Design of Orthonormal Haar-like Features for Fast Pattern Matching

Izumi Ito
School of Engineering
Tokyo Institute of Technology
Tokyo, Japan
Email: ito@ict.e.titech.ac.jp

Karen Egiazarian
Signal Processing Laboratory
Tampere University of Technology,
Tampere, Finland
Email: karen.egiazarian@tut.fi

***Abstract*—The goal of pattern matching is to find small parts of an image that are similar to a given template. Matching in transform-domain (such as Haar, Walsh-Hadamard, etc.) is more efficient that matching in the spatial domain. However, it has a limitation: the template size is restricted to be a power of two to apply fast computational algorithms of transforms. In this paper, fast pattern matching method based on orthonormal tree-structured Haar transform (OTSHT) is proposed. It allows to overcome the above-mentioned limitation of the template size. Two types of tree structures are considered in this paper: balanced tree and logarithmic tree. It is demonstrated that the proposed method with the balanced tree structure is computationally more efficient.**

I. Introduction

Pattern matching, also known as a template matching, is a classic and fundamental method to locate small parts in an image that match a given template. It has a lot of applications such as object detection [1] and image registration [2] and broadens the variety of applications by being used as a pre- or post-processing steps in computer vision and novel image analysis [3][4]. A number of pattern matching methods have been proposed as a full search equivalent method based on orthogonal transforms [5][6][7][8]. Fast pattern matching method based on orthogonal Haar transform (OHT) [5] uses strip sums and has demonstrated to be more efficient than those based on Walsh-Hadamard transform [7] and Gray-Code kernels [8]. However, the size of the template is restricted to be a power-of-two. In the present paper, we propose the design of Haar-like features for fast pattern matching with arbitrary size using orthonormal tree-structured Haar transform (OTSHT). Since OTSHT [9], proposed by one of the authors, has a freedom of the design and its two-dimensional expression has not been considered yet, we present the mathematical expression for two-dimensional OTSHT basis images. We construct two types of OTSHT and evaluate them for pattern matching.

II. Preliminaries

A. Fast pattern matching algorithm using orthogonal Haar transform

Let us consider finding a part of image similar to a given template, where the template is compared with all of the candidate windows in the image by sliding window manner and the squared sum of difference (SSD) is used in order to find the similarity between the template and a candidate window, which is referred to as full search (FS).

In fast pattern matching algorithm using OHT, the similarity is calculated in the transformed domain while pruning candidate windows. The pruning process is crucial to reduce the computational cost.

The underlying theory of the pruning phase is following inequality proved in [10]:

$$||\mathbf{x_1}^{(L)} - \mathbf{x_2}^{(L)}||^2 \geq ||\mathbf{H}^{(L\times u)T}\mathbf{x_1}^{(L)} - \mathbf{H}^{(L\times u)T}\mathbf{x_2}^{(L)}||^2 \tag{1}$$

where the superscript represents the size of vector or matrix, $\mathbf{x_1}$ and $\mathbf{x_2}$ are the vector expression of the template and a candidate window, respectively, $\cdot^T$ represents transposition, and $\mathbf{H}^{(L\times u)}$ is a part of matrix $\mathbf{H}^{(L\times U)}$ consists of U orthonormal basis vectors of length L:

$$\mathbf{H}^{(L\times U)} = [\mathbf{V}_0, \mathbf{V}_1, \dots, \mathbf{V}_u, \dots, \mathbf{V}_{U-1}]. \tag{2}$$

Therefore, when we use the appropriate threshold, we can prune the unmatched candidate windows effectively. We do not need to calculate all of the projection value, i.e., $\mathbf{H}^{(L\times U)T}\mathbf{x}^{(L)}$, of all the candidate windows but calculate part of the projection value of the part of the candidate windows. Moreover, since classical Haar matrix consists of 1, -1, and 0, the projection value corresponds to the sum of pixel value in a rectangle area, which can be calculated using integral image with a few operations. Note that normalization is needed on top of that.

B. Strip sum via integral image

A rectangle sum is defined as the sum of all pixel value in a rectangle area in an image. Let $RS(j_1, j_2, w, h)$ be a rectangle sum in the area located at upper left (j_1, j_2) with width w and height h. The rectangle sum is computed by three additions using an integral image:

$$\begin{aligned} RS(j_1, j_2, w, h) &= \sum_{u=j_1}^{j_1+w-1}\sum_{v=j_2}^{j_2+h-1} x(u,v) \\ &= S(j_1+w, j_2+h) + S(j_1, j_2) \\ &\quad -S(j_1, j_2+h) - S(j_1+w, j_2) \end{aligned} \tag{3}$$

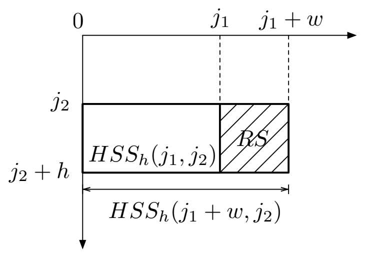

Fig. 1. Relation between rectangle sum (RS) and horizontal strip sum (HSS).

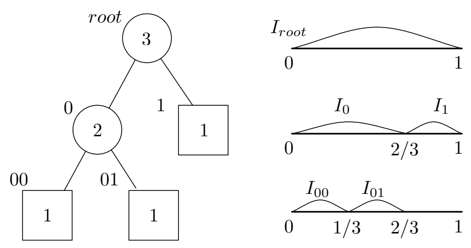

Fig. 2. Binary interval splitting tree (BIST) having 3 leaves with 2 depth and its interval.

where $S(j_1, j_2)$ is the integral image given as

$$S(j_1, j_2) = \sum_{u=0}^{j_1-1} \sum_{v=0}^{j_2-1} x(u, v). \tag{4}$$

Strip sum is a special case of rectangle sum including horizontal strip sum and vertical strip sum. We explain horizontal strip sum due to space limitation. Let $HSS_h(j_1, j_2)$ be horizontal strip sum with height h, which corresponds to integral image as

$$HSS_h(j_1, j_2) = RS(0, j_2, j_1, h). \tag{5}$$

The horizontal strip sum is computed using an integral image by one addition as

$$HSS_h(j_1, j_2) = S(j_1, j_2 + h) - S(j_1, j_2). \tag{6}$$

Therefore, we can calculate a rectangle sum using strip sum by one addition as

$$RS(j_1, j_2, w, h) = HSS_h(j_1 + w, j_2) - HSS_h(j_1, j_2). \tag{7}$$

Figure 1 illustrates the relation between rectangle sum (RS) and horizontal strip sum (HSS).

C. Tree-structured Haar transform

Let us consider a binary interval splitting tree (BIST) having N leaves with d depth. Each inner node is labeled by $\boldsymbol{\alpha}$, $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \cdots, \alpha_k)$, $\alpha_k \in \{0, 1\}, k = 1, 2, \ldots, N-1$. The label of each node starts with root. The label of left child of root is $\boldsymbol{\alpha} = (root, 0)$ also simply represented as $\boldsymbol{\alpha} = (0)$ and the label of the right child of root is $\boldsymbol{\alpha} = (root, 1)$ also simply represented as $\boldsymbol{\alpha} = (1)$. In the same manner, the left child of node $\boldsymbol{\alpha}$ is labeled as $(\boldsymbol{\alpha}, 0)$ by adding 0 to the right end of $\boldsymbol{\alpha}$ and its right child is labeled as $(\boldsymbol{\alpha}, 1)$ by adding 1 to the end of $\boldsymbol{\alpha}$. Let $\nu(\boldsymbol{\alpha})$ be the number of leaves the node has.

The interval, $I_{\boldsymbol{\alpha}}$, of node $\boldsymbol{\alpha}$ in BIST is assigned as follows:

$$I_{root} = [0, 1) \tag{8}$$

$$I_0 = I_{root,0} = \left[0, \frac{\nu(0)}{\nu(root)}\right) \tag{9}$$

$$I_1 = I_{root,1} = \left[\frac{\nu(0)}{\nu(root)}, 1\right) \tag{10}$$

For child of $\boldsymbol{\alpha}$ with interval $I_{\boldsymbol{\alpha}} = [a, b)$, $0 \le a < b < 1$, $a, b \in \mathbb{R}$,

$$I_{\alpha,0} = \left[a, a + \frac{\nu(\alpha, 0)}{\nu(\alpha)}(b-a)\right) \tag{11}$$

and

$$I_{\alpha,1} = \left[a + \frac{\nu(\alpha, 0)}{\nu(\alpha)}(b-a), b\right). \tag{12}$$

Figure 2 shows an example of BIST. A circle and square represent a node and a leaf, respectively. The top-left number of the circle is the label of the node. The number in the circle is $\nu(\alpha)$.

From the interval, the set of the basis functions f is defined as

$$f_{root,0}(t) = \frac{1}{\sqrt{N}},\ t \in [0, 1) \tag{13}$$

$$f_{root,1}(t) = \begin{cases} \sqrt{\frac{\nu(1)}{N\nu(0)}}, & \text{if } t \in I_0 \\ -\sqrt{\frac{\nu(0)}{N\nu(1)}}, & \text{if } t \in I_1 \end{cases} \tag{14}$$

$$f_\alpha(t) = \begin{cases} \sqrt{\frac{\nu(\alpha,1)}{\nu(\alpha)\nu(\alpha,0)}} & \text{if } t \in I_{\alpha,0} \\ -\sqrt{\frac{\nu(\alpha,0)}{\nu(\alpha)\nu(\alpha,1)}} & \text{if } t \in I_{\alpha,1} \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

Note that there are some weights for OTSHT.

III. Proposed method

We extend one-dimensional tree-structured Haar transform to two-dimensional Haar-like features, which enables us to use the sprit sum. Two types of orthonormal tree-structured Haar transforms are considered for fast pattern matching.

A. Orthonormal Tree-structured Haar transform basis image

Let us consider the design of OTSHT basis images from tree-structured Haar transform. We take a way of the two-dimensional decomposition according to the interval of BIST for holding orthonormal in order to compute the projected value using strip sum. Here, we introduce new label β which is especially expressed for the intervals of BIST in the vertical direction.

The basis image function of the space $(I_{root} \times I_{root})$ is defined as

$$\varphi_0(s,t) = \tfrac{1}{N} \quad (s,t) \in I_{root} \times I_{root} \tag{16}$$

Subsequently, the set of basis image functions of the space $(I_\alpha \times I_\beta)$ is given as

$$\varphi_1(s,t) = \begin{cases} \frac{\nu(\beta,1)}{\sqrt{\nu(\alpha)\nu(\beta)\nu(\beta,0)\nu(\beta,1)}}, & (s,t) \in I_\alpha \times I_{\beta,0} \\ -\frac{\nu(\beta,0)}{\sqrt{\nu(\alpha)\nu(\beta)\nu(\beta,0)\nu(\beta,1)}}, & (s,t) \in I_\alpha \times I_{\beta,1} \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

$$\varphi_2(s,t) = \begin{cases} \frac{\nu(\alpha,1)}{\sqrt{\nu(\alpha)\nu(\alpha,0)\nu(\alpha,1)\nu(\beta,0)}}, & (s,t) \in I_{\alpha,0} \times I_{\beta,0} \\ -\frac{\nu(\alpha,0)}{\sqrt{\nu(\alpha)\nu(\alpha,0)\nu(\alpha,1)\nu(\beta,0)}}, & (s,t) \in I_{\alpha,1} \times I_{\beta,0} \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

$$\varphi_3(s,t) = \begin{cases} \frac{\nu(\alpha,1)}{\sqrt{\nu(\alpha)\nu(\alpha,0)\nu(\alpha,1)\nu(\beta,1)}}, & (s,t) \in I_{\alpha,0} \times I_{\beta,1} \\ -\frac{\nu(\alpha,0)}{\sqrt{\nu(\alpha)\nu(\alpha,0)\nu(\alpha,1)\nu(\beta,1)}}, & (s,t) \in I_{\alpha,1} \times I_{\beta,1} \\ 0 & \text{otherwise} \end{cases} \tag{19}$$

where (α, β) starts with $(root, root)$ and then (α, β) are replaced by $((\alpha, 0), (\beta, 0))$, $((\alpha, 1), (\beta, 0))$, $((\alpha, 0), (\beta, 1))$, and $((\alpha, 1), (\beta, 1))$ until all node of BIST is used. When the space is indivisible, i.e., $\nu(\alpha) = 1$ and $\nu(\beta) = 1$, no more decomposition is applied. If a node α has child and node β does not have child i.e., $\nu(\alpha) > 1$ and $\nu(\beta) = 1$, then β is replaced as an inner node succeeding to the leaf, i.e., $\nu(\beta, 0) = 1$, and vice versa. A set of N^2 basis images of size $N \times N$ is generated. Note that $(root, 0)$ and $(root, 1)$ are replaced by (0) and (1), respectively.

Figure 3 illustrates how to divide into a positive and a negative value areas according to the intervals of BIST. The white and black parts represent a positive and a negative value areas, respectively, which make Haar-like features. The space $(I_\alpha \times I_\beta)$ is vertically divided into two, $(I_\alpha \times I_{\beta,0})$ and $(I_\alpha \times I_{\beta,1})$, and the value at each space is assigned by (17). Then the upper space $(I_\alpha \times I_{\beta,0})$ is horizontally divided into two, $(I_{\alpha,0} \times I_{\beta,0})$ and $(I_{\alpha,1} \times I_{\beta,0})$, by (18), while the lower space divided into two, $(I_{\alpha,0} \times I_{\beta,1})$ and $(I_{\alpha,1} \times I_{\beta,1})$ by (19).

Figure 4 shows an example of constructing the set of basis images based on BIST shown in Fig. 2. The white, black and grey areas represent a positive, a negative, and zero value areas, respectively. The first basis image is given by (16). When $(\alpha, \beta) = (root, root)$, the second one is given by (17), and the third and forth are given by (18) and (19), respectively. When $(\alpha, \beta) = (0, 0)$, the fifth is given by (17), and then the seventh and eighth are given by (18) and (19), respectively. When $(\alpha, \beta) = (1, 0)$, the sixth is given by (17), but no more decomposition is applied because α has no child, i.e., $\nu(\alpha) = 1$. When $(\alpha, \beta) = (0, 1)$, since α has child and β does not have child, i.e., $\nu(\alpha) > 1$ and $\nu(\beta) = 1$, the leaf β is replaced by an inner node succeeding to the leaf as shown in Fig. 5 where the leaf represents in dashed square is labeled as

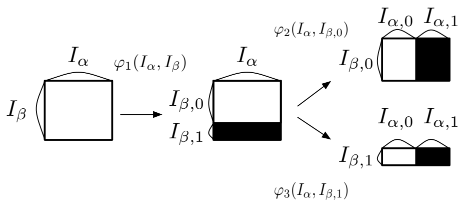
I_α
$\varphi_1(I_\alpha, I_\beta)$
I_β
$I_{\beta,0}$
$I_{\beta,1}$
$\varphi_2(I_\alpha, I_{\beta,0})$
$I_{\alpha,0}$ $I_{\alpha,1}$
$\varphi_3(I_\alpha, I_{\beta,1})$

Fig. 3. Decomposition from a node having the space $I_\alpha \times I_\beta$

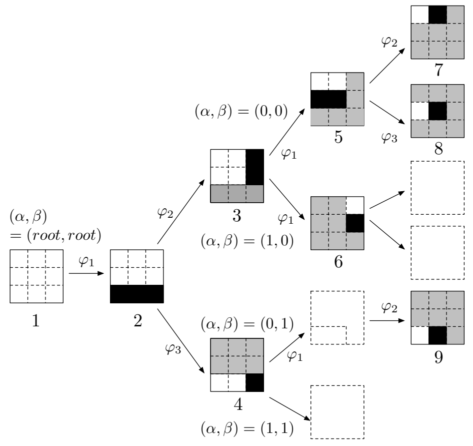

Fig. 4. Basis images based on BIST having 3 leaves.

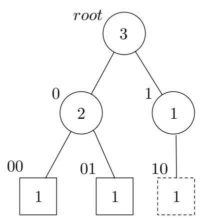

Fig. 5. Succession to the leaf.

(10), and the ninth are given by (18). When $(\alpha, \beta) = (1, 1)$, since both α and β do not have child, no more decomposition is applied. Thus, a set of 9 basis images of size 3×3 is generated.

B. Balanced binary tree and logarithmic binary tree

Tree structured Haar transform has a freedom of the design. Here, we consider the two structures, balanced binary tree-based (B-) and logarithmic binary tree-based (L-) OTSHT.

The balanced binary tree is depth-balanced where the depths of left and right subtrees of each node are within 1. Figure 6 shows an example of BIST generated from a balanced binary tree of the depth 3 having $N = 5$ leaves. Figure 9 shows the set of 25 OTSHT basis images of size 5×5 generated by (16), (17), (18), and (19) from the BIST. There are a total of $r = 11$ rectangle sums with different sizes: 5×5, 5×3, 5×2, 3×3, 3×2, 3×1, 2×3, 2×2, 2×1, 1×2, and 1×1 and $N_h = 4$

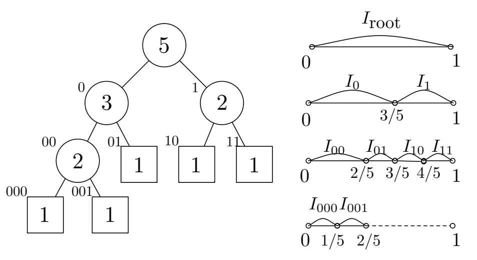

Fig. 6. Balanced binary tree-based BIST.

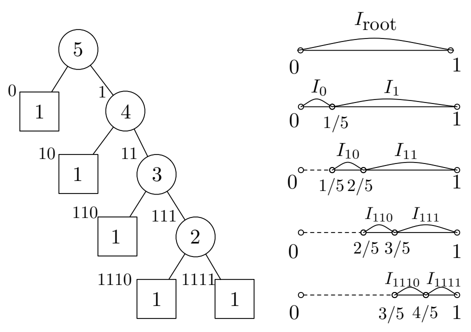

Fig. 8. Logarithmic binary tree-based BIST.

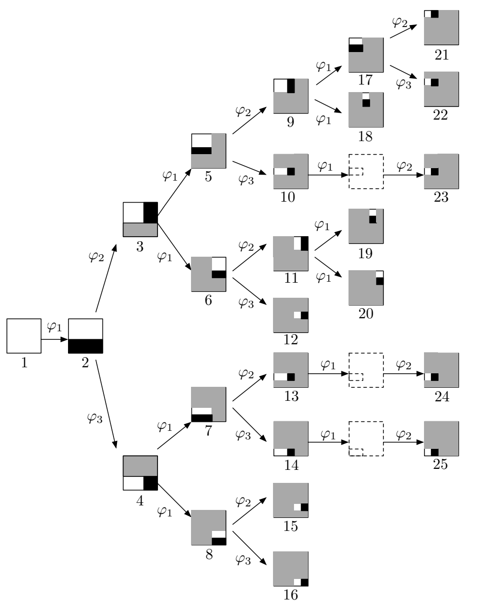

Fig. 7. B-OTSHT basis images having 5 leaves.

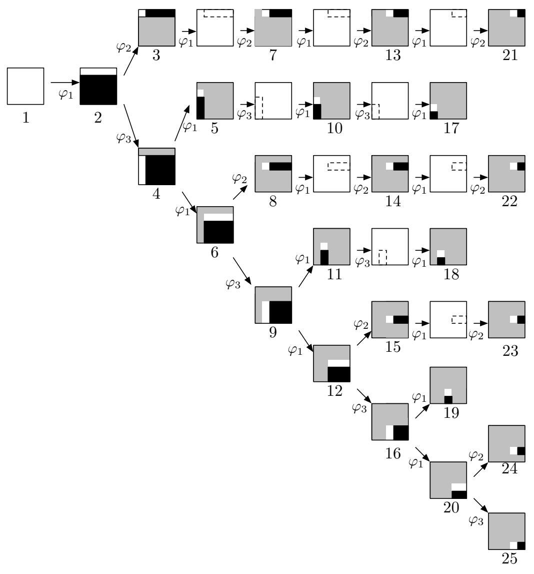

Fig. 9. L-OTSHT basis images having 5 leaves.

different heights: 5, 3, 2, and 1. Note that when N is power of 2, B-OTSHT reduces to OHT.

The logarithmic binary tree is the special case of the Fibonacci p-tree when $p \to \infty$. Figure 8 shows an example of BIST generated from a logarithmic binary tree of the depth 4 having $N = 5$ leaves. Figure 9 shows the set of 25 basis images of size 5×5 generated by (16), (17), (18), and (19) from the BIST. There are a total of $r = 15$ rectangle sums with different sizes: 5×5, 5×4, 5×1, 4×4, 4×3, 4×1, 3×3, 3×2, 3×1, 2×2, 2×1, 1×4, 1×3, 1×2, and 1×1 and $N_h = 5$ different heights: 5, 4, 3, 2, and 1.

C. Computational complexity

The orthonormal tree-sturctured Haar transform has the weight where the elements do not consist of only 1, -1, and 0. As a results, multiplication of the weight is required in some basis images. The weight is multiplied for the partial SSD of candidate windows, i.e., $||(w_p(RS^{+}_{\mathbf{x}_1} - RS^{+}_{\mathbf{x}_2}) - w_n(RS^{-}_{\mathbf{x}_1} - RS^{-}_{\mathbf{x}_2})||^2/\xi^2$, where w_p and w_n are the weight in a positive and a negative value areas, respectively, $RS^{+}_{\mathbf{x}_i}$ and $RS^{-}_{\mathbf{x}_i}$ are rectangle sum of a positive and a negative value areas of $\mathbf{x}_i, (i = 1, 2)$, respectively, and ξ is the normalization factor. if $w_n = w_p$, the weights are included in a normalization factor.

A rectangle sum is calculated by one addition using strip sum. It requires $2 + r + N_h$ additions per pixel for calculating all the projection values of N^2 OTSHT basis images, where r and N_h are the number of different size and different heights, respectively, of rectangle sums in OTSHT. Each of numbers depends on the structure of BIST, which is shown in Table I with the number of the weight, N_{weight}.

D. Simulations

We performed B-OTSHT and L-OTSHT for pattern matching using a template with arbitrary size comparing to OHT. The size of template, $N \times N$, was changed from $N = 5$ to 15. The size of OHT is 8×8 for $N < 8$ and 16×16 for $8 < N$, in which the templates and the candidate windows were zero-padded to adjust the size. All of the results found the the same result as FS. We evaluated the average total number of candidate windows and the average number of basis images for projection values. We used 'Lena' of size

TABLE I
THE NUMBER OF DIFFERENT SIZE, DIFFERENT HEIGHT, AND WEIGHT OF RECTANGLE SUMS IN A SET OF OTSHT BASIS IMAGES

	B-OTSHT			L-OTSHT		
N	r	N_h	N_{weight}	r	N_h	N_{weight}
9	19	6	30	28	9	63
10	13	5	48	32	10	81
11	19	6	54	37	11	101
12	11	5	48	39	12	123
13	23	7	54	42	13	147
14	17	6	48	46	14	173
15	23	7	30	50	15	201

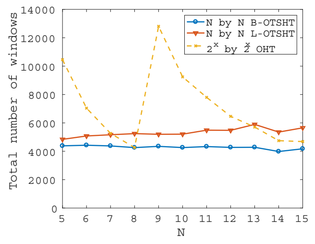

Fig. 10. Average total number of candidate windows for OTSHT and OHT

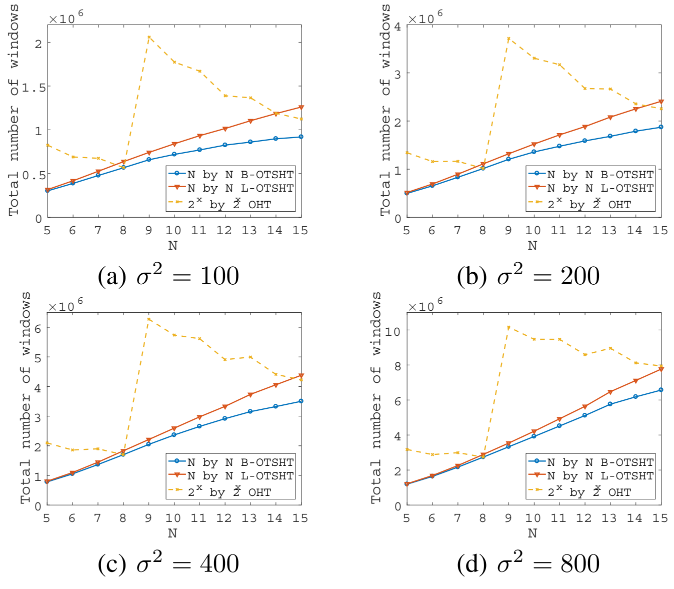

Fig. 11. Average total number of candidate windows for OTSHT and OHT in noisy conditions.

512 $\times$ 512 and set templates whose upper-left position is every 100 pixels in raster scanning. When $N = 5$, a total of 2,581 templates were set and each template is compared with $(512 - N + 1)^2 = 258,064$ candidate windows. If candidate windows are not rejected at all by a threshold in pruning process, a total of $(512 - N + 1)^2 N^2 = 6,451,600$ windows is supposed to be operated. The threshold was set to

$$Th = 1.1 \times SSD_{min} + N^2 \tag{20}$$

where SSD_{min} is the minimum SSD among candidate windows with a template by FS [5]. When the number of remaining candidate windows is less than 0.02% of all the candidate windows, we found the matched window directly.

Figure 10 shows the average total number of candidate windows except for $u = 1$. We observe that the number of windows for B-OTSHT is less than that for L-OTSHT. We confirmed that average number of basis images used in pruning process is $u = 4$ for B-OTSHT, $u = 6$ to 7 for L-OTSHT, and $u = 10$ to 5 for OHT. Figure 11 shows the average total number of candidate windows in noisy condition where the noise is Gaussian noise with zero mean and variance, 100, 200, 400, and 800. We observed that B-OTSHT is more efficient than L-OTSHT and OHT. We confirmed that the average number of basis images used in pruning process for B-OTSHT is less than that for L-OTSHT and OHT, e.g., when $N = 9$ and $\sigma^2 = 100$, the average number of basis images is $u = 64$ for B-OTSHT, $u = 77$ for L-OTSHT, and $u = 195$ for OHT.

IV. Conclusion

We have presented the mathematical expression for two-dimensional orthonormal Haar-like features based on tree-structured Haar transform in order to achieve fast pattern matching using a template with arbitrary size. We have discussed two types of tree structures, balanced tree and logarithmic tree. We have demonstrated that the proposed method with the balanced tree structure is computationally more efficient.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 15K06055

References

[1] R. Dufour, E.Miller, and N.Galatsanos, “Template matching based object recognition with unknown geometric parameters, ” IEEE Trans. Image processing, vol.11, no.12, pp.1385-1396, 2002
[2] L. Ding, A. Goshtasby, and M.Satter, “Volume image registration by template matching,” in Image Visual Computing, vol. 19, no. 12, pp.821-832, 2001
[3] S.Sarraf, C.Saverino, and A.M.Colestani, “A robust and adaptive decision-making algorithm for detecting brain networks using functional mri within the spatial and frequency domain,” IEEE-EMBS International Conference on Biomedical and Health Informatics, pp.53-56, 2016
[4] S.Sarraf, G.Tofighi, et al., “Deepad:Alzheimer disease classification via deep convolutional neural networks using mri and fmri, bioRxiv, p.070441, 2016
[5] W. Ouyang, R. Zhang, and W-K. Cham, “Fast pattern matching using orthogonal Haar transform” IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp.3050-3057, 2010
[6] Y. Li, H.Li, and Z.Cai, “Fast orthogonal Haar transform pattern matching via image square sum,” IEEE Trans. Pattern Anal. Mach. Intell., vol.36, no.9, 2014
[7] W. Ouyang, and W.K. Cham, “Fast algorithm for Walsh Hadamard transform on sliding windows,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no.1, pp.1650171, Jan. 2010
[8] Y. Moshe and H. Hel-Or, “Video block motion estimation based on Gray-code kernels,” IEEE Trans. Image Process., vol.18, pp.2243-2254, 2009
[9] K. Egiazarian and J. Astola, “Tree-structured Haar Transform,” Journal of Mathematical Imaging and Vision, 16: pp.269-279, 2002
[10] Y. Hel-Or and H.Hel-Or, “Real time pattern matching using projection kernels”, IEEE Trans. Pattern Anal. Mach. Intell., 27(9): pp.1430-1445, 2005.