Fast Full Search Equivalent Block Matching for Multichannel Images

Izumi Ito[1] and Aleksandra Pižurica[2]

Abstract – **Block matching is a fundamental tool to search blocks (patches) similar or identical to a given query in image processing. Generally, a full search (FS) algorithm is the most accurate but requires vast computation especially in multichannel images, where the data volume is increasing due to higher definition and more channels. In this paper, we present a fast FS-equivalent algorithm using orthonormal tree-structured Haar transform (OTSHT) for multichannel images. We demonstrate the superior performance of three-dimensional OTSHT comparing with state-of-the-art algorithms. This significant speedup can enable new applications of block matching in multichannel images.**

Keywords – **Block matching, Tree-structured Haar transform, Multichannel image, Full search algorithm.**

I. Introduction

Block matching is a fundamental tool to search blocks (patches) similar or identical to a given query, and as such has been widely used in solving various image processing problems, such as object recognition and tracking [1], image registration [2], analysis [3], and restoration [4]. Generally, a full search (FS) algorithm is the most accurate, which exhaustively compares all pixel intensities of all candidates overlapping each other in sliding window manner, but requires vast computation.

To reduce the computational complexity of the FS algorithm, several fast FS-equivalent algorithms have been studied [5]. The orthogonal Haar transform (OHT) is one of the fastest algorithms, where similarity measure is efficiently evaluated in the transformed domain, but has a limitation that patch size must be power-of-2 [6]. The two-dimensional tree-structured Haar transform (2D-OTSHT) is a generalization of OHT which can apply patches with arbitrary sizes [7], but must apply to each channel separately for multichannel images.

In this paper, we present an FS-equivalent block matching algorithm using three-dimensional orthonormal tree-structured Haar transform (3D-OTSHT) for multichannel images to reduce huge amount of computation involved in increase of the number of channels with higher definition. Using a three-dimensional (3D) integral image, similarity measure is calculated in the transformed domain in order from low to high frequency-like levels, and at each level, candidates are reduced. We focus on reduction of candidates in 3D-OTSHT. Comparing with the basis of state-of-the-art algorithms, 2D-OTSHT and cube matching [8], superior performance of 3D-OTSHT is demonstrated using multichannel images. This paper thus extends our recent work [9]. We provide here additional insights and more extensive analysis of 3D-OTSHT showing its clear potential for fast block matching in multichannel images.

[1]Izumi Ito is with Tokyo Institute of Technology, Information and Communications Engineering, 152-8550 Tokyo, Japan, E-mail: ito@ict.e.titech.ac.jp

[2]Aleksandra Pižurica is with Gent University, Department Telecommunications and Information Processing, 9000 Gent, Belgium, E-mail: Aleksandra.Pizurica@UGent.be

II. Preliminaries

A. Tree-Structured Haar Transform

Tree-structured Haar transform (TSHT) is a generalization of the Haar transform, which can be applied to signals with arbitrary length [10]. TSHT functions are composed of square waves, and a function has an interval at most with a positive constant, an interval with a negative constant, and otherwise zero values. A set of TSHT functions is generated by dividing an interval into two sub-intervals. The complete division of intervals can be expressed by a binary tree structure.

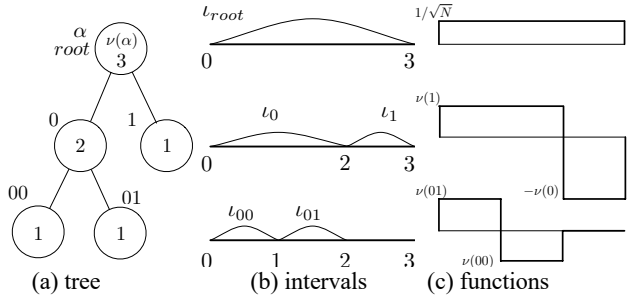
Let α be a node of a binary tree having N leaves for a signal of length N. Let α_0 and α_1 be the left child node and the right child node of α. The TSHT function for interval ι_{root} is given as

$$h(t) = \frac{1}{N}, t \in \iota_{root} \tag{1}$$

and the other TSHT functions for other intervals are

$$h(t) = \begin{cases} \nu(a_1), t \in \iota_{\alpha_0} \\ \nu(a_0), t \in \iota_{\alpha_1} \\ 0, otherwise \end{cases} \tag{2}$$

where $\nu(\alpha)$ represents the number of leaves that α has, and ι_α is the interval derived from α. Fig.1 shows a binary tree having $N = 3$ leaves, the intervals, and TSHT functions, where in the tree, a circle represents a node, and the number in a circle is $\nu(\alpha)$.



(a) tree (b) intervals (c) functions

Fig. 1. A binary tree, the intervals, and TSHT functions.

B. 3D Integral Image

A 3D integral image, $J(x,y,z)$, is built from an image, $I(u,v,w)$, of size $L \times M$ having B bands, for $x = 0,1,2 \dots, L-1$, $y = 0,1,2 \dots, M-1$, and $z = 0,1,2 \dots, B-1$, as

$$J(x+1,y+1,z+1) = \sum_{u=0}^{x}\sum_{v=0}^{y}\sum_{w=0}^{z} I(u,v,w) \tag{3}$$

where $J(0,y,z) = J(x,0,z) = J(x,y,0) = 0$. Observe that $(3B-1)LM$ additions are required to build a 3D integral image.

The region sum (RS) is the sum of all pixel intensities in a region whose diagonal starts at location (sX, sY, sZ) and ends at location (eX, eY, eZ), which can be calculated by seven additions via a 3D integral image regardless of region size as

$$\begin{aligned} RS(region) = {} & J(eX+1, eY+1, eZ+1) \\ & - J(eX+1, eY+1, sZ) \\ & - J(eX+1, sY, eZ+1) \\ & - J(sX, eY+1, eZ+1) \\ & + J(sX, sY, eZ+1) + J(sX, eY+1, sZ) \\ & + J(ex+1, sY, sZ) - J(sX, sY, sZ). \end{aligned} \tag{4}$$

This property is key to significant speedup for calculating transform coefficients.

III. BLOCK MATCHING FOR MULTICHANNEL IMAGES

A.3D-OTSHT

3D-OTSHT consists of a set of basis blocks that forms a basis. For rapid calculation, a basis block of 3D-OTSHT is designed to have at most two regions in it, a positive constant region and a negative constant region.

Let T_X, T_Y, and T_Z be binary trees having N, N, and B leaves for complete division of the X, Y, and Z axes, respectively, for a query of size $N \times N$ having B bands. We use α, β, and γ to denote nodes of T_X, T_Y, and T_Z, respectively. By subdividing a region whose sides are intervals $\iota_\alpha \times \iota_\beta \times \iota_\gamma$, a set of basis blocks is built, which is defined by the following basis block functions.

The basis block function for $\iota_{root} \times \iota_{root} \times \iota_{root}$ is

$$\varphi_0(x,y,z) = \frac{1}{N\sqrt{B}}, (x,y,z) \in \iota_{root} \times \iota_{root} \times \iota_{root} \tag{5}$$

and the other basis block functions are

$$\varphi_1(x,y,z) = \begin{cases} c^+_{\varphi_1}, (x,y,z) \in \iota_{\alpha_0} \times \iota_\beta \times \iota_\gamma \\ c^-_{\varphi_1}, (x,y,z) \in \iota_{\alpha_1} \times \iota_\beta \times \iota_\gamma \\ 0, otherwise \end{cases} \tag{6}$$

$$\varphi_2(x,y,z) = \begin{cases} c^+_{\varphi_2}, (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_0} \times \iota_\gamma \\ c^-_{\varphi_2}, (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_1} \times \iota_\gamma \\ 0, otherwise \end{cases} \tag{7}$$

$$\varphi_3(x,y,z) = \begin{cases} c^+_{\varphi_3}, (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_0} \times \iota_\gamma \\ c^-_{\varphi_3}, (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_1} \times \iota_\gamma \\ 0, otherwise \end{cases} \tag{8}$$

$$\varphi_4(x,y,z) = \begin{cases} c^+_{\varphi_4}, (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_0} \times \iota_{\gamma_0} \\ c^-_{\varphi_4}, (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_0} \times \iota_{\gamma_1} \\ 0, otherwise \end{cases} \tag{9}$$

$$\varphi_5(x,y,z) = \begin{cases} c^+_{\varphi_5}, (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_1} \times \iota_{\gamma_0} \\ c^-_{\varphi_5}, (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_1} \times \iota_{\gamma_1} \\ 0, otherwise \end{cases} \tag{10}$$

$$\varphi_6(x,y,z) = \begin{cases} c^+_{\varphi_6}, (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_0} \times \iota_{\gamma_0} \\ c^-_{\varphi_6}, (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_0} \times \iota_{\gamma_1} \\ 0, otherwise \end{cases} \tag{11}$$

$$\varphi_7(x,y,z) = \begin{cases} c^+_{\varphi_7}, (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_1} \times \iota_{\gamma_0} \\ c^-_{\varphi_7}, (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_1} \times \iota_{\gamma_1} \\ 0, otherwise \end{cases} \tag{12}$$

where

$$c^+_{\varphi_1} = \frac{\nu(\alpha_1)}{\sqrt{\nu(\alpha)\nu(\beta)\nu(\gamma)\nu(\alpha_0)\nu(\alpha_1)}} \tag{13}$$

$$c^-_{\varphi_1} = \frac{-\nu(\alpha_0)}{\sqrt{\nu(\alpha)\nu(\beta)\nu(\gamma)\nu(\alpha_0)\nu(\alpha_1)}} \tag{14}$$

$$c^+_{\varphi_2} = \frac{\nu(\beta_1)}{\sqrt{\nu(\alpha_0)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}} \tag{15}$$

$$c^-_{\varphi_2} = \frac{-\nu(\beta_0)}{\sqrt{\nu(\alpha_0)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}} \tag{16}$$

$$c^+_{\varphi_3} = \frac{\nu(\beta_1)}{\sqrt{\nu(\alpha_1)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}} \tag{17}$$

$$c^-_{\varphi_3} = \frac{-\nu(\beta_0)}{\sqrt{\nu(\alpha_1)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}} \tag{18}$$

$$c^+_{\varphi_4} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_0)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{19}$$

$$c^-_{\varphi_4} = \frac{-\nu(\gamma_0)}{\sqrt{\nu(\alpha_0)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{20}$$

$$c^+_{\varphi_5} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_0)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{21}$$

$$c^-_{\varphi_5} = \frac{-\nu(\gamma_0)}{\sqrt{\nu(\alpha_0)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{22}$$

$$c^+_{\varphi_6} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_1)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{23}$$

$$c^-_{\varphi_6} = \frac{-\nu(\gamma_0)}{\sqrt{\nu(\alpha_1)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{24}$$

$$c^+_{\varphi_7} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_1)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{25}$$

$$c^-_{\varphi_7} = \frac{-\nu(\gamma_0)}{\sqrt{\nu(\alpha_1)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}} \tag{26}$$

Eq. (6) through Eq. (12) are repeatedly used for all nodes, and a total of $K_{\max} = N^2B$ basis blocks are built. Each basis block is assigned a number $k, (k = 1, 2, \dots, K_{\max})$ in building order, where $c^+_{\varphi_i}$ and $c^-_{\varphi_i}, (i = 1, 2, \dots, 7)$ are rewritten as c^+_k and c^-_k, respectively. Fig.2 illustrates the appearance of 3D-OTSHT.

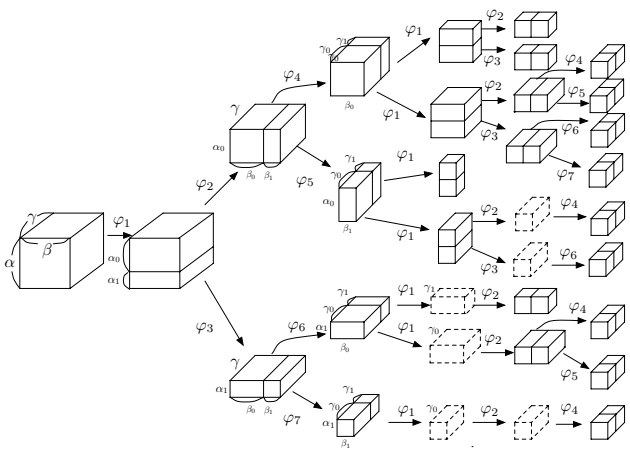


Fig. 2. 3D-OTSHT basis blocks built by subdivision.

B. Block Matching Using 3D-OTSHT via 3D Integral Image

In block matching using 3D-OTSHT via 3D integral image, the sum of square differences (SSD) of 3D-OTSHT coefficients is used as similarity measure to reject patches that do not match.

From Eq. (1), the k-th 3D-OTSHT coefficient, $P_i(k)$, of the i-th patch, $\mathbf{p}_i$, is obtained by

$$P_i(k) = c_k^+ \times RS(region^+) + c_k^- \times RS(region^-) \quad (27)$$

where $region^+$ and $region^-$ are corresponding regions assigned to c_k^+ and c_k^-, respectively, in a 3D integral image.

The SSD of $\mathbf{p}_i$ is calculated at K, $(K = 1,2,\dots,K_{\max})$ as

$$SSD^K(\mathbf{p}_i) = \sum_{k=1}^{K} \big(P_i(k) - Q(k)\big)^2 \quad (28)$$

where $Q(k)$ represents the k-th 3D-OTSHT coefficient of a query. If $SSD^K(\mathbf{p}_i)$ is beyond a threshold, $\mathbf{p}_i$ is rejected from the search, and neither OTSHT coefficients nor SSD is calculated thereafter, which is called pruning.

IV. Evaluation

A. Methods and environments

We performed three fast block matching algorithms, 2D-OTSHT [7], a limited cube matching (LCM), i.e., cube matching [8] without the FS phase, and 3D-OTSHT to evaluate the pruning performance of the basis. These algorithms reduce the number of candidates by pruning based on SSD in the transformed domain in order from low to high frequency-like levels. In 2D-OTSHT, the transformed coefficients are obtained by 2D-OTSHT basis images via a 2D integral image, and SSD is evaluated on each channel separately. In LCM, the transformed coefficients are obtained by 2D-OTSHT basis images via a 3D integral image. Note that other fast FS-equivalent algorithms cannot be performed due to the limitation of patch sizes, and that FS-equivalent algorithms applicable to any patch sizes are desired for high definition images and patch-based denoising.

We used Standard Image Data Base (SIDBA) [11] (dataset 1) containing 12 scenes of size 256×256 with 3 channels, TokyoTech five-band image dataset [12] (dataset 2) containing 11 scenes of size 1824×1368, and TokyoTech 31-band image, 'butterfly' of size 500×500 [13].

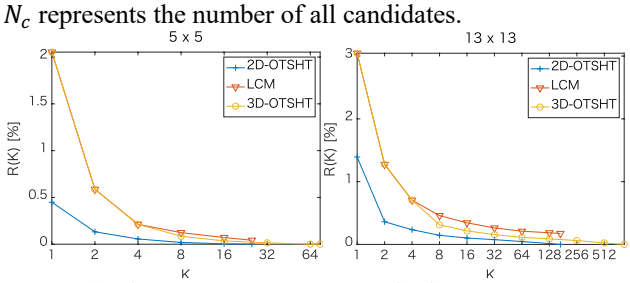
B. Reduction ratio

We chose randomly $N_Q = 10$ queries per patch size, $N \times N$, in a scene of datasets 1 and 2, and set a threshold $Th = N^2B$. Note that in 2D-OTSHT basis images $K_{\max} = N^2$.

The pruning performance is evaluated by the reduction ratio, $R(k)$, of the number of remaining extra patches detected by K basis images/blocks to the number of all candidates, which is defined by

$$R(K) = \frac{N_d(K) - N_g}{N_c} \quad (29)$$

where $N_d(K)$ refers to the number of patches detected by K basis images/blocks, N_g is the number of ground truth patches, and N_c represents the number of all candidates.



(a) dataset 1 (b) dataset 2

Fig. 3. Reduction ratio

Figs. 3(a) and 3(b) show the mean $R(k)$ of in datasets 1 and 2, respectively. Initially, the mean $R(k)$ decreases rapidly, then decreases slowly, as K increases. In 2D-OTSHT and 3D-OTSHT, $R(K_{\max}) = 0$, i.e., the results are the same as the FS algorithm, while in LCM, $R(K_{\max}) \neq 0$, i.e., the results of LCM are not the same as those of the FS. Note that the results of LCM include all the results of the FS. 3D-OTSHT is better than LCM, but worse than 2D-OTSHT in terms of K. However, 2D-OTSHT requires operations on each channel, and takes more time, which will be shown in the next section.

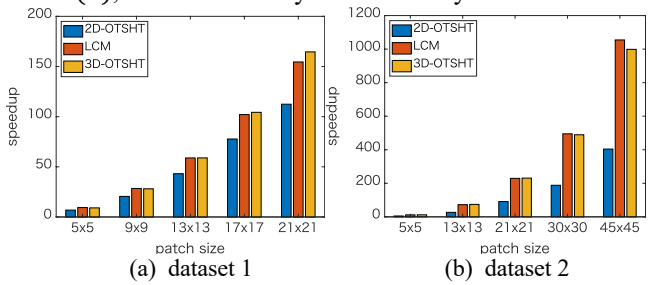
C. Speedup by pruning

We measured speedup over the FS algorithm from the elapsed time required for the three algorithms. All algorithms were written in C as single thread task, compiled with Xcode and ran on a macOS system with 2.5 GHz Intel Core i7 and 16 GB RAM. The speedup over the FS algorithm is defined as

$$Speedup = \frac{T_{\mathrm{FS}}}{T_{\mathrm{ALG}}} \quad (30)$$

where T_{FS} and T_{ALG} represent the elapsed time required for the FS algorithm and an algorithm, respectively. Figs. 4(a) and 4(b) show the mean speedup in datasets 1 and 2, respectively, where $N_Q = 10$ queries were randomly chosen per patch size, $N \times N$, in a scene of datasets 1 and 2, $Th = N^2B$, and $K = K_{\max}$. In dataset 1, 3D-OTSHT is around 9 to 165 times faster than the FS algorithm, and in dataset 2, 3D-OTSHT ran

around 12 and 1,000 times faster, i.e., the larger patch sizes, the higher the efficiency. When $N = 45$, 3D-OTSHT was slightly slower than LCM, because in 3D-OTSHT, $K_{\max} = 10{,}125$, and there are no extra patches, while in LCM, $K_{\max} = 2{,}025$, and there remained extra patches. Together with $R(k)$, the results testify to the efficacy of 3D-OTSHT.



(a) dataset 1 (b) dataset 2

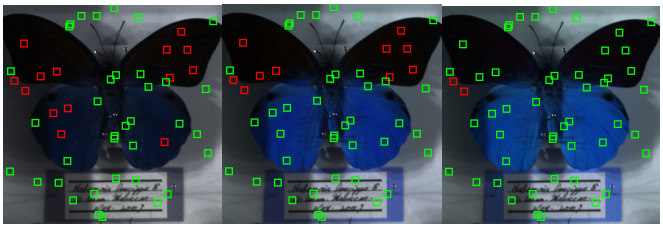
Fig. 4. Speedup over FS algorithm

D. Pruning details

Reduction ratio of candidates varies depending on queries. We focus on each query in pruning with condition that candidates are reduced to $S = 0.02$ percent of all candidates to identify the patch. We performed LCM and 3D-OTSHT in limited B-band images generated from image 'butterfly', where $Th = N^2B$ and $B = 5$, 10, 15, 20, 25, and 31. 50 queries of size $N \times N$, $(N = 16)$ were randomly chosen.

In some queries, LCM failed to reduce candidates. Fig. 5 shows 50 queries where queries that LCM succeeded/failed to reduce are expressed in green/red squares in $B = 5, 10$, and 15. The results in $B = 20$, 25, and 31 are the same as those in $B = 15$. We observed that LCM failed when queries are homogeneous areas. 3D-OTSHT, on the other hand, succeeded to reduce in all queries. In the queries that LCM failed to reduce, the mean K-values of 3D-OTSHT in $B = 5$ and 10 were 980 and 1,776 (around 77 and 70 percent of $K_{\max}$), respectively, and $K = K_{\max}$, in $B = 15$ to 31.

Practically, 3D-OTSHT is more efficient when used in combination with the FS algorithm than 3D-OTSHT alone, because direct calculation is more efficient than calculation via a 3D integral image in higher K-values, as shown in [10]. However, the threshold to reject candidates, and parameters (K and S) to terminate the pruning process should be carefully chosen depending on queries and machine capability for significant speedup. For example, when queries are homogeneous, a lower Th or an appropriate S-value with $K = K_{\max}$ works, and when queries have distinct contrast, a lower K-value can work well.



$B = 5$ $B = 10$ $B = 15$

Fig. 5. 50 queries. Queries that LCM succeeded/failed to reduce are expressed in green/red squares.

V. Conclusion

In this paper we presented new insights into our 3D-OTSHT approach for block matching in multichannel images with more extensive analysis. Using a 3D integral image, the similarity measure is evaluated in the transformed domain in order from low to high frequency-like levels to reduce the number of candidates, and thereby achieves significant speedup. Comparing 3D-OTSHT with state-of-the-art algorithms, 2D-OTSHT and LCM, in terms of reduction ratio, speedup, and pruning details, we showed that 3D-OTSHT is advantageous over the others. Fast search of patches in multichannel images is a fundamental problem and hence 3D-OTSHT has many applications. Moreover, 3D-OTSHT can enable new applications in multichannel images that were not feasible before due to prohibitive computational complexity.

References

[1] R. Dufour, R, E. Miller, N. Galatsanos, "Template matching based object recognition with unknown geometric parameters." IEEE Transactions on Image Processing. vol.11, pp.1385-1396, 2002.

[2] L. Ding, A. Goshtasby, M. Satter, "Volume image registration by template matching." Image and Vision Computing, vol. 19, pp. 821-832, 2001.

[3] S. Sarraf, C. Saverino, A.M. Colestani, "A robust and adaptive decision-making algorithm for detecting brain networks using functional MRI within the spatial and frequency domain." In Proceedings of the IEEE-EMBS International Conference on Biomedical and Health Informatics, pp. 53-56, 2016.

[4] V. Papyan, M. Elad, "Multi-scale patch-based image restoration." IEEE Transactions on Image Processing, vol.25, pp. 249-261, 2016.

[5] W.Ouyang, F. Tombari, S. Mattoccia, L.D.Stefano, and W.-K. Cham, "Performance evaluation of full search equivalent patten matching algorithms," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 34, no. 1, pp.127-143, 2012.

[6] W. Ouyang, W. Zhao, W.-K. Cham, L. WeiFast, "Fast full-search-equivalent pattern matching using asymmetric Haar wavelet packets." IEEE Transactions on Circuits and Systems for Video Technology, vol. 28, pp.819-833, 2018.

[7] I. Ito, K. Egiazarian, "Two-dimensional orthonormal tree-structured Haar transform for fast block matching." Journal of Imaging, 4, pp.1-18, 2018.

[8] I. Ito, A. Pižurica, "Fast cube matching using orthogonal tree-structured Haar transform for multispectral images." In Proceedings of the 11th International Symposium on Image and Signal Processing and Analysis, pp. 70-75, 2019.

[9] I. Ito, A. Pižurica, "Three-dimensional block matching using orthonormal tree-structured Haar transform for multichannel images," Journal of Imaging, vol.6, issue 4, pp.1-18, 2020.

[10] K. Egiazarian, J. Astola, "Tree-structured Haar transform." Journal of Mathematical Imaging and Vision, 16, pp.269-279, 2002.

[11] Standard Image Data Base (SIDBA). Available online: http://www.ess.ic.kanagawa-it.ac.jp/app_images_j.html

[12] Y. Monno, M.Tanaka, M.Okutomi, TokyoTech 5-Band Multispectral Image Dataset and Demosaiking Codes. Available online: http://www.ok.sc.e.titech.ac.jp/res/MSI/MSIdata.html

[13] --, TokyoTech 31-Band Hyperspectral Image Dataset. Available online: http://www.ok.sc.e.titech.ac.jp/res/MSI/MSIdata31.html