This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2629621, IEEE Transactions on Circuits and Systems for Video Technology

Fast Full-Search Equivalent Pattern Matching Using Asymmetric Haar Wavelet Packets

Wanli Ouyang*, *Member, IEEE,* Tianle Zhao*, Wai-kuen Cham, *Senior Member, IEEE* and Liying Wei

***Abstract*—Pattern matching is widely used in signal processing, computer vision, image and video processing. One efficient approach is to perform pattern matching in a transform domain that has good energy packing ability and so allows early rejection of most mismatched candidates. Calculating the transforms of pixels in sliding windows requires much computation, and so fast algorithms are employed. Existing methods require $O(u)$ additions per pixel for projecting input pixels onto u 2D basis vectors. In this paper, we propose a new 2D transform, called asymmetric 2D Haar transform (A2DHT), and extend it to wavelet packets that contain exponentially large number of bases. A basis selection algorithm is then proposed to search for the optimal basis in the wavelet packets. A fast algorithm is also developed which can compute u projection coefficients with only $O(\log u)$ additions per pixel. Results of experiments show that the proposed fast algorithm and the proposed transform can significantly accelerate the full-search equivalent pattern matching process and outperform state-of-the-art methods.**

***Index Terms*—Fast algorithm, pattern matching, Haar wavelet, wavelet packets.**

I. Introduction

PATTERN matching, also called template matching, is the procedure of seeking a required pattern or template in a given signal. For pattern matching in images, this procedure is illustrated by Fig. 1. Pattern matching has applications in manufacturing for quality control [1], image based rendering [2], image compression [3], object detection [4], super resolution [5], texture synthesis [6], block matching in motion estimation [7], [8], image denoising [9], [10], [11], road/path tracking [12], mouth tracking [13], image matching [14], action recognition [15] and tone mapping [16].

Since pattern matching is a time-consuming task, many fast algorithms have been proposed. These fast algorithms can be divided into full-search-equivalent algorithms and full-search-nonequivalent algorithms. *Full-search-equivalent* algorithms guarantee to obtain the same result as that of a brute-force full search, which examines all possible candidates, while *full-search-nonequivalent* algorithms do not have this guarantee. *Full-search-equivalent* algorithms run much faster than a brute-force full search. The algorithms proposed in [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27] and [28] are FS-equivalent algorithms. *Full-search-nonequivalent* algorithms generally run even faster. The algorithms proposed in [29], [30], [31], [32], [33], [34] and [35] are FS-nonequivalent algorithms. The scope of this paper is limited to the study of full-search-equivalent algorithms.

In this paper, we use a new asymmetric two-dimensional Haar transform (A2DHT). Since the sums of pixel values within rectangular regions of an image are buildings block of this transform, a method called fast strip sum is proposed which computes each of the sums by one addition. With the fast strip sum method, we develop an algorithm that computes u A2DHT coefficients by $O(\log u)$ additions per pixel on sliding windows. A FS-equivalent pattern matching algorithm in the transform domain is then developed, which allows early rejection of most mismatched candidates and hence runs very fast. Besides, we extend the A2DHT to wavelet packets that contain exponentially large number of bases. Calculating transforms specified by these bases has almost the same computationally complexity as that of the original A2DHT. Further more, we propose a selection algorithm to choose the transform that best fits the pattern matching task. Results of experiments show that pattern matching using the proposed algorithm is faster than existing FS-equivalent algorithms, and for pattern matching of images of natural scenes the A2DHT is optimal among all bases contained in the wavelet packets mentioned above.

The main contributions of this paper include: 1) the proposal of the A2DHT and its extension to wavelet packet bases; 2) the development of the fast strip sum method and the corresponding buffering strategy that are used to calculate the sums of pixel values within rectangular regions of an image; 3) the proposal of an algorithm that requires $O(\log u)$ additions per pixel to compute u A2DHT coefficients on sliding windows; 4) the proposal of a very fast FS-equivalent algorithm in the transform domain; and 5) the proposal of a basis selection algorithm, which when applied to natural scene images proves the optimality of the A2DHT.

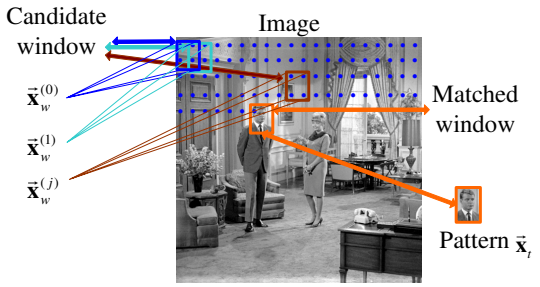


Fig. 1. Pattern matching in image 'couple'. We represent the pattern by vector $\vec{\mathbf{x}}_t$ and represent the candidate windows by vectors $\vec{\mathbf{x}}_w^{(0)}$, $\vec{\mathbf{x}}_w^{(1)}, \ldots, \vec{\mathbf{x}}_w^{(j)}, \ldots, \vec{\mathbf{x}}_w^{(W-1)}$.

The authors are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong.

E-mail: {wlouyang,tlzhao,wkcham}@ee.cuhk.edu.hk, weiliying101@gmail.com.

* indicates equal contribution.

Manuscript received ...

The paper is organized as follows. Section II introduces some related works and defines rectangle sum, integral image as well as the pattern matching problem. Section III discusses in detail the strip sum method and analyses the computation and memory required by the strip sum method when computing rectangle sums. Section IV proposes the A2DHT and analyses its computational complexity. In the same section, we show in detail the extension of the A2DHT to wavelet packet bases as well as the selection algorithm that can efficiently search the wavelet packet bases. Section V gives experimental results. Finally, Section VI draws conclusions.

II. Related work

In this section, we introduce existing techniques and concepts commonly used by many fast algorithms, which include integral image and rectangle sum techniques [36], [33], [37], as well as problem settings and basic ideas of transform domain pattern matching. These techniques and concepts are preliminaries to understand the algorithms in Section III and Section IV.

A. Integral Image and Rectangle Sum

Denote by $x(j_1, j_2)$ the pixel value at position (j_1, j_2) of a $J_1 \times J_2$ image, where $0 \le j_1 < J_1$ and $0 \le j_2 < J_2$. The integral image $ii(j_1, j_2)$ is defined as,

$$ii(j_1, j_2) = \sum_{u=0}^{j_1-1} \sum_{v=0}^{j_2-1} x(u, v). \tag{1}$$

A rectangle in an image is specified by,

$$rect = (j_1, j_2, N_1, N_2), \tag{2}$$

where (j_1, j_2) is the coordinate of the upper left corner of the rectangle in the image, N_1 and N_2 denote the width and height of the rectangle, respectively, $0 \le j_1 < J_1 - N_1$, $0 \le j_2 < J_2 - N_2$, and $N_1, N_2 > 0$. See Fig. 2 for a graphical illustration. The sum of pixel values within a rectangle $rect$ is called rectangle sum, denoted by $rs(rect)$, in this paper.

Viola and Jones found in [36] that the integral image can be computed at the cost of 2 additions per pixel. Moreover, given the integral image, the rectangle sum $rs(rect)$ can be computed at the cost of 3 additions per pixel as follow,

$$\begin{aligned} rs(rect) &= \sum_{u=j_1}^{j_1+N_1-1} \sum_{v=j_2}^{j_2+N_2-1} x(u, v) \\ &= ii(j_1 + N_1, j_2 + N_2) + ii(j_1, j_2) \\ &\quad - ii(j_1, j_2 + N_2) - ii(j_1 + N_1, j_2). \end{aligned} \tag{3}$$

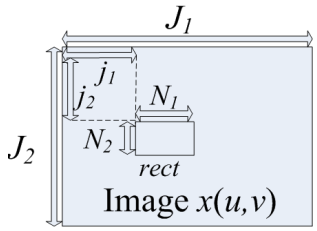


Fig. 2. An example of a rectangle $rect = (j_1, j_2, N_1, N_2)$ in a J_1 by J_2 image $x(u, v)$, where (j_1, j_2) is the coordinate of the upper left corner, N_1 and N_2 are the width and height of the rectangle, respectively.

Note that one subtraction is considered to be one addition regarding the computational complexity in this paper.

Rectangle sums and Haar-like features have been widely used in many applications such as object detection [36], [38], [39], [40], [41], object classification [42], pattern matching [24], [23], [33], feature point based image matching [43] and texture mapping [44]. Porikli utilized the integral image method to design a fast algorithm for computing the histogram in [45]. Since the work in [44], it has been considered that at least 3 additions are required to obtain the rectangle sum using the summed area table in [44], [37] or the integral image method in [36]. The authors of [28] proposed the image square sum technique, which is specifically designed for the A2DHT to accelerate the calculation of the projection values. Calculation and caching of these square sums require lots of time and memory. The fast pattern matching algorithm proposed in [28] is in some cases comparable to the fast pattern matching algorithm proposed in this paper.

B. Transform Domain Pattern Matching

Suppose an $N_1 \times N_2$ pattern is to be sought in a given image as shown in Fig. 1. The pattern will be compared with candidate windows of the same size in the image. Denote by W the number of candidate windows. We represent a pattern as a vector $\vec{\mathbf{x}}_t$ of length N and represent jth candidate windows as $\vec{\mathbf{x}}_w^{(j)}$, where $j = 0, 1, \ldots, W-1$ and $N = N_1 N_2$. Subscripts $\cdot_t$ and $\cdot_w$ denote pattern and window, respectively. For example, if a 32×32 pattern is sought in a 256×256 image, we have $N = 1024$ and $W = (256 - 32 + 1)^2 = 50625$. The distance between $\vec{\mathbf{x}}_t$ and $\vec{\mathbf{x}}_w^{(j)}$, which is denoted by $d(\vec{\mathbf{x}}_t, \vec{\mathbf{x}}_w^{(j)})$, is a measure of the dissimilarity between $\vec{\mathbf{x}}_t$ and $\vec{\mathbf{x}}_w^{(j)}$. The smaller is $d(\vec{\mathbf{x}}_t, \vec{\mathbf{x}}_w^{(j)})$, the more similar are $\vec{\mathbf{x}}_t$ and $\vec{\mathbf{x}}_w^{(j)}$.

There are two widely used goals for FS in pattern matching:
Goal 1. find all candidate windows with $d(\vec{\mathbf{x}}_t, \vec{\mathbf{x}}_w^{(j)}) < T$ for a given threshold T;
Goal 2. find the window that has the minimum $d(\vec{\mathbf{x}}_t, \vec{\mathbf{x}}_w^{(j)})$ among all candidate windows.
Goal 1 is adopted by [19], [23], [20] and [24]. Goal 2 is adopted by [25], [27], [46], [47] and in the source code of [19][1]. Note that algorithms adopting Goal 1 can be modified to deal with Goal 2 using the approaches proposed in [25], [19], [27]. In this paper, we focus on Goal 1.

Most transform domain pattern matching algorithms use sum of squared differences (SSD) as distance measure [19], [20], [21]. As pointed out in [19], although there are arguments against SSD as a dissimilarity measure for images, it is still widely used due to its simplicity. Discussions on SSD as a dissimilarity measure can be found in [48], [49], [50].

A transform that projects a vector $\vec{\mathbf{x}} \in \mathbb{R}^N$ onto a linear subspace spanned by $U(\ll N)$ basis vectors $\vec{\mathbf{v}}^{(0)}, \ldots \vec{\mathbf{v}}^{(U-1)}$ can be represented as follows:

$$\vec{\mathbf{y}} = \mathbf{V}\vec{\mathbf{x}} = [\vec{\mathbf{v}}^{(0)} \ \ldots \ \vec{\mathbf{v}}^{(U-1)}]^T \vec{\mathbf{x}}, \tag{4}$$

where $\cdot^T$ denotes matrix transposition, vector $\vec{\mathbf{x}}$ of length N is called input window, vector $\vec{\mathbf{y}}$ of length U is called projection

[1] Available on http://www.faculty.idc.ac.il/toky/software/software.htm.

TABLE I
TRANSFORM DOMAIN PATTERN MATCHING

Overall procedure:
Initialise S_w as the set of all candidate windows $\vec{\mathbf{x}}_w^{(j)}$;
For $u = 1$ to U_M:{
  For $\vec{\mathbf{x}}_w^{(j)}$ in S_w: {
    Project $\vec{\mathbf{x}}_t$ and $\vec{\mathbf{x}}_w^{(j)}$ to obtain $\mathbf{V}\vec{\mathbf{x}}_t$ and $\mathbf{V}\vec{\mathbf{x}}_w^{(j)}$, respectively;
    Remove $\vec{\mathbf{x}}_w^{(j)}$ from S_w, if $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2 > T$;
  }
}
The candidate windows remained in S_w undergo full search.

value vector, the U elements in vector $\vec{\mathbf{y}}$ are called projection values and $\mathbf{V}$ is a $U \times N$ matrix of which the rows are the U orthogonal basis vectors $\vec{\mathbf{v}}^{(i)}$ of length N, $i = 0, \ldots, U-1$. In [19], [51], the transform is called projection. Basis vector is called projection kernel in [19] and called filter kernel in [20]. In the followings, we show how to use the transform in pattern matching tasks.

As proved in [19], the following inequality holds when the u basis vectors in $\mathbf{V}$ are orthonormal,

$$||\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w^{(j)}||^2 \geq ||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2. \quad (5)$$

Thus $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2$ is the lower bound of $||\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w^{(j)}||^2$. Denote the set of candidates as S_w, which initially contains all candidates. According to (5), if $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2 > T$, then $||\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w^{(j)}||^2 > T$, and we can safely remove candidate window $\vec{\mathbf{x}}_w^{(j)}$ from S_w. Therefore, $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2 > T$ acts as the rejection condition for rejecting mismatching windows.

The procedure of transform domain pattern matching is summarized in Table I. At each iteration of u, where u increases from 1, $\vec{\mathbf{x}}_t$ and the remaining candidates $\vec{\mathbf{x}}_w^{(j)}$ in S_w are projected onto transform domain using $\mathbf{V}$ and the rejection condition $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2 > T$ is checked. Finally, when u reaches a sufficiently large number which is denoted as U_M, the iteration terminates and then the remaining candidate windows in S_w undergo FS to find out the matched windows. Since the matched windows will not be rejected by the rejection condition and will be found by FS, such pattern matching approach is FS-equivalent. The u basis vectors $\vec{\mathbf{v}}^{(0)}, \ldots, \vec{\mathbf{v}}^{(u-1)}$ in $\mathbf{V}$ are selected from the U orthonormal vectors $\vec{\mathbf{v}}^{(0)} \ \ldots \ \vec{\mathbf{v}}^{(U-1)}$. For example, the pattern matching algorithm in [8] selects the u Walsh Hadamard Transform (WHT) basis vectors having the lowest frequencies.

According to [19], pattern matching using WHT as the transform is almost two orders of magnitude faster than FS. Besides, with proper modifications, transform domain pattern matching are able to deal with distortions such as illumination changes, contrast changes and geometric distortions such as rotations and small affine coordinate transforms [14]. Because of these advantages, transform domain pattern matching has been used for block matching in motion estimation for video coding [7], [8], road/path tracking [12], wide baseline image matching [14], texture synthesis [52] and augmented reality [53].

Hel-Or and Hel-Or's algorithm in [19] requires $2N - 2$ additions per pixel to compute all WHT projection values in each window of size N. In the worst case, however, their algorithm requires $O(\log N)$ additions per pixel to obtain one projection value. Following this work, the Gray-Code Kernel (GCK) algorithm proposed in [20] requires $2u$ additions per pixel to obtain u projection values. Later we developed a fast algorithm that requires about $3u/2 + 1$ additions per pixel to obtain u projection values [21]. However, the computational complexity of these algorithms are $O(u)$ per pixel. In this paper, we propose a new transform whose computational complexity is $O(\log u)$ per pixel.

A FS nonequivalent pattern matching algorithm using nonorthogonal Haar-like features is proposed in [35]. However, [20] shows that WHT is better than the nonorthogonal Haar-like features in pattern matching. In this paper, we propose an orthogonal Haar-like transform A2DHT for FS-equivalent pattern matching.

III. THE FAST ALGORITHM FOR COMPUTING RECTANGLE SUM

In this section, the fast algorithm for computing rectangle sum is introduced. Rectangle sum is a building block of the Haar-like transform proposed in Sec. IV. Analysis of its computational complexity and memory requirement provided in this section shows that using this fast algorithm can reduce the computational complexity of the matching process.

A. Computation of Rectangle Sum by Strip Sum

Define horizontal strip sum $hss(j_1, j_2, N_2)$ as:

$$hss(j_1, j_2, N_2) = \sum_{u=0}^{j_1-1} \sum_{v=j_2}^{j_2+N_2-1} x(u, v). \quad (6)$$

Hence, hss is represented by the upper right corner (j_1-1, j_2) and height N_2. Fig. 3 shows the $hss(j_1 + N_1, j_2, N_2)$ and $hss(j_1, j_2, N_2)$. $hss(j_1, j_2, N_2)$ can be obtained by one addition per pixel as follows using the integral image $ii(j_1, j_2)$:

$$\begin{aligned} hss(j_1, j_2, N_2) &= \sum_{u=0}^{j_1-1} \sum_{v=0}^{j_2+N_2-1} x(u, v) - \sum_{u=0}^{j_1-1} \sum_{v=0}^{j_2-1} x(u, v) \\ &= ii(j_1, j_2 + N_2) - ii(j_1, j_2). \end{aligned} \quad (7)$$

We can compute rectangle sums $rs(j_1, j_2, N_1, N_2)$ and $rs(j_1, j_2, N_1', N_2)$ $(N_1 \neq N_1')$ from strip sums as follows using (3) and (7):

$$\begin{aligned} &rs(j_1, j_2, N_1, N_2) \\ &= hss(j_1 + N_1, j_2, N_2) - hss(j_1, j_2, N_2), \\ &rs(j_1, j_2, N_1', N_2) \\ &= hss(j_1 + N_1', j_2, N_2) - hss(j_1, j_2, N_2). \end{aligned} \quad (8)$$

As shown in (8), only one addition is required to compute the rectangle sum $rs(j_1, j_2, N_1, N_2)$ from the strip sums $hss(j_1 + N_1, j_2, N_2)$ and $hss(j_1, j_2, N_2)$. Fig. 3 illustrates the computation.

The $rs(j_1, j_2, N_1, N_2)$ and $rs(j_1, j_2, N_1', N_2)$ in (8) have the same height N_2 but have different width $(N_1 \neq N_1')$. They both use the $hss(j_1, j_2, N_2)$ for the computation. Fig. 4 shows the relationship between $rs(j_1, j_2, N_1, N_2)$ and $rs(j_1, j_2, N_1', N_2)$. Thus one strip sum $hss(j_1, j_2, N_2)$ is utilized for the computation of two rectangle sums of sizes $N_1 \times N_2$ and $N_1' \times N_2$ in (8). In general, if $hss(j_1, j_2, N_2)$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2629621, IEEE Transactions on Circuits and Systems for Video Technology

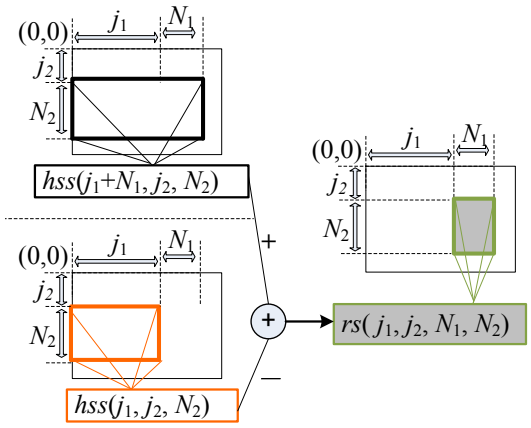


Fig. 3. Horizontal strip sum hss and rectangle sum rs on the image. Only one addition is required to compute $rs(j_1, j_2, N_1, N_2)$ from the two hss using (8).

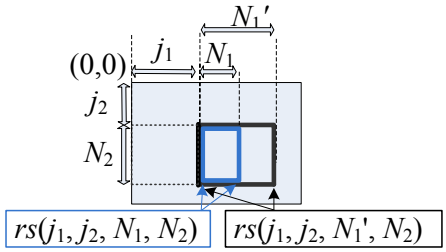


Fig. 4. Rectangle sums sharing the same height N_2. The two rectangle sums can use the same strip sum for computation.

TABLE II
PSEUDO-CODE SHOWING THE UTILIZATION OF STRIP SUM FOR COMPUTING RECTANGLE SUMS SHARING THE SAME HEIGHT N_2.

1. $Temp_{hss}$ is a 1-D array buffer of size J_1. N_2 is a fixed value. j_2 is the row index and j_1 is the column index;
2. **for** j_2 from 0 to $J_2 - 1$,
3. Compute the $hss(j_1, j_2, N_2)$ for $0 \le j_1 \le J_1 - 1$ at the j_2-th row from integral image using (7) and store them into $Temp_{hss}$;
4. **for** j_1 from 0 to $J_1 - 1$,
5. Compute $rs(j_1, j_2, N_1, N_2)$ using (8) for rectangles having any N_1. The $hss(j_1, j_2, N_1, N_2)$ and $hss(j_1 + N_1, j_2, N_2)$ in (8) are stored in $Temp_{hss}$;
6. **end**
7. **end**

TABLE III
THE STEPS AND NUMBER OF OPERATIONS REQUIRED BY STRIP SUM METHOD TO OBTAIN r RECTANGLE SUMS.

1. Prepare the integral image by 2 additions and 4 memory fetch operations (M-ops) per pixel;
2. prepare the $\min\{n_W, n_H\}$ strip sums using (7) or (9) by $\min\{n_W, n_H\}$ additions and $2\min\{n_W, n_H\}$ M-ops per pixel;
3. obtain the r rectangle sum from strip sum using (8) or (9) by r additions and $2r$ M-ops per pixel.

has been provided on all pixel locations (j_1, j_2), we can use it to obtain rectangle sums having any width N_1 and the fixed height N_2 at any pixel position by one addition. This procedure of using strip sums for computing a rectangle sum is shown in Table II.

hss is used for computing rectangle sums having the same height N_2 in (8). Similarly, strip sum can be used for computing rectangle sums having the same width N_1. And we can use the vertical strip sum vss for computing rectangle sums having any height N_2 and fixed width N_1 as follows:

$$\begin{aligned} &rs(j_1, j_2, N_1, N_2) = vss(j_1, j_2 + N_2, N_1) - vss(j_1, j_2, N_1), \\ &\text{where } vss(j_1, j_2, N_1) = \sum_{u=j_1}^{j_1+N_1-1} \sum_{v=0}^{j_2-1} x(u, v) \\ &\qquad\qquad = ii(j_1 + N_1, j_2) - ii(j_1, j_2). \end{aligned} \tag{9}$$

B. Computational Complexity Analysis

The algorithm and the overall computation required by strip sum for computing one rectangle sum are summarised as follows:

1) prepare the integral image by two additions per pixel;
2) prepare the strip sum using (7) by one addition per pixel;
3) obtain the rectangle sum from strip sum using (8) by one addition per pixel.

Suppose r rectangle sums of different sizes are computed on each pixel position. Let these r rectangle sums have n_W different widths and n_H different heights. As illustrated in Table II, one horizontal strip sum is required for computing rectangle sums sharing the same height. Hence, n_H horizontal strip sums are required for computing r rectangle sums having n_H different heights. Alternatively, we can compute these r rectangle sums using n_W vertical strip sums. Therefore, $\min\{n_W, n_H\}$ strip sums are required for computing these r rectangle sums. The overall algorithm and the computation required for computing r rectangle sums is shown in Table III. In summary, the strip sum method requires $2+\min\{n_W, n_H\}+r$ additions and $4 + 2\min\{n_W, n_H\} + 2r$ memory fetch operations (M-ops) per pixel for computing r rectangle sums.

The box filtering technique in [54] requires $4r$ additions and $6r$ M-ops per pixel for computing r rectangle sums. The integral image method in [36] requires $2 + 3r$ additions and $4 + 4r$ M-ops per pixel, where 2 additions and 4 M-ops are used for preparing the integral image, $3r$ additions and $4r$ M-ops are used for computing r rectangle sums from integral image using (3). The computational complexity of these methods is summarized in Table IV. As a solid example, suppose we want to calculate the 4×4 A2DHT of an $N_1 \times N_2$ image, then $\min\{n_W, n_H\} = 3$. There are around $5N_1N_2$ rectangles of 5 different sizes. To calculate these rectangle sums, the box filtering technique requires $20N_1N_2$ additions, the integral image method requires $17N_1N_2$ additions, and the proposed strip-sum technique requires only $10N_1N_2$ additions. Therefore, the proposed strip-sum technique requires only around 59% of the additions required by the integral image technique.

C. Buffering Strip Sum

When r rectangle sums having n_H different heights are computed for a $J_1 \times J_2$ image, memory of size $J_1J_2n_H$ will

TABLE IV
THE ADDITIONS (*adds*) AND MEMORY FETCH OPERATIONS (*M-op*) PER PIXEL REQUIRED FOR COMPUTING r RECTANGLE SUMS HAVING n_H DIFFERENT HEIGHTS AND n_W DIFFERENT WIDTHS.

	Box	Integral Image	Strip sum
Adds	$4r$	$2+3r$	$2+\min\{n_W,n_H\}+r$
M-ops	$6r$	$4+4r$	$4+2\min\{n_W,n_H\}+2r$

be required if all of the n_H horizontal strip sums are stored in memory. Actually, buffering strategy can be used to reduce the memory requirement. In the following, we propose a buffering strategy for computing rectangle sums having one height. And then we generalize this proposed strategy for computing r rectangle sums having n_H different heights.

We use the computation of rectangle sums of the same height N_2 but different widths as the example to illustrate the buffering strategy. The procedure of memory usage is described in Table II. When we use (8) to compute the rectangle sums $rs(j_1,0,N_1,N_2)$ of any width N_1 at row $j_2=0$ for any j_1, we only need the strip sums $hss(j_1,0,N_2)$ and $hss(j_1+N_1,0,N_2)$ at row 0, but not the strip sums at other rows. Thus we can compute the strip sums at row 0 and store them using a buffer $Temp_{hss}$ of size J_1, i.e. Step 3 in Table II. As the row index j_2 increases from 0 to 1, the strip sums at row 0 is no longer required. We can reuse the buffer $Temp_{hss}$ to store the strip sums at row 1, which is then used for computing rectangle sums at row 1. Therefore, the same buffer $Temp_{hss}$ can be used to store strip sums as the row index j_2 increases from 0 to J_2-1.

In general, if r rectangle sums of n_H different heights are computed, we need n_H buffers and memory of size $J_1 n_H$. As $n_H \le J_2$, the memory required by the strip sum using above buffering strategy will not exceed $J_1 J_2$. The integral image technique utilizes a buffering strategy and stores intermediate results in the calculation of projection coefficients and hence accelerates the algorithm. The proposed buffering strategy is extended directly from the idea of integral image. The proposed buffering strategy saves the memory required by the strip sum for computing the rectangle sum.

IV. THE ASYMMETRIC 2D HAAR TRANSFORM

A. The Haar Transform

The level 1 decomposition for the conventional 2D discrete Haar transform (HT) of 2D input data $\mathbf{X}^{(N)}$ can be represented as follows:

$$\begin{aligned}\mathbf{Y}^{(N)}_{HT,1} &= \mathbf{W}^{(N)}_1 \mathbf{X}^{(N)} \mathbf{W}^{(N)^T}_1 = \left[\frac{\mathbf{H}^{(N)}}{\mathbf{G}^{(N)}}\right]\mathbf{X}^{(N)}\left[\frac{\mathbf{H}^{(N)}}{\mathbf{G}^{(N)}}\right]^T \\ &= \left[\begin{array}{c|c} \mathbf{H}^{(N)}\mathbf{X}^{(N)}\mathbf{H}^{(N)^T} & \mathbf{H}^{(N)}\mathbf{X}^{(N)}\mathbf{G}^{(N)^T} \\ \hline \mathbf{G}^{(N)}\mathbf{X}^{(N)}\mathbf{H}^{(N)^T} & \mathbf{G}^{(N)}\mathbf{X}^{(N)}\mathbf{G}^{(N)^T} \end{array}\right] \\ &= \left[\begin{array}{c|c} \mathbf{A}^{(N/2)} & \mathbf{B}^{(N/2)} \\ \hline \mathbf{C}^{(N/2)} & \mathbf{D}^{(N/2)} \end{array}\right], \\ \mathbf{H}^{(N)} &= \mathbf{I}_{N/2} \otimes [1\ 1], \mathbf{G}^{(N)} = \mathbf{I}_{N/2} \otimes [1\ -1],\end{aligned} \tag{10}$$

where $\mathbf{I}_{N/2}$ denotes the identity matrix of size $N/2 \times N/2$, $\mathbf{H}$ is the averaging matrix and $\mathbf{G}$ is the detail matrix, $\otimes$ denotes the matrix Kronecker product. For example,

$$\mathbf{W}^{(4)}_1 = \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ \hline 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{array}\right] = \left[\frac{\mathbf{H}^{(4)}}{\mathbf{G}^{(4)}}\right]. \tag{11}$$

The level L decomposition for 2D HT is as follows:

$$\mathbf{Y}^{(N)}_{HT,L} = \left[\begin{array}{cc|c} \mathbf{A}^{(N/2^L)} & \mathbf{B}^{(N/2^L)} & \cdots \\ \mathbf{C}^{(N/2^L)} & \mathbf{D}^{(N/2^L)} & \\ \hline \cdots & & \cdots \\ \hline \multicolumn{2}{c|}{\mathbf{C}^{(N/2)}} & \mathbf{D}^{(N/2)} \end{array}\right] \mathbf{B}^{(N/2)},$$

$$\begin{aligned} \text{where } \mathbf{A}^{(N/2^l)} &= \mathbf{H}^{(N/2^{l-1})}\mathbf{A}^{(N/2^{l-1})}\mathbf{H}^{(N/2^{l-1})^T}, \\ \mathbf{B}^{(N/2^l)} &= \mathbf{H}^{(N/2^{l-1})}\mathbf{A}^{(N/2^{l-1})}\mathbf{G}^{(N/2^{l-1})^T}, \\ \mathbf{C}^{(N/2^l)} &= \mathbf{G}^{(N/2^{l-1})}\mathbf{A}^{(N/2^{l-1})}\mathbf{H}^{(N/2^{l-1})^T}, \\ \mathbf{D}^{(N/2^l)} &= \mathbf{G}^{(N/2^{l-1})}\mathbf{A}^{(N/2^{l-1})}\mathbf{G}^{(N/2^{l-1})^T}, \\ l &= 1,\ldots,L, \mathbf{A}^{(N)} = \mathbf{X}^{(N)}. \end{aligned} \tag{12}$$

B. The Proposed Asymmetric 2D Haar Transform

The level 1 decomposition for the proposed A2DHT for 2D input data $\mathbf{X}^{(N)}$ can be represented as follows:

$$\begin{aligned}\mathbf{Y}^{(N)}_{A2DHT,1} &= \left[\begin{array}{c|c} \mathbf{H}^{(N)}\mathbf{X}^{(N)}\mathbf{H}^{(N)^T} & \mathbf{X}^{(N)}\mathbf{G}^{(N)^T} \\ \hline \mathbf{G}^{(N)}\mathbf{X}^{(N)}\mathbf{H}^{(N)^T} & \end{array}\right] \\ &= \left[\begin{array}{c|c} \mathbf{A}^{(N/2)} & \mathbf{E}^{(N/2)} \\ \hline \mathbf{C}^{(N/2)} & \end{array}\right].\end{aligned} \tag{13}$$

Thus, the level L decomposition for A2DHT is:

$$\mathbf{Y}^{(N)}_{A2DHT,L} = \left[\begin{array}{cc|c} \mathbf{A}^{(N/2^L)} & \mathbf{E}^{(N/2^L)} & \cdots \\ \mathbf{C}^{(N/2^L)} & & \\ \hline \cdots & & \\ \hline \multicolumn{2}{c|}{\mathbf{C}^{(N/2)}} & \end{array}\right] \mathbf{E}^{(N/2)},$$

$$\text{where } \mathbf{E}^{(N/2^l)} = \mathbf{A}^{(N/2^{l-1})}\mathbf{G}^{(N/2^{l-1})^T}, \text{for } l = 1,\ldots L. \tag{14}$$

$\mathbf{A}^{(N/2^l)}$ and $\mathbf{C}^{(N/2^l)}$ for $l=1,\ldots L$ are given in (12). Fig. 5 shows the level 1 decomposition filter bank representation of HT and A2DHT. The $\mathbf{B}^{(N/2^l)}$ and $\mathbf{D}^{(N/2^l)}$ in (12) for HT are replaced by the $\mathbf{E}^{(N/2^l)}$ in (14) for A2DHT. Similar to conventional HT, the A2DHT is applicable to multiresolution analysis. In Section IV-E, we also provide an explicit construction of the A2DHT basis. Fig. 6 shows the 512×512 image 'Barbara' decomposed by HT and A2DHT at 7 levels.

Fig. 7 shows the proposed 2D 4×4 and 8×8 A2DHT. Normally, the bases in form of 2D images are called basis images. In this paper, a 2D image is represented by a 1D vector. For example, a 2D candidate window and a pattern are represented as 1D vectors $\vec{\mathbf{x}}^{(j)}_w$ and $\vec{\mathbf{x}}_t$ respectively. Hence, the bases in Fig. 7 are called basis vectors. Also, a 2D candidate window represented by 1D vector is said to be projected onto a basis vector instead of onto a basis image. It is easy to see that the A2DHT basis vectors in Fig. 7 are orthogonal to each other. We can normalize the basis vectors to form orthonormal A2DHT basis vectors. Thus A2DHT can be applied for transform domain pattern matching shown in Table I.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCSVT.2016.2629621, IEEE Transactions on Circuits and Systems for Video Technology

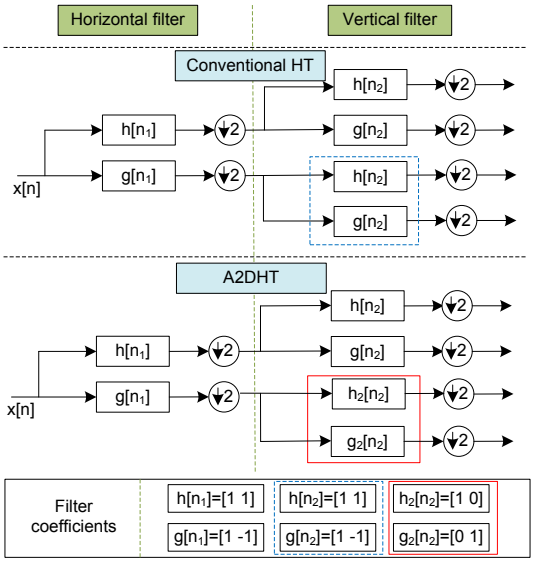


Fig. 5. The level 1 decomposition filter bank for 2D A2DHT and conventional 2D HT. $h[n_2]$ and $g[n_2]$ are used for conventional HT while $h_2[n_2]$ and $g_2[n_2]$ are used for A2DHT.

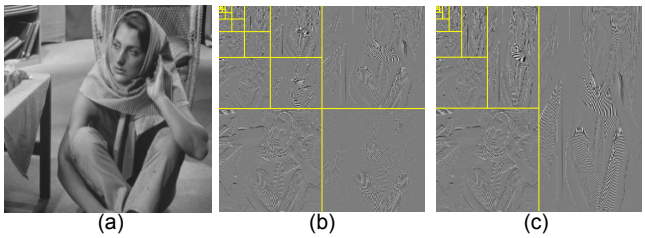


Fig. 6. The image 'Barbara' (left) decomposed at 7 levels using conventional Haar transform (*middle*) and the proposed A2DHT (*right*). The image 'Barbara' is high-pass filtered, yielding the two large images for A2DHT, each describing local changes in brightness (details) in the original image. It is then low-pass filtered and downscaled, yielding an approximation image. Then this procedure is repeated for 6 times on the approximation image.

The GCK algorithm requires $2u$ additions per pixel for obtaining u GCK projection values. If we directly use the integral image for computing Haar-like features as the implementation in OpenCV [55], 7 additions per pixel are required to obtain a Haar-like feature and about $7u$ additions per pixel are required to obtain u A2DHT projection values. Hence, direct use of integral image method makes A2DHT less efficient than WHT and GCK. In the followings, we propose a fast A2DHT algorithm that requires $O(\log u)$ additions per pixel for the computation of the u A2DHT projection values.

C. The Fast A2DHT Algorithm

The 4×4 A2DHT in Fig. 7 is used as an example to illustrate the proposed fast A2DHT algorithm. The 16 4×4 A2DHT basis vectors can be divided into 5 groups which are:

- Group 0: basis vector 0;
- Group 1: basis vector 1;
- Group 2: basis vectors 2 and 3;
- Group 3: basis vectors 4 to 7;
- Group 4: basis vectors 8 to 15;

Each group has its unique feature. Fig. 8 shows the 5 group features and their corresponding basis vectors. Each of these 5

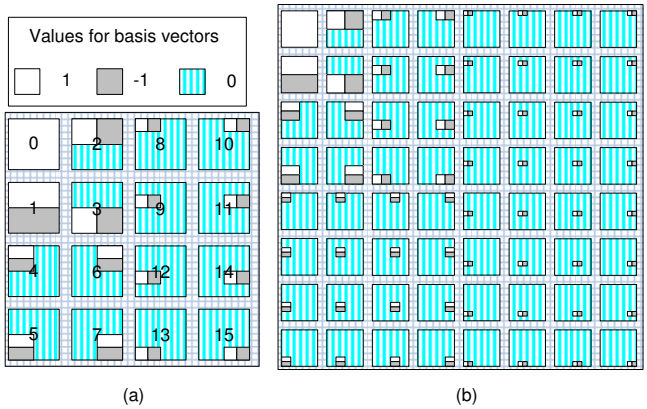


Fig. 7. (a): The 2D 4×4 A2DHT basis; (b): the 2D 8×8 A2DHT basis. White represents value +1, grey represents value -1 and vertical strips represent value 0. The numbers for 4×4 A2DHT basis denote the order when they are computed in pattern matching. Construction of the basis is illustrated in IV-E.

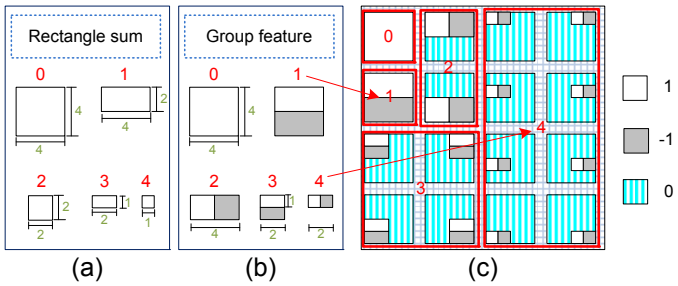


Fig. 8. The 5 rectangle sums, the 5 group features and their corresponding basis vectors for the 4×4 A2DHT.

group features can be computed from a rectangle sum. Since basis vectors in a group have the same feature, their projection values can be simultaneously obtained by computing one feature in a sliding window manner on the basis image. Therefore, the essence of the proposed algorithm is that *a single group feature contributes to multiple basis vectors*. For example, basis vectors 2 and 3 which have the same group 2 feature can be computed as follows:

1) Compute group 2 feature in a sliding window manner for all pixel locations and store them in $\mathbf{F}_2$.
2) The projection values for basis vectors 2 and 3 are copied from $\mathbf{F}_2$. Denote $f_2(j_1, j_2)$ as the element at horizontal location j_1 and vertical location j_2 for $\mathbf{F}_2$. As shown in Fig. 9, $f_2(0, 2)$ is considered as the basis vector 3 for window $(0, 0)$ and the basis vector 2 for window $(0, 2)$.

The steps for computing A2DHT and the number of operations required are shown in Table V. For 4×4 A2DHT as shown in Fig. 8(c), we compute the $g = 5$ group features from $r = 5$ rectangle sums with sizes 4×4, 4×2, 2×2, 2×1 and 1×1 as shown in Fig. 8(b). These rectangle sums have $n_H = 3$ different heights: 4, 2 and 1.

For the 8×8 A2DHT as shown in Fig. 7(b), there are 7 group features. These group features are computed from 7 rectangle sums with sizes 8×8, 8×4, 4×4, 4×2, 2×2, 2×1 and 1×1. Hence the heights are 8, 4, 2 and 1, and the number of heights is 4. As the number of basis vectors increases from 16 for the 4×4 A2DHT to 64 for the 8×8 A2DHT as shown in Fig. 7, we can see that when n_H increases by 1, g and r

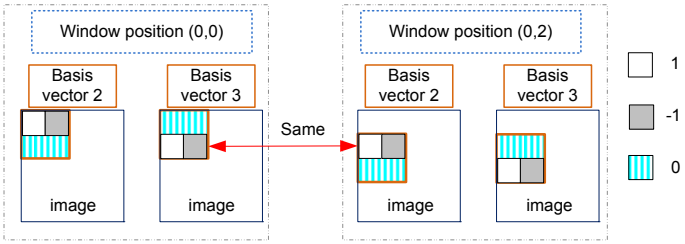


Fig. 9. The basis vector 3 of the window at position (0, 0) and the basis vector 2 of the window at position (0, 2) sharing the group feature 2 at the same position for 4×4 A2DHT. Generally, the basis vector 3 of the window at position (0, 0) and the basis vector 2 of the window at position (0, $N/2$) share the same group feature for $N \times N$ A2DHT, $N = 4, 8, 16 \ldots$

TABLE V
THE STEPS AND NUMBER OF OPERATIONS REQUIRED TO OBTAIN u A2DHT PROJECTION VALUES FROM g GROUP FEATURES.

1. Compute the rectangle sums with r different sizes n_H different heights. According to Table IV, $2 + n_H + r$ additions per pixel are required by the strip sum method.
2. The group feature 0 is the pre-computed rectangle sum. No computation is required.
3. Compute the remaining $g-1$ group features. $g-1$ additions per pixel are required.
4. The u basis vectors are the g group features in different positions. No computation required.

Overall, $1 + r + n_H + g$ additions per pixel.

will increase by 2.

Generally, when the $N_1 \times N_1$ input data is projected onto the first $u = 4^n, n = 0, 1, ...$ A2DHT bases, there are $g = 1 + \log_2 u$ group features. For these group features, there are $r = \log_2 u + 1$ rectangle sums having different sizes: $N_1 \times N_1$, $N_1 \times \frac{N_1}{2}$, $\frac{N_1}{2} \times \frac{N_1}{2}$, $\frac{N_1}{2} \times \frac{N_1}{4}, \cdots, \frac{N_1}{\sqrt{u}} \times \frac{N_1}{\sqrt{u}}$. And we have $n_H = 0.5log_2u + 1$. As given in Table V, $1 + r + n_H + g$ additions per pixel are required for computing these u bases. For the u basis vectors, since $g = 1 + \log_2 u$, $r = 1 + \log_2 u$ and $n_H = 1 + 0.5log_2u$, the proposed method requires $4 + 2.5 \log_2 u$ additions per pixel for obtaining u A2DHT projection values. Table VI summarizes the corresponding r, n_H, g and number of additions per pixel for 16, 64 and u A2DHT basis vectors. When $u = 16$, we have the example for the 4×4 A2DHT.

As for memory required in computing A2DHT, we can use the buffering strategy in Section III-C and require J_1 memory for storing strip sum.

D. A2DHT for Pattern Matching

Since A2DHT bases have varying norms, normalization is required for obtaining SSD. Because the normalization factors are power of 2, normalization can be computed by shift operations which has the same cost as additions. This normalization is computed for the partial SSD of the remaining candidates, i.e. $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2$ for $\vec{\mathbf{x}}_w^{(j)} \in S_w$, but not for A2DHT on the entire image, i.e. $\mathbf{V}\vec{\mathbf{x}}_w^{(j)}$ for $j = 0, 1, 2, \ldots, W-1$. For $u = 4^n$ A2DHT bases, there are $\log_2 u$ different normalization factors. In the worst case, the normalization requires $\log_2 u$ shifts per pixel when no candidate is rejected at all. This normalization procedure requires little computation in practical cases because many candidates are rejected in early stages.

TABLE VI
NUMBER OF ADDITIONS PER PIXEL REQUIRED BY THE A2DHT ALGORITHM FOR COMPUTING A2DHT.

Number of basis vectors	16	64	u
Number of rectangle sum r	5	7	$1+log_2u$
Number of heights n_H	3	4	$1+0.5log_2u$
Number of group features g	5	7	$1+\log_2 u$
Number of additions	14	19	$4 + 2.5 \log_2 u$

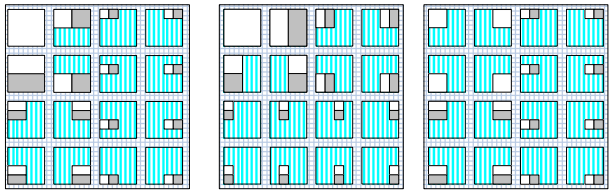


Fig. 10. (a) The 4×4 A2DHT basis $\mathcal{B}^{(4)}$; (b) the transposed version $\mathcal{B}_T^{(4)}$; (c) one of direct sum variants $\oplus_{l=0}^{3} \mathcal{B}_l^{(2)}$.

Projections are computed in pattern matching in two ways:
1) The sliding window way, which computes projection for all window positions over the whole image.
2) The random access way, which computes projections only for candidate windows remained.

A2DHT can be computed efficiently in both ways. For example, if we want to project 500 candidate windows for checking the rejection condition $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2 > T$ at various locations of a 256×256 image, A2DHT can be computed in the random access way on the 500 candidate windows with the aid of integral image. However, the GCK algorithm has to compute the transformation in the sliding window way on the entire image such that GCK is computed for about 256^2 windows instead of for the remaining 500 candidate windows. The corners method introduced in [25] for WHT may be used to compute the WHT for the 500 candidate windows. However, the corners method requires $O(N)$ additions per pixel for obtaining one WHT coefficient even if the strip sum algorithm is used while the A2DHT algorithm require $O(1)$ additions per pixel for the calculation of one A2DHT coefficient.

As a transform domain matching algorithm, A2DHT share the good properties of other transform domain algorithms, such as the ability to handle various distortions after proper modifications. Besides, A2DHT have all the merits possessed by wavelet transforms. For instance, basis vectors of the A2DHT are more locally supported and thus better adapted to local structures of an image.

E. Extension of the A2DHT to wavelet packets

In complement to the discussion on the A2DHT from the perspective of multi-resolution analysis in Section IV-B, we provide in this section an explicit construction of the A2DHT basis. The A2DHT is in nature a 2D wavelet transform, therefore it is possible to extend it to wavelet packet bases, which are regarded as variants of the proposed A2DHT basis, and on which it has the same computational complexity to calculate the corresponding projection coefficients. The A2DHT is asymmetric, meaning that it is not a separable 2D transform. So the extension differs from the classical extension proposed in [56] while resembling it. Besides, we propose in this section a basis selection algorithm, which finds the basis

that best fit a given set of patterns and reference images among all the bases included in the wavelet packets. When applied to natural scene images, the selection algorithm proves that the proposed A2DHT is a best choice.

For simplicity of notation, let $N_1 = N_2 = M$, where M is some positive integer power of 2. For any linear real space $\mathcal{H}^{(M)} = \mathbb{R}^{M\times M}$, we can define the A2DHT basis $\mathcal{B}^{(M)} = \left\{\psi_{i,j}^{(M)}\right\}_{i,j=0}^{M-1}$ recursively as follow. Denote by $\otimes$ the matrix Kronecker product, then for $M = 2$,

$$\psi_{0,0}^{(2)} = \tfrac{1}{2}\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, \quad \psi_{1,0}^{(2)} = \tfrac{1}{\sqrt{2}}\begin{bmatrix} 1 & -1 \\ 0 & 0 \end{bmatrix},$$
$$\psi_{0,1}^{(2)} = \tfrac{1}{2}\begin{bmatrix} 1 & 1 \\ -1 & -1 \end{bmatrix}, \quad \psi_{1,1}^{(2)} = \tfrac{1}{\sqrt{2}}\begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}.$$

For $M > 2$, if $i, j \in \left\{n \in \mathbb{N} \,\middle|\, 0 \le n < \frac{M}{2}\right\}$, then

$$\psi_{i,j}^{(M)} = \frac{1}{2}\psi_{i,j}^{(M/2)} \otimes \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix};$$

and if $(i,j) \in \left\{(i,j) \in \mathbb{N}^2 \,\middle|\, 0 \le i < \frac{M}{4}, \frac{M}{4} \le j < \frac{M}{2}\right\} \cup \left\{(i,j) \in \mathbb{N}^2 \,\middle|\, \frac{M}{4} \le i < \frac{M}{2}, 0 \le j < \frac{M}{2}\right\}$, then

$$\psi_{2i,2j}^{(M)} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \otimes \psi_{i,j}^{(M/2)},$$
$$\psi_{2i,2j+1}^{(M)} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \otimes \psi_{i,j}^{(M/2)},$$
$$\psi_{2i+1,2j}^{(M)} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \otimes \psi_{i,j}^{(M/2)},$$
$$\psi_{2i+1,2j+1}^{(M)} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \otimes \psi_{i,j}^{(M/2)}.$$

Next, we construct the wavelet packet bases of $\mathcal{H}^{(M)}$ by recursively and iteratively using the following two properties.

Property 1. *For any M, if $\mathcal{B}^{(M)}$ is the A2DHT basis, the transposes of the 2D basis vectors from $\mathcal{B}^{(M)}$ also form an orthonormal basis $\mathcal{B}_T^{(M)} = \left\{\psi_{i,j}^{(M)^T}\right\}_{i,j=0}^{M-1}$ of $\mathcal{H}^{(M)}$.*

Property 2. *For any M, if $\mathcal{A}_l^{(M/2)}$, $l = 0,1,2,3$, are orthonormal bases of $\mathcal{H}^{(M/2)} = \mathbb{R}^{M/2\times M/2}$, then their direct sum $\oplus_{l=0}^{3}\mathcal{A}_l^{(M/2)}$ defined as*

$$\left\{\begin{bmatrix} \psi_0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ \psi_1 & 0 \end{bmatrix}, \begin{bmatrix} 0 & \psi_2 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 0 & \psi_3 \end{bmatrix} \in \mathcal{H}^{(M)} \,\middle|\, \psi_l \in \mathcal{A}_l^{(M/2)}, l = 0,1,\ldots,3\right\} \tag{15}$$

is an orthonormal basis of $\mathcal{H}^{(M)}$.

Since Property 2 can be recursively used for each of the linear subspaces $\mathcal{H}^{(m)}$ where $m = 2^j, j = 1,2,\ldots,J = \log_2 M$, and since Property 1 is valid for arbitrary $m = 2^j$, we have the following properties.

Property 3. *The bases constructed by iteratively and recursively using Property 1 and Property 2 are structured in a quad-tree.*

Property 4. *For any basis included in the quad-tree, calculating the projection value onto any basis vector has the same computational complexity, which requires the calculation of at most two rectangle sums.*

TABLE VII
NUMBER OF VARIANTS FOR DIFFERENT PATTERN SIZES.

J	2	3	4	5
M	4	8	16	32
N_J	83	4.7×10^7	5.1×10^{30}	6.6×10^{122}

One can verify that the number of bases included in the quad-tree is exponentially large, which is stated by the proposition below.

Proposition 1. *Let N_J with $J = \log_2 M$ be the number of variants of the A2DHT basis included in the quad-tree, then N_J satisfies*

$$3^{\frac{M^2}{4}} = 3^{4^{J-1}} \le N_J \le 3^{\frac{245}{243}\cdot 4^{J-1}} = 3^{\frac{245}{243}\cdot\frac{M^2}{4}}. \tag{16}$$

Detailed proof can be found in Appendix B, which is similar to the proof provided in [57]. Now, we have constructed an exponentially large number of variants of the A2DHT basis. Table IV-E shows N_J for different pattern sizes. When $M = 4$, Fig. 10(a) shows the proposed A2DHT basis $\mathcal{B}^{(4)}$, and Fig. 10(b) shows its transposed version $\mathcal{B}_T^{(4)}$. Fig. 10(c) shows the variant $\oplus_{l=0}^{3}\mathcal{B}_l^{(2)}$ that corresponds to decomposing $\mathcal{H}^{(4)}$ into 4 disjoint subspaces $\mathcal{H}_l^{(2)}$ ($l = 0,1,2,3$) of dimensionality 2×2.

With the presence of Property 4, we can conclude that the optimal basis included in the quad-tree is the one with the best energy-packing ability, so that it allows the earliest rejection of the majority of the mismatched candidates. We develop a modified version of the Best Orthogonal Basis (BOB) selection algorithm proposed in [56] to find the basis which has the best energy packing ability and hence yields the lowest overall complexity.

To simplify the notation, let $\mathbf{V}^{(M)} = \left[\vec{\mathbf{v}}_0^{(M)}, \vec{\mathbf{v}}_1^{(M)}, \ldots, \vec{\mathbf{v}}_{M^2-1}^{(M)}\right]^T$, where $\vec{\mathbf{v}}_k^{(M)}$ is the vectorisation of $\psi_{i,j}^{(M)} \in \mathcal{B}^{(M)}$ for each $k = iM + j$, and let $\vec{\mathbf{c}} = [c_0, c_1, \ldots, c_{M^2-1}]^T = \mathbf{V}^{(M)}(\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w) \in \mathbb{R}^{M^2}$ be the projection coefficient vector of the difference $\vec{\mathbf{d}} = \vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w$. Then better energy packing ability of a basis amounts to lower entropy of the corresponding coefficient sequence $\{c_k\}_{k=0}^{M^2-1}$ defined by

$$\epsilon^2(\vec{\mathbf{c}}) = -\sum_k \frac{c_k^2}{||\vec{\mathbf{c}}||_2^2}\log\frac{c_k^2}{||\vec{\mathbf{c}}||_2^2}, \tag{17}$$

where $||\cdot||_2$ denotes the ℓ_2-norm of a vector. The algorithm we use to find the optimal basis is similar to the BOB algorithm proposed in [56]. It runs from the leaves to the root of the quad-tree. Suppose $\mathcal{A}_l^{(m)}$ denotes the best basis of subspace $\mathcal{H}_l^{(m)}$ with $m = 1,2,2^2,\ldots,M$ and $l = 0,1,2,\ldots,4^{M-m}-1$. In each subspace $\mathcal{H}_l^{(m)}$, we examine three bases $\mathcal{B}^{(m)}$, $\mathcal{B}_T^{(m)}$ and $\oplus_{k=0}^{3}\mathcal{A}_{4l+k}^{(m/2)}$ and then choose the one with the minimum entropy as the best one of this subspace. After that, the algorithm goes "upwards" to a "coarser" level, i.e. changing from m to $2m$, and examines bases for the 4^{M-2m} subspaces $\mathcal{H}_l^{(2m)}$ with $l = 0,1,\ldots,4^{M-2m}-1$. The algorithm runs very fast, since it benefits from the additivity of the entropy measure. Please refer to Appendix C for details.

Equation (17) defines the entropy of a single coefficient vector. Different from the case in [56], in our case the energy

packing ability of a basis $\mathcal{B}$ is closely related to the specific probability distributions of the pattern and the candidates. Hence, we consider the pattern $\vec{\mathbf{x}}_t$ and the candidate $\vec{\mathbf{x}}_w$ as random vectors, then the expectation of c_k^2 is $\bar{c_k^2} = \mathcal{E}\left\{\left(\vec{\mathbf{v}}_k^T\left(\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w\right)\right)^2\right\} = \vec{\mathbf{v}}_k^T\mathcal{E}\left\{\left(\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w\right)\left(\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w\right)^T\right\}\vec{\mathbf{v}}_k$, where $\mathcal{E}\{\cdot\}$ is the mathematical expectation operator. In experiments, we estimate $\mathcal{E}\left\{\left(\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w\right)\left(\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w\right)^T\right\}$ with

$$\hat{C} = \frac{1}{P}\sum_{p=1}^{P}\left(\vec{\mathbf{x}}_{t,p} - \vec{\mathbf{x}}_{w,p}\right)\left(\vec{\mathbf{x}}_{t,p} - \vec{\mathbf{x}}_{w,p}\right)^T \tag{18}$$

by taking samples $\vec{\mathbf{x}}_{t,p}$ and $\vec{\mathbf{x}}_{w,p}$ of the pattern and the candidate, respectively. Then c_k^2 is estimated by

$$\hat{c_k^2} = \vec{\mathbf{v}}_k^T\hat{C}\vec{\mathbf{v}}_k = \sum_i \lambda_i\left(\vec{\mathbf{v}}_k^T\vec{\mathbf{u}}_i\right)^2, \tag{19}$$

where $\hat{C} = \sum_i \lambda_i\vec{\mathbf{u}}_i\vec{\mathbf{u}}_i^T$, and λ_i and $\vec{\mathbf{u}}_i$ are the i-th eigenvalue and eigen-vector of $\hat{C}$, respectively. Notice that eigen-decomposition of $\hat{C}$ is utilised so that we can use the fast algorithm described in Section III when calculating the projection coefficients $\vec{\mathbf{v}}_k^T\vec{\mathbf{u}}_i$.

Now we have provided a selection algorithm that finds the basis that best fits a given set of patterns and candidates among the bases contained in the whole wavelet packets. This could be useful. Because the users of our fast matching algorithm can adaptively choose the best basis for their own data to be processed.

To find the optimal basis for pattern matching of natural scene images, we used samples from the ImageNet dataset [58] to estimate the expectations of the coefficient squares c_k^2. Experiments were conducted for $M = 16, 32$, and 64. For each size more than 120 million pairs $(\vec{\mathbf{x}}_{t,p}, \vec{\mathbf{x}}_{w,p})$ were sampled from 24605 images which are selected from the ImageNet dataset. Then the coefficient squares c_k^2 are estimated using Equation (18) and Equation (19).

For $M = 16$, Fig. 11 shows some bases included in the quad-tree and the corresponding entropies. Fig. 11 (a) shows the first 10 vectors of the proposed 16×16 A2DHT basis $\mathcal{B}^{(16)}$, which corresponds to the minimum entropy 1.6320; Fig. 11 (b) shows the first 10 vectors of the transposition variant $\mathcal{B}_T^{(16)}$, which corresponds to a slightly higher entropy 1.6351; Fig. 11 (c) and Fig. 11 (d) show the first 10 vectors of other two bases that correspond to significantly higher entropies. Notice that all bases shown in Fig. 11 are in their optimal orders which are obtained by sorting the corresponding coefficient squares $\hat{c_k^2}$ in descending order. The results of the experiments conducted for all $M = 16, 32$ and 64 show that the coefficient sequence when using the A2DHT basis has the lowest entropy, which indicates that the A2DHT basis is the optimal one among all its variants. The A2DHT basis is slightly better than its transposed version. This is probably because that there are a little bit more horizontal edges than vertical ones in natural scene images. We remark here that above result is valid for pattern matching tasks of natural scene images. For other types of images, fingerprint images for instance, the optimal basis may not be the proposed A2DHT basis. However, the basis selection algorithm described in this section can still be used to find the most suitable basis included in the quad-tree, and hence enable fast matching of the pattern.

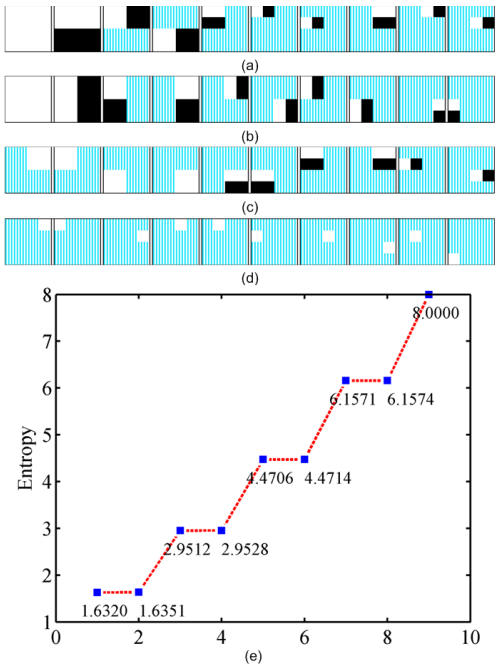


Fig. 11. (a) First 10 vectors of the proposed A2DHT basis $\mathcal{B}^{(16)}$; (b) first 10 vectors of the transposition variant $\mathcal{B}_T^{(16)}$;(c) and (d) first 10 vectors of other two bases included in the quad-tree; (e) Entropies corresponding to several 16×16 bases that are included in the quad-tree.

F. Comparison of A2DHT with Other Transforms

1) A2DHT and GCK: The elements in A2DHT are only 1, -1 and 0. The elements in GCK [20] are real numbers. A2DHT is more efficient when computed on isolated windows. It is because efficient computation of GCK and generalized GCK requires that they are computed in sliding window manner. Among the families of the GCK and the generalized GCK, the most efficient transform domain pattern matching reported uses WHT. A2DHT and WHT are compared in the next section.

2) A2DHT and WHT: The following theorem describes the relationship between WHT and A2DHT:

Theorem 1. *If the 2D $N_1 \times N_2$ WHT bases are in the same order as A2DHT bases, then: 1) the subspace spanned by the first $u = 4^n, n = 0, 1, \ldots$ WHT bases is equal to the subspace spanned by the first u A2DHT bases; 2) the first u orthonormal WHT bases and the first u orthonormal A2DHT bases extract the same energy from any input data; 3) if WHT is computed by the approach in [21], the computational complexity for the u WHT bases is $3u/2 + 1$ additions per pixel; if A2DHT is computed by the approach proposed in this paper, the computational complexity for the u A2DHT bases is $4 + 2.5\log_2 u$ additions per pixel.*

The proof for the theorem above is provided in the appendix. Here, we use the 4×4 A2DHT and 4×4 WHT shown in Fig. 13 as an example for illustration. Let $\vec{\mathbf{v}}_{WHT}^{(i)}$ be the ith 2D

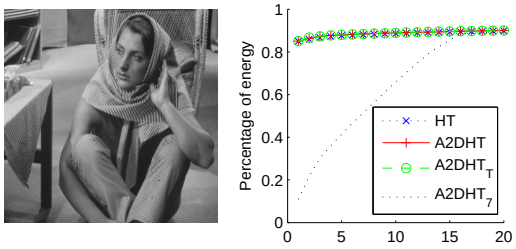


Fig. 12. The percentage of extracted energy as a function of the number of basis (*right*) from 512×512 image 'Barbara' (*left*), where HT denotes the HT for 9 levels, A2DHT denotes the A2DHT for 9 levels, A2DHT_T denotes the transposed A2DHT for 9 levels and A2DHT_7 denotes the A2DHT for 7 levels.

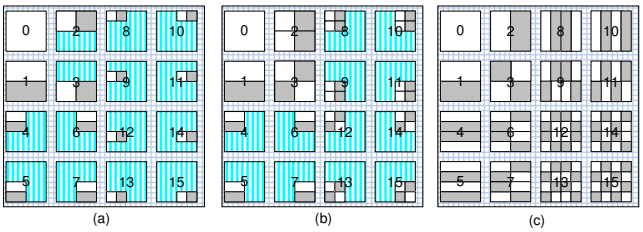


Fig. 13. 2D 4×4 transforms: (a) the proposed A2DHT, (b) conventional Haar transform, (c) WHT. White represents +1, grey represents -1 and green vertical strips represent 0. The numbers for 4×4 A2DHT basis denote the order when they are computed in pattern matching.

WHT basis and $\vec{\mathbf{v}}_{A2DHT}^{(i)}$ be the ith 2D A2DHT basis. We have:

$$\begin{bmatrix} \vec{\mathbf{v}}_{WHT}^{(0)^T} \\ \vec{\mathbf{v}}_{WHT}^{(1)^T} \\ \vec{\mathbf{v}}_{WHT}^{(2)^T} \\ \vec{\mathbf{v}}_{WHT}^{(3)^T} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} \vec{\mathbf{v}}_{A2DHT}^{(0)^T} \\ \vec{\mathbf{v}}_{A2DHT}^{(1)^T} \\ \vec{\mathbf{v}}_{A2DHT}^{(2)^T} \\ \vec{\mathbf{v}}_{A2DHT}^{(3)^T} \end{bmatrix}. \quad (20)$$

The first 4 WHT bases can be linearly represented by the first 4 A2DHT bases and vice versa. Thus the subspace spanned by the first 4 orthogonal WHT bases is equal to the subspace spanned by 4 A2DHT bases. The WHT bases are orthogonal to each other and can be normalized to be orthonormal bases, so are the A2DHT bases. Thus the energy extracted from input data by the first 4 orthonormal WHT bases is equal to that extracted by the first 4 orthonormal A2DHT bases. As for computational complexity, the fastest WHT algorithm in [21] requires $3u/2+1 = 25$ additions per pixel to compute the $u = 16$ WHT bases, the proposed fast A2DHT algorithm requires 14 additions per pixel to compute the 16 A2DHT bases.

3) A2DHT and HT: Fig. 13 shows the 4×4 A2DHT and 2D HT. In appendix A, we prove that the first $u = 4^n, n = 0, 1, \ldots$ conventional HT bases and the first u proposed A2DHT bases span the same subspace. As shown in Fig. 12, HT and A2DHT have very close energy extraction ability. However, as shown in Table VIII, A2DHT has lower computational complexity than the HT. For example, A2DHT bases 2 and 3 can be obtained at the same time when computed on sliding windows. On the other hand, the conventional HT bases 2 and 3 are required to be computed independently.

V. Experimental Results

This section evaluates the performance of pattern matching using A2DHT by comparing it with FS and the other fast FS-equivalent algorithms. All of the experiments were conducted on a 2.13GHz PC using C on windows XP system with compiling environment VC 6.0. Sum of squared difference (SSD) is used as the measure of dissimilarity between a pattern and candidate windows. Since all of the algorithms compared in this section are FS-equivalent and hence have *the same* accuracy, which is equal to the accuracy of a brute-force full search, we only focus on the speed.

TABLE VIII
Computational complexity of A2DHT and HT for u basis vectors. The integral image (II) method is compared with our A2DHT algorithm (Our alg.) in Section IV-C.

		$u = 4^n$	u=16	u=64	u=256
HT	II	5+29$(u-1)/3$	150	614	2470
	Our alg.	5+4.5$log_2 u$	23	32	41
A2DHT	II	5+7$(u-1)$	110	446	1790
	Our alg.	5+2.5$log_2 u$	15	20	25

TABLE IX
Datasets and corresponding sizes of images and patterns used in the experiments.

Dataset	Image Size	Pattern Size
$S1$	160×120	16×16
$S2$	320×240	32×32
$S3$	640×480	64×64
$S4$	1280×960	128×128
$S5$	1280×960	64×64
$S6$	1280×960	32×32

A. Dataset and algorithms used for pattern matching experiments

We compare the following 6 fast algorithms in this section,

1) WHT: the WHT algorithm in [19];
2) GCK: the GCK algorithm for WHT in [20];
3) SEGGCK: the Segmented GCK algorithm in [59]
4) IDA: the recently proposed IDA algorithm in [23];
5) A2DHT_I: the A2DHT using the integral image;
6) A2DHT: the A2DHT using the strip sum.

The code of WHT is available online [60] and the code of IDA is provided by the authors of [23]. The parameters of WHT use the default values in [60] and those of IDA are chosen according to [23]. For GCK, we choose the sequency ordered WHT and the code is based on the code used for motion estimation in [8]. In the experiments, we examine the overall computational efficiency of different algorithms instead of any single step if not specified. Thus all preprocessing, e.g. calculation of integral image or strip sum, are included for evaluation.

As pointed out in [19], when the percentage of remaining candidate windows is below a certain threshold ϵ, it is more efficient to directly use the FS rather than using any transform domain algorithm. In the experiments, we set ϵ to 0.02% for A2DHT, and 2 for WHT and GCK according to the source code in [60]. We shall discuss the effects of ϵ in more details in Section V-D.

Table IX shows the sizes of patterns and images in the six datasets, $S1$ to $S6$, used for evaluating the performance of

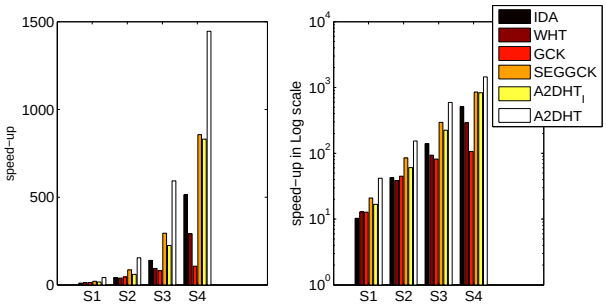


Fig. 14. Speed-up in execution time over FS on datasets $S1-S4$ for different algorithms measured by normal scale (*Left*) and log scale (*right*). The bars for each dataset from left to right correspond to algorithms IDA, WHT, GCK, SEGGCK, A2DHT$_I$ and A2DHT.

the six fast algorithms. 120 images of 4 different resolutions, ranging from 160×120 to 1280×960, are chosen from three databases: MIT [61], medical [62], and remote sensing [63]. The MIT database mainly contain indoor, urban, and natural environments, and some object categories such as cars and fruits. The two other databases contain radiographs and Landsat satellite images. There are 30 images in each resolution, and 10 patterns for each image. Hence, there are 300 image-pattern pairs in each of the six datasets. Note that the standard deviations of pixel intensities of the patterns are all above a threshold (45, in our experiments) to eliminate flat patterns. The OpenCV function 'cvResize' with linear interpolation is used to produce the images of desired resolutions. Datasets $S1$ to $S4$ are the same as those in [23]. Datasets $S5$ and $S6$ are used to investigate the effect of pattern size on the performance of different matching algorithms.

In the experiments, a candidate window is regarded as a correct match if the SSD between the candidate window and the pattern is below a threshold T. For an $N_1 \times N_2$ pattern, the threshold T is set as follow,

$$T = 1.1 \cdot SSD_{min} + N_1 N_2, \tag{21}$$

where SSD_{min} is the SSD between the pattern and the best matched window.

B. Experiment 1 – Influence of image and pattern sizes

In this experiment, we compare the speed-ups yielded by the 6 fast algorithms on datasets $S1$-$S4$ which contain images and patterns of different sizes. The *speed-up* in execution time (resp. number of operations) of an algorithm A over another algorithm B is measured by the execution time (resp. number of operations) required by B divided by that required by A. The larger the speed-up is, the faster the algorithm runs. The speed-ups in execution time yielded by IDA, WHT, GCK, SEGGCK and A2DHT over FS are shown in Fig. 14. It can be seen that A2DHT outperforms the other fast algorithms on all of the 4 datasets. To compute the projection coefficients, using strip sum only requires around 50% of the execution time required by methods using integral image. Hence, for fast matching algorithms adopting the same transform, the A2DHT, using strip sum saves around 40-50% of the execution time.

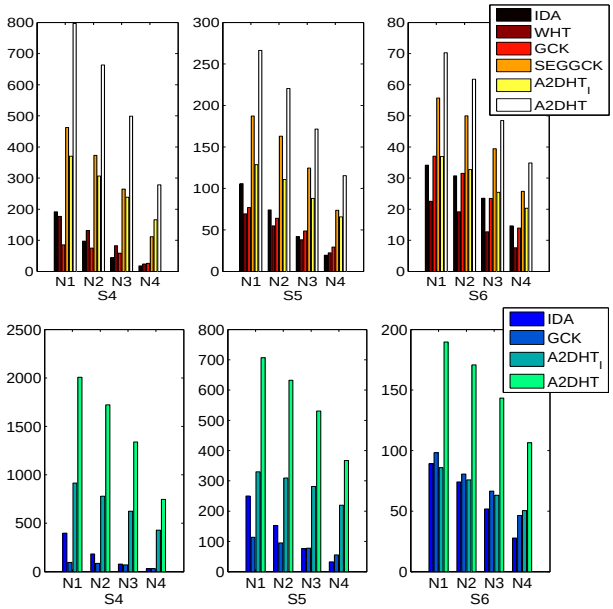


Fig. 15. Speed-ups in execution time (upper row) and speed-ups in number of operations (bottom row) yielded by different algorithms over FS for different noise levels and sizes of image-pattern pairs in pattern matching.

C. Experiment 2 – Influence of pattern size and noise

To evaluate the performance of the algorithms for patterns of various sizes and images of a fixed size, we conduct experiments on datasets $S4$-$S6$. In $S4$-$S6$, the image size is always 1280×960, while the pattern size changes from 128×128 to 32×32. Moreover, i.i.d. zero-mean Gaussian noise of 4 different levels are added to each image. The noise of 4 different levels, denoted by $N1$, $N2$, $N3$ and $N4$ (from low to high), has variance 100, 200, 400 and 800, respectively. They correspond to PSNR 28.1, 25.1, 22.1 and 19.2 when applied on the 512×512 image "Lena".

Fig. 15 shows the speed-ups in execution time (and in number of operations if applicable) yielded by the fast algorithms over FS. It can be seen that the A2DHT runs the fastest. We can also see from Fig. 16, the speed-ups in execution time yielded by A2DHT over IDA and GCK are about 4 - 15 and 8 - 10, respectively, the speed-ups in number of operations yielded by A2DHT over IDA and GCK are about 5 - 24 and 19 - 24, respectively. The speed-up in number of operations for WHT is not provided because the bottom up approach of WHT method in [19] is too complicated to analyse.

D. Experiment 3 – Influence of parameter ϵ

As explained in [19], when the percentage of remaining candidate windows is smaller than a certain threshold ϵ, it is more efficient to directly use SSD for finding the matched windows instead of using transformation. Fig. 17(a) shows the influence of ϵ for both GCK and A2DHT on dataset $S4$ with Gaussian noise $N4$. It can be seen that 2% is better for GCK while 0.02% is better for A2DHT. Fig. 17(b) shows that the computation of A2DHT is obviously faster than the computation of GCK for different situations of ϵ.

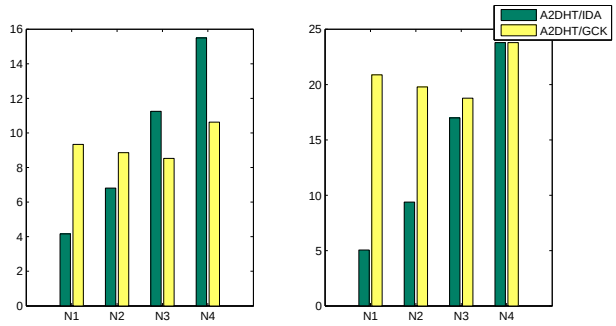


Fig. 16. Speed-up of A2DHT over IDA and GCK at 4 noise levels on dataset $S4$. The y-axis denotes the ratio between the execution time (left) (and the number of operations, right) required by the IDA or the GCK and that required by the A2DHT. The A2DHT runs faster if the ratio is greater than 1.

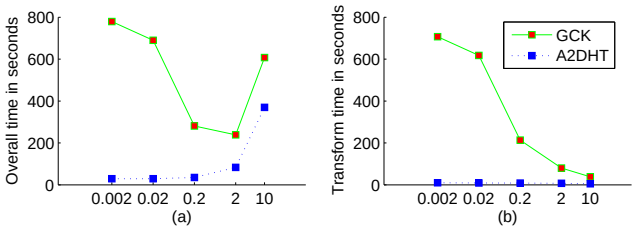


Fig. 17. **(a)** The overall execution time in seconds as a function of ϵ (left) and **(b)** the transformation time in seconds as a function of ϵ (right). Experiments are done on dataset $S4$ with noise $N4$. ϵ denotes the percentage of remaining window below which FS is used for pattern matching, e.g. 2 and 10 in the X axis correspond to 2% and 10%, respectively.

E. Experiment 4 – Energy packing ability of A2DHT and WHT

Experiments 1, 2 and 3 evaluate the execution time required for the whole pattern matching procedure. In this experiment, we evaluate the energy packing ability of A2DHT and WHT. As analyzed in [19], the computational efficiency of transform domain pattern matching is dependent on two factors: 1) the rejection power of projection values, which is dependent on the energy packing ability of transformation; 2) the cost of computing transformation. In summary, transform domain pattern matching requires that the transformation should be computationally efficient in packing energy.

We examine the energy packing ability of A2DHT and WHT on dataset $S1$. This dataset contains 300 image-pattern pairs, i.e. 4,567,500 window-pattern pairs of size 16×16. The SSD between the window and pattern were computed. The percentage of energy packed by the first u basis vectors is measured by:

$$PER^{(u)} = E\left[\frac{||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2}{||\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w^{(j)}||^2}\right], \tag{22}$$

where $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2$ is the partial energy packed by transformation $\mathbf{V}$ and $||\vec{\mathbf{x}}_t - \vec{\mathbf{x}}_w^{(j)}||^2$ is the total energy.

Fig. 18(a) shows the $PER^{(u)}$ in (22) as a function of the number of bases u. It can be seen that A2DHT is close to WHT in energy packing ability. These results, however, do not exhibit the runtime required in extracting energy. Fig. 18(b) shows the $PER^{(u)}$ in (22) as a function of the number of operations per pixel required to obtain the partial energy $||\mathbf{V}\vec{\mathbf{x}}_t - \mathbf{V}\vec{\mathbf{x}}_w^{(j)}||^2$. It can be seen that A2DHT can extract energy from input data using much fewer number of additions compared with WHT.

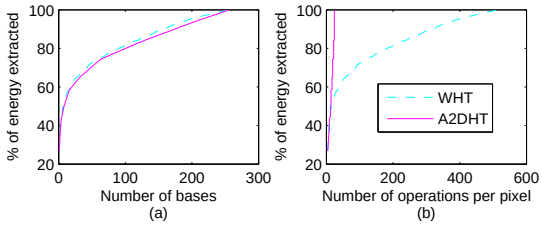


Fig. 18. Transformation using WHT and A2DHT. WHT is in sequency order and A2DHT is in the order illustrated in Fig. 7. The energy is given as the average percentage of the actual energy between pattern and window. All values are the average over 4,567,500 window-pattern pairs.

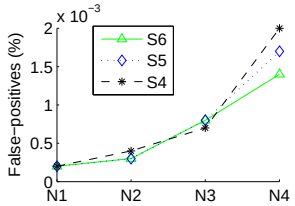


Fig. 19. False-positives (%) for noises $N1$-$N4$ in dataset $S4$-$S6$.

F. Discussion on full search scheme

As illustrated in Section II-B, there are two goals for FS in pattern matching:

Goal 1. find windows having distance $d(\vec{\mathbf{x}}_t, \vec{\mathbf{x}}_w^{(j)})$ smaller than a given threshold T;

Goal 2. find the window having the minimum distance $d(\vec{\mathbf{x}}_t, \vec{\mathbf{x}}_w^{(j)})$.

In experiments 1 to 4, this paper follows the approach in [23] and use goal 1. The choice of T for the previous experiments follows the experiments in [23], in which IDA [23] is compared with WHT [19]. As stated in [23], T is chosen to be very close to the best matching window. As shown in Fig. 19, the false positives using the threshold in (21) are not greater than 0.0025% in $S4$-$S6$ for the 4 noise levels.

In practice, T is dependent on specific applications. If the SSD_{min} in (21) is not known, T can be set as a fixed value. Fig. 20 shows the experimental results on dataset $S4$ with noise $N1$. The speed-ups of IDA, GCK and WHT over FS with different values of T are tested. As shown in Fig. 20, as T increases, the number of matched windows increases, while the speed-ups of IDA, GCK and A2DHT decreases. Nevertheless, A2DHT outperforms the GCK and IDA algorithm when the number of matched windows varies in a large range (from 10 to 10^4).

Fig. 21 shows the experimental results when adopting goal 2. A2DHT is faster than GCK and IDA under different levels of noise and for patterns of different sizes.

VI. Conclusions

This paper utilises the integral image to compute the strip sum with one addition per pixel. Then the strip sum method is used for computing the rectangle sums with one addition per pixel independent of the size of the rectangle. The rectangle sums are building blocks of the Haar-like features proposed in [35], [64], [37]. We then develop a new transform, the A2DHT, which can be computed efficiently using the fast strip sum method. Existing fast algorithms for computing

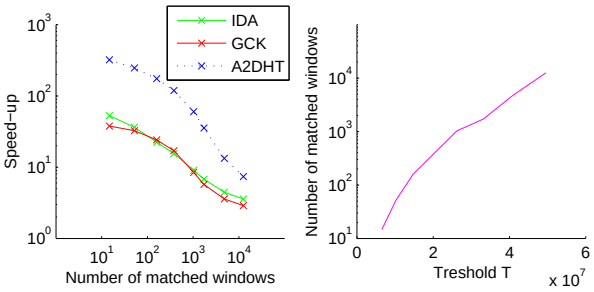


Fig. 20. Speed-up in execution time over FS as a function of the number of matching windows (left) and the the number of matching as a function of threshold T.

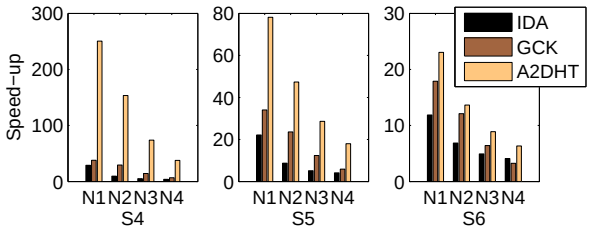


Fig. 21. Speed-up in execution time over FS for finding the best matching window for noises $N1$-$N4$ in datasets $S4$-$S6$. SSD is used as the matching criteria.

transforms on sliding windows [20], [19], [21] require $O(u)$ additions per pixel for projecting input data onto u 2D basis. The A2DHT requires only $O(\log u)$ for the same task when using the strip sum. Besides, we extend the A2DHT to wavelet packets that contain an exponentially large number of bases. Projections on all of these bases can be calculated as efficiently as A2DHT. Further more, we propose a selection algorithm which can find among all the variants the best transform (equivalently, the optimal basis of the space) for a given set of patterns and candidates. For natural scene images, the selection algorithm shows that the proposed A2DHT is the best. Experimental results show that the proposed algorithm can significantly accelerate the FS-equivalent pattern matching process and outperforms state-of-the-art methods. The fast strip sum method and the A2DHT have potential applications in feature extraction, block matching in motion estimation, texture synthesis and analysis, super resolution, image based rendering, image de-noising, object detection and tracking.

ACKNOWLEDGEMENT

The authors wish to thank Professor Yacov Hel-Or at the Interdisciplinary Center and Professor Hagit Hel-Or at the University of Haifa for providing the thesis on generalized GCK and their code implementing GCK and WHT, Dr. Federico Tombari and Prof. Luigi Di Stefano at the University of Bologna for providing their code implementing IDA, image datasets and helpful discussion, Professor Antonio Torralba and CSAIL at the MIT for the use of the MIT database, Professor Rainer Koster and the Institute for Clinical Radiology and Nuclear Medicine of the Lukas Hospital Neuss for the use of the medical image database, and NASA for the use of the remote sensing image database.

REFERENCES

[1] M. S. Aksoy, O. Torkul, and I. H. Cedimoglu, "An industrial visual inspection system that uses inductive learning," *J. of Intelligent Manufacturing*, vol. 15, no. 4, pp. 569–574, 2004.

[2] A. Fitzgibbon, Y. Wexler, and A. Zisserman, "Image-based rendering using image-based priors," in *ICCV*, vol. 2, 2003, pp. 1176–1183.

[3] T. Luczak and W. Szpankowski, "A suboptimal lossy data compression based on approximate pattern matching," *IEEE Trans. Information Theory*, vol. 43, no. 5, pp. 1439–1451, 1997.

[4] R. M. Dufour, E. L. Miller, and N. P. Galatsanos, "Template matching based object recognition with unknown geometric parameters," *IEEE Trans. Image Process.*, vol. 11, no. 12, pp. 1385–1396, Dec. 2002.

[5] W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, vol. 22, no. 2, pp. 56–65, Mar./Apr 2002.

[6] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, Sept. 1999, pp. 1033–1038.

[7] C. M. Mak, C. K. Fong, and W. K. Cham, "Fast motion estimation for H.264/AVC in Walsh Hadamard domain," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 5, pp. 735–745, Jun. 2008.

[8] Y. Moshe and H. Hel-Or, "Video block motion estimation based on Gray-code kernels," *IEEE Trans. Image Process.*, vol. 18, no. 10, pp. 2243–2254, Oct. 2009.

[9] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, vol. 2, Jun. 2005, pp. 60– 65.

[10] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Trans. Image Process.*, vol. 16, no. 8, pp. 2080–2095, Aug. 2007.

[11] R. Zhang, W. Ouyang, and W. Cham, "Image deblocking using dual adaptive fir wiener filter in the DCT transform domain," in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing (ICASSP)*, Taiwan, April 19-24 2009, pp. 1181–1184.

[12] Y. Alon, A. Ferencz, and A. Shashua, "Off-road path following using region classification and geometric projection constraints," in *CVPR*, vol. 1, Jun. 2006, pp. 689–696.

[13] Y. Shina, J. S. Jua, and E. Y. Kim, "Welfare interface implementation using multiple facial features tracking for the disabled people," *Pattern Recognition Letters*, vol. 29, no. 13, pp. 1784–1796, Oct. 2008.

[14] Q. Wang and S. You, "Real-time image matching based on multiple view kernel projection," in *CVPR*, 2007.

[15] X. Wu, "Template-based action recognition: Classifying hockey players' movement," Master's thesis, The University of British Columbia, 2005.

[16] Y. Hel-Or, H. Hel-Or, and E. David, "Fast template matching in non-linear tone-mapped images," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1355–1362.

[17] J. Lewis, "Fast template matching," in *Vision Interface 95*, Quebec City, Canada, May 15-19 1995, pp. 120–123.

[18] M. G. Alkhansari, "A fast globally optimal algorithm for template matching using low-resolution pruning," *IEEE Trans. Image Process.*, vol. 10, no. 4, pp. 526–533, Apr 2001.

[19] Y. Hel-Or and H. Hel-Or, "Real time pattern matching using projection kernels," *IEEE Trans. PAMI*, vol. 27, no. 9, pp. 1430–1445, Sept. 2005.

[20] G. Ben-Artz, H. Hel-Or, and Y. Hel-Or, "The Gray-code filter kernels," *IEEE Trans. PAMI*, vol. 29, no. 3, pp. 382–393, Mar. 2007.

[21] W. Ouyang and W. K. Cham, "Fast algorithm for Walsh Hadamard transform on sliding windows," *IEEE Trans. PAMI*, vol. 32, no. 1, pp. 165–171, Jan. 2010.

[22] W. Ouyang, R. Zhang, and W.-K. Cham, "Fast pattern matching using orthogonal haar transform," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3050–3057.

[23] F. Tombari, S. Mattoccia, and L. D. Stefano, "Full search-equivalent pattern matching with incremental dissimilarity approximations," *IEEE Trans. PAMI*, vol. 31, no. 1, pp. 129–141, Jan. 2009.

[24] S. Mattoccia, F. Tombari, and L. D. Stefano, "Fast full-search equivalent template matching by enhanced bounded correlation," *IEEE Trans. Image Process.*, vol. 17, no. 4, pp. 528–538, Apr. 2008.

[25] H. Schweitzer, R. Deng, and R. F. Anderson, "A dual bound algorithm for very fast and exact template-matching," *IEEE Trans. PAMI*, vol. 33, no. 3, pp. 459–470, Mar. 2011.

[26] W. Pan, S. Wei, and S. Lai, "Efficient NCC-based image matching in Walsh-Hadamard domain," in *ECCV*, D. Forsyth, P. Torr, and A. Zisserman, Eds., 2008.

[27] S.-D. Wei and S.-H. Lai, "Fast template matching based on normalized cross correlation with adaptive multilevel winner update," *IEEE Trans. Image Process.*, vol. 17, no. 11, pp. 2227–2235, Nov. 2008.

[28] Y. Li, H. Li, and Z. Cai, "Fast orthogonal haar transform patternmatching via image square sum," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 36, no. 9, pp. 1748–1760, 2014.

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

[30] B. Zitova and J. Flusser, "Image registration methods:a survey," *Image Vis. Comput.*, vol. 21, no. 11, pp. 977–1000, 2003.

[31] W. Krattenthaler, K. Mayer, and M. Zeiler, "Point correlation: A reduced-cost template matching technique," in *Proc. 1st IEEE Int. Conf. Image Processing*, vol. 1, Austin, TX, 1994, pp. 208–212.

[32] K. Briechle and U. D. Hanebeck, "Template matching using fast normalized cross correlation," in *Proc. SPIE AeroSense Symp.*, vol. 4387. SPIE, 2001, pp. 95–102. [Online]. Available: http://link.aip.org/link/?PSI/4387/95/1

[33] H. Schweitzer, J. W. Bell, and F. Wu, "Very fast template matching," in *ECCV*, 2002, pp. 358–372.

[34] P. Simard, L. Bottou, P. Haffner, and Y. LeCun, "Boxlets: A fast convolution algorithm for signal processing and neural networks," *Adv. Neural Inf. Process. Syst.*, vol. 11, pp. 571–577, 1999.

[35] F. Tang, R. Crabb, and H. Tao, "Representing images using nonorthogonal Haar-like bases," *IEEE Trans. PAMI*, vol. 29, no. 12, pp. 2120–2134, Dec. 2007.

[36] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *CVPR*, 2001, pp. I:511–I:518.

[37] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *DAGM 25th Pattern Recognition Symposium*, 2003, pp. 297–304.

[38] P. Viola, M. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *ICCV*, 2003, pp. II:734–II:741.

[39] O. Tuzel, F. Porikli, and P. Meer, "Pedestrian detection via classification on riemannian manifolds," *IEEE Trans. PAMI*, vol. 30, no. 10, pp. 1713–1727, Oct. 2008.

[40] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *CVPR*, 2006.

[41] M. Pham, Y. Gao, V. Hoang, and T. Cham, "Fast polygonal integration and its application in extending haar-like features to improve object detection," in *CVPR*, 2010, pp. 942–949.

[42] C. Lampert, M. Blaschko, and T. Hofmann, "Beyond sliding windows: object localization by efficient subwindow search," in *CVPR*, 2008.

[43] H. Bay, T. Tuytelaars, and L. Gool, "Surf: Speeded up robust features," in *ECCV*, vol. 1, 2006, pp. 404–417.

[44] F. Crow, "Summed-area tables for texture mapping," in *Proc.11th Ann. Conf. Computer Graphics and Interactive Techniques*, 1984, pp. 207–212.

[45] F. Porikli, "Integral histogram: a fast way to extract histograms in cartesian spaces," in *CVPR*, 2005.

[46] A. Mahmood and S. Khan, "Exploiting transitivity of correlation for fast template matching," *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 2190–2200, 2010.

[47] ——, "Correlation-coefficient-based fast template matching through partial elimination," vol. 21, no. 4, pp. 2099–2108, 2012.

[48] B. Girod, *Whats Wrong with Mean-Squared Error?* MIT Press, 1993, ch. 15.

[49] S. Santini and R. Jain, "Similarity measures," *IEEE Trans. PAMI*, vol. 21, no. 9, pp. 871–883, Sept. 1999.

[50] A. Ahumada, "Computational image quality metrics: A review," in *Proc. Soc. Information Display Intl Symp.*, vol. 24, 1998, pp. 305–308.

[51] M. Ben-Yehuda, L. Cadany, and H. Hel-Or, "Irregular pattern matching using projections," in *Proc. 12th Int'l Conf. Image Processing (ICIP)*, vol. 2, 2005, pp. 834–837.

[52] Y. Hel-Or, T. Malzbender, and D. Gelb, "Synthesis and rendering of 3d textures," in *Texture 2003 Workshop accomp. ICCV 2003,*, 2003, pp. 53–58.

[53] Q. Wang, J. Mooser, S. You, and U. Neumann, "Augmented exhibitions using natural features," *Int'l. J. Virtual Reality*, vol. 7, no. 4, pp. 1–8, 2008.

[54] M. J. McDonnell, "Box-filtering techniques," *Comput. Graph. Image Process.*, vol. 17, pp. 65–70, 1981.

[55] "Opencv library," accessed in 2012. [Online]. Available: http://sourceforge.net/projects/opencvlibrary

[56] R. R. Coifman and M. V. Wickerhauser, "Entropy-based algorithms for best basis selection," *Information Theory, IEEE Transactions on*, vol. 38, no. 2, pp. 713–718, 1992.

[57] S. Mallat, *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[58] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, pp. 1–42, April 2015.

[59] W. Ouyang, R. Zhang, and W.-K. Cham, "Segmented gray-code kernels for fast pattern matching," *IEEE Transactions on Image Processing*, vol. 22, no. 4, pp. 1512–1525, 2013.

[60] Y. Hel-Or, "Webpage of real-time pattern matching using projection kernels," accessed in 2012. [Online]. Available: www.faculty.idc.ac.il/toky/Software/software.htm

[61] A. Torralba, "Mit-csail database of objects and scenes," accessed in 2012. [Online]. Available: http://people.csail.mit.edu/torralba/images

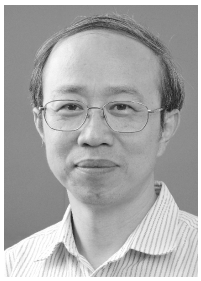
[62] J. Abel, "Data compression resource," accessed in 2012. [Online]. Available: www.data-compression.info/Corpora/LukasCorpus

[63] NASA, "Nasa mrsid image database," accessed in 2012. [Online]. Available: http://zulu.ssc.nasa.gov/mrsid

[64] P. Viola and M. Jones, "Robust real-time face detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.

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

Tianle Zhao Tianle Zhao received his Bachelor's degree from Shanghai Jiao Tong University, Shanghai, China, in 2013 in Information Engineering. He is now a Ph.D. candidate in the Department of Electronic Engineering, The Chinese University of Hong Kong. His research interests include image processing and pattern recognition.



Wai-kuen Cham Wai-kuen Cham graduated from The Chinese University of Hong Kong in 1979 in Electronics. He received his M.Sc. and Ph.D. degrees from Loughborough University of Technology, U.K., in 1980 and 1983 respectively. From June 1984 to April 1985, he was a senior engineer in Datacraft Hong Kong Limited and a lecturer in the Department of Electronic Engineering, Hong Kong Polytechnic (now The Polytechnic University of Hong Kong). Since May 1985, he has been with the Department of Electronic Engineering, the Chinese University of Hong Kong.

Liying Wei Liying Wei received the B.E. degree from Xidian University, China, in 2002, the M.E. degree from Beijing University of Posts and Telecommunications, Beijing, China, in 2006, and the Ph.D. degree from The Australian National University, Australia, in 2011. She worked as a postdoc researcher in the Australian National University from September 2010 to March 2011, the Chinese University of Hong Kong from June 2011 to May 2013, the Telecom-ParisTech from June 2014 to May 2015 and Institut de Recherche et Coordination Acoustique/Musique (IRCAM) from June 2015 to November 2015. She is a postdoc researcher in IBM Netherlands. Her current research interests are in the field of digital signal processing, acoustic signal processing and image processing.