*remote sensing*

MDPI

*Article*

# On-Board Detection and Matching of Feature Points

**Jingjin Huang [1,2,3] and Guoqing Zhou [1,2,3,*]**

[1] School of Precision Instrument & Opto-Electronics Engineering, Tianjin University, Tianjin 300072, China; jingjin_huang@tju.edu.cn
[2] GuangXi Key Laboratory for Spatial Information and Geomatics, Guilin University of Technology Guilin, Guangxi 541004, China
[3] The Center for Remote Sensing, Tianjin University, Tianjin 300072, China
* Correspondence: gzhou@glut.edu.cn; Tel.: +86-022-2740-7952

Received: 13 April 2017; Accepted: 9 June 2017; Published: 13 June 2017

**Abstract:** This paper presents a FPGA-based method for on-board detection and matching of the feature points. With the proposed method, a parallel processing model and a pipeline structure are presented to ensure a high frame rate at processing speed, but with a low power consumption. To save the FPGA resources and increase the processing speed, a model which combines the modified SURF detector and a BRIEF descriptor, is presented as well. Three pairs of images with different land coverages are used to evaluate the performance of FPGA-based implementation. The experiment results demonstrate that (1) when the image pairs with artificial features (such as buildings and roads), the performance of FPGA-based implementation is better than those image pairs with natural features (such as woods); (2) the proposed FPGA-based method is capable of ensuring the processing speed at a high frame rate, such as the speed of can achieve 304 fps under a 100 MHz clock frequency. The speedup of the proposed implementation is about 27 times higher than that when using the PC-based implementation.

**Keywords:** onboard; detection; image matching; parallel processing; feature points

## 1. Introduction

The detection and matching of feature points are used as the first step of satellite image processing such as attitude estimation, geometrical calibration, object detection and tracking, and ortho-rectification [1]. While the traditional way that implementation detection and matching on ground and/or in indoor is not suitable for the increasing requirement for real-time or near real-time remotely sensed imagery applications [2,3]. To meet the requirement of real-time processing, various detection and matching algorithms are implemented on Field Programmable Gate Array (FPGA), which can offer highly flexible designs and scalable circuits [4]. Meanwhile, the parallel processing characteristic and the pipeline structure of FPGA allow data to be processed more quickly with a lower power consumption than a similar microprocessor implementation and/or CPU [5,6]. In addition, several FPGA-based implementations are proposed with soft cores microprocessors to realize the complex algorithms [7], such as, motion estimation algorithms [8–10] and epsilon quadratic sieve algorithm [11]. Hence, the FPGA-based implementation of detection and matching algorithms are widely researched.

Among the detection and matching algorithms, the Scale-Invariant Feature Transform (SIFT), Speed-Up Robust Feature (SURF), Oriented-FAST and Rotated BRIEF (ORB), and theirs modified algorithms have excellent performance in computer vision applications [12], and these algorithms also have been implemented on FPGAs to achieve various applications. For example, Yao L. et al. (2009) proposed a Xilinx Virtex-5 FPGA implementation of optimized SIFT feature detection. The proposed FPGA implementation can carry out the feature detection of a typical image of 640 × 480 pixels within 31 ms [13]. Svab, J. et al. (2009) presented a hardware implementation of the SURF on

FPGA. The implementation achieves about 10 frames per second at 1024 × 768 resolution and the total power consumption is less than 10 W [14]. Schaeferling M. et al. (2010) proposed a hardware architecture to accelerate the SURF algorithm on Virtex-5 FPGA [15]. Lentaris G. et al. (2013) proposed a hardware-software co-design scheme using Xilinx Virtex 6 FPGA to speed-up the SURF algorithm for the ExoMars Programme [16]. Schaeferling M. et al. (2011) implemented a complete SURF-based system on Xilinx Virtex 5 FX70T FPGA for object recognition. The average time was 481 ms for one frame [17]. Sledevic T. et al. (2012) proposed an FPGA-based implementation of a modified SURF algorithm. The proposed architecture achieves real-time orientation and the descriptor calculation can achieve on 60 fps 640 × 480 video stream only on a 25 MHz clock [18]. Battezzati N. et al. (2012) proposed a FPGA architecture for implementation SURF algorithm. The entire system is about 340 ms for one frame [19]. Zhao et al. (2013) proposed a real-time SURF-based traffic sign detection system by exploiting parallelism and rich resources in FPGAs. The proposed hardware design is able to accurately process video strams of 800 × 600 resolution at 60 frame fps [20]. Fan X. et al. (2013) proposed a high performance hardware implemented of OpenSURF [21] algorithm on XC6VSX475T FPGA. The proposed implementation achieved 356 fps with 156 MHz clock [22]. Krajnik T. et al. (2014) presented a complete hardware and software solution of an FPGA-based computer vision embedded module capable of carrying out SURF image features extraction algorithm [23]. Chen et al. (2015) proposed an FPGA architecture of OpenSURF. The result found that the architecture can detect feature and extract descriptors from video streams of 800 × 600 resolutions at 60 fps [24]. Gonzalez C. et al. (2016) proposed an FPGA implementation of an algorithm for automatically detecting targets in remotely sensed hyperspectral images [25]. Weberruss J. et al. (2015) proposed a hardware architecture of ORB on FPGA, which offer lower power consumption and higher frame rates than general hardware [26]. Among of them, the SIFT algorithm which has scale invariance, rotational invariance and affine invariance is the most famous and achieve best performance. However, some characteristics of SIFT descriptor, such as large computational burden, floating point arithmetic, poor real-time performance, limit the FPGA implementation of it. While SURF algorithm has less computation burden and quicker calculating speed, especially, its detector is ease to implement in hardware with fixed-point arithmetic and parallel characteristic. The feature points also can be detected at different scales. In contrast, its descriptor is poor real-time performance and large consumption of hardware resources. Therefore, the SURF algorithm is not suitable for on-board processing directly where a high real time requirement environment. On the other hand, the ORB algorithm consists of oriented FAST detector and rotated BRIEF (Binary Robust Independent Elementary Features) descriptor [27] which are ease to be implemented in FPGA. While the FAST detector has no scale invariant and need to be further considered especial when series images are not in the same scale.

Thus, to implement on-board detection and matching with a high frame per second (fps) which usually up to hundreds of fps, a modified SURF detector and a BRIEF descriptor are proposed in this paper. The SURF detector has scale invariant and parallel computing characteristics which is easy to be implemented on FPGA. The BRIEF descriptor consists of binary vectors, and the Hamming distance of two descriptors is computed by the XOR operation. The SURF detector and the BRIEF descriptor are easy to implement on hardware platform, such as FPGA.

The rest of this paper is organized as follows: Section 2 briefly reviews the proposed method; FPGA-based implementation is presented in Section 3. Experiment, performance evaluation, and discussions are presented in Section 4. Finally, Section 5 draws up some conclusions.

## 2. Detection and Matching Algorithm

### *2.1. PC-Based Detection and Matching Algorithm*

#### 2.1.1. SURF Detector

The SURF detector firstly proposed by Bay [28] performs a multi-scale characteristic. The basic idea and steps of the detector are summarized as follows. More details can refer [28].

(1) Integral image generation

Integral image is a novel method to improve the performance of the subsequent steps for SURF detector. It is defined by Equation (1). By using the integral image, it is efficient to calculate the summation of pixels in an upright rectangle area of image.

$$I(x,y)=\sum_{r=1}^{y}\sum_{c=1}^{x} i(r,c) \tag{1}$$

where $I(x,y)$ represents the integral value at location $(x,y)$ of the image, $i(r,c)$ represents the gray value at the location $(r,c)$ of the image.

(2) Hessian matrix responses generation

The expression of Hessian matrix is presented in Equation (2). The determinant of Hessian matrix is calculated by Equation (3) in an approximation way. In Equation (3), $\omega^2$ is a weight coefficient equal to 0.91 [28]. The $D_{xx}$, $D_{yy}$ and $D_{xy}$ are computed respectively by the integral image and the box filters (see Figure 1a–c). Figure 1a is a box filter in X direction, Figure 1b is a box filter in Y direction, and Figure 1c is a box filter at XY direction. N can be selected at 9, 15, 21...; m can be selected at 2, 3, 4..., and k can be selected at 3, 5, 7..., respectively. In addition, the rectangles with gray mean their value with 0; the rectangles with white mean their values with 1; and the rectangles with black mean their value with −2.

$$H(x,\sigma)=\begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{2}$$

$$\det H_{approx}=L_{xx}L_{yy}-L_{xy}^2=D_{xx}D_{yy}-\omega^2 D_{xy}^2 \tag{3}$$

(3) Using the 3D non-maximal suppression

A 3D non-maximal suppression is performed to find a set of candidate points. To keep the feature point with a strong robustness, a window with a size of 5 × 5 is used instead of a 3 × 3 window. To do this each pixel in the scale space is compared to its 74 neighbors (see black points in Figure 1d), comprised of the 24 points in the native scale and the 25 points in each of the scales above and below.

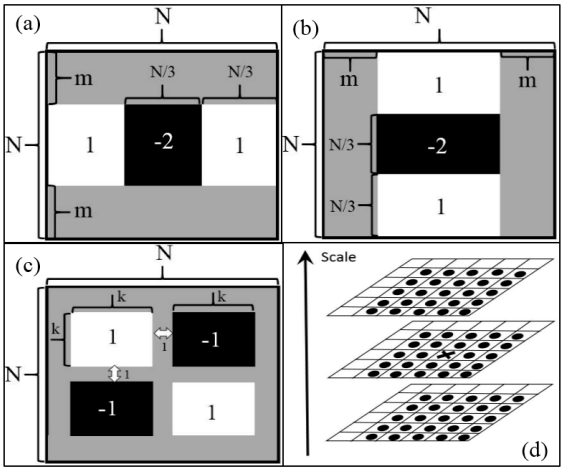


**Figure 1.** (**a**) Filter box in X direction; (**b**) filter box in Y direction; (**c**) filter box in 45° direction; (**d**) 3D non-maximal suppression.

2.1.2. BRIEF Descriptor

A BRIEF descriptor [27] of the feature point consists of a binary vector. The length of the binary vector is generally defined as 128 bits, 256 bits, and 512 bits. According to the comparative results analyzed by Calonder et al. [27], the performance with the 256 bits was similar to that with 512 bits,

while only the marginally worse in other cases. To save the FPGA resources, the 256 bits is suggested in this paper. The following equation presents the definition of a BRIEF descriptor:

$$\tau(p;r_1,c_1,r_2,c_2)=\begin{cases}1: p(r_1,c_1)<p(r_2,c_2)\\ 0: p(r_1,c_1)\geq p(r_2,c_2)\end{cases} \quad (4)$$

where $(r_1, c_1)$ and $(r_2, c_2)$ represent the rows and columns of one point-pair, respectively. $p(r_1, c_1)$ and $p(r_2, c_2)$ represent the intensity values at location $(r_1, c_1)$ and $(r_2, c_2)$. If the value at $p(r_1, c_1)$ is less than one at $p(r_2, c_2)$, then value of $\tau = 0$, otherwise, $\tau = 1$.

Because of the BRIEF descriptor is sensitive to noise, the intensity value of patch-pair is computed by an smoothing filtering with a 5 × 5 sub-window centred on ($r_i$, $c_i$, i = 1,2..., and 512) (see Figure 2a). Meanwhile, $\{(r_1, c_1), (r_2, c_2)\}$ is defined as a patch-pair, $\{(r_3, c_3), (r_4, c_4)\}$ is defined as another patch-pair, meanwhile, there are 256 patch-pairs in total (see Figure 2b). The locations $(r_i, c_i)$ of the 256 point pairs are determined by the Gaussian distribution. $(r_i, c_i)$–i.i.d. Gaussian (0, $S^2/25$). $(r_i, c_i)$ are determined from an isotropic Gaussian distribution; S is the size of a patch [27]. For the details, it can be referenced to Calonder et al. [27].

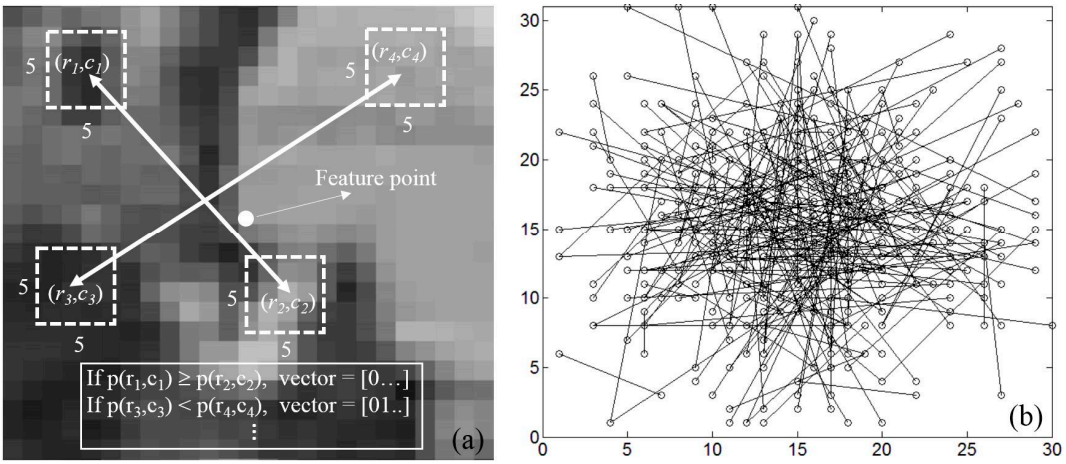


**Figure 2.** (**a**) Operation processing of the BRIEF descriptor; (**b**) its 256 pixel-pairs.

### 2.1.3. Matching

To match two feature points in one pair of images, the corresponding descriptors of the feature points are firstly generated. Then a Hamming distance of the descriptors is computed by XOR operation (1 XOR 1 = 0 and 1 XOR 0 = 1). A threshold, *(t)* is setup to observe if the feature points are successfully matched when using the Hamming distance. The results and the matching process are listed in Table 1. As observed from Table 1, if the Hamming distance is less than 2, the two feature points are successfully matched, otherwise, they are unmatched.

**Table 1.** Matching process (Note: the length of a BRIEF descriptor is assumed 6, and the threshold *(t)* is given 2).

| Descriptor 1 | Descriptor 2 | XOR Operation | Hamming Distance | Result |
|---|---|---|---|---|
| 110011 | 110011 | 000000 | 0 | Matched |
| 110011 | 110000 | 000011 | 2 | Matched |
| 110011 | 110100 | 000111 | 3 | Unmatched |
| 110011 | 001100 | 111111 | 6 | Unmatched |

### *2.2. FPGA-Based Detection and Matching Algorithm*

Implementation of the proposed algorithm on the FPGA increases the speed of execution and provides flexibility, due to the pipelining and reconfigurable nature of the FPGA. If the proposed algorithm is implemented on FPGA directly, the speed and the clock frequency will be restricted greatly, and the usage of FPGA resources will be increased significantly. Hence, to guarantee the accuracy and speed, some modifications are essential. In this paper, the modifications that how to reduce the usage of memory, multipliers, and dividers, are mainly focused on the SURF detector. Due to the excellent performance of BRIEF descriptor and matching on FPGA implementation, the modifications of BRIEF descriptor and matching are not necessary.

#### 2.2.1. Modification of Integral Image

Although the integral image can speed up the calculation of summation of pixels, the large bit width of integral image may seriously impact the memory of FPGA where the whole integral image are stored on. For example, as an 8 bits format gray image with a size of 512 × 512, the bit width of the integral image is 28 bits. To store a frame of such integral image, 512 × 512 × 28 bits (about 7.0 Mb) memory are required. That may be unacceptable in many cases, especially if the size of image is larger. To deal with this problem, several work focus on how to reduce the bit width on FPGA implementation, such as Ma et al. (2014, 2015, 2016) proposed a full-image evaluation methodology to reduce bit width when implement the histogram of oriented gradients on FPGA [29–31], Sousa et al. (2013) presented a fixed-point algorithm when implemented a Harris corner detector on Tightly-Coupled Processor Arrays [32]. In this paper, we use a computing through the overflow technique proposed by Belt [33] to reduce the bit width of integral image and maintain the accuracy of the calculated results. According to Belt, if the width and the height of the maximal rectangular is $W_{\max}$ and $H_{\max}$, the bit width $L_{ii}$ of the final integral image is decided by Equation (5).

$$\left(2^{L_{ii}}-1\right) \geq \left(2^{L_i}-1\right) W_{\max} H_{\max} \tag{5}$$

where $L_i$ is the bit width of input image. Although there are some intermediate overflowing results during the process of computing integral image, final summation of pixels in any rectangular whose size is less than $W_{max} \times H_{max}$ is still correct. In our FPGA implementation, two octaves (see Table 2) are suggested in the FPGA-based implementation [20,26], because of a higher resources consumption and a lesser feature points in higher scale space. As seen from Table 2 and Figure 1a–c, the maximal size of box filter is 51 × 51 (namely, N = 51 and m = 7). The box filter with a size of 51 consists of gray rectangle, white rectangle, and black rectangle. In these rectangles, the maximal size of these rectangles is 17 × 37 or 37 × 17 without consider the gray rectangle. Thus, the $W_{\max}$ = 17 (or 37), and the $H_{\max}$ = 37 (or 17) in this paper, and the bit width of integral image is 18 bits if $L_i$ equal to 8 according to Equation (5).

**Table 2.** The relationship between scale and the size of box filter.

| **Octave** | **1** | | | | **2** | | | |
|---|---|---|---|---|---|---|---|---|
| Size of box filter | 9 | 15 | 21 | 27 | 15 | 27 | 39 | 51 |
| scale | 1.2 | 2 | 2.8 | 3.6 | 2 | 3.6 | 5.2 | 6.8 |

#### 2.2.2. Modification of Hessian Matrix Responses

To reduce the use of multipliers or dividers, the value of $\omega^2$ is changed from the original value 0.91 [28] to 0.875 [15,18], as the latter is more suitable to be calculated by FPGA. Then, the Equation (3) is translated into Equation (6):

$$\det H_{approx} = D_{xx}D_{yy} - D_{xy}^2 + D_{xy}^2 \Big/ 8 = D_{xx}D_{yy} - D_{xy}^2 + D_{xy}^2 >> 3 \tag{6}$$

where only two multipliers are needed and a right shift for division, while three multipliers are needed in Equation (3).

## 3. FPGA-Based Implementation

### *3.1. The Whole FPGA Architecture*

To achieve the on-board detection and matching, a FPGA-based method is proposed in Figure 3. In the proposed method, a pipeline structure and a parallel processing model are proposed to ensure the real-time processing performance. The detailed descriptions of each module are given below:

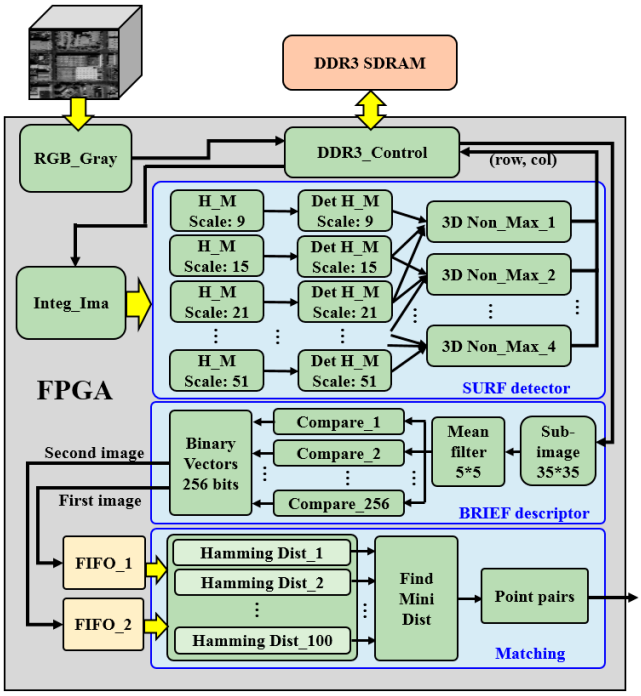


**Figure 3.** Hardware architecture of FPGA-based detection and matching.

Firstly, a series of color images are converts into the corresponding gray images by "RGB_Gray" module, and then the gray images are sent to "DDR3_Control" module to generate the write data and the write the address. The data and the address are sent to DDR3 SDRAM (DDR3), a memory with 512 M outside of the FPGA chip. The stored data are used for the feature point detection and the descriptor generation.

Secondly, an integral image of a gray image are generated by "Inte_Ima" module, and the Hessian matrixes are computed by "H_M Scale: i" modules (i = 9, 15, 21, 27, 35, and 51) in parallel. Then, the determinants of Hessian matrixes computed in parallel by "Det H_M Scale: i" modules (i = 9, 15, 21, 27, 35, and 51) are sent to "3D Non_Max_j" modules (j = 1, 2, 3, and 4). The locations (rows and columns) of feature points in different scales are determined by the 3D non-maximal suppression.

Thirdly, a sub-image with the size of 35 × 35 pixels$^2$ centered on the feature point is reading out from the "DDR3_Control" module. To reduce the sensitiveness to noise, a mean filter with a size of 5 × 5 pixels$^2$ is used. The filtered results are sent to the "compare_k" module (k = 1, 2..., 256) to compute the binary vectors in parallel. The 256 binary vectors are used to create a BRIEF descriptor. The generated descriptors in the first image and the second image are sent to "FIFO_1"and "FIFO_2" IP cores, respectively.

Fourthly, when the computation of the descriptors in the first image is finished, the descriptors in the second image start to be computed. All descriptors in the first image and the first descriptor in the second image are sent to "Hamming Dist_g" modules (g = 1, 2..., 100), which are used to compute Hamming distances at the same clock period. The computed Hamming distances are sent to "Find

Mini Dist" module, which is used to find out the minimum value of Hamming distances. The matching points can be outputted by the minimum value and a threshold *(t)*. Similarly, the processes of the late descriptor in the second image are the same as that in the first descriptor.

### *3.2. Implementation of DDR3 Write-Read Control*

To write the gray images into DDR3 and then read them out successfully, a write-read control module need to be redesigned. In this module, we need to re-design these six signals:

(1) "app_cmd". When "app_cmd" = 3'b000, the write signal is active-high. When "app_cmd" = 3'b001, the read signal is active-high;
(2) "app_addr". This input indicates the address for the current request;
(3) "app_en". This is the active-high strobe for the "app_cmd", "add_addr" et al.;
(4) "app_wdf_data". This provides the gray image data for write commands;
(5) "app_wdf_wren". This is the active-high strobe for "app_wdf_data";
(6) "app_wdf_end". This signal equals to "app_wdf_wren".

To achieve a DDR3 write-read control successfully, the six signals must meet the requirements of sequential relationships which are presented in Figure 4. In Figure 4a, the six signals are all tightly aligned with clock signal when write data into DDR3. In Figure 4b, the "app_cmd", "app_addr", and "app_en" are also tightly aligned with clock signal when read data out of DDR3. More details can refer to [34].

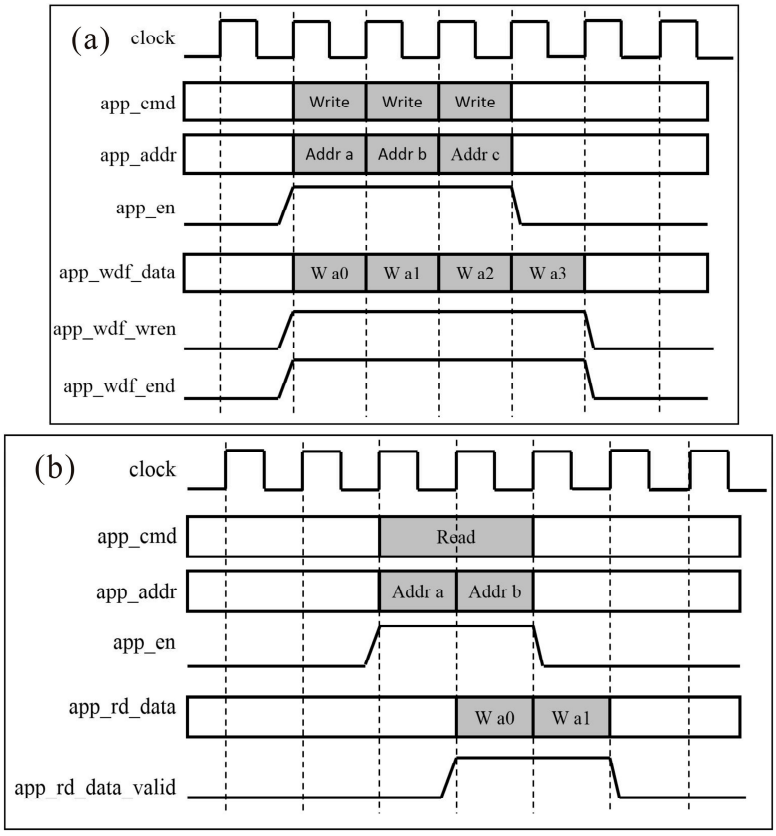


**Figure 4.** Sequential relationship of writing (**a**) and reading (**b**).

### *3.3. FPGA Implementation of Integral Image*

To accelerate the image processing speed, an integral image is adopted in this paper. The FPGA implementation of integral image computation is illustrated in Figure 5, in which the values of integral image in first row are firstly computed, then the results are sent into a line buffer in which

its depth is 512 bits. For the values of integral image in later rows are computed by two parts, part one is the sum of the gray values in native row, the other one is the value of integral image which locate at the last row and the same column. According to the Section 2.2.1, if the bit width of gray image is 8 bits, the bit width of integral image should be 18 bits.

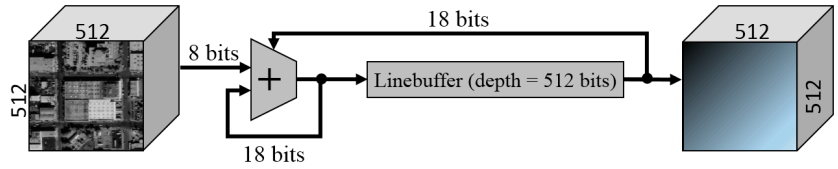


**Figure 5.** FPGA implementation of integral image.

### *3.4. FPGA Implementation of Hessian Matrix Responses*

Feature extraction is regarded to be the most time consuming stage in SURF when implemented by PC [35]. To accelerate the processing speed of the determinants of Fast-Hessian matrix at different scales, a parallel sliding window method is proposed. As depicted in Figure 6a, the $D_{xx}$, $D_{yy}$, and $D_{xy}$ are computed by the convolution of the integral image and the filter boxes (Figure 1a–c) in different scales in parallel. In addition, the FPGA implementation of Equation (6) is shown in Figure 6b. As seen from Figure 6b, the hessian matrix responses are computed by two multipliers, two subtracters, and a right shifting operation.

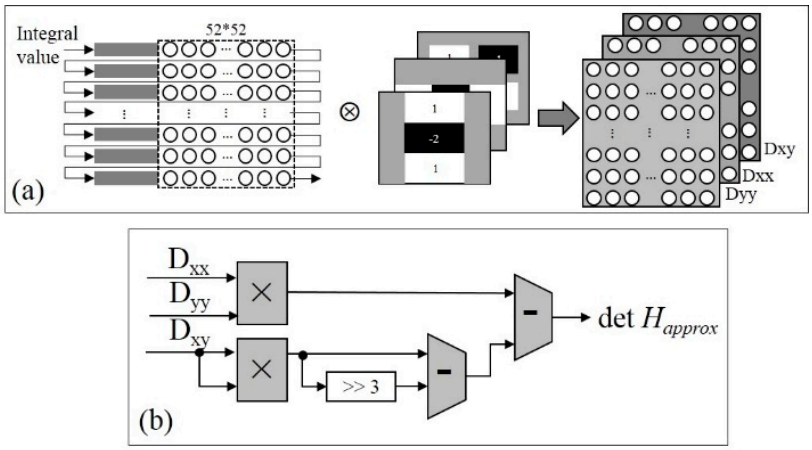


**Figure 6.** (**a**) Parallel computation of $D_{xx}$, $D_{yy}$, and $D_{xy}$ in different scales; (**b**) FPGA implementation of determinant of Hessian matrix.

### *3.5. FPGA Implementation of 3D Non-Maximal Suppression*

After finishing the computation of Hessian matrix responses in parallel, a 3D non-maximal suppression method is adopted to determine the locations of the feature points. The FPGA implementation of 3D non-maximal suppression is presented in Figure 7. As seen from Figure 7, a candidate point (i.e., $b_1$) is compared with its 74 neighbors (Figure 1d) in parallel. An "AND" operation is adopted in the 74 comparative results. If "AND" operation result is "1", then the candidate point is regarded as a feature point, otherwise, the candidate point is not a feature point. In this implementation, 74 comparators and an "AND" operation are used to determine the feature points.

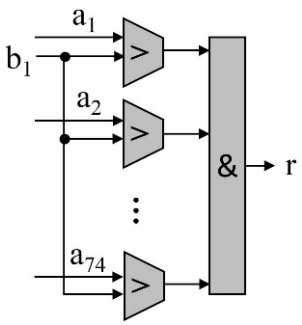


**Figure 7.** FPGA implementation non-maximal suppression.

### *3.6. FPGA Implementation of BRIEF Descriptor*

Once the location of feature point is determined, the BRIEF descriptor is generated in this section. In BRIEF descriptor module, a sub-image (35 × 35) centred on the feature point are sent to 35 line buffers, and a mean filter with a size of 5 × 5 is implemented on the sub-image (as presented in Figure 8a. Then the filter values of 256 patch-pairs are selected according to Figure 2b. The FPGA implementation of Equation (4) is presented in Figure 8b. As seen from Figure 8b, the 256 patch-pairs are compared to generate a 256 bits binary vector, and 256 comparators and a combination operation are adopted. The generated descriptors are stored into "FIFO" IP core waiting for matching.

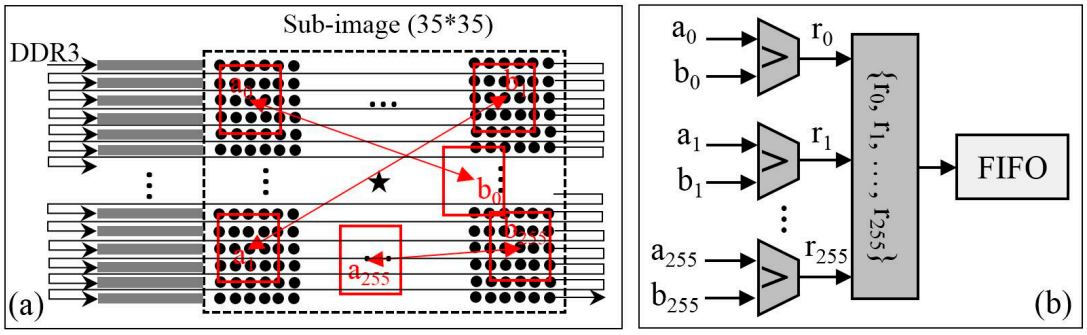


**Figure 8.** (**a**) A sub-image extraction by line-buffer; (**b**) generation of a BRIEF descriptor.

### *3.7. FPGA Implementation of Matching*

The FPGA implementation of matching is presented in Figure 9. As seen from Figure 9a, the BRIEF descriptors with 256 bits of the first image and the second image are stored into "FIFO_1" IP core and "FIFO_2" IP core, respectively. To reduce running time and save hardware resources, the maximum number of descriptors is defined as 100. The first "XOR" operations of the first descriptor in the second image and the 100 descriptors in the first image are implemented in parallel, and the second "XOR" operations are implemented between the second descriptor in the second image and the 100 descriptors in the first image, the later "XOR" operation is in order. As seen from Figure 9b, the hamming distance of two descriptors is computed by "+" operation. Meanwhile, the 100 hamming distances are computed in parallel and multilevel. To determine a point-pair, a minimum value can be determined from the 100 hamming distances by compare operations (see Figure 9c) in parallel.

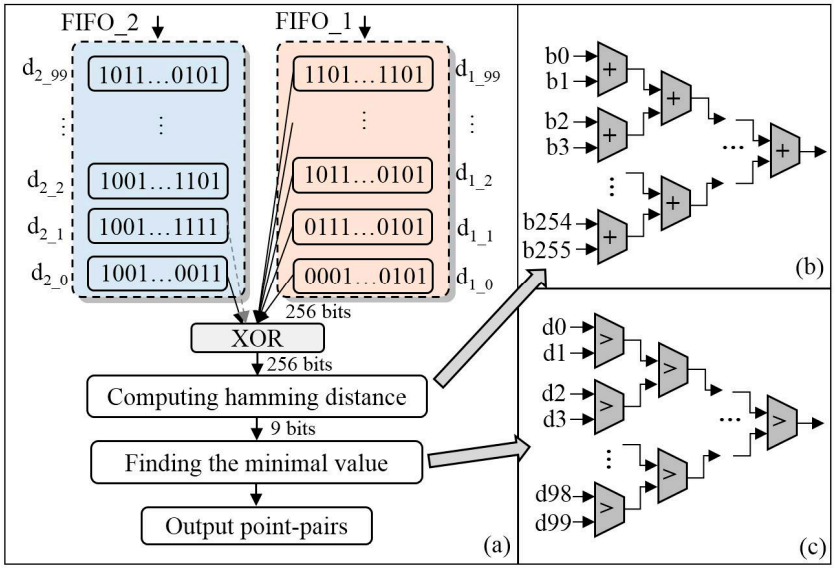


**Figure 9.** (**a**) Implementation process of matching; (**b**) Hamming distance computation; (**c**) Locating of the minimal value.

## 4. Experiments and Discussion

### *4.1. Hardware Environment and Data Set*

We implement the proposed architecture on a custom-designed board which contains a Xilinx XC72K325T FPGA. The selected FPGA has 326,080 Logic Cells, 4000 kb Block RAMS and 840 DSP Slices. The resources of board are enough to implement the whole design. Detail resource utilization is listed in Table 3. In addition, the design tool is Vivado (2014.2 version), the simulation tool is Modelsim-SE 10.4, and the hardware design language is Verilog HDL. Microsoft Visual Studio software (2015 version) and OpenCV library (2.4.9 version) are used for comparison on PC.

A pair of images with known homography matrix has widely been used to evaluate the performance of the object detection and image matching algorithm [36]. In this paper, three pairs of images with different land coverages (buildings, roads, and woods) are downloaded from [37], which are captured by IKONOS sensor with 1-meter black-and-white (panchromatic) resolution. Homograhpies of each pair of images are computed using a "findHomography" function which is called from OpenCV library (2.4.9 version) [38]. The computed results of the homographies for the three pairs of images are described as follows:

$$H_{Building} = \begin{bmatrix} 0.938443 & -0.029160 & 5.169672 \\ -0.008925 & 0.924411 & 9.490082 \\ -0.000121 & -0.000222 & 1 \end{bmatrix} \quad (7)$$

$$H_{Road} = \begin{bmatrix} 0.945168 & -0.020939 & 3.765825 \\ -0.055158 & 0.924799 & 12.459159 \\ -0.000151 & -0.000221 & 1 \end{bmatrix} \quad (8)$$

$$H_{Wood} = \begin{bmatrix} 1.064960 & 0.049669 & -3.542474 \\ -0.054367 & 1.120186 & -21.377591 \\ -1.031071 & 0.000489 & 1 \end{bmatrix} \quad (9)$$

### *4.2. Experiment Results*

To evaluate the performance of FPGA-based implementation, the image pairs are sent to the proposed FPGA architecture as the originated input, and the locations of point pairs are outputted to the final results. To compare the proposed method, the experimental results are computed by Matlab R2014a version software on PC is depicted in Figure 10. As seen from Figure 10a, when the image pair consists of the buildings, most of the point pairs are successfully matched except several mismatches**.** In Figure 10b, when the image pairs consist of roads, most of the point pairs are still successfully matched but the rate is less than amount in Figure 10a. In Figure 10c, although lots of the point pairs are effectively matched, the number of mismatches is increasing. To quantitatively analyze the performance of the FPGA-based implementation, a comprehensive analysis, which includes accuracy and speed, are presented in the Section 4.3.

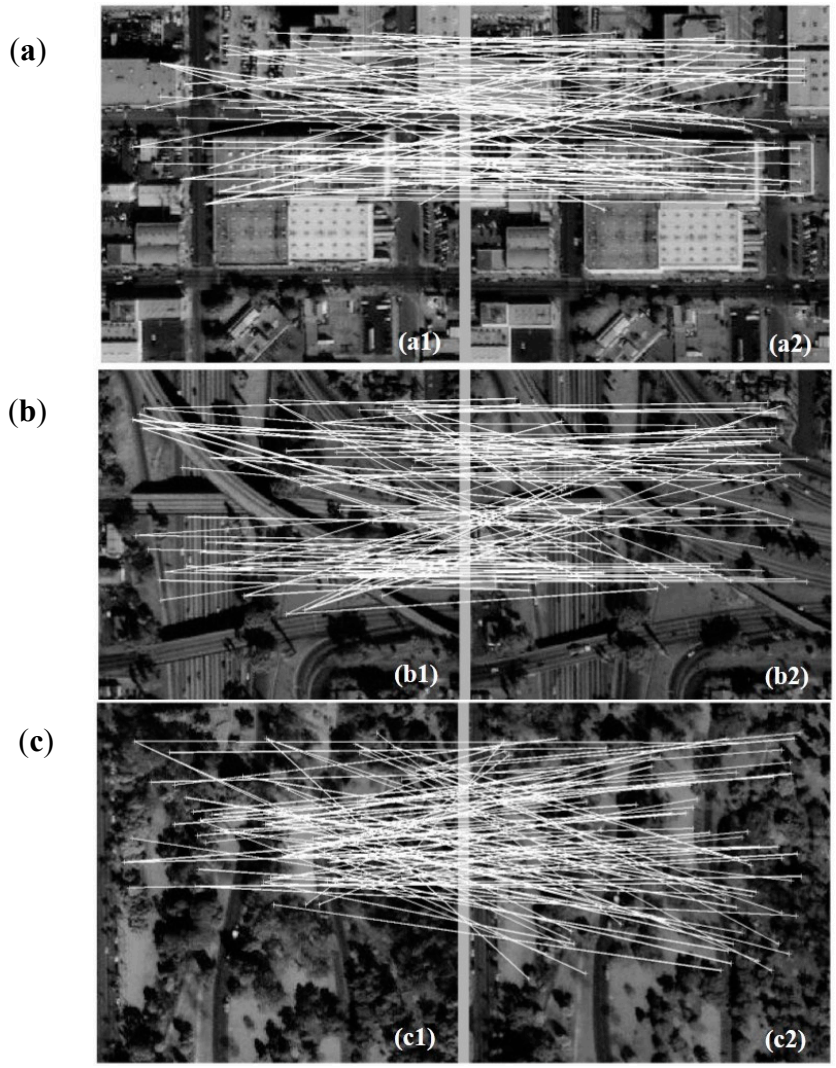


**Figure 10.** Detection and matching in image pairs with building (**a**), road (**b**), and wood (**c**).

### *4.3. Performance Evaluation*

The performance of the FPGA-based implementation is evaluated from the accuracy and the speed. On accuracy side, we will focus on the differences of the locations matched by FPGA and the real locations computed with the known homography matrixes (Equations (7)–(9)). On the speed side, we will focus on the speedup of the FPGA-based operation by comparing with PC-based operation, and the previous work. Details of evaluation are listed as follows.

4.3.1. Accuracy Analysis

A criterion based on the number of correct matches and the number of false matches is firstly proposed by Mikolajczyk [36]. The criterion is widely used to assess the performance of detection and matching algorithm. The evaluation method is presented as a curve of *recall* vs. *1-precision* [36]. The curve is generated below a threshold $t$ which determined if two descriptors are matched. Given one image pair covering the same land coverages, the *recall* as shown in Equation (10) is the ratio of the number of the correctly matched points to the number of corresponding matched points:

$$recall = \#correct\_matches/\#correspondence \quad (10)$$

In practice, the number of corresponding points is determined by the overlapping of the points in the image pair. The *1-precision* as shown in Equation (11) is the ratio of the total number of the falsely matched points to the sum of the number of the correctly matched points and the number of the falsely matched points:

$$1\text{-}precision = \#false\_matches/(\#correct\_matches + \#false\_matches) \quad (11)$$

For the purpose of comparison, the corresponding OpenCV-based models of the SURF + BRIEF and SURF have also been implemented on PC. In the comparison, the Microsoft Visual Studio software (2015 version) and C++ programming language are used in this paper. Certainly, the corresponding detection and matching functions are called from the OpenCV2.4.9 library [38] which is added into the source library of Microsoft Visual Studio 2015. Three image pairs processed by FPGA are also selected as the input data of PC-based implementation, and the number of detection and matching is about 100. The matching performance for the two methods are also obtained and presented with the curves of *recall* vs. *1-precision* as shown in Figure 11. As seen from Figure 11, no matter the image pair with buildings, roads, or woods, the red curve is always keep the similar level, which means that the SURF + BRIEF has a better matching performance in different land coverages. While the black curve is getting lower, which means that the matching performance of SURF is getting worse when the image pair covering the natural features, such as woods. The reasons are that the descriptor of SURF + BRIEF is more robust. Hence, the SURF + BRIEF is feasible to be implemented in FPGA.

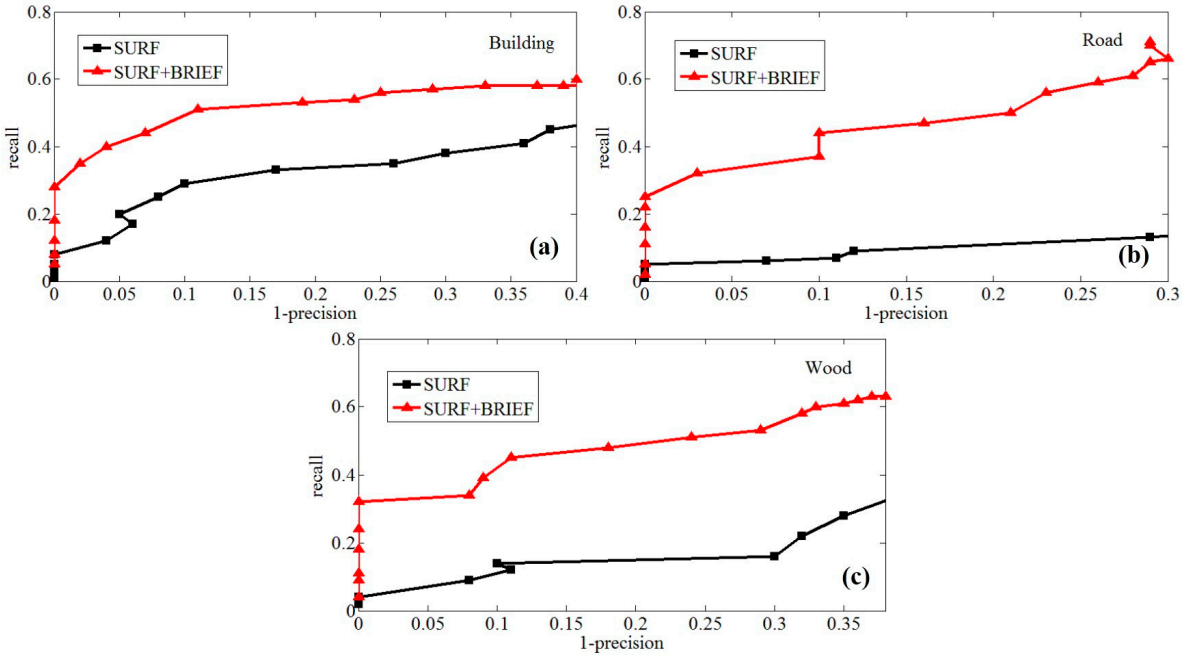


**Figure 11.** Accuracy comparison of SURF + BRIEF and SURF in image pairs with building (**a**), road (**b**), and wood (**c**).

After evaluating the performance of SURF + BRIEF and SURF implemented on PC, the performance evaluation of the FPGA-based implementation is also conducted. The results of PC-based implementation (namely the red curve in Figure 11) are used in comparison. The curves of *recall* vs. *1-precision* of FPGA-based implementation and PC-based implementation are presented in Figure 12. As seen from Figure 12a, when the image pair consists of the buildings, the FPGA-based implementation can achieve a very close performance to the PC-based implementation. In Figure 12b, when the image pair consists of roads, the performance of FPGA-based implementation is slightly worse than that of the PC-based implementation. When the image pair consists of woods, the performance of the FPGA-based implementation is worse than that of PC-based implementation (see Figure 12c). The reasons may be caused by:

(1) Only two octaves are used to extract the feature points on FPGA implementation, which will inevitably lead to performance degradation;
(2) Some divisions of the algorithm are implemented on FPGA by right shift operation which may cause some error. Additionally, fix points are adopted in the whole system. Some calculation error may propagate and accumulate which finally lead to some false matching points;
(3) Because of the different land coverages, the descriptors generated by these image pairs with artificial features (such as buildings and roads) are more robust than these generated by the image pairs with natural features (such as woods).

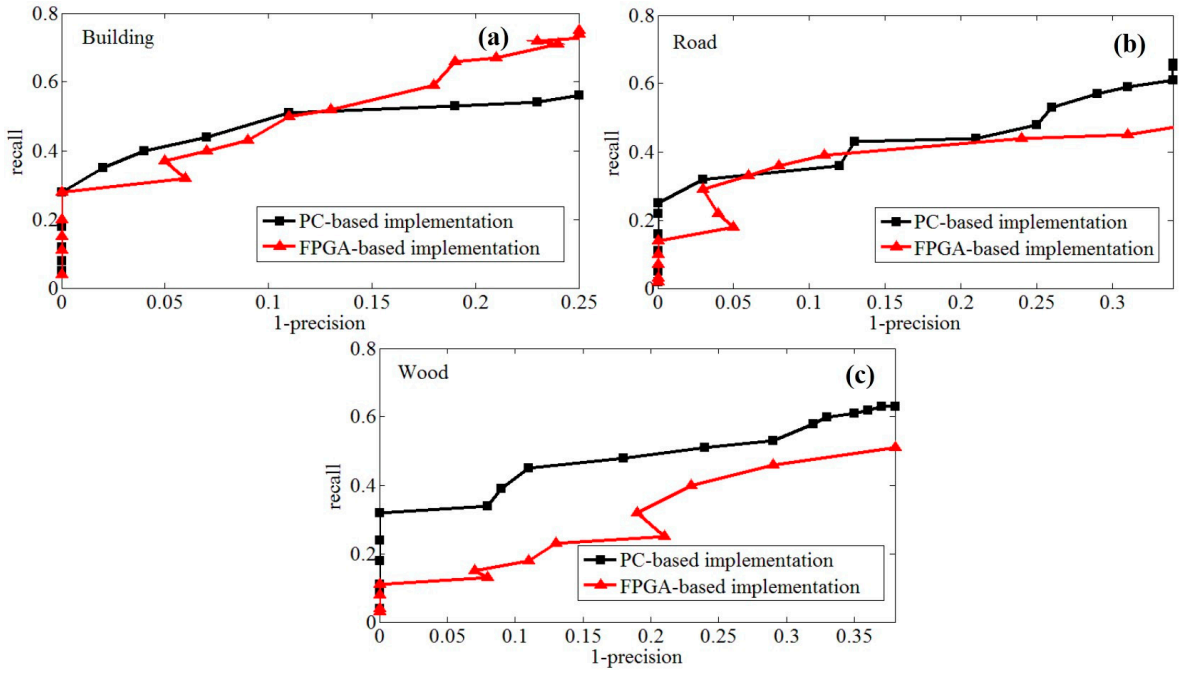


**Figure 12.** Comparison of FPGA-based and PC-based implementation of SURF + BRIEF algorithm in image pairs with building (**a**), road (**b**), and wood (**c**).

### 4.3.2. Speed Comparison

The speed is one of the most importance factors on on-board detection and matching. In this section, the comparison consists of two aspects. The first aspect is a comparison of the FPGA-based implementation and the PC-based implementation. The other aspect is the comparison of the FPGA-based implementation and the previous work.

The proposed method is running a PC with a Windows 7 (64 bit) operation system, which is equipped with an Intel(R) Core(TM) i7-4790 CPU @ 3.6GHz and a RAM 8 GB. To keep the same situations in comparison, the size of the image pair is same as the image pair processed by FPGA, and the number of point pairs is also about 100. The run time of the PC-based implementation and the FPGA-based implementation are 90 ms (11 fps) and 3.29 ms (304 fps), respectively. The results

show that the FPGA-based implementation can achieve 27 times speedup compared with the PC-based implementation.

Considering the SURF algorithm and FAST + BRIEF algorithm are successfully implemented on FPGA by the previous work, the proposed algorithm is compared with SURF algorithm and FAST + BRIEF algorithm. The comparison results are listed in Table 3. As listed in Table 3, the SURF [22] algorithm has a highest fps, while the clock frequency is also highest. If the clock frequency is defined as 100 MHz, the fps can't achieve 356. The FAST + BRIEF [39] reach 325 fps, while the clock frequency is only 100 MHz. The proposed method reaches 304 fps with the same clock frequency as [39]. In addition, the speedup of the proposed method is most significant. The reasons why the fps of the proposed method is lower than the [39] are that

(1) In detection phase, the SURF detector is more time-consuming than FAST detector;
(2) The image column in this paper is larger than the image column of [39]. The larger column will take more time in operation on FPGA.

While the speed of the proposed method is still acceptable and satisfactory when implement on FPGA.

**Table 3.** Comparison with previous work.

| Algorithm | Row × Column | Clock (MHz) | Feature Points | fps | Speedup (time) | FPGA Type |
|---|---|---|---|---|---|---|
| SURF [22] | 640 × 480 | 156 | 100 | 356 | 6 | XC6CSX475T, Xilinx |
| FAST + BRIEF [39] | 640 × 480 | 100 | / | 325 | 15–25 | XC7Z020, Xilinx |
| Proposed Method | 512 × 512 | 100 | 100 | 304 | 27 | XC72K325T, Xilinx |

#### 4.3.3. FPGA Resources Utilization Analysis

In this FPGA-based implementation of the proposed method, the utilization of FPGA resources is presented in Table 4. As seen from Table 4, the maximal utilization is LUTs, which is about 43%, while only 19% in [39]. The minimal utilization is BRAMs, which is about 0%, while increases to 27% and 28% in [39] and [22], respectively. The utilization of DSPs is 59% in [22], while are 0 in [39] and this work. Furthermore, the FPGA resource of this work will reduce when the whole system is further optimized.

**Table 4.** The resource utilization by entity (XC7325T FPGA).

| Resources | SURF [22] | SURF + BRIEF [39] | Proposed Method |
|---|---|---|---|
| FFs | / | 17,412 (16%) | 122,000 (30%) |
| LUTs | 107,873 (36%) | 9866 (19%) | 88,462 (43%) |
| BRAMs | 295 (28%) | 38 (27%) | 1 (~0%) |
| DSPs | 1185 (59%) | / | / |

### *4.4. Discussion*

Here, we present a FPGA-based implementation of SURF + BRIEF algorithm and firstly adopted three image pairs with different land coverages to evaluate the FPGA-based implementation. As the experiment results indicate, we believe that the performance of the FPGA-based implementation is satisfied. Especially when the image pairs with the artificial features (such as buildings and roads), the accuracy of the FPGA-based implementation is similar to the PC-based implementation. However, when the image pair with woods, the accuracy of the FPGA-based implementation is lower than the PC-based implementation.

Furthermore, no matter which land coverages are covered in image pairs, the accuracy of the FPGA-based implementation still can't reach the rigorous same as the PC-based implementation [39]. The main reasons are that (1) only two octaves are used in FPGA, while more octaves are adopted in PC; (2) the fix-point and shift operation are used, while floating point with 64 bits and multiplier/divider are adopted directly in PC.

Based on the hardware characteristic of FPGA, the FPGA architecture can achieve task parallel processing and pipeline processing. For instance, the task parallel means that each modules are operating in parallel and independent, and the pipeline processing means that each modules can deal with different parts of the same image. Meanwhile, the next module is not affected by the last module.

In sum, because of this characteristic of FPGA, the on-board detection and matching of feature points are feasible.

## 5. Conclusions

This paper presents a FPGA-based method for on-board detection and matching. A pipeline structure and a parallel computation are presented in this paper. A model which combines the modified SURF detector and a BRIEF descriptor is presented and implemented in FPGA. During the process of implementation, (1) a computing through the overflow technique is used to reduce bit width of integral image and a right shift operation is used instead of a divider; (2) the responses of Hessian matrix in different scales are computed in parallel. The parallel processing also can be found in the 74 comparators in 3D non-maximal suppression module, the 256 comparators in BRIEF generator module, and 100 comparators in matching module.

Three pairs of images with different land coverages are applied to evaluate the performance of FPGA-based implementation. The experiment results demonstrate that (1) when the image pairs consists of artificial features (such as buildings and roads), the performance from FPGA-based implementation is very close to that from the PC-based implementation. When the image pairs consists of natural features (such as woods), the performance by FPGA-based implementation is getting worse; (2) the FPGA-based implementation can achieve a 304 fps under a 100 MHz clock frequency, which is about 27 times comparing with PC-based implementation.

**Acknowledgments:** This paper is financially supported by grand of China Natural Science Foundation (Grant No. 41431179). The National Key Research and Development Program of China under Grant numbers 2016YFB0502501 and The State Oceanic Administration under Grant numbers [2014]#58. GuangXi Key Laboratory for Spatial Information and Geomatics Program (Contract No. 151400725, GuiKeHe 14123001-4). Tianjin Research Program of Application Foundation and Advanced Technology (Contract No. 14JCYBJC41700).

**Author Contributions:** G. Zhou conceived and designed the experiments. J. Huang performed the experiments and wrote the initial paper.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Mair, E.; Hager, G.D.; Burschka, D.; Suppa, M.; Hirzinger, G. Adaptive and generic corner detection based on the accelerated segment test. In Proceedings of the 11th European Conference on Computer Vision, Crete, Greece, 5 September 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 183–196.
2. Zhou, G.Q.; Baysal, O.; Kaye, J.; Habib, S.; Wang, C. Concept design of future intelligent earth observing satellites. *Int. J. Remote Sens.* **2004**, *25*, 2667–2685.
3. Zhang, B. Intelligent remote sensing satellite system. *J. Remote Sens.* **2011**, *15*, 415–431.
4. Pingree, P. Chapter 5: Advancing NASA's on-board processing capabilities with reconfigurable FPGA Technologies. In *Aerospace Technologies Advancements*; InTech: Rijeka, Croatia, 2010; pp. 69–86.
5. Dawood, A.S.; Visser, S.J.; Williams, J.A. Reconfigurable FPGAs for real time image processing in space. In Proceedings of the 14th International Conference on Digital Signal Processing, Santorini, Greece, 1–3 July 2002; Volume 2, 845–848.
6. Mueller, R.; Teubner, J.; Alonso, G. Data processing on FPGAs. *Proc. VLDB Endow.* **2009**, *2*, 910–921.
7. Ball, J. The nios II family of configurable soft-core processors. In Proceedings of the 2005 IEEE Hot Chips XVII Symposium (HCS), Stanford, CA, USA, 14–16 August 2005; pp. 1–40.
8. González, D.; Botella, G.; García, 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**, *2013*, 1–20.

9. González, D.; Botella, G.; García, C.; Bäse, M.; Bäse, U.M.; Prieto-Matías, M. Customized Nios II multi-cycle instructions to accelerate block-matching techniques. In Proceedings of the SPIE/IS&T Electronic Imaging. International Society for Optics and Photonics, San Francisco, CA, USA, 8–12 February 2015; pp. 940002-1–940002-14.
10. 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.
11. Meyer-Bäse, U.; Botella, G.; Castillo, E.; García, A. Nios II hardware acceleration of the epsilon quadratic sieve algorithm. In Proceedings of the SPIE Defense, Security, and Sensing, Orlando, FL, USA, 5–9 April 2010; pp. 77030M-1–77030M-10.
12. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. In Proceedings of the 2011 IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 2564–2571.
13. Yao, L.; Feng, H.; Zhu, Y.; Jiang, Z.; Zhao, D. An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher. In Proceedings of the International Conference on Field-Programmable Technology, Sydney, Australia, 9–11 Decenmber 2009; pp. 30–37.
14. Svab, J.; Krajnik, T.; Faigl, J.; Preucil, L. FPGA based speeded up robust features. In Proceedings of the IEEE International Conference on Technologies for Practical Robot Applications, Woburn, MA, USA, 9–10 November 2009; pp. 35–41.
15. Schaeferling, M.; Kiefer, G. Flex-SURF: A flexible architecture for FPGA-based robust feature extraction for optical tracking systems. In Proceedings of the 2010 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 13–15 December 2010; pp. 458–463.
16. Lentaris, G.; Stamoulias, I.; Diamantopoulos, D.; Siozios, K.; Soudris, D. An FPGA implementation of the SURF algorithm for the ExoMars programme. In Proceecdings of the 7th HiPEAC Workshop on Reconfigurable Computing (WRC2013), Berlin, Germany, 21–23 January 2013.
17. Schaeferling, M.; Kiefer, G. Object recognition on a chip: A complete SURF-based system on a single FPGA. In Proceedings of the 2011 International Conference on Reconfigurable Computing and FPGAs, Cancun, Mexico, 30 November–2 December 2011; pp. 49–54.
18. Sledevič, T.; Serackis, A. SURF algorithm implementation on FPGA. In Proceedings of the 2012 13th Biennial Baltic Electronics Conference, Tallinn, Estonia, 3–5 October 2012; pp. 291–294.
19. Battezzati, N.; Colazzo, S.; Maffione, M.; Senepa, L. SURF algorithm in FPGA: A novel architecture for high demanding industrial applications. In Proceedings of the Conference on Design, Automation and Test in Europe, Dresden, Germany, 12–16 March 2012; pp. 161–162.
20. Zhao, J.; Zhu, S.; Huang, X. Real-time traffic sign detection using SURF features on FPGA. In Proceedings of the 2013 IEEE High Performance Extreme Computing Conference (HPEC), Waltham, MA, USA, 10–12 September 2013; pp. 1–6.
21. Evans, C. *Notes on the Opensurf Library*; Tech. Rep. CSTR-09-001; University of Bristol: Bristol, UK, 2009.
22. Fan, X.; Wu, C.; Cao, W.; Zhou, X.; Wang S.; Wang L. Implementation of high performance hardware architecture of OpenSURF algorithm on FPGA. In Proceedings of the 2013 International Conference on Field-Programmable Technology, Kyoto, Japan, 9–11 December 2013; pp. 152–159.
23. Krajnik, T.; Svab, J.; Pedre, S.; Cizek, P.; Preucil, L. FPGA-based module for SURF extraction. *Mach. Vis. Appl.* **2014**, *25*, 787–800.
24. Chen, C.; Yong, H.; Zhong, S.; Yan, L. A real-time FPGA-based architecture for OpenSURF. In Proceedings of the MIPPR 2015: Pattern Recognition and Computer Vision, Enshi, China, 31 October 2015; pp. 1–8.
25. Gonzalez, C.; Bernabe, S.; Mozos, D.; Plaza A. FPGA implementation of an algorithm for automatically detecting targets in remotely sensed hyperspectral images. *IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens.* **2016**, *9*, 4334–4343.
26. Weberruss, J.; Kleeman, L.; Drummond, T. ORB feature extraction and matching in hardware. In Proceedings of the Australasian Conference on Robotics and Automation, the Australian National University, Canberra, Australia, 2–4 December 2015; pp. 1–10.
27. Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P. Brief: Binary robust independent elementary features. In Proceedings of the 11th European Conference on Computer Vision, Crete, Greece, 5 September 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 778–792.

28. Bay, H.; Tuytelaars, T.; Gool, L.V. SURF: Speeded up robust features. In Proceedings of the 9th European Conference on Computer Vision, Graz, Austria, 7–13 May 2006; Springer Berlin/Heidelberg, Germany; pp. 404–417.
29. Ma, X.; Borbon, J.R.; Najjar, W.; Roy-Chowdhury, A.K. Optimizing hardware design for Human Action Recognition. In Proceedings of the 2016 26th International Conference on Field Programmable Logic and Applications (FPL), Lausanne, Switzerland, 29August–2 September 2016; pp. 1–11.
30. Ma, X.; Najjar, W.; Chowdhury, A.R. High-Throughput Fixed-Point Object Detection on FPGAs. In Proceedings of the 2014 IEEE 22nd Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Boston, MA, USA, 11–13 May 2014; pp. 107–107.
31. Ma, X.; Najjar, A.; Roy-Chowdhury, A.K. Evaluation and Acceleration of High-Throughput Fixed-Point Object Detection on FPGAs. *IEEE Trans. Circuits Syst. Video Technol.* **2015**, *25*, 1051–1062.
32. Sousa, R.; Tanase, A.; Hannig, F.; Teich, J. Accuracy and performance analysis of Harris Corner computation on tightly-coupled processor arrays. In Proceedings of the 2013 Conference on Design and Architectures for Signal and Image Processing (DASIP), Cagliari, Italy, 8–10 October 2013; pp. 88–95.
33. Belt, H.J. Word length reduction for the integral image. In Proceedings of the 15th IEEE International Conference on image processing, San Diego, CA, USA, 12–15 October 2008; pp. 805–808.
34. Xilinx. 7 Series FPGAs Memory Interface Solutions. Available online: http://www.xilinx.com/support/documentation/ip_documentation/ug586_7Series_MIS.pdf (accessed on 1 March 2011).
35. Cornelis, N.; Gool, L.V. Fast scale invariant feature detection and matching on programmable graphics hardware. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition Work-Shops, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
36. Mikolajczyk, K.; Schmid, C. A performance evaluation of local descriptors. *IEEE Trans. Pattern Anal. Mach. Intell*. **2005**, *27*, 1615–1630.
37. Ikonos Data Set—High-Resolution Satellite Data. Available online: http://www.isprs.org/data/ikonos/default.aspx (accessed on 12 June 2017).
38. OpenCV library. Available online: http://opencv.org/ (accessed on 12 June 2017).
39. Fularz, M.; Kraft, M.; Schmidt, A.; Kasinski, A. A high-performance FPGA-based image feature detector and matcher based on the fast and brief algorithms. *Int. J. Adv. Robot. Syst.* **2015**, *12*, 10, 1–15.