Journal of *Imaging*

MDPI

Article

Three-Dimensional Block Matching Using Orthonormal Tree-Structured Haar Transform for Multichannel Images

Izumi Ito [1,*] and Aleksandra Pižurica [2]

[1] Information and Communications Engineering, Tokyo Institute of Technology, Tokyo 152-8552, Japan
[2] Department Telecommunications and Information Processing, Ghent University, 9000 Gent, Belgium; Aleksandra.Pizurica@UGent.be
* Correspondence: ito@ict.e.titech.ac.jp; Tel.: +81-3-5734-2997

Received: 11 November 2019; Accepted: 6 February 2020; Published: 11 February 2020



Abstract: Multichannel images, i.e., images of the same object or scene taken in different spectral bands or with different imaging modalities/settings, are common in many applications. For example, multispectral images contain several wavelength bands and hence, have richer information than color images. Multichannel magnetic resonance imaging and multichannel computed tomography images are common in medical imaging diagnostics, and multimodal images are also routinely used in art investigation. All the methods for grayscale images can be applied to multichannel images by processing each channel/band separately. However, it requires vast computational time, especially for the task of searching for overlapping patches similar to a given query patch. To address this problem, we propose a three-dimensional orthonormal tree-structured Haar transform (3D-OTSHT) targeting fast full search equivalent for three-dimensional block matching in multichannel images. The use of a three-dimensional integral image significantly saves time to obtain the 3D-OTSHT coefficients. We demonstrate superior performance of the proposed block matching.

Keywords: block matching; Haar transform; multichannel image; multispectral image; color image

1. Introduction

Block matching is a fundamental tool used to search for blocks (patches) similar to a given query. It has been widely used in solving various image processing problems, like object recognition and tracking [1], image registration [2], image analysis [3], image restoration [4], to name a few. Block matching searches for patches of the same size as a query and is sensitive to deformation. In this sense, it is different from some common image descriptors such as scale-invariant feature transform (SIFT) [5] and speed-up robust features (SURF) [6], which extract features robust to deformation. In block matching, generally, a full search (FS) algorithm that exhaustively compares all the pixel intensities of all the candidates overlapping each other is the most accurate, but requires vast computation due to the huge number of candidates in a large search space. The larger the image size and number of image bands, the harder it is to use FS.

Fast block matching has been studied from algorithms and architectures. There are several works of architecture such as hardware acceleration and configuration by custom instructions specially for motion estimation in video coding [7–11]. There are also two categories in the algorithms, FS-equivalent algorithms and non-FS-equivalent algorithms. Some non-FS-equivalent algorithms such as three-step-search [12] and diamond search [13] reduce the amount of computation by limiting search areas, and others do so by approximating patterns [14,15], where there is a trade-off between accuracy and efficiency. The scope of this paper is limited to the FS-equivalent algorithm.

Fast FS-equivalent algorithms have been intensively developed in order to address the computational complexity with FS [16–19]. Although these methods can be applied only to the patches of size power-of-2, they prove that the search in a transformed domain is much more efficient than the search in a spatial domain. The additional calculation to obtain the transform coefficients is fully compensated by the reduction of candidates. The orthogonal Haar transform (OHT) reportedly performs fastest in this setting [20,21]. One of the reasons is that there is a unique way to calculate the transform coefficients using an integral image [22,23]. Once an integral image is built, the transform coefficients can be calculated with a few arithmetic operations. This is based on the fact that the Haar transform matrix is sparse and composed of rectangular functions, which is much different from other transforms. In order to overcome the limitation of patch sizes, two-dimensional orthonormal tree-structured Haar transform (2D-OTSHT), the generalization of OHT, was proposed [24]. Compared to OHT, the normalization factor of 2D-OTSHT is not an integer number, but this has little effect in the whole speedup.

In this paper, we propose three-dimensional orthonormal tree-structured Haar transform (3D-OTSHT) with three-dimensional integral image for multichannel images. In the proposed 3D-OTSHT, the transform coefficients can be obtained by a few arithmetic operations regardless of patch size. Focusing on the pruning performance in the transformed domain, we consider the combination with FS, where the pruning process is stopped at a certain level of reduction of candidates. We demonstrate superior results regarding the savings in computation time compared to the straightforward solution, where the fast FS-equivalent method for grayscale images is applied to each band separately. Experimental results are obtained using a standard dataset for color images and a five-band multispectral image dataset.

The paper is organized as follows: In Section 2, we provide the problem setting and required techniques as preliminaries. We present the design of 3D-OTSHT for three-dimensional (3D) integral image and the 3D block matching using 3D-OTSHT in Section 3. Our evaluations of the pruning performance and speedup are detailed in Section 4. Finally, in Section 5, we conclude our study.

2. Preliminaries

First, the problem targeted in this paper is stated. Next, a couple of techniques required for the proposed method are briefly described.

2.1. Problem Statement

Consider the problem of searching for patches similar to a given query in a multichannel image. In the full search (FS) algorithm, the matching patches are detected with a threshold in a sliding window manner by measuring the similarity of all the candidates in the whole search space. Generally, the sum of squared differences (SSD) of all the intensities in a candidate is used as the similarity.

Let $q(x,y,z)$ be a query patch of size $N \times N$ having B bands, and $I(u,v,w)$ measure an image of size $M \times M$ having B bands. The SSD of all the intensities of all the candidates is calculated, for $0 \leq u < M-N$, $0 \leq v < M-N$, and $0 \leq w < B$, as

$$SSD(u,v,w) = \sum_{u=0}^{M-N} \sum_{v=0}^{M-N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \sum_{z=0}^{B-1} \left(I(u+x,v+y,w+z) - q(x,y,z)\right)^2. \quad (1)$$

It turns out that $(2(M-N+1)^2N^2B-1)$ additions and $(M-N+1)^2N^2B$ multiplications are required for the search. The aim of this paper is to reduce the computational complexity for search in multichannel images while keeping the same accuracy as FS using the same threshold as FS uses.

A part of $I(u,v,w)$, i.e., the i-th candidate, and a query are hereafter simply expressed as $\mathbf{p}_i$ and $\mathbf{q}$, respectively, in a vector form, e.g., the SSD of the i-th candidate is represented as

$$SSD_i = ||\mathbf{p}_i - \mathbf{q}||_2^2. \quad (2)$$

2.2. 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 [25]. In the conventional Haar transform [26], the basis is built by dividing an interval equally into two intervals. In TSHT, on the other hand, the basis is built by dividing an interval unequally into two intervals putting weights on them. The complete division of the intervals can be expressed by a binary tree structure.

Figure 1a shows an example of a binary tree having 3 leaves. A circle represents a node. The topmost node is the root. The node having no connection below (no child node) is a leaf. The number inside a circle represents the number of leaves the node has, which determines the internal division ratio when dividing an interval and the weight of intervals in the basis function. Figure 1b shows the intervals corresponding to the nodes of the binary tree. The interval corresponding to a node is divided internally in two subintervals, in a ratio equal to the number of leaves of the left child node to that of the right child node.

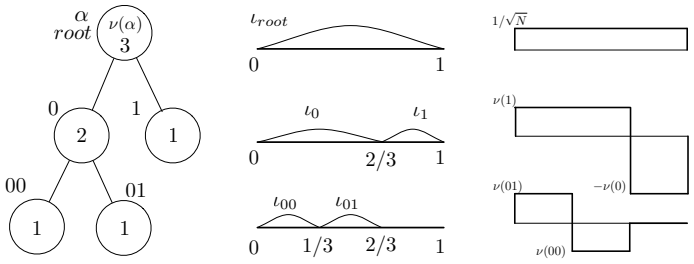
Let α be a node of a binary tree having N leaves. Let α_0 and α_1 be the left child node and the right child node of α, respectively. We denote by ι_α the interval corresponding to α. The basis function for interval ι_{root} is given as

$$h(t) = \frac{1}{\sqrt{N}}, \quad t \in \iota_{root} \tag{3}$$

and

$$h(t) = \begin{cases} \nu(\alpha_1), & t \in \iota_{\alpha_0} \\ -\nu(\alpha_0), & t \in \iota_{\alpha_1} \\ 0, & \text{otherwise,} \end{cases} \tag{4}$$

where $\nu(\alpha)$ represents the number of leaves that α has. Thus, except for ι_{root}, the absolute value of the weight of two intervals is inversely proportional to the number of leaves. Figure 1c shows the basis.



(**a**) a tree (**b**) intervals (**c**) basis

Figure 1. Binary tree and corresponding intervals for the basis. (**a**) Binary tree having $N = 3$ leaves, where each node is labeled with 0 and 1 outside the circle except for root; (**b**) intervals corresponding to the nodes; (**c**) the basis built from the intervals.

2.3. Three-Dimensional Integral Image

The three-dimensional (3D) integral image $J(x, y, z)$ is generated from an image, $I(u, v, w)$, of size $M \times M$ having B bands, for $x = 0, 1, 2, \ldots, M$; $y = 0, 1, 2, \ldots, M$; and $z = 0, 1, 2, \ldots, B$, as

$$J(x, y, z) = \sum_{u=0}^{x-1} \sum_{v=0}^{y-1} \sum_{w=0}^{z-1} I(u, v, w), \tag{5}$$

where $J(0, y, z) = 0$, $J(x, 0, z) = 0$, and $J(x, y, 0) = 0$. Observe that $(3B - 1)M^2$ additions are required to build a 3D integral image.

With the 3D integral image, the sum of all the intensities in region A (ABCD-EFGH) whose diagonal starts at the location (sX, sY, sZ) and ends at (eX, eY, eZ), as shown in Figure 2, is calculated by seven additions regardless of region size as

$$\begin{aligned} regionSum(A) = {} & J(eX+1, eY+1, eZ+1) - J(eX+1, eY+1, sZ) - J(eX+1, sY, eZ+1) \\ & - J(sX, eY+1, eZ+1) + J(sX, sY, eZ+1) + J(sX, eY+1, sZ) + J(eX+1, sY, sZ) \\ & - J(sX, sY, sZ). \end{aligned} \tag{6}$$

This property is a key to significant speedup of the proposed method.

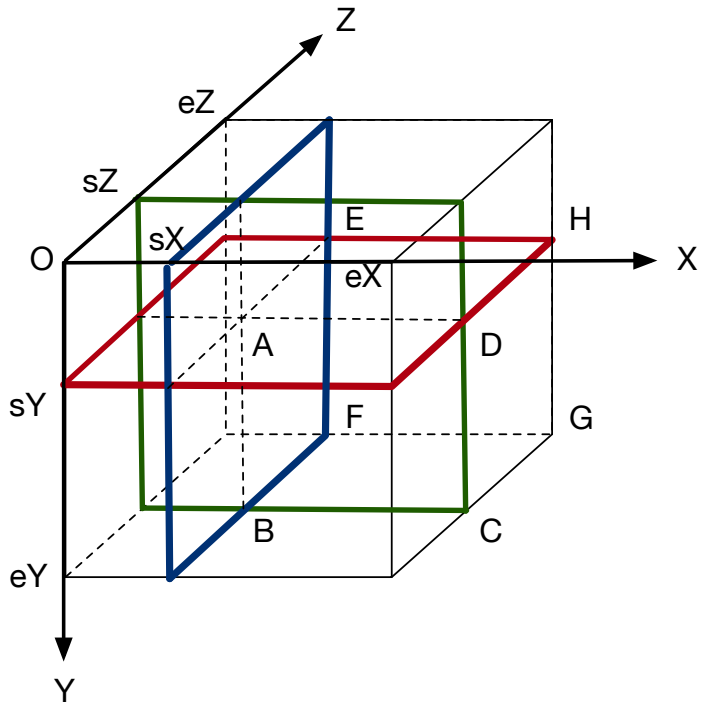


Figure 2. Specific property of 3D integral image. The sum of the intensities in region ABCD-EFGH whose diagonal starts at A:(sX, sY, sZ) and ends at G:(eX, eY, eZ) is obtained by seven additions.

3. Three-Dimensional Block Matching for Multichannel Images

We propose here fast FS-equivalent three-dimensional (3D) block matching for multichannel image using three-dimensional orthonormal tree-structured Haar transform (3D-OTSHT). First, we construct 3D-OTSHT to simplify the computation to obtain its coefficients. Next, we describe the 3D block matching using 3D-OTSHT.

3.1. Three-Dimensional Orthonormal Tree-Structured Haar Transform

One of the vectors forming the basis of the vector space for a three-dimensional region is referred to in this paper as a basis block. The 3D-OTSHT consists of the basis blocks built by subdividing a three-dimensional region formed by intervals along X, Y, and Z axis. For rapid calculation of 3D-OTSHT coefficients via integral image, we design the basis block to have at most two regions in it, each of which is assigned to a constant.

Let us consider the basis blocks for a query of size $N \times N$ having B bands. We generate the set of basis blocks by basis block functions. Let T_X, T_Y, and T_Z be the binary trees for X axis having N

leaves, Y axis having N leaves, and Z axis having B leaves, respectively. The nodes of T_X, T_Y, and T_Z are denoted by α, β, and γ, respectively.

We define the basis block function for region $(\iota_{root} \times \iota_{root} \times \iota_{root})$ as

$$\varphi_0(x,y,z) = \frac{1}{N\sqrt{B}}, \quad (x,y,z) \in \iota_{root} \times \iota_{root} \times \iota_{root} \tag{7}$$

and the basis block functions for the other regions as

$$\varphi_1(x,y,z) = \begin{cases} c^+_{\varphi 1}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_\beta \times \iota_\gamma \\ c^-_{\varphi 1}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_\beta \times \iota_\gamma \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

$$\varphi_2(x,y,z) = \begin{cases} c^+_{\varphi 2}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_0} \times \iota_\gamma \\ c^-_{\varphi 2}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_1} \times \iota_\gamma \\ 0, & \text{otherwise} \end{cases} \tag{9}$$

$$\varphi_3(x,y,z) = \begin{cases} c^+_{\varphi 3}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_0} \times \iota_\gamma \\ c^-_{\varphi 3}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_1} \times \iota_\gamma \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

$$\varphi_4(x,y,z) = \begin{cases} c^+_{\varphi 4}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_0} \times \iota_{\gamma_0} \\ c^-_{\varphi 4}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_0} \times \iota_{\gamma_1} \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

$$\varphi_5(x,y,z) = \begin{cases} c^+_{\varphi 5}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_1} \times \iota_{\gamma_0} \\ c^-_{\varphi 5}, & (x,y,z) \in \iota_{\alpha_0} \times \iota_{\beta_1} \times \iota_{\gamma_1} \\ 0, & \text{otherwise} \end{cases} \tag{12}$$

$$\varphi_6(x,y,z) = \begin{cases} c^+_{\varphi 6}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_0} \times \iota_{\gamma_0} \\ c^-_{\varphi 6}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_0} \times \iota_{\gamma_1} \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$\varphi_7(x,y,z) = \begin{cases} c^+_{\varphi 7}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_1} \times \iota_{\gamma_0} \\ c^-_{\varphi 7}, & (x,y,z) \in \iota_{\alpha_1} \times \iota_{\beta_1} \times \iota_{\gamma_1} \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

where $c^+_{\varphi n}$ and $c^-_{\varphi n}$, $(n = 1, 2, \ldots, 7)$ are a positive constant and a negative constant, respectively, which includes normalization factor and weight:

$$c^+_{\varphi 1} = \frac{\nu(\alpha_1)}{\sqrt{\nu(\alpha)\nu(\beta)\nu(\gamma)\nu(\alpha_0)\nu(\alpha_1)}}, \quad c^-_{\varphi 1} = -\frac{\nu(\alpha_0)}{\sqrt{\nu(\alpha)\nu(\beta)\nu(\gamma)\nu(\alpha_0)\nu(\alpha_1)}}, \tag{15}$$

$$c^+_{\varphi 2} = \frac{\nu(\beta_1)}{\sqrt{\nu(\alpha_0)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}}, \quad c^-_{\varphi 2} = -\frac{\nu(\beta_0)}{\sqrt{\nu(\alpha_0)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}}, \tag{16}$$

$$c^+_{\varphi 3} = \frac{\nu(\beta_1)}{\sqrt{\nu(\alpha_1)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}}, \quad c^-_{\varphi 3} = -\frac{\nu(\beta_0)}{\sqrt{\nu(\alpha_1)\nu(\beta)\nu(\gamma)\nu(\beta_0)\nu(\beta_1)}}, \tag{17}$$

$$c^+_{\varphi 4} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_0)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \quad c^-_{\varphi 4} = -\frac{\nu(\gamma_0)}{\sqrt{\nu(\alpha_0)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \tag{18}$$

$$c^+_{\varphi 5} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_0)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \quad c^-_{\varphi 5} = -\frac{\nu(\gamma_0)}{\sqrt{\nu(\alpha_0)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \tag{19}$$

$$c^+_{\varphi 6} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_1)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \quad c^-_{\varphi 6} = -\frac{\nu(\gamma_0)}{\sqrt{\nu(\alpha_1)\nu(\beta_0)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \tag{20}$$

$$c^+_{\varphi 7} = \frac{\nu(\gamma_1)}{\sqrt{\nu(\alpha_1)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}, \quad c^-_{\varphi 7} = -\frac{\nu(\gamma_0)}{\sqrt{\nu(\alpha_1)\nu(\beta_1)\nu(\gamma)\nu(\gamma_0)\nu(\gamma_1)}}. \tag{21}$$

A series of functions (8) to (14) is repeated at all the intervals corresponding to the nodes of three binary trees. Eventually, N^2B basis blocks are generated. Each basis block is indicated as $\mathbf{G}_k$, ($k = 1, 2, \ldots, N^2B$) in a vector form. As such, $c^+_{\varphi n}$ and $c^-_{\varphi n}$ are replaced by c^+_k and c^-_k, respectively. Let $\mathbf{G}$ be the set of basis blocks, i.e., $\mathbf{G} = [\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_{N^2B}]^T$, where $.^T$ denotes the transposition. $\mathbf{G}$ is orthonormal.

Figure 3 shows the appearance of 3D-OTSHT. Function φ_1 divides a region along X axis into two regions, c^+_* and c^-_*, where c^+_* and c^-_* are the region assigned to $c^+_{\varphi n}$ and $c^-_{\varphi n}$ as in (15) through (21), respectively. Functions φ_2 and φ_3 divide c^+_* and c^-_*, respectively, built by φ_1 along Y axis; φ_4 and φ_5 divide c^+_* and c^-_*, respectively, built by φ_2 along Z axis; φ_6 and φ_7 divide c^+_* and c^-_*, respectively, built by φ_3 along Z axis.

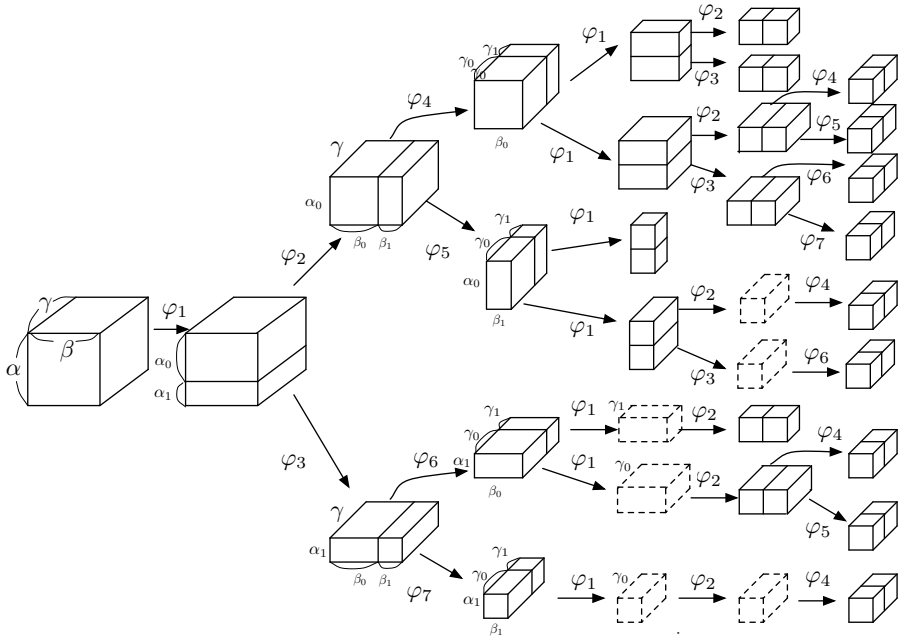


Figure 3. Three-dimensional orthonormal tree-structured Haar transform (3D-OTSHT) basis blocks built by subdivision.

3.2. 3D Block Matching Using 3D-OTSHT with 3D Integral Image

In the proposed 3D block matching, SSD of 3D-OTSHT coefficients is calculated as similarity, but not all the transform coefficients of all the candidates are used. The candidate that does not match is rejected from the search in the middle of processing, which is called pruning.

The transform coefficients are obtained efficiently with a 3D integral image. From (6), the k-th 3D-OTSHT coefficient, $P(k)$, of a patch, $\mathbf{p}$, in a vector form is obtained as

$$\begin{aligned} P(k) &= \mathbf{G}_k \mathbf{p} \\ &= c^+_k \times regionSum(\mathbf{p}^+_*) + c^-_k \times regionSum(\mathbf{p}^-_*), \end{aligned} \tag{22}$$

where $\mathbf{p}_*^+$ and $\mathbf{p}_*^-$ are the regions in the patch corresponding to the regions to which c_k^+ and c_k^- are assigned in the k-th basis block, respectively.

For pruning, the similarity of candidates is calculated using a subset of 3D-OTSHT. Let $\mathbf{G}^k$ be the subset of $\mathbf{G}$ that contains the first k basis blocks, i.e., $\mathbf{G}^k = [\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_k]^T$. The similarity of the i-th candidate is calculated with $\mathbf{G}^k$ as

$$SSD_i^k = ||\mathbf{P}_i^k - \mathbf{Q}^k||_2^2, \tag{23}$$

where $\mathbf{P}_i^k = \mathbf{G}^k\mathbf{p}_i$ and $\mathbf{Q}^k = \mathbf{G}^k\mathbf{q}$. At every k ($k = 1, 2, \ldots, N^2B$), SSD_i^k is judged with a threshold. Once SSD_i^k is beyond the threshold, the i-th candidate is rejected from the search and neither OTSHT coefficient nor SSD is calculated afterward. As long as using the same threshold as FS uses, the unmatched candidates are securely rejected. The theory behind this is that if the transform is orthonormal, the energy of a signal is not changed in the transformed domain. Therefore, it holds that

$$||\mathbf{P}_i - \mathbf{Q}||_2^2 = ||\mathbf{p}_i - \mathbf{q}||_2^2, \tag{24}$$

where $\mathbf{P}_i = \mathbf{G}\mathbf{p}_i$ and $\mathbf{Q} = \mathbf{G}\mathbf{q}$. From $||\mathbf{P}_i^k - \mathbf{Q}^k||_2^2 \leq ||\mathbf{P}_i - \mathbf{Q}||_2^2$ for $k = 1, 2, \ldots, N^2B$, secure rejection is made. For this reason, the transform should be orthonormal and SSD used as the similarity measure.

3.3. 3D Block Matching Using Limited 3D-OTSHT

All the basis blocks of 3D-OTSHT can detect patches with the same accuracy as FS. However, it is inefficient to use all the basis blocks because $\mathbf{G}_k$ becomes sparser as k increases. To avoid this, the limited number of basis blocks is used for pruning, and after the number of candidates is reduced, the remaining candidates are scrutinized by FS. That is, $\mathbf{G}^K$ with a certain K is used instead of $\mathbf{G}$ for SSD_i^k shown in (23), and at every k, ($k = 1, 2, \ldots, K$), SSD_i^k is judged with a threshold for pruning.

4. Evaluation

We performed 3D block matching in order to evaluate the pruning performance and elapsed time of the proposed method using multichannel images.

4.1. Methods and Environments

We compared the performance of the following five methods for search: FS, two-dimensional OTSHT with a two-dimensional integral image by single judge (2D-OTSHT-2DI-S) [24], two-dimensional OTSHT with two-dimensional integral image by whole judge (2D-OTSHT-2DI-W), two-dimensional OTSHT with a 3D integral image (2D-OTSHT-3DI) [27], and the proposed 3D-OTSHT with 3D integral image (3D-OTSHT-3DI). 2D-OTSHT-2DI-S and -W use the OTSHT for grayscale images on each channel, and judge the candidates in different ways: The single judge (-S) decides whether to reject the candidate or not based on a single channel, while the whole judge (-W) evaluates the candidate based on all the channels. In 2D-OTSHT-3DI, a basis image forms a basis block for the 3D integral image so that the same basis image is piled up.

We use two image data sets, color image data set and five-band multispectral image data set. The color image data set, called SIDBA, contains 12 scenes of size 256 × 256 having 3 color channels [28]. The 5-band image data set contains 11 scenes of size 1824 × 1368 having 5 bands [29]. For each dataset, five patch sizes are used. We chose 10 queries randomly in an image per patch size. Then, we obtain the ground truth patches similar to the query by FS with a threshold. If the SSD of the i-th patch holds

$$||\mathbf{p}_i - \mathbf{q}||_2^2 <= threshold, \tag{25}$$

we set the i-th patch a ground truth. In this experiment, the threshold is $10N^2B$ for the color images and $2N^2B$ for the 5-band multispectral images. The same threshold is used in all the methods. Table 1 summarizes the number of ground truth patches. From Table 1, it can be seen that the mean number

of ground truth patches tends to decrease as the patch size increases and increase as the image size increases. We confirmed that the number of ground truth patches chosen by the queries with low standard deviations of pixel intensities is likely to be large, and that the number of ground truth patches chosen by the queries with high standard deviations is likely to be one. Therefore, the threshold should be set appropriately considering the size and characteristics of images in practical applications. Generally, the larger threshold yields more ground truth patches. However, a threshold that is too large will be meaningless.

All the algorithms are written in C as single thread tasks, compiled with Xcode 10.1, and run on a macOS system with 4 GHz Intel core i7 and 16 GB RAM, where eight active processor cores with hardware multithreading are used.

Table 1. The number of ground truth patches.

Data Set	Image Size	Band	Scenes	Patch Size	Samples	Number of Patches		
						Min.	Mean	Max.
1	256 × 256	3	12	5 × 5	120	1	265	7541
				9 × 9	120	1	308	6097
				13 × 13	120	1	234	4937
				17 × 17	120	1	51	1564
				21 × 21	120	1	4	127
2	1824 × 1368	5	11	5 × 5	110	1	6158	103,161
				13 × 13	110	1	5634	113,573
				21 × 21	110	1	2508	89,534
				30 × 30	110	1	1543	85,131
				45 × 45	110	1	1882	74,967

4.2. Pruning Performance

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

$$R(K) = \frac{N_d(K) - N_g}{N_c} \times 100, \tag{26}$$

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 the candidates. Lower $R(K)$ means better performance.

Figures 4 and 5 show the pruning performance, $R(K)$, $(K = 2, 4, 8, \ldots, K_{max})$, in the color images and the 5-band multispectral images, respectively, where K_{max} refers to the maximum number of basis images/blocks. In the proposed 3D-OTSHT-3DI, $K_{max} = N^2B$, while in the other methods, $K_{max} = N^2$. We confirmed that all the methods detect every ground truth patch at any K, i.e., there are no false negatives. Therefore, the final accuracy (the accuracy after the remaining candidates are scrutinized by FS) is the same as the accuracy of FS. The proposed 3D-OTSHT-3DI method yields the best pruning performance both on 3-band and 5-band images for all the patch sizes. In both image data sets, we observe that when K is greater than or equal to 8, 3D-OTSHT-3DI is better than the other methods and that as K increases, the rate of change reduces. These facts suggest that not all the basis blocks are required for the whole speedup.

Figure 6 shows an example of patches detected at K basis images/blocks by different methods and for different values of K. Every one of the detected patches of size 13 × 13 is shown in an orange square, and the detected overlapping patches cover different areas depending on the employed method and the K-value. It can be seen that the amount of candidate patches reduces as K increases. The result of 3D-OTHST-3DI does not differ significantly between $K = 8$ and $K = 16$ and, clearly, these results agree best with the ground truth.

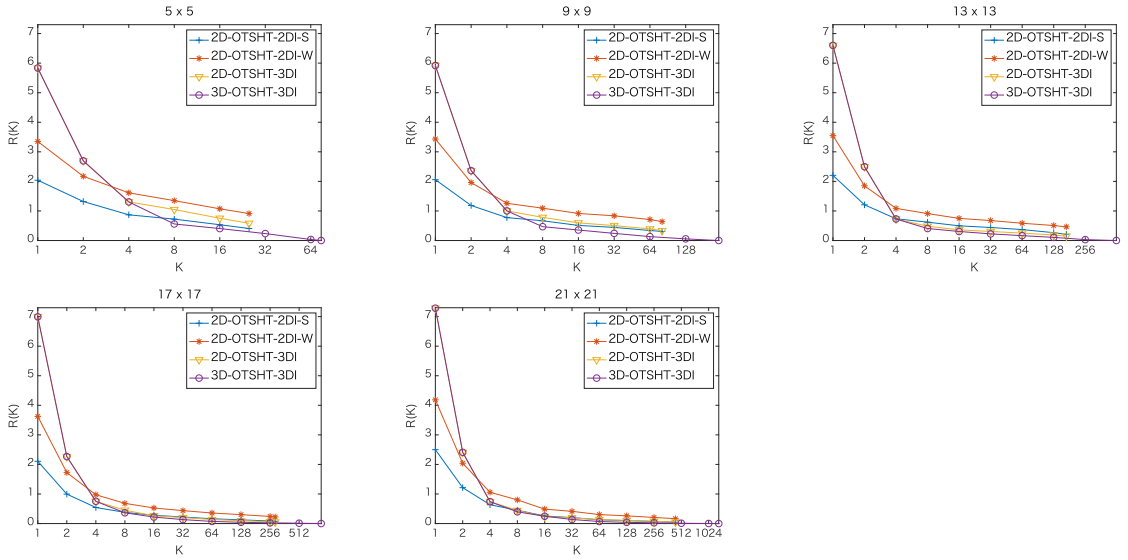


Figure 4. Percentage of extra patches remaining at K basis images/blocks in color images of size 256×256.

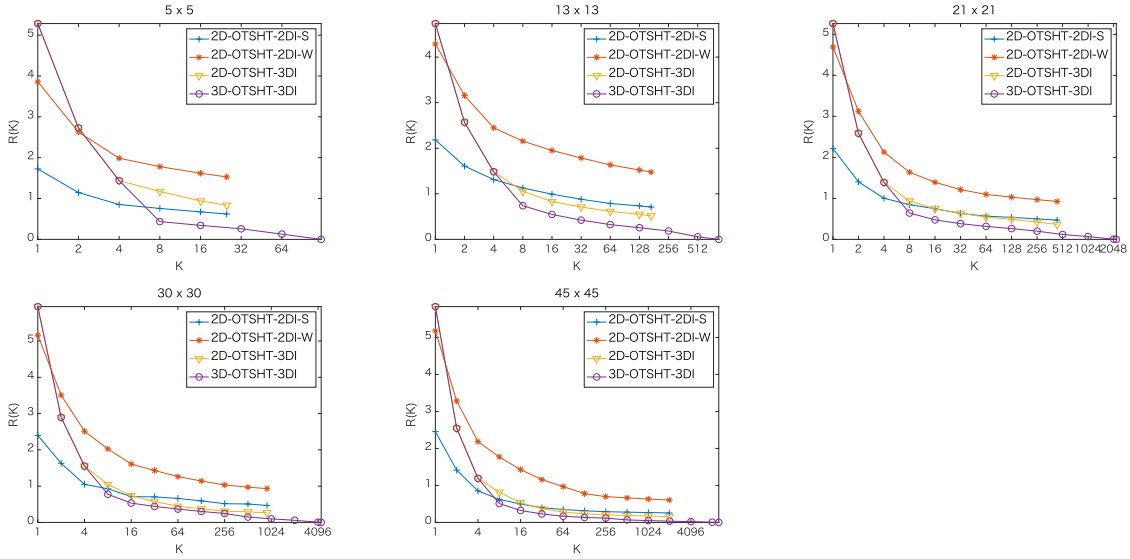


Figure 5. Percentage of extra patches remaining at K basis images/blocks in 5-band images of size 1824×1368.

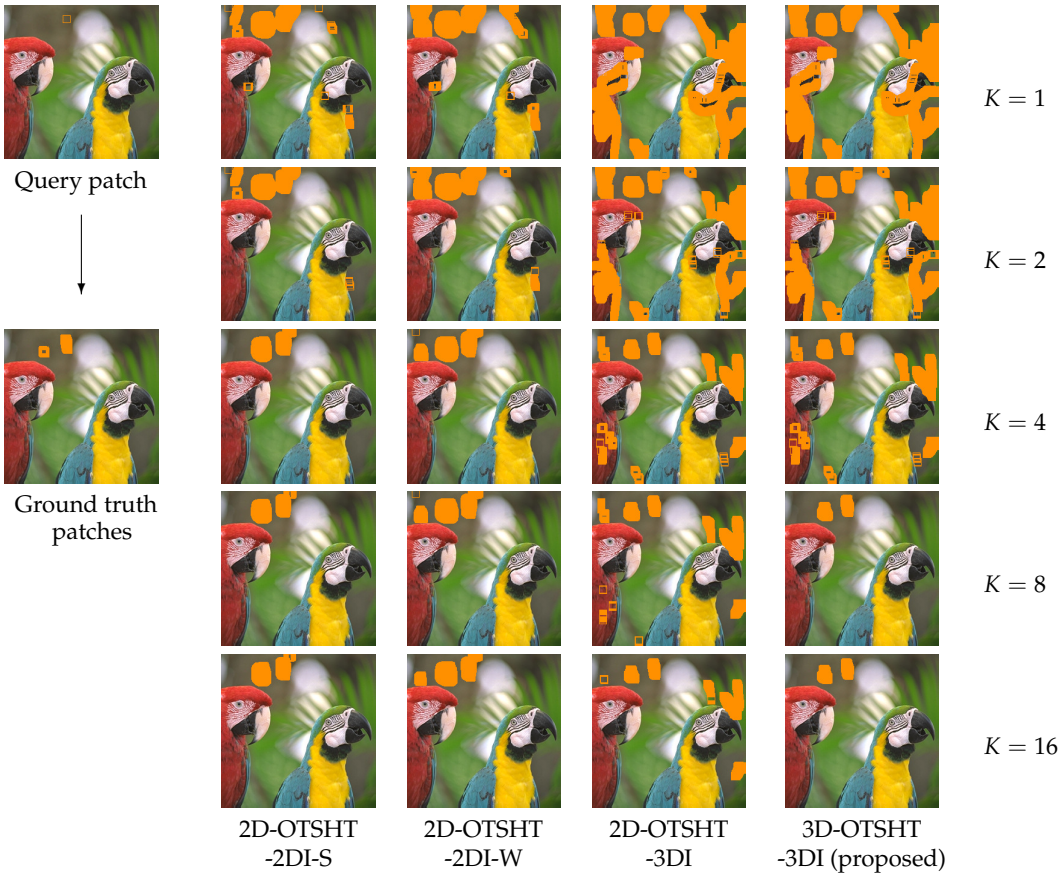


Figure 6. An example of the patches detected by a query patch in different methods at K basis images/blocks. The query patch of size 13×13 and the detected patches are shown in orange squares.

4.3. Computational Complexity

Here, we consider the number of arithmetic operations with respect to the number of basis images/blocks when we use only OTSHT not included in the arithmetic operations of FS. Table 2 shows the number of additions and multiplications per pixel for search for patches similar to a query of size $N \times N$ having B bands including building an integral image. It shows the worst case, where no candidates are rejected at any K basis images/blocks. In 2D-OTSHT-2DI-S and -W, two additions are needed per pixel for building a 2D-integral image in each band, and $(8K-1)$ additions and $3K$ multiplications are required per pixel for SSD in each band. Thus, in total, $(2+8k-1)B$ additions and $3KB$ multiplications per pixel. In 2D- and 3D-OTSHT-3DI, $(3B-1)$ additions are needed per pixel for a 3D-integral image, and $(16K-1)$ additions and $3K$ multiplications are required per pixel for SSD. For the same K, the number of additions for 2D- and 3D-OTSHT-3DI is smaller than that for 2D-OTSHT-2DI-S and -W, except for the images having 2 bands; while the number of multiplications for 2D- and 3D-OTSHT-3DI is smaller than that for 2D-OTSHT-2DI-S and -W in any images with more than one band.

Table 2. The number of additions and multiplications per pixel for searching patches having B bands with K basis images/blocks.

Method	Additions	Multiplications
2D-OTSHT-2DI-S and -W	$(2+8K-1)B$	$3KB$
2D- and 3D-OTSHT-3DI	$3B-1+16K-1$	$3K$

4.4. Speedup

In this section, we analyze when to stop the pruning process for the whole speedup in combination with FS, showing the mean elapsed time at K basis images/blocks in the methods, and compare the final performance of the five methods. Figures 7 and 8 show the mean ratio (%) of the elapsed time of each method at K basis images/blocks to the elapsed time of FS. Tables 3 and 4 detail the mean elapsed time and the ratio to the elapsed time of FS when we use G^K, $(K = 1, 2, 4, \ldots, 32)$ for the color images and the 5-band multispectral images, respectively. The fastest time at each patch size is expressed in bold. From Tables 3 and 4, it can be seen that the proposed method outperforms the other methods except for the cases of patches of size 9×9 in the color images and patches of size 5×5 in the 5-band images. In the color images, for patch size 5×5, the fastest mean elapsed time was 1.03 (ms) which is 23.17 % of the mean elapsed time of FS; while for 21×21, the fastest mean elapsed time was 1.30 (ms), which is 1.78 % of the mean elapsed time of FS. In the 5-band images, for patch size 5×5, the fastest mean elapsed time was 0.050 (s) which is 17.44 % of the mean elapsed time of FS; while for 45×45, the fastest mean elapsed time was 0.164 (s), which is 0.78 % of the mean elapsed time of FS. The larger the patch size and the larger the number of bands, the higher the efficiency of the proposed method.

We confirmed that the mean ratio did not differ much between eight active processor cores with hardware multithreading and one core in the same system.

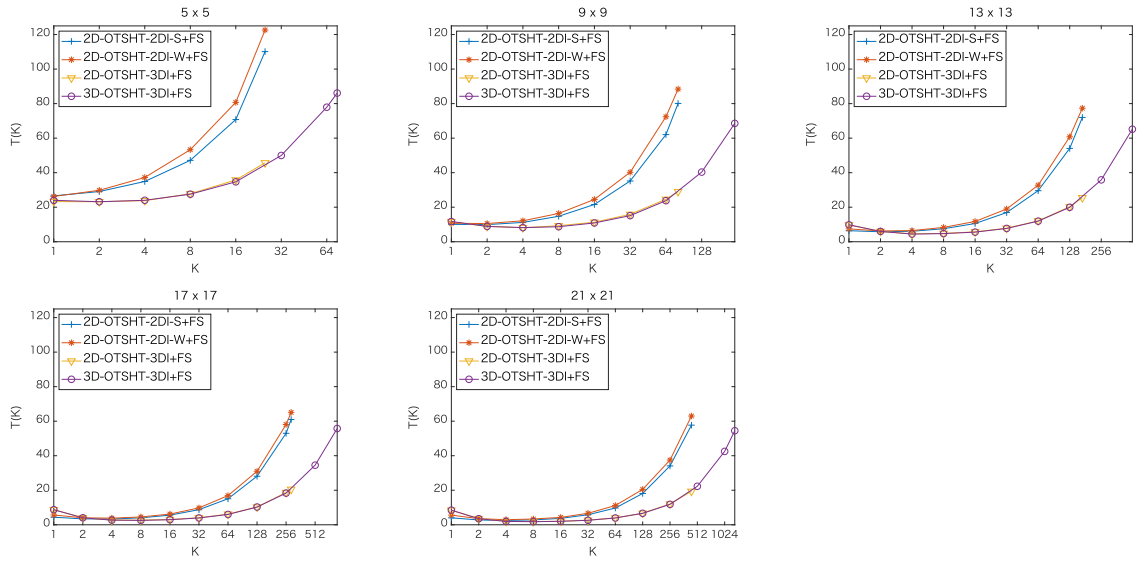


Figure 7. Mean ratio [%] of the elapsed time of each method at K basis images/blocks to the elapsed time of FS in color images of size 256×256.

Table 3. Mean elapsed time and ratio to full search (FS) time in color images of size 256 × 256.

	FS		2D-OTSHT-2DI -S+FS [24]		2D-OTSHT-2DI -W+FS		2D-OTSHT-3DI +FS [27]		3D-OTSHT-3DI +FS (Proposed)	
Size	**[ms]**	**K**	**Time [ms]**	**Ratio [%]**	**Time [ms]**	**Ratio [%]**	**Time [ms]**	**Ratio [%]**	**Time [ms]**	**Ratio [%]**
		1	1.18	26.57	1.17	26.26	1.04	23.23	1.07	23.98
5		2	1.30	29.07	1.33	29.78	1.03	23.18	**1.03**	**23.17**
×	4.46	4	1.55	34.89	1.66	37.21	1.06	23.79	1.07	24.00
5		8	2.10	47.12	2.38	53.30	1.24	27.78	1.23	27.53
		16	3.15	70.77	3.60	80.71	1.59	35.75	1.55	34.71
		32	—	—	—	—	—	—	2.23	49.96
		1	1.46	10.07	1.55	10.67	1.68	11.60	1.70	11.72
9		2	1.44	9.92	1.53	10.54	1.29	8.91	1.29	8.87
×	14.51	4	1.62	11.18	1.76	12.12	**1.18**	**8.14**	1.19	8.18
9		8	2.14	14.74	2.38	16.38	1.34	9.24	1.26	8.70
		16	3.13	21.58	3.55	24.50	1.64	11.31	1.58	10.88
		32	5.11	35.19	5.84	40.22	2.30	15.86	2.19	15.12
		1	1.82	6.39	2.11	7.43	2.77	9.76	2.78	9.80
13		2	1.64	5.77	1.77	6.24	1.71	6.03	1.70	5.97
×	28.39	4	1.72	6.06	1.83	6.45	1.29	4.54	**1.26**	**4.44**
13		8	2.14	7.53	2.35	8.29	1.38	4.85	1.34	4.70
		16	3.00	10.55	3.33	11.73	1.62	5.71	1.58	5.58
		32	4.80	16.90	5.38	18.96	2.22	7.81	2.19	7.70
		1	2.19	4.39	2.82	5.66	4.35	8.74	4.35	8.72
17		2	1.71	3.42	2.01	4.03	2.04	4.09	2.01	4.04
×	49.84	4	1.66	3.34	1.91	3.82	1.34	2.68	1.34	2.69
17		8	2.00	4.02	2.26	4.53	1.33	2.67	**1.27**	**2.55**
		16	2.78	5.58	3.09	6.19	1.51	3.04	1.46	2.93
		32	4.36	8.74	4.87	9.77	2.01	4.04	1.97	3.96
		1	2.94	4.02	4.10	5.60	6.17	8.44	6.19	8.46
21		2	2.04	2.79	2.60	3.55	2.63	3.59	2.57	3.51
×	73.16	4	1.81	2.48	2.15	2.94	1.44	1.97	1.43	1.95
21		8	2.08	2.84	2.44	3.33	1.38	1.88	**1.30**	**1.78**
		16	2.71	3.71	3.13	4.28	1.49	2.03	1.47	2.01
		32	4.24	5.80	4.87	6.66	1.95	2.66	1.90	2.60

All the algorithms are written in C as single thread tasks, compiled with Xcode 10.1, and run on a macOS system with 4 GHz Intel core i7 and 16 GB RAM, where eight active processor cores with hardware multithreading are used.

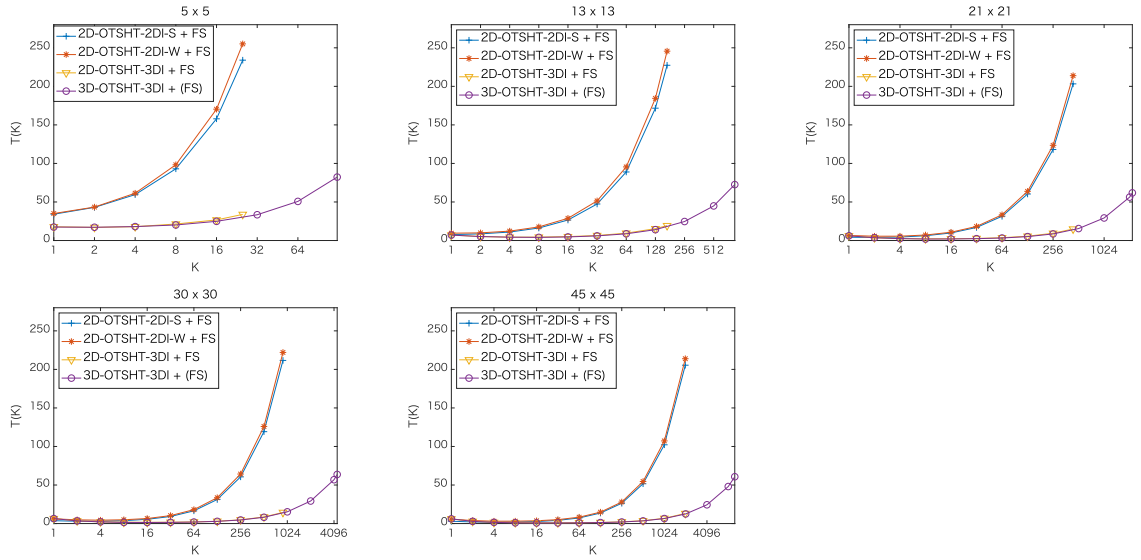


Figure 8. Mean ratio [%] of the elapsed time of each method at K basis images/blocks to the elapsed time of FS in 5-band images of size 1824 × 1368.

Table 4. Mean elapsed time and ratio to FS time in 5-band images of size 1824 × 1368.

	FS		2D-OTSHT-2DI -S+FS [24]		2D-OTSHT-2DI -W+FS		2D-OTSHT-3DI +FS [27]		3D-OTSHT-3DI +FS (Proposed)	
Size	**[s]**	**K**	**Time [s]**	**Ratio [%]**	**Time [s]**	**Ratio [%]**	**Time [s]**	**Ratio [%]**	**Time [s]**	**Ratio [%]**
		1	0.099	34.913	0.102	35.845	0.052	18.239	0.051	17.873
5		2	0.125	44.007	0.126	44.373	**0.050**	**17.437**	0.050	17.604
×	0.285	4	0.173	60.934	0.179	62.793	0.052	18.389	0.053	18.487
5		8	0.271	95.202	0.286	100.577	0.062	21.896	0.059	20.621
		16	0.460	161.597	0.496	174.368	0.078	27.316	0.073	25.471
		32	—	—	—	—	—	—	0.097	34.203
		1	0.137	7.792	0.169	9.646	0.126	7.192	0.125	7.141
13		2	0.153	8.742	0.176	10.065	0.092	5.250	0.092	5.231
×	1.753	4	0.196	11.160	0.217	12.361	0.077	4.392	0.077	4.401
13		8	0.288	16.453	0.314	17.910	0.078	4.473	**0.072**	**4.134**
		16	0.473	26.997	0.512	29.224	0.090	5.112	0.083	4.724
		32	0.843	48.094	0.914	52.168	0.115	6.587	0.107	6.101
		1	0.211	4.360	0.324	6.696	0.302	6.250	0.301	6.235
21		2	0.195	4.035	0.275	5.681	0.177	3.658	0.176	3.637
×	4.833	4	0.221	4.577	0.276	5.719	0.122	2.516	0.122	2.525
21		8	0.306	6.322	0.351	7.266	0.108	2.228	**0.094**	**1.939**
		16	0.483	9.983	0.529	10.936	0.111	2.299	0.098	2.034
		32	0.834	17.252	0.899	18.607	0.133	2.747	0.117	2.428
		1	0.352	3.742	0.589	6.252	0.616	6.547	0.616	6.547
30		2	0.287	3.053	0.459	4.871	0.332	3.527	0.330	3.503
×	9.415	4	0.286	3.034	0.418	4.437	0.209	2.222	0.207	2.195
30		8	0.357	3.788	0.470	4.990	0.169	1.798	0.142	1.512
		16	0.527	5.595	0.623	6.620	0.154	1.631	**0.133**	**1.407**
		32	0.874	9.284	0.985	10.460	0.163	1.730	0.147	1.559
		1	0.642	3.057	1.202	5.724	1.294	6.163	1.297	6.178
45		2	0.444	2.113	0.832	3.961	0.606	2.888	0.605	2.883
×	20.993	4	0.371	1.768	0.651	3.102	0.328	1.565	0.327	1.556
45		8	0.409	1.948	0.659	3.138	0.257	1.224	0.193	0.920
		16	0.554	2.638	0.768	3.659	0.211	1.004	**0.164**	**0.781**
		32	0.876	4.173	1.076	5.124	0.200	0.952	0.168	0.802

All the algorithms are written in C as single thread tasks, compiled with Xcode 10.1 and run on a macOS system with 4 GHz intel core i7 and 16 GB RAM, where eight active processor cores with hardware multi-threading are used.

We also compared the five methods with the best performed K-value to A2DHT [20]. A2DHT is an FS-equivalent algorithm for grayscale images and reportedly performs fastest in FS-equivalent algorithms, whose source code is provided by the authors [30]. We modified the code so that the method was applied to each band separately. Since A2DHT has a limitation that the patch size is power-of-2, the queries of size 16×16 were used for the color image dataset and the queries of size 16×16 and 32×32 were used for the 5-band image dataset. The number of ground truth patches is summarized in Table 5. We confirmed that all the methods detected every ground truth patch. Table 6 shows the mean elapsed time and mean ratio to FS time, where the fastest time and its ratio at each patch size is expressed in bold. It can be seen that the proposed method outperforms state-of-the-art methods.

Table 5. The number of ground truth patches of size power-of-2.

Dataset	Image Size	Band	Scenes	Patch Size	Samples	Number of Patches		
						Min.	Mean	Max.
1	256×256	3	12	16×16	120	1	15	732
2	1824×1368	5	11	16×16	110	1	4880	100,278
				32×32	110	1	1370	91,325

Table 6. Mean elapsed time and ratio to FS time for patches of size power-of-2.

Size	Data Set	FS		A2DHT [20]		2D-OTSHT -2DI-S+FS [24]		2D-OTSHT -2DI-W+FS		2D-OTSHT -3DI+FS [27]		3D-OTSHT-3DI +FS (Proposed)	
		[ms]	[%]	[ms]	[%]	[ms]	[%]	[ms]	[%]	[ms]	[%]	[ms]	[%]
16×16	1	46.04	100	2.65	5.76	1.69 ($K = 4$)	3.68	1.94 ($K = 4$)	4.22	1.33 ($K = 4$)	2.89	**1.23** ($K = 8$)	**2.67**
		[s]	[%]	[s]	[%]	[s]	[%]	[s]	[%]	[s]	[%]	[s]	[%]
16×16	2	2.517	100	0.099	3.933	0.136 ($K = 2$)	5.398	0.166 ($K = 2$)	6.614	0.069 ($K = 4$)	2.728	**0.067** ($K = 8$)	**2.647**
32×32	2	10.854	100	0.124	1.142	0.246 ($K = 4$)	2.262	0.342 ($K = 4$)	3.155	0.120 ($K = 16$)	1.105	**0.108** ($K = 16$)	**0.995**

All the algorithms are written in C as single thread tasks, compiled with Xcode 10.1, and run on a macOS system with 4 GHz Intel core i7 and 16 GB RAM, where eight active processor cores with hardware multithreading are used.

5. Conclusions

We proposed a fast FS-equivalent 3D block matching method in order to search for the patch(es) similar to a given query for multichannel images. The proposed method uses 3D-OTSHT and reduces the number of candidates with SSD in the transformed domain. The pruning process rejects unmatched candidates during the block matching processing. Moreover, in combination with FS, the pruning process is stopped during the processing for the whole speedup. We designed 3D-OTSHT making the most of 3D integral image rather than the extension of one-dimensional TSHT. Unmatched patches are securely rejected due to being orthonormal and using SSD as the similarity measure. We have analyzed the pruning performance and mean elapsed time using a color image dataset and a 5-band multispectral image dataset, and demonstrated that the proposed method outperforms state-of-the-art methods. In the color images, the mean elapsed time was shortened up to less than 2% of the FS time. In the 5-band multispectral images, the search time was shortened up to around 0.8% of the FS time, hence allowing more than 100-times faster processing without sacrificing the accuracy. We believe that these huge savings in computation time can enable new applications of patch matching in multichannel images, which were not feasible before due to the prohibitive computational complexity.

Author Contributions: Conceptualization, I.I., A.P.; methodology, I.I.; software, I.I.; validation, I.I.; investigation, I.I.; resources, I.I. ; data curation, I.I.; writing—original draft preparation, I.I.; writing—review and editing, A.P.; visualization, I.I.; project administration, I.I. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest: The authors declare no conflict of interest. The founding sponsors had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, and in the decision to publish the results.

References

1. Dufour, R.; Miller, E.; Galatsanos, N. Template matching based object recognition with unknown geometric parameters. *IEEE Trans. Image Process.* **2002**, *11*, 1385–1396. [CrossRef] [PubMed]
2. Ding, L.; Goshtasby, A.; Satter, M. Volume image registration by template matching. *Image Vis. Comput.* **2001**, *19*, 821–832. [CrossRef]
3. Sarraf, S.; Saverino, C.; Colestani, A.M. 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, Las Vegas, NV, USA, 24–27 February 2016; pp. 53–56.
4. Papyan, V.; Elad, M. Multi-scale patch-based image restoration. *IEEE Trans. Image Process.* **2016**, *25*, 249–261. [CrossRef] [PubMed]
5. Lowe, D.G. Object recognition from local scale-invariant features. In Proceedings of the International Conference of Computer Vision, Kerkyra, Greece, 20–27 September 1999; pp. 1150–1157.
6. Bay, H.; Tuytelaars, T.; Gool, L.V. *SURF: Speeded Up Robust Features*; Lecture Notes in Computer Science 2006; Springer: Berlin/Heidelberg, Germany, 2006; Volume 3951. pp. 404–417.
7. Pastuszak, G.; Trochimiuk, M. Architecture design of the high-throughput compensator and interpolator for the H.265/HEVC Encoder. *J. Real-Time Image Process.* **2016**, *11*, 663–673. [CrossRef]
8. Gonźalez, D.; Botella, G.; Garcia, C.; Prieto, M.; Tirado, F. Acceleration of block-matching algorithms using a custom instruction-based paradigm on a Nios II microprocessor. *EURASIP J. Adv. Signal Process.* **2013**, *118*. [CrossRef]
9. González, D.; Botella, G.; Meyer-Baese, U.; García, C.; Sanz, C.; Prieto-Matías, M.; Tirado, F. A low cost matching motion estimation sensor based on the NIOS II microprocessor. *Sensors* **2012**, *12*, 13126–13149. [CrossRef]
10. Nguyen, A.H.; Pickering, M.R.; Lambert, A. The FPGA implementation of a one-bit-per-pixel image registration algorithm. *J. Real-Time Image Process.* **2016**, *11*, 799–815. [CrossRef]
11. Li, D.X.; Zheng, W.; Zhang, M. Architecture design for H.264/AVC integer motion estimation with minimum memory bandwidth. *IEEE Trans. Consum. Electoron.* **2007**, *53*, 1053–1060. [CrossRef]
12. Koga, T.; Iinuma, K.; Hirano, A.; Iijima, Y. Motion compensated interframe coding for video conferencing. In Proceedings of the National Telecommunications Conference, New Orleans, LA, USA, 29 November–3 December 1981; pp. G5.3.1–G5.3.5.
13. Zhu, S.; Ma, K. A new diamond search algorithm for fast block motion estimation. *IEEE Trans. Image Process.* **2000**, *9*, 287–290. [CrossRef]
14. Simard, P.; Bottou, L.; Haffner, P.; Cun, Y.L. Boxlets: A fast convolution algorithm for signal processing and neural networks. *Adv. Neural Inf. Process. Syst.* **1999**, *11*, 571–577.
15. Tang, F.; Crabb, R.; Tao, H. Representing images using non-orthogonal Haal-like bases. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 2120–2134. [CrossRef] [PubMed]
16. Tombari, F.; Mattoccia, S.; Stefano, L.D. Full search-equivalent pattern matching with incremental dissimilarity approximations. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 129–141. [CrossRef]
17. Ouyang, W.; Cham, W.K. Fast algorithm for Walsh Hadamard transform on sliding windows. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 165–171. [CrossRef] [PubMed]
18. Ouyang, W.; Zhang, R.; Cham, W.-K. Segmented gray-code kernels for fast pattern matching. *IEEE Trans. Image Process.* **2013**, *22*, 1512-1525. [CrossRef] [PubMed]
19. Ouyang, W.; Zhang, R.; Cham, W.-K. Fast pattern matching using orthogonal Haar transform. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp.3050–3057.

20. Ouyang, W.; Zhao, T.; Cham, W.-K.; WeiFast, L. Full-search-equivalent pattern matching using asymmetric Haar wavelet packets. *IEEE Trans. Circuits Syst. Video Technol.* **2018**, *28*, 819–833. [CrossRef]
21. Li, Y.; Li, H.; Cai, Z. Fast orthogonal Haar transform pattern matching via image square sum. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 1748–1760. [CrossRef]
22. Crow, F. Summed-area tables for texture mapping. *SIGGRAPH* **1984**, *18*, 207–212. [CrossRef]
23. Viola, P.; Jones, M. Robust real-time object detection. *Int. J. Comput. Vis.* **2001**, *57*, 37–154.
24. Ito, I.; Egiazarian, K. Two-dimensional orthonormal tree-structured Haar transform for fast block matching. *J. Imaging* **2018**, *4*, 1–18. [CrossRef]
25. Egiazarian, K.; Astola, J. Tree-structured Haar transform. *J. Math. Imaging Vis.* **2002**, *16*, 269–279. [CrossRef]
26. Haar, A. Zur theorie der orthogonalen functionsysteme. *Math. Annal.* **1910**, *69*, 331–371. [CrossRef]
27. Ito, I.; Pižurica, A. 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, Dubrovnik, Croatia, 23–25 September 2019; pp. 70–75.
28. Standard Image Data BAse. Available online: http://www.ess.ic.kanagawa-it.ac.jp/app_images_j.html (accessed on 10 February 2020).
29. Monno, Y.; Tanaka, M.; Okutomi, M. TokyoTech 5-Band Multisupectral Image Dataset and Demosaicking Codes. Available online: www.ok.sc.e.titech.ac.jp/res/MSI/MSIdata.html (accessed on 10 February 2020).
30. Fast Full-Search Equivalent Pattern Matching Using Asymmetric Haar Wavelet Packets. Available online: https://wlouyang.github.io (accessed on 10 February 2020).



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).