by almost 2 orders of magnitude as shown in Table 1. The approach is based on a projection scheme where tight 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.

This approach can also be used for *Classification* in which projection values are used as features which can be extracted very fast. In recent approaches, the projection scheme coupled with rejection has been studied. These approaches aim to increase classification efficiency by choos-

	Naive	Fourier	New Approach
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 1: A comparison between existing pattern matching approaches and the proposed approach.

ing projection kernels that are optimal with respect to discrimination abilities [4, 1, 2]. 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.

The idea of choosing projection kernels that are fast to apply is not new. In [9] Viola uses 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 an integral image scheme [3]. 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 kernels are restricted to those applicable in the integral image scheme. In some classification problems, however, such kernels can produce poor results as they may form non-informative feature inputs. In our method, this behavior is avoided since the projection kernels form a complete representation of the input.

2. Bounding the Euclidean Distance

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

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

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

$$d_E^2(\mathbf{p},\mathbf{w})\geq b^2/(\mathbf{u}^T\mathbf{u}) \tag{3}$$

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:

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

where,

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

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. Note, that if the projection vectors are orthonormal, the lower bound reduces to $\mathbf{b}^T\mathbf{b}$.

An iterative scheme for calculating the lower bound is also possible and is elaborated in [6] together with additional extensions and proofs relating to this section.

3. Finding Efficient Projection Vectors

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 is 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 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 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.

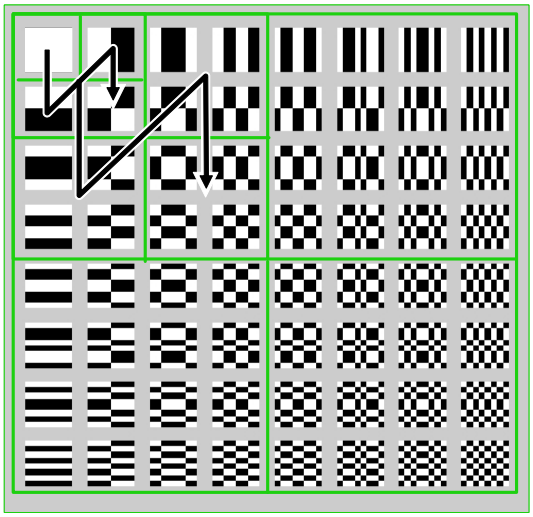


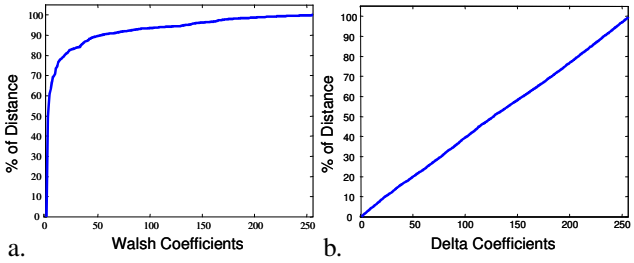
Figure 1: The projection vectors of the Walsh-Hadamard transform of order $n = 8$ ordered with increasing spatial frequency. White represents the value 1, and black represents the value -1. A diadic ordering of these basis vectors is shown by the overlayed arrows.

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. Additionally, a technique is presented here, to calculate the projection values very efficiently.

4. The Windowed Walsh-Hadamard Basis Vector for Pattern Matching

The Walsh-Hadamard (WH) transform has long been used for image representation under numerous applications. The elements of the (non-normalized) basis vectors contain only binary values (± 1). Thus, computation of the transform requires only integer additions and subtractions. The WH transform of an image window of size $k \times k$ ($k = 2^m$) is obtained by projecting the window onto the k^2 WH basis vectors [8]. In our case, it is required to project *each* $k \times k$ window of the image onto the vectors. This results in a highly over-complete image representation, with $k^2 n^2$ projection values for the entire image.

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



a. b.

Figure 2: 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 100 16×16 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).

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 [7]. In terms of pattern matching, the lower bounds on distances between window and pattern, are shown to be tight after very few projections. Figure 2a 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% of the distance. For comparison, Figure 2b shows the same lower bound when using the standard basis for projection (delta functions), i.e. when calculating the Euclidean distance by accumulating squared 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 WH vectors is introduced in the following section and is the essence of the pattern matching scheme presented in this paper.

4.1 The Walsh-Hadamard Tree Structure

We introduce a novel scheme for computing the projection of all image windows onto WH basis vectors (this may also be considered as the *Windowed WH Transform*). 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. Towards this end, we project *each* window of the signal, of length k, onto a set of 1D WH basis vectors. These projections can be computed efficiently using the tree scheme de-

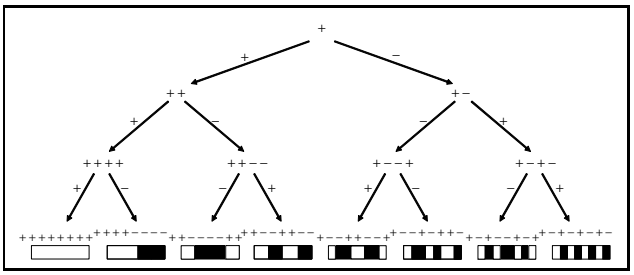


Figure 3: Tree-scheme for computing projections of all signal windows onto all Walsh-Hadamard vectors of order 8.

picted in Figure 3. 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 WH basis vector. The symbols $+$ and $-$ represent the operations performed at a 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 x of the signal, the value of entry $x+\Delta$ 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. The operations $+/-$ on the tree branches were designed to create projections onto WH vectors ordered in increasing diadic sequency (as shown in Figure 1).

Computations within the tree are typically performed by descending from a node in the tree to its child. The following lists 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 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 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 2 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 WH tree is traversed in pre-order. 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 pre-order tree traversal. Thus, projecting all windows onto *all* projection vectors requires only $2k-2$ operations per window (about 2 operations per window per projection vector, independently 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.

4.2. Using The Walsh-Hadamard Tree Structure for Pattern Matching

The pattern matching approach we propose, uses the WH tree structure and follows the projection schemes 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, complexity and run time of the pattern matching process can be significantly 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, a *single* branch of the WH tree is maintained. This branch includes all nodes from the root to the current leaf. The algorithm proceeds as follows:

The Pattern Matching Algorithm:

Assume the sought pattern $\mathbf{p}$ is projected (in a preprocessing stage) onto the set of k WH 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 $\{\mathbf{w}_x\}$ onto the first WH projection vector $\mathbf{u}_1$: $\hat{w}_x^1 = \mathbf{u}_1^T\mathbf{w}_x$

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. 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 vectors 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 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 4.1). In practice, when only 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 [6].

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. 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. Practically, 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.

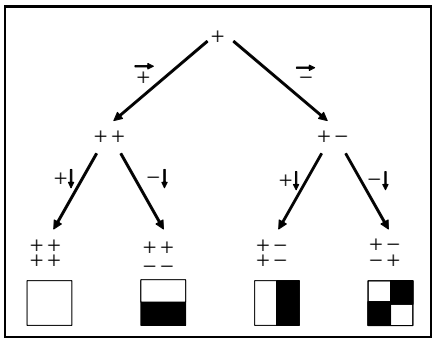


Figure 4: The tree-scheme for computing projections of all 2D image windows onto all 2×2 Walsh-Hadamard kernels.

Efficiency of computation is due to three main factors:

1. Using the recursive structure of the WH Tree, Calculations applied to one window are exploited when computing neighboring windows.

2. Using the structure of the Walsh-Hadamard Tree, calculations applied to one vector, are exploited in the computation of the next vector.

3. As discussed above, the first few WH 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. Traditional approaches, performing in-signal computations require memory on the order of n (signal size). The proposed approach maintains a branch of the WH 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.

4.3. 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 WH kernels[1] using a WH 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. Figure 4 depicts such a tree for the WH transform of order 2×2. Each node represents an $n \times n$ image of intermediate values used in the computation. 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 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 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

[1] The notion 'kernels' in 2D replaces the notion 'vectors' in 1D

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 (Figure 1). The order of kernels is determined by the operations (+ or -) assigned to each edge. The diadically increasing order is obtained as follows: Let $s = (+-+--+-+)$ be a seed sequence, and $S = \{s\}^*$ an arbitrary number of repetitions of s. At each level of the tree, scanning the operations applied at each node, from left to right, yields a sequence equal to a prefix string of S. From this rule it is straightforward to calculate the operation that is to be applied at any given tree level for any given branch.

Extensions to higher dimensional data (e.g. for spatio-temporal pattern matching in video sequences) is straightforward and can be performed in a similar manner.

5. Experimental Results

The proposed scheme was tested on real images and patterns. The results show that the suggested approach reduces run time by almost 2 orders of magnitude compared to the naive and FFT approaches.

As an example, Figure 5 shows an original 256×256 image and the 16×16 pattern that is to be matched. Figures 6a-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 WH kernel were calculated. All windows with projection values above a given threshold were rejected. After the first projection, only 602 candidate windows remained (Figure 6a), i.e. only 0.92% of the original windows. Following the second projection only 8 candidate windows remained (Figure 6b) and following the third projection a single window containing the pattern remained (Figure 6c).

Figure 5: Inputs for the pattern matching example. a. Original 256×256 image. b. 16×16 pattern shown at large scale.

The performance shown in this example is typical and is attained over many such examples. Figure 7 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 per pixel required to complete the process was calculated and found to equal 8.0154 which is slightly over $2\log_2(16)$ as expected (see Section 4.3).

Run time 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 64×64. The experiments were performed on a PIII processor, 1.8 GHz. The average number of operations and run times are summarized in Table 1. It can be seen that the suggested approach is advantageous over the traditional approaches especially for small pattern size. For larger pattern sizes run time increases, but still maintains a speedup of orders of magnitude over the two other methods.

5.1. Robustness of the Approach

Noise

Appearances of a pattern in an image, may vary due to noise, quantization, digitization and transformation errors. In the following experiment an original 256×256 image was embedded with 10 occurrences of a 16×16 pattern. Various levels of noise were added to the image. Figure 10a shows a noisy version of the image with noise level of 40 (i.e. each pixel value in the image was increased/decreased by a random amount uniformly chosen in $[-40 \ldots 40]$). All 10 patterns were detected even under these very noisy conditions. Under these conditions a

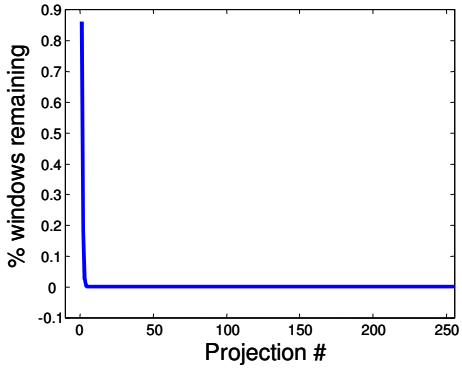


Figure 7: Percentage of image windows remaining after each projection for images of size 256×256 and patterns of size 16×16. Results are averaged over 100 image-pattern pairs.

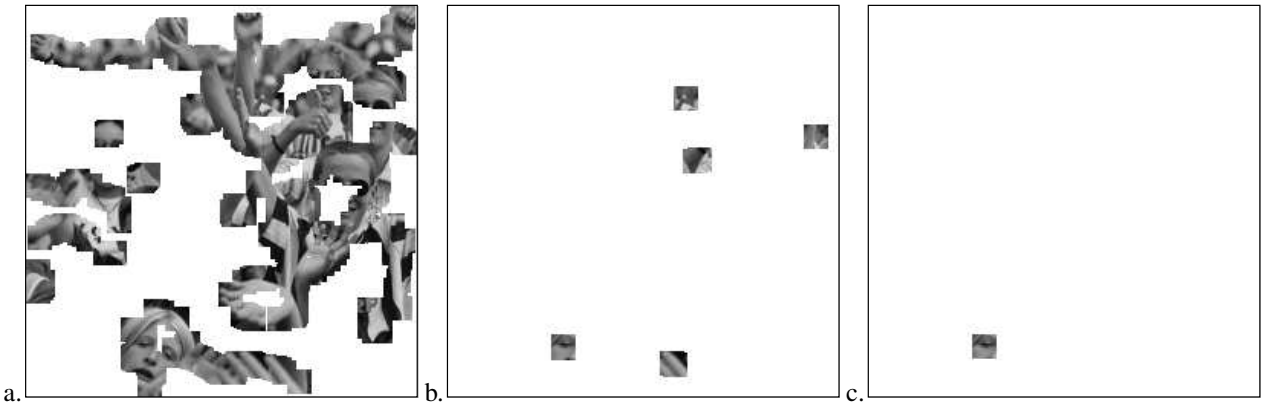


Figure 6: Three steps in the pattern matching process. a. Following the first projection only 602 candidate windows remain (0.92% of the windows). b. Following the second projection only 8 candidate windows remain. c. After the third projection a single window remains corresponding to the correct pattern location.

larger threshold for rejection is required. This implies that fewer image windows are rejected after each projection and that the overall number of required projections increases. Figure 8 shows this behavior where the minimum number of projections required to find all noisy patterns is shown as a function of the noise level. In all cases, the threshold used is the minimum value that produces no miss-detections. Although the number of required projections increases with noise level, efficiency of the pattern matching process is maintained as shown in Figure 9. The Figure shows the percentage of image windows remaining after each projection for noise levels varying between 0 and 40. As the noise level increases, fewer windows are rejected after each projection, however the decreasing profile for the various thresholds is similar (Figure 9 inset shows a scaled version of the values for the first few projections). The average number of operations per pixel was calculated and at maximal noise level was found to be 10.0623 which is only slightly higher than the value obtained for non-noisy images of the same size (see above).

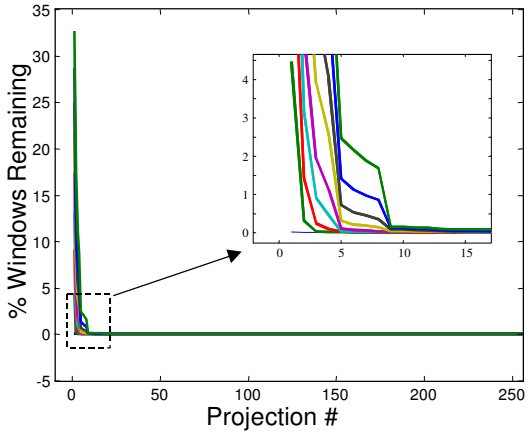


Figure 9: Percentage of image windows remaining after each projection for images at various noise levels (e.g. Figure 10a). Image size is 256×256 and pattern size is 16×16.

Invariance to Illumination

The WH tree has the additional advantage that it easily enables pattern matching that disregards the DC values.

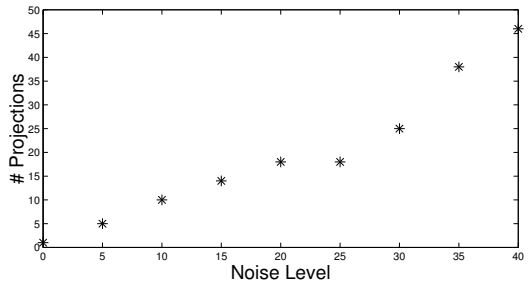


Figure 8: The minimum number of projections required to find all patterns in an image as a function of the noise level.

This can be used to compensate for illumination variations. The DC value of all windows is given by their projection onto the first WH kernel. In order to perform DC invariant pattern matching, the first branch of the WH tree (Section 4.1) may be skipped and the process initialized by computing the projections onto the second kernel. The rest of the process continues as before. As an example, in Figure 10b an illumination gradient was added to the original image in which 10 occurrences of the pattern were embedded. The illumination gradient produces large variance in the appearance of the pattern in the image (Figure 10b bottom). Using the original approach the patterns were not detected (even using a very high threshold only 2 of the occurrences were detected and many false alarms resulted as well). Performing DC invariant pattern matching, all 10 occurrences of the pattern were detected using a small threshold and only 5 projections (Figure 10c).

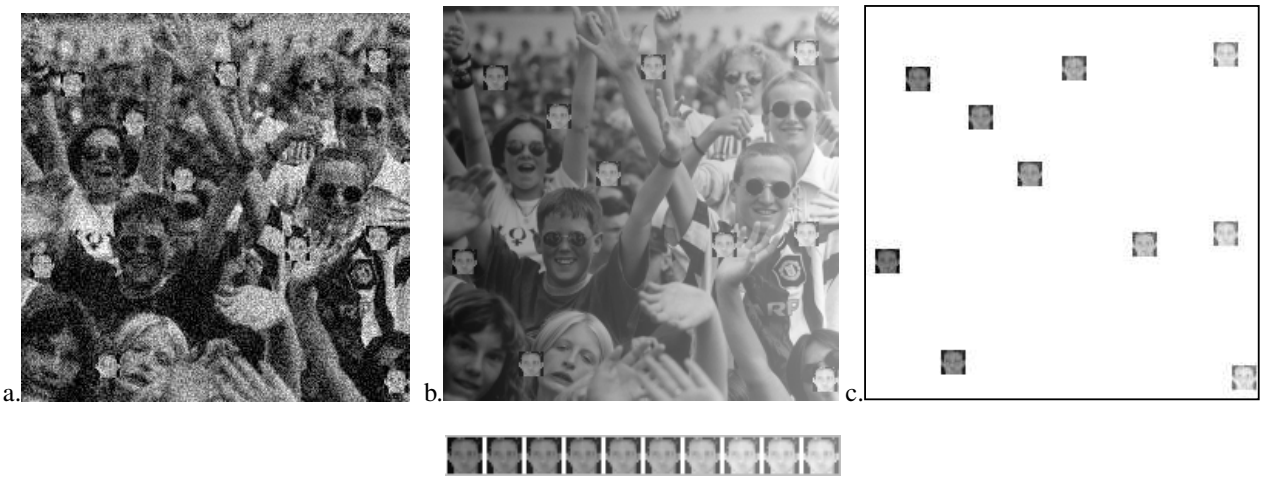


Figure 10: An original image was embedded with 10 occurrences of a pattern. a) Noise was added to the image at a level of 40 (see text). b. A gradient of illumination was added to the image. The 10 patterns as appearing in the image are shown below. c. All 10 patterns of varying illuminations were found. Five projections were required to find all 10 patterns.

6. Conclusion

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 traditional schemes first and foremost due to the reduction in time complexity of 2 orders of magnitude. The efficiency due to the projection technique suggested in this paper, is amplified by the fact that all computations of the transform coefficients are performed using integer additions and subtractions. In addition to the ability of the suggested method to cope with various illuminations, it also allows multi-scale pattern matching, with patterns scaled by powers of 2. This can be performed at almost no extra cost since the appropriate WH kernels for these scaled patterns are already given at the intermediate tree nodes. The scheme was described for 1D and 2D data however extensions to higher dimensional data (including image sequences) is straightforward.

The projection approach described here, independent of the rejection scheme, can be used also in classification systems where the projection values serve as input features that are informative and can be extracted in real-time. However, 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 is the classification technique we showed a method that improves run-time and efficiency over other known techniques.

Limitations of the suggested approach includes the fact that memory of size $2n^2 \log k$ is required during the process and that pattern sizes are limited to powers of 2.

Finally, although the Pattern Matching approach described in this paper, was developed for the Euclidean norm, extensions to other norms are straightforward. The method has already been extended to normalized correlation scheme, which allows matching of patterns of any size by spatial masking.

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

References

[1] S. Baker and S.K. Nayar. Pattern rejection. In *IEEE-CVPR - San Francisco, CA, USA, 18-20 June*, 1996.

[2] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

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

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

[5] 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.

[6] Y. Hel-Or and H. Hel-Or. Real time pattern matching using projection kernels. Technical Report CS-2002-1, Interdisciplinary Center Technical Report, 2002.

[7] H. Kitajima. Energy packing efficiency of the hadamard transform. *IEEE T-Comm.*, pages 1256–1258, 1976.

[8] D. Sundararajan and M.O. Ahmad. Fast computation of the discrete walsh and hadamard transforms. *IEEE T-IP*, 7(6):898–904, 1998.

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