Ground Control Point Automatic Extraction for Spaceborne Georeferencing Based on FPGA

Dequan Liu, Guoqing Zhou, *Senior Member, IEEE*, Dianjun Zhang, Xiang Zhou, and Chenyang Li

***Abstract*—Feature points that are obtained from the combined speeded-up robust feature (SURF) detector and binary robust independent elementary features (BRIEF) descriptor have a highly robust performance. These points are previously considered the ground control points (GCPs) for building a connection between the image coordinates and the corresponding geodetic coordinates. This article proposes a novel architecture to automatically and intelligently extract GCPs based on field programmable gate arrays (FPGAs). The parallelization SURF detector, BRIEF descriptor, and BRIEF matching are implemented in a single Xilinx XC7VX980T FPGA system. Word length reduction, memory-efficient parallel architecture, shift and subtraction strategies, a sliding window for separable convolution, and an optimized multispacer-scale are used to optimize the SURF detector. Improved parallel adder trees are used to accelerate the BRIEF matching. The proposed system achieves 380 frame per second (fps) with a 100 MHz clock frequency, which satisfies the real-time and low-power requirements of embedded devices. The results of the experiment demonstrate that the proposed architecture, when mapped onto a Xilinx Virtex-7 XC7VX980T FPGA device, can select the robust feature points.**

***Index Terms*—Binary robust independent elementary features (BRIEF), field programmable gate arrays (FPGAs), georeferencing, ground control points (GCPs), speeded-up robust feature (SURF).**

I. Introduction

GEOREFERENCING is an important technique of remote sensing (RS) image for an extensive variety of tasks, and estimation of the mathematical geometric calibration function of RS image based on the ground control points (GCPs) is necessary. Traditionally, GCPs are obtained by special equipment or manually selected from a reference image or topographic, which is computationally expensive. Furthermore, the traditionally method cannot satisfy the practical real-time performance [1]–[3] of RS image processing. The primary task of the RS image matching is to find the correct GCPs correspondences on the reference image and the warped image [4]. Therefore, an automatic method for selecting GCPs is desired. This article aims to automatically extract GCPs between the remotely sensed image and the corresponding reference image in real-time.

In image matching, the keypoint can be recognized between two or more images of the same scene. The images can be employed in frequent matching for different times, perspectives, and scales [5]. The local invariant feature has an important role in image matching. The literature on local feature detection is vast and dates to 1954, when Attneave [6] observed that information about shape was concentrated at dominant points with high curvature [7]. In 1980, Moravec [8] proposed a detector that was repeatable for small variations and near edges and was applied for stereo image matching. Harris and Stephens [9] improved the prior Moravec detector in 1988. The Harris corner detector includes gradient information and eigenvalues of symmetric positive, which are defined as a 2×2 matrix to make it more repeatable. The Harris corner detector is a prevalent feature detection technique that combines a corner detector and edge detector based on a local autocorrelation function. However, it is not scale-invariant and sensitive to noise [10]. Smith and Brady [11] developed the smallest univalue segment assimilating nucleus (SUSAN) detector in 1997. SUSAN is not sensitive to local noise and high anti-interference ability [4]. To obtain the scale-invariant feature, Lindeberg [12]–[14] investigated the scale-invariant theory and presented a framework for selecting local appropriate scales that can be used for automatic scales selection. Mikolajczyk *et al.* [15] presented the Harris–Laplace and Harris–Affine detectors and gradient location and orientation histogram detector [16]. Lowe [17] proposed the scale-invariant feature transform (SIFT) algorithm. Since each layer relied on the previous layer and images had to be resized, it was not computationally efficient [18]. Many SIFT variants, such as PCA-SIFT [19], GSIFT [20], CSIFT [21], and ASIFT [22], are relatively highly efficient. The speeded-up robust features (SURF) is a local descriptor that is inspired by SIFT. SURF was first introduced in [23] and fully explained in [24].

Recently, binary descriptors are being developed for image processing. The binary robust invariant scalable keypoints (BRISK) detector is a novel method for keypoint detection, and description, and matching method that was proposed by

Manuscript received January 17, 2020; revised May 15, 2020; accepted May 26, 2020. Date of publication June 1, 2020; date of current version June 29, 2020. This work was supported in part by the National Natural Science of China under Grants 41961065 and 41431179, in part by the Guangxi Innovative Development Grand Program under Grants Guike AD19254002, GuikeAA18118038, and GuikeAA18242048, in part by the Guangxi Natural Science Foundation for Innovation Research Team under Grant 2019GXNSFGA245001, in part by the Guilin Research and Development Plan Program under Grant 20190210-2, in part by National Key Research and Development Program of China under Grant 2016YFB0502501, and in part by the BaGuiScholars Program of Guangxi. (*Corresponding author: Guoqing Zhou*.)

Dequan Liu is with the School of Microelectronics, Tianjin University, Tianjin 300072, China (e-mail: 1017232001@tju.edu.cn).

Guoqing Zhou and Xiang Zhou are with the School of Microelectronics, Tianjin University, Tianjin 300072, China, and also with the GuangXi Key Laboratory for Spatial Information and Geomatics, Guilin University of Technology, Guilin, Guangxi, 541004, China (e-mail: gzhou@glut.edu.cn; zqx0711@tju.edu.cn).

Dianjun Zhang and Chenyang Li are with the School of Marine Science and Technology, Tianjin University, Tianjin 300072, China (e-mail: zhangdj@tju.edu.cn; lichenyang_1008@tju.edu.cn).

Digital Object Identifier 10.1109/JSTARS.2020.2998838

Leutenegger *et al.* [25]. BRISK is constructed by pixel comparisons, whose distribution forms a concentric circle surrounding the feature. Calonder *et al.* [26] investigated the binary robust independent elementary features (BRIEF) to efficiently extract features. The BRIEF descriptor vector consists 512-, 256-, and 128-bit vectors. Hence, this feature substantially reduces the memory required to store the feature descriptor and the time consumed to match the features, while yielding comparable recognition accuracy [27]. Rublee *et al.* [28] proposed a very fast binary descriptor, named ORB, which was rotation invariant and resistant to noise. The main contribution of this work is the addition of an orientation component to the feature from accelerated segment test (FAST) [29] feature detector and to propose a learning method for choosing pairwise tests with excellent discrimination power and a low correlation response among the tests [30]. In [31], Alahi *et al.* suggested the fast retina keypoint (FREAK) as a fast compact and robust keypoint descriptor.

Feature-based matching algorithms have been extensively employed for a variety of applications, such as object localization, object recognition, motion estimation, and 3-D reconstruction. These algorithms exhibit high-performance in image matching. However, they cannot achieve a performance that is sufficiently high to satisfy practical real-time requirements due to the computational complexity and vast memory consumption [32]–[35]. Therefore, researchers have applied the matching algorithm to real-time applications. These studies can be grouped into two main approaches: The first approach aims to reduce the complexity of the matching algorithm without losing precision. A principal component analysis (PCA) [19] and linear discriminant analysis [36] are dimensionality reduction techniques that reduce the size of the original descriptor, such as SIFT or SURF [26], [37]. Calonder *et al.* [38] proposed a concept that used a shorten descriptor to quantize its floating-point coordinates into integers codes on fewer bits; the same result was proposed in [39] and [40]. This concept is an effective way to replace the original complex detector or descriptor with a speeded-up detector or binary description, such as FAST [29], BRISK [25], BRIEF [26], ORB [28], and FREAK [31]. Lowe [17] approximated the Laplacian of Gaussian (LoG) via the difference of Gaussians filter. Bay *et al.* [23] proposed approximation to the LoG by using box filter representations of the respective kernels based on SIFT.

The second method is focused on improving the processing speed using dedicated hardware such as multicore central processing units (CPUs), graphic processing units (GPUs), application specific integrated circuits, and field programmable gate arrays (FPGAs). Čížek *et al.* [41] proposed a processor-centric FPGA-based architecture for a latency reduction in the vision-based robotic navigation. Krajník *et al.* [42] presented a complete hardware and software solution of an FPGA-based computer vision embedded module that can carry out the SURF image feature extraction algorithm. Yao *et al.* [43] proposed an architecture of optimized SIFT feature detection for an FPGA implementation of image matching. The total dimension of the feature descriptor had been reduced to 72 from 128 of the original SIFT. In [44], a parallelization and optimization method to effectively accelerate the SURF was proposed and achieved a maximum of 83.80 frame per second (fps) in a real-machine experiment, which enables real-time processing. Cheon *et al.* [45] analyzed the SURF algorithm and presented a fast descriptor extraction method that eliminated redundant operations in the Haar wavelet response step without additional resources. Kim *et al.* [46] presented a parallel processing technique for real-time feature extraction in object recognition by autonomous mobile robots, which utilized both CPU and GPU by combining OpenMP, Streaming SIMD Extension and CUDA programming. Schaeferling *et al.* [47] described two embedded systems (ARM-based microcontroller and intelligent FPGA) for object detection and pose estimation using sophisticated point features. The feature detection step of the SURF algorithm was accelerated by a special IP core. Huang *et al.* [48] designed an architecture that combined the FAST detector and the BRIEF descriptor for detection and matching with subpixel precision. Lima *et al.* [30] proposed a hardware architecture based on the BRIEF descriptor. This approach contributed to reducing the number of memory accesses required to obtain the descriptor while maintaining its discrimination quality. Zhao *et al.* [49] presented an efficient real-time FPGA implementation for object detection. The system employed the SURF algorithm to detect keypoints in every video frame and applied the FREAK method to describe the keypoints. In [50] and [72], a modified SURF detector and BRIEF descriptor matching algorithm based on FPGA were presented. To accelerate the SURF algorithm, an improved FAST feature point combined with the SURF descriptor matching algorithm was proposed in [51], which realized the real-time matching of target images.

Inspired by previous research, this article proposes a hardware architecture to automatically extract GCPs for the RS image that is georeferenced based on the FPGA, which significantly reduces the computational requirement. The main contribution of this article is considered as follows.

1) A parallelization SURF detector and BRIEF descriptor are designed in an FPGA.
2) Five approaches are applied to optimize the SURF detector for hardware implementation. Moreover, a novel suppressed method for candidate points that are not local maxima in the 3-D scale-space domain is suppressed zero in the nonmaximal suppression step, which ensures that feature points do not overlap and are evenly spread over the input image.
3) The BRIEF descriptor is adopted in the proposed system, whose memory footprint is only a 256-bit vector, i.e., 32-byte. However, the original SURF (O-SURF) descriptor is a 512-bit vector of floating points, representing it still requires a 64-byte. Moreover, the implementation of the BRIEF matching is highly efficient in an FPGA, which has the minimum Hamming distance between the reference image and the sensed image. Improvement adder trees are employed to reduce the complexity of the BRIEF matching step.

The rest of this article is organized as follows. Section II provides an overview of the SURF detector and BRIEF descriptor. Five approaches for optimizing SURF feature detection

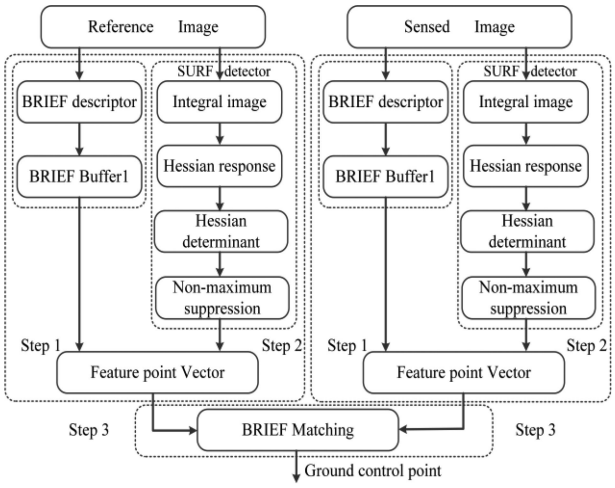

Fig. 1. Diagram of ground control point extraction for the remote sensing images.

is introduced in Section III. Section IV describes the parallel implementation on the SURF detector, the BRIEF descriptor, and the BRIEF matching, which are implemented in a Xilinx XC7VX980T FPGA. The evaluation experimental results and comparison with existing schemes are reported in Section V. Section VI concludes this article.

II. Feature Detector and Descriptor Algorithm

The process of feature-based matching has three steps: feature detection, feature description, and feature matching. In this article, an efficient real-time FPGA-based system, which comprises SURF feature detection, BRIEF descriptor, and BRIEF matching, in a single chip, is investigated. The diagram of the proposed process is presented in Fig. 1. The feature detection and feature description module automatically extract GCPs from the RS image. The feature matching module automatically matches the image pairs based on the ground control point [4].

A. SURF Feature Detector

SURF [23], [24] is scale- and rotation-invariant, which takes a grayscale image as an input. As shown in Fig. 1, SURF detector can be divided into three steps, integral image, Hessian response, and nonmaximum suppression. SURF feature detector will be briefly summarized in this section.

1) Integral Image: Integral image is a novel method to improve the performance of the subsequent steps of SURF detector [32]. The integral image is used as a rapid and effective way to calculate summations over image subregions [24]. Given the pixel value $i(x, y)$ for the coordinates (x, y) of an image with width *W* and height *H*, the value of coordinates (x, y) in the integral image $ii(x, y)$ can be defined as follows [23], [52]:

$$ii\,(x,y) = \sum_{x'=0}^{x}\sum_{y'=0}^{y} i\,(x',y') \quad 0 \le x \le W,\ 0 \le y \le H. \quad (1)$$

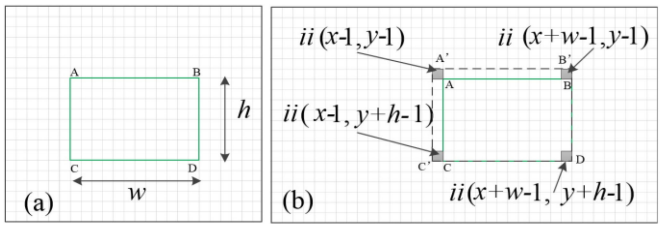

Fig. 2. Calculation cumulative sum of all pixels in the rectangular region by (2). With an integral image, the sum values of the green rectangular region that is bounded with A, B, C, and D are calculated by $ii(A') + ii(D) - ii(C') - ii(B')$.

Through the concept of integral image (as shown in Fig. 2), the cumulative sum of all the pixels in the rectangular region with the coordinates (x, y) of the top-left pixel, width *w*, and height *h* can be stated mathematically as in [53]

$$S_{w,h}(x,y) = \sum_{x'=x}^{x+w-1}\sum_{y'=y}^{y+h-1} i(x',y') = ii(x-1,y-1) + ii(x+w-1,y+h-1) - ii(x-1,y+h-1) - ii(x+w-1,y+h-1) \quad (2)$$

the initial condition of (2) is

$$ii(-1,y) = ii(x,-1) = ii(-1,-1) = 0. \quad (3)$$

According to (2), integral image provides a fast way to get the sum histogram of an arbitrary-sized rectangle, requiring only three adders and calculating near-constant-time.

2) Hessian Response: In the O-SURF [23], [24], the scale space was established by the purpose of the scale invariance. The scale space can be divided into $o(o \ge 1)$ octaves, and each octave is further divided into $v(v \ge 3)$ intervals to obtain a total of $o \times v$ scale-levels. Each interval represents the response of the Hessian determinant, which can be approximately defined as (4) [23], [24], where $\omega = 0.912$ is a weight coefficient that is used to correct the error caused by approximation. The approximated D_{xx}, D_{yy}, and D_{xy} box filter kernels are depicted in Fig. 3 [54], where white, gray, and black pixels refer to the weight values of $\{1, 0, -2\}$ for the D_{xx} and D_{yy} box filters and $\{1, 0, -1\}$ for the D_{xy} box filter, respectively [55]. By the concept of the integral image, the calculation of D_{xx} or D_{yy} requires 8 memory accesses [54], while the calculation of D_{xy} requires 16 memory accesses. The 32 memory accesses are marked with a dot in Fig. 3. The scale σ is defined in the O-SURF method by an analogy with the linear scale space. The distances between the marked points increase with an increase of σ. However, the number of points to be accessed remains constant [56]

$$\det(A) = D_{xx} \times D_{yy} - \omega^2 \times {D_{xy}}^2. \quad (4)$$

In the O-SURF method, the 9×9 box filter is defined for the first scale. However, Bay *et al.* did not specify exact values at the remaining scales [57], [58]. Hence, numerous parameters for the particular scale-levels (i, j), $i \in [1, o]$ and $j \in [1, v]$ need to be defined. Lobe l, which is one-third of L—the size of the D_{xx},

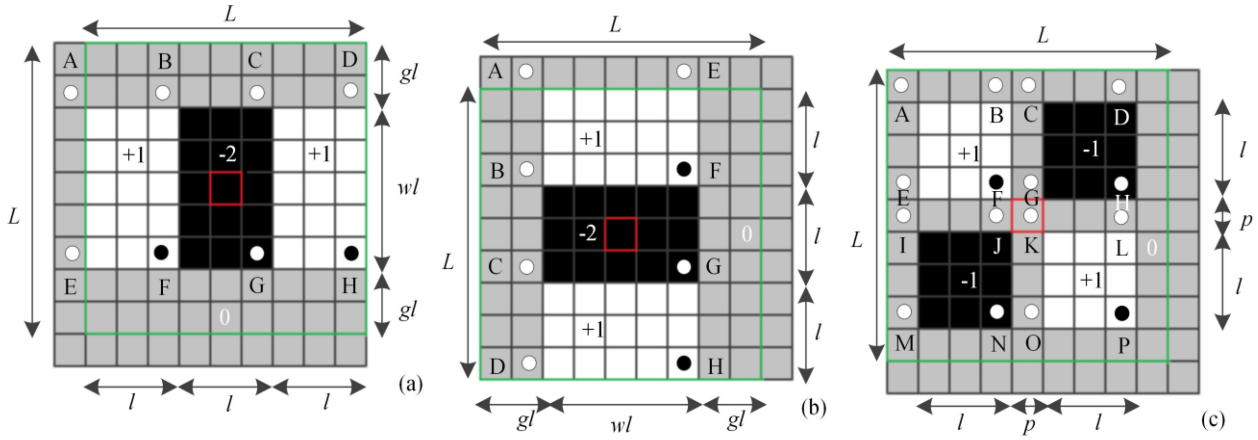

Fig. 3. 9×9 box filter that approximates the second-order Gaussian derivative in the x, y, and xy directions to obtain the D_{xx}, D_{yy}, and D_{xy} box filters. (a) D_{xx} box filter. (b) D_{yy} box filter. (c) D_{xy} box filter.

TABLE I
BOX-SPACE SAMPLING VALUES

o	*v*	*l*	*L*	*σ*	2*l*+1	(*l*-1)/2	2*l*-1	(*l*+1)/2	(3*l*+1)/2	(3*l*-1)/2
1	1	3	9	1.2	7	1	5	2	5	4
	2	5	15	2.0	11	2	9	3	8	7
	3	7	21	2.8	15	3	13	4	11	10
	4	9	27	3.6	19	4	17	5	14	13
2	1	5	15	2.0	11	2	9	3	8	7
	2	9	27	3.6	19	4	17	5	14	13
	3	13	39	5.2	27	6	25	7	20	19
	4	17	51	6.8	35	8	33	9	26	25
3	1	9	27	3.6	19	4	17	5	14	13
	2	17	51	6.8	35	8	33	9	26	25
	3	25	75	10	51	12	49	13	38	37
	4	33	99	13.2	67	16	65	17	50	49
4	1	17	51	6.8	35	8	33	9	26	25
	2	33	99	13.2	67	16	65	17	50	49
	3	49	147	19.6	99	24	97	25	74	73
	4	65	195	26.0	131	32	129	33	98	97

D_{yy}, and D_{xy} box filters, is $2^o \times v + 1$. The scale σ is $(2^o \times v + 1) \times 1.2/3 = 0.4l$. wl, which is the length of the white area in the D_{xx} and D_{yy} box filters, is $2l{+}1$ in [58] or $2l - 1$ in [59] or $(3l + 1)/2$ in MATLAB with OpenSURF by Evans [60]. gl, which is the length of the side of the gray area in the D_{xx} and D_{yy} box filters, is $(l - 1)/2$ in [58] or $(l + 1)/2$ in [59] or $(3l + 1)/4$ in [60]. The size of the octave increases by $6 \times 2^{o-1}$ pixels per interval. p is a constant of one pixel in the D_{xy} box filter. Table I shows the values of the box filters.

According to (2), the separable convolution response of the D_{xx}, D_{xy}, and D_{yy} can be computed by the following equations:

$$D_{xx} = (A + F - B - E) - 2 \times (B + G - C - F) + (C + H - D - G) \quad (5)$$

$$D_{yy} = (A + F - B - E) - 2 \times (B + G - C - F) + (C + H - D - G) \quad (6)$$

$$D_{xy} = (A + F - B - E) - (C + H - G - D) - (I + N - J - M) + (K + P - L - O). \quad (7)$$

3) Nonmaximum Suppression: To localize the interest point, the nonmaximum suppression (NMS) is applied using the three adjacent scales. The NMS compares a determinant with its 8 direction neighbors in its native scale interval, and 9 direction neighbors in each of the intervals above and below for a total of 26 direction neighbors. Moreover, a threshold is employed to determine only the most distinctive image point as a candidate point [24].

B. BRIEF Descriptor

BRIEF is a descriptor that uses binary tests between two pixels in a smoothed image patch. More specifically, if p is a smoothed image patch, the corresponding binary test τ is defined as follows [26]:

$$\tau(p; x, y) = \begin{cases} 1 & \text{if } I(p, x) < I(p, y) \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

where $p(x)$ is the intensity of p at the point x. The descriptor is defined as a vector of n_d binary tests [26]

$$f_{nd}(p) = \sum_{1<i\leq n_d} 2^{i-1} \tau(p; x_i, y_i). \quad (9)$$

The length of n_d is generally defined as 128-, 256-, and 512-bit vectors. The result of the experiment in [26] demonstrated that the performance of 256-bit vector was similar to that of the 512-bit vector, while only marginally worse in other cases [48]. Due to the limited hardware resources, a 256-bit vector is employed in this article.

C. Hamming Distance Matching

In the BRIEF descriptor [26], the Hamming distance is used to match. The Hamming distance can be efficiently calculated by the *XOR* operation. Considering the two existing descriptors S_1 and S_2, the corresponding Hamming distance (256-bit) can be defined as $D_{hd}(S_1, S_2) = \sum_{i=1}^{256} (a_i \oplus b_i)$, where $S_1 = a_1 a_2, \ldots, a_{256}$, $S_2 = b_1 b_2, \ldots, b_{256}$, and the value of a_i and b_i is 0 or 1. The smaller is the value of D_{hd}, the higher is the matching rate. Moreover, a threshold is used to check whether points truly correspond to each other. If the Hamming distance is less than a threshold, the feature points-pair are corresponding points; otherwise, they are nonmatching points [50].

III. Optimization of SURF Detector

To ensure efficient implementation of the SURF detector in an FPGA, five approaches are applied to optimize the SURF detector. The first approach, which is word length reduction (WLR), is used to reduce the word length of the integral image without a loss of accuracy. The second approach, which is a memory-efficient parallel architecture (MEPA), is used to parallel compute the output of FIFO. The third approach, which consists of shift and subtraction strategies (SAS), is adopted to simplify the response of the Hessian determinant. The SAS transforms the floating-point operation into a shift and subtraction operation. The fourth approach utilizes sliding widow, is used to parallel compute the D_{xx}, D_{yy}, and D_{xy} box filters. The fifth approach is referred to a parallel multiscale-space. Five approaches are introduced in this section.

A. Word Length Reduction (WLR)

SURF is a detector and descriptor of local scale- and rotation-invariant image features. By using integral image for image convolution, SURF computes faster than other state-of-the-art algorithms but produces comparable or even better results by utilizing repeatability, distinctiveness, and robustness [59]. However, the word length of the integral image substantially impacts on the performance of designed hardware, especially for implementations that need to store the entire integral image on FPGA [55]. To solve this problem, Hsu *et al.* [61] presented a row-based stream processing (RBSP) method that only needs a 34-row memory foot to simultaneously calculate the response of two octaves box filters. The overflow based on two's complement-coded arithmetic and rounding with error diffusion techniques were proposed by Belt [62], which can work on a face detector for a VGA resolution with a 16-bit vector. However, this approach has drawbacks, including rounding errors and an additional constraint of a fixed size for a box filter. Lee and Jeong [63] proposed a new structure for memory size reduction which includes four types of image information: an integral image, a row integral image, a column integral image, and an input image. Using this method, the integral image memory can be reduced to 42.6% for a 640 × 480 8-bit grayscale image. Ehsan *et al.* [64]–[66] proposed a parallel recursive equation to compute the integral image. This method not only substantially decreases the operation and memory requirements (by at least 44.44%) but also maintains the accuracy.

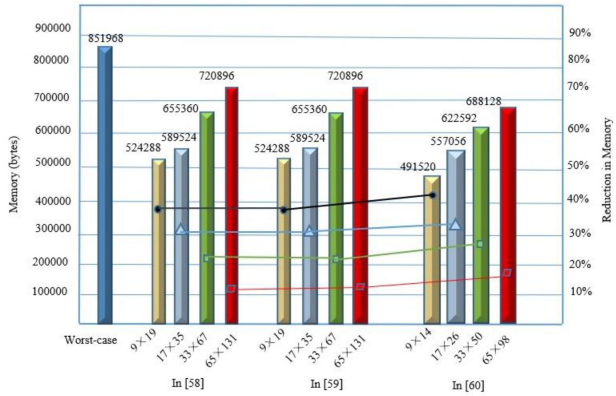

Fig. 4. Comparison of memory between worst-case and maximum width and height of the box filter.

This article focuses on reducing the size of the memory and parallelization computation. In [64], the maximum binary word length of the integral image in the worst-case (WS) is stated as follows:

$$ii_{\max} = (2^{L_i} - 1) \times W \times H \tag{10}$$

where ii is the word length of the integral image; $ii_{\max}$ is the value of WS; i is the input image; L_i is the word length per pixel of the input image; and W and H are the width and height, respectively, of the input image. According to [62], the number of bits L_{ii} required for representing the WS integral image value is $(2^{L_{ii}} - 1) \geq (2^{Li} - 1) \times W \times H$. The total memory in bytes required to store the integral image is $(W \times H) \times L_{ii}/8$. In general, the maximum width and height of the box filter are known, and the word length for the integral image using the exact method [66] with complement-coded arithmetic needs to satisfy

$$(2^{L_{ii}} - 1) \geq (2^{L_i} - 1) \times W_{\max} \times H_{\max} \tag{11}$$

where $W_{\max}$ and $H_{\max}$ are the maximum width and maximum height, respectively, of the box filter (i.e., $l_{\max} \times wl_{\max}$ or $wl_{\max} \times l_{\max}$ in Fig. 3). For example, the input images comprise 8-bit gray data with resolution $512 * 512$ pixels. Table II lists the WLR values of the different sizes of the box filter with four octaves. The total memory in bytes is shown in Fig. 4. As shown in Table II and Fig. 4, the box filters in [58] and [59] have the same memory. The OpenSURF [60] has the smallest memory compared with that of [58] and [59]. However, the value of $l/2$ is not an integer and is unsuitable for the FPGA. Compared with the WS, WLR method substantially reduces the space of memory.

TABLE II
WORD LENGTH (BIT) OF THE INTEGRAL IMAGE WITH THE MAXIMUM WIDTH AND HEIGHT OF THE BOX FILTER

Octave	WS L_{ii}	$l_{\max} \times wl_{\max}$	L_{ii} [58]	$l_{\max} \times wl_{\max}$	L_{ii} [59]	$l_{\max} \times wl_{\max}$	L_{ii} [60]
1	26	9×19	16	9×17	16	9×14	15
2		17×35	18	17×33	18	17×26	17
3		33×67	20	33×65	20	33×50	19
4		65×131	22	65×129	22	65×98	21

B. Parallel Computation Integral Image

Equation (1) can be transformed into the pipeline recursive equation that was presented by Viola–Jones [67]

$$S(x,y) = i(x,y) + S(x,y-1) \tag{12}$$

$$ii(x,y) = ii(x-1,y) + S(x,y) \tag{13}$$

where $S(x,y)$ is the cumulative row sum value at the image location (x,y).

In (1), $M^2N^2/4$ adders are used to compute the integral image for an image with resolution $M \times N$ pixels [68]. Apparently, (1) is not suitable for a medium- or high-resolution image. In the Viola–Jones parallel recursive (12) thru (13), the number of additions is $2MN$. However, the Viola–Jones method has time delay drawbacks. To accelerate the processing of an integral image, Ehsan *et al.* [66] proposed an n stage of a pipelined system that processes n rows of an input image in parallel, providing n integral image value per clock cycles without delay when the pipeline is full. This method can be mathematically defined as follows:

$$S(x+j,y) = ii(x+j,y) + S(x+j,y-1). \tag{14}$$

For odd rows

$$ii(x+2k,y) = ii(x+2k-1,y) + S(x+2k,y). \tag{15}$$

For even rows

$$ii(x+2m+1,y) = ii(x+2m-1,y) + S(x+2m,y) + S(x+2m+1,y) \tag{16}$$

where n is the number of rows to be calculated (always a multiple of 2),$j = 0,\ldots,n-1, k = 0,\ldots,n/2-1$, and $m = 0,\ldots,n/2-1$. This set of equations requires $2MN + MN/2$ addition operation for an input image with the resolution $M \times N$ pixels [66]. Compared with Viola–Jones equation, the increase is not significant.

To ensure a trade-off between the computation time and consumption memory, the four-row parallel method is adopted in the integral image module.

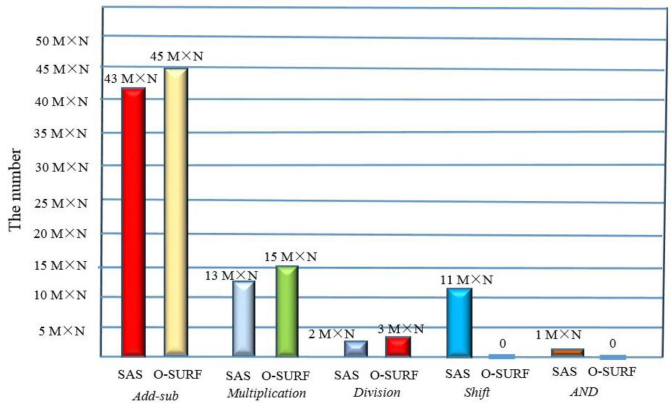

Fig. 5. Comparison between the SAS and the O-SURF for an image with the resolution pixels.

C. Shift and Subtraction (SAS)

In (4), the weight coefficient $\omega^2 = 0.831744$ $(\omega = 0.912)$ is derived to minimize the approximation error caused by the box filters in the O-SURF. Hence, a floating-point architecture is required to compute $\det(H)$. OpenSURF does not use this value, but instead applies 0.81 $(\omega = 0.9)$. However, the floating-point operation is more complex than that of a fixed point. To overcome this problem, in Flex-SURF [55], the value of $\omega^2 = 0.875$ is employed. The same strategy is adopted to simplify the processing in [35], [48], [50], [69], [70]. Equation (4) can be replaced by a subtraction and a shift operation [69]

$$\begin{aligned}\det(H_{\text{approx}}) &= D_{xx} \times D_{xy} - 0.875(D_{xy})^2 \\ &= D_{xx} \times D_{xy} - ({D_{xy}}^2 - {D_{xy}}^2/8) \\ &= D_{xx} \times D_{xy} - {D_{xy}}^2 + ({D_{xy}}^2 \gg 3).\end{aligned} \tag{17}$$

Fig. 5 shows the number of operations which are employed to calculate the integral image using the SAS and the O-SURF. As observed in Fig. 5, the SAS highly requires a larger number of shift operations than O-SURF. However, the shift operation consumes one clock period in the FPGA architecture. Compared with the O-SURF, addition/subtraction (add-sub), multiplication and division are reduced by 4.44%, 13.33%, and 33.33%, respectively.

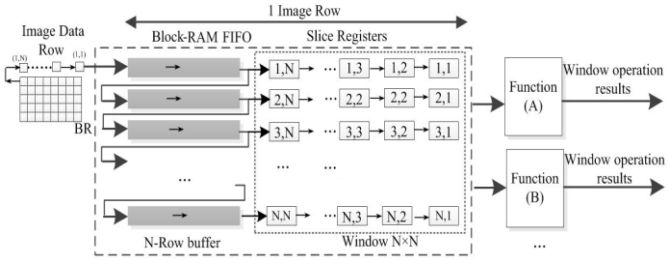

Fig. 6. Architecture of the sliding window.

D. Sliding Window

The sliding window technique has been indicated to be a good choice for parallel computer data [30], [49], [71], and [72], which includes four portions: the stream of an input image, a buffer, slice registers, and a function module. The described fabric is depicted in Fig. 6. The input data stream is buffered in a custom pipeline structure and organized in pixel rows. The buffer is implemented by a combination of block random-access memory first in, first out (Block-RAM FIFO) and slice registers (SRs). The SR sections enable access to all pixels in the respective pipeline elements compared with a window. With each incoming pixel, data are shifted by one pixel, which causes the window to virtually slide forward. Hence, while the input image is streamed into the pipeline, the window moves grid by grid from the top left to the bottom right in the integral image of the original image and reads the corresponding data from the slice register. A function module, which is determined for a certain window operation, interconnects to a subset of pixel registers to simultaneously fetch all data required for one calculation. With each pixel shifted into the structure, the function module calculates a new result and produces an output data stream [54]. The module has three types of sliding windows ($N \times N$) in the total system, where N is equal to 52, 5, and 35 in the Hessian response, non-max suppression and BRIEF descriptor, respectively.

E. Parallel Multiscale-Space for Hessian Determinant

The larger is the number of octaves, the larger is the number of hardware resources that will be consumed [73]. Avoiding consumes more resources in the FPGA, and the parallel multi-scale space architecture is simultaneously designed to compute the Hessian determinant. The recommended two octaves and six scales [24] are implemented to extract feature points, which corresponds to the scales {9, 15, 21, 27} and {15, 27, 39, 51}, respectively.

The interpolation step in the Hessian determinant of the O-SURF is computationally expensive because it requires the calculation of the first- and second-order derivatives of the Hessian matrices and their inverses. Two box filters with sizes 33 ($l = 11, L = 33, 2l - 1 = 21, (2l + 1)/2 = 7$), and 45 ($l = 15, L = 45, 2l - 1 = 29, (2l + 1)/2 = 8$) are added to calculate the Hessian determinants to create a scale-space with higher granularity and remove the interpolation step without sacrificing the accuracy [74]. A total of 8 scales {9, 15, 21, 27, 33, 39, 45, 51} are used to compute the Hessian determinants.

To efficiently calculate the Hessian determinant, multiple integral images need to be accessed by RBSP [61] in parallel to perform the separable convolution of the integral image with 24 box filters (8×3). The RBSP cores are visualized in Fig. 7. Fig. 7(a) lists the 10-line buffers for the box filter of size 15×15. Fig. 7(b) reveals that the memory foot of 32 ($8 + 8 + 16$) points of the box filters in Fig. 7(a), where Wx, Wy, and W represent the corresponding points in the D_{xx}, D_{yy}, and D_{xy} box filters. Fig. 7(c) shows the memory foot of 24 box filters with the size {9, 15, 21, 27, 33, 39, 45, 51}.

The 32 sampled points are a separable convolution with the 15×15 box filter as follows: L_0 thru L_{15} data are parallel input to the r-line buffer cores. Sixteen registers (R_0 thru R_{15}) are used to store one-line data, and 32 points can be selected to calculate the determinant of the Fast-Hessian. The box filter responses are given as follows:

$$\begin{aligned} D_{xx} &= (L_3\_R_{15} + L_{12}\_R_{10} - L_3\_R_{10} - L_{12}\_R_{15}) \\ &\quad - 2 \times (L_3\_R_{10} + L_{12}\_R_5 - L_3\_R_5 - L_{12}\_R_{10}) \\ &\quad + (L_3\_R_5 + L_{12}\_R_0 - L_3\_R_0 - L_{12}\_R_5) \\ &= (L_3\_R_{15} - L_3\_R_{10}) - (L_{12}\_R_{15} - L_{12}\_R_{10}) \\ &\quad - 2 \times (L_3\_R_{10} - L_3\_R_5) + 2 \times (L_{12}\_R_{10} - L_{12}\_R_5) \\ &\quad + (L_3\_R_5 - L_3\_R_0) - (L_{12}\_R_5 - L_{12}\_R_0) \end{aligned} \tag{18}$$

$$\begin{aligned} D_{yy} &= (L_0\_R_{12} + L_5\_R_3 - L_0\_R_3 - L_5\_R_{12}) \\ &\quad - 2 \times (L_5\_R_{12} + L_{10}\_R_3 - L_5\_R_3 - L_{10}\_R_{12}) \\ &\quad + (L_{10}\_R_{12} + L_{15}\_R_3 - L_{10}\_R_3 - L_{15}\_R_{12}) \\ &= (L_0\_R_{12} - L_0\_R_3) - (L_5\_R_{12} - L_5\_R_3) \\ &\quad - 2 \times (L_5\_R_{12} - L_5\_R_3) + 2 \times (L_{10}\_R_{12} - L_{10}\_R_3) \\ &\quad + (L_{10}\_R_{12} - L_{10}\_R_3) - (L_{15}\_R_{12} - L_{15}\_R_3) \end{aligned} \tag{19}$$

$$\begin{aligned} D_{xy} &= (L_2\_R_{13} - L_2\_R_2) - (L_7\_R_{13} - L_7\_R_8) \\ &\quad - (L_2\_R_6 - L_2\_R_2) + (L_7\_v_7 - L_7\_R_2) \\ &\quad - (L_8\_R_{13} - L_8\_R_8) + (L_{13}\_R_{13} - L_{13}\_R_8). \end{aligned} \tag{20}$$

Because the values of R_0 thru R_{15} are integral, then the multiply operation can be transformed into a shift operation. Equations (18)–(20) are decomposed into the vertical and horizontal convolution, which are implemented by addition/subtraction and a shift operation. The proposed separable convolution method is simpler than proposed by Čížek [75]. The proposed separable convolution utilization is only $O(n)$ and is not $O(n^2)$ for parallel implementation of the full sliding window [75].

In Fig. 7(c), the color dots are the selected positions for the convolution operation. The sliding window shifts from the left to the right and from the top to the bottom of the image. After scanning through the whole image, the Hessian determinants are simultaneously computed at the same period clock.

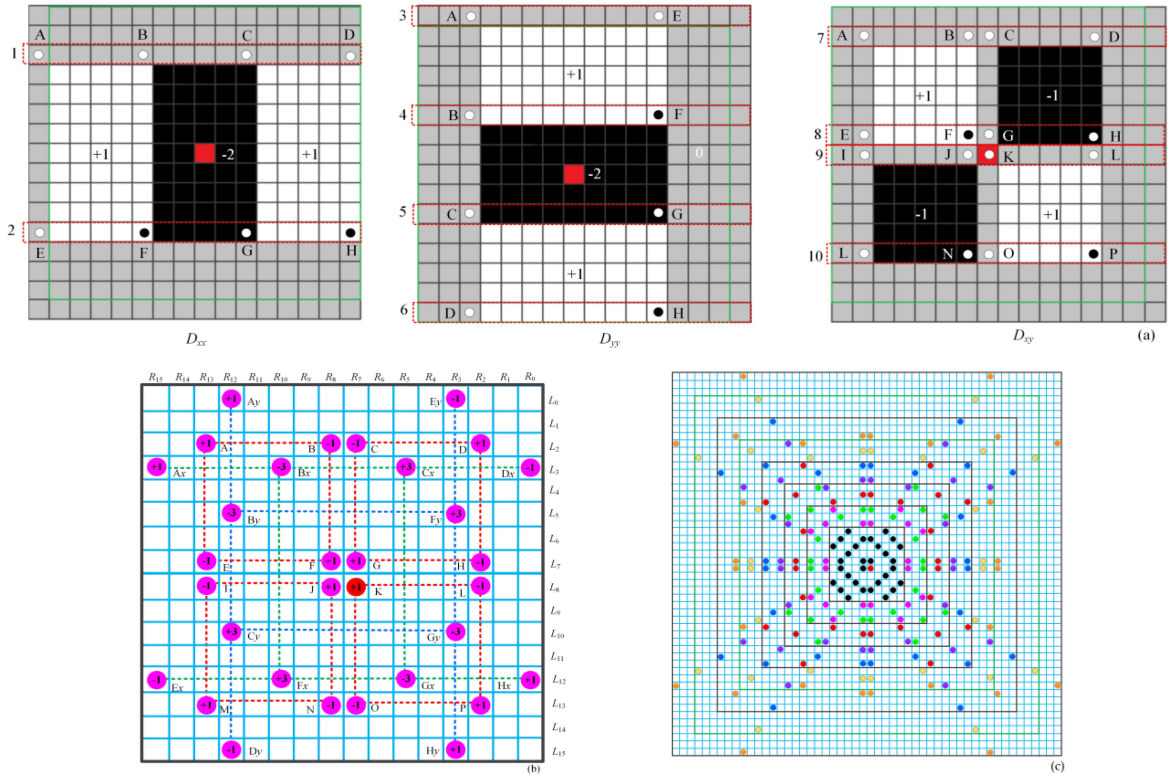

Fig. 7. Parallel multiscale space hessian detector. (a) Ten-line buffers for the box filter of size 15 × 15. (b) Sampled points of the box filter. (c) Pixel-access window is required for parallel convolution with the 24 box filters.

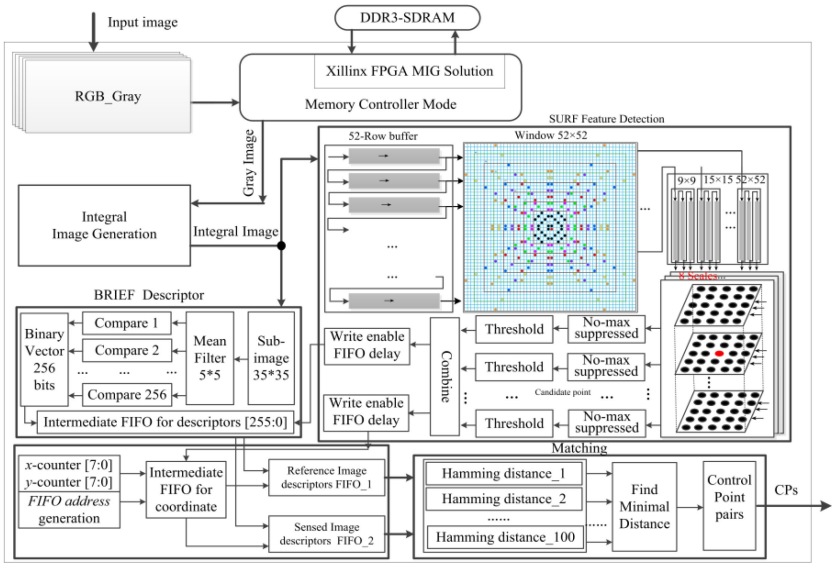

Fig. 8. Proposed hardware architecture.

IV. Hardware Implementation

A. Total System Architecture

Considering the memory-space, power, and real-time constraints of embedded systems, the FPGA is selected to ensure the system performing in real-time. The proposed total hardware architecture, which is shown in Fig. 8, contains memory controller module, integral image generation (IIG) module, SURF detector module, BRIEF descriptor module, and BRIEF matching module.

1) Memory Controller Module: To drive onboard, DDR3 and Xilinx IP memory interface generator (MIG) are chosen to create a logical connection with DDR3 [40].

2) IIG Module: An integral image is a novel method for improving the performance of the SURF detector. The WLR algorithm and four-row parallel method are adopted to optimize

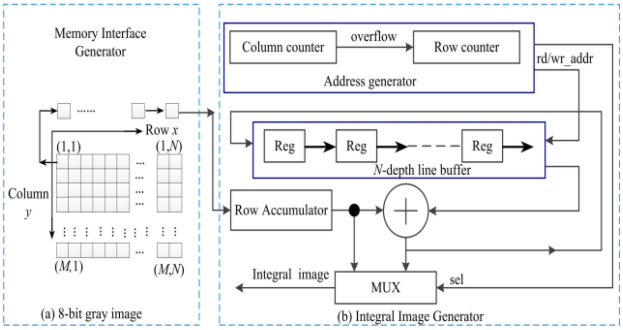

Fig. 9. Hardware architecture for the integral image generator. (a) Memory interface generator. (b) Integral image generator.

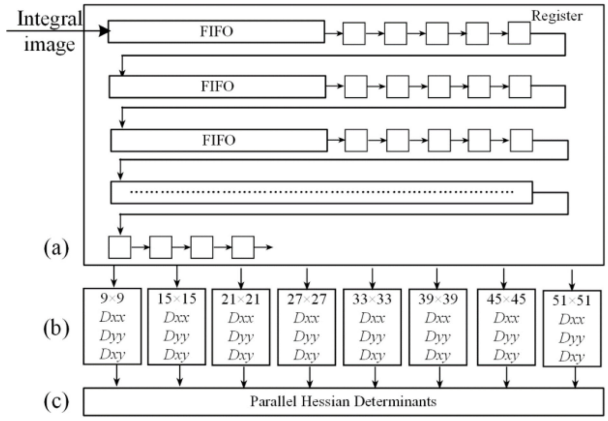

Fig. 10. Parallel multiscale-space for fast Hessian response. (a) Sliding window. (b) Parallel implementation D_{xx}, D_{yy}, and D_{xy}. (c) Parallel calculation Hessian determinant.

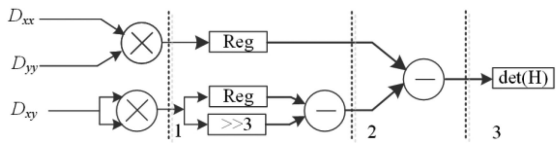

Fig. 11. Architecture for the pipelined Hessian determinant calculation.

the integral image. The IIG module transforms the integral image to the SURF feature detector module, memory controller module, and BRIEF descriptor module. Therefore, the IIG module is separated from the SURF feature detector module.

3) SURF Detector Module: The SURF detector extracts the local maxima Fast-Hessian determinant as candidate points on the multiscale and then locates the corresponding index and scale. The FPGA-based SURF feature detection architecture is divided into two submodules: Fast-Hessian response generation and the location of an interesting point. The location of interest point is divided into three steps: nonmaximal suppression, threshold, and interpolation. To solve these problems, a parallel architecture for the modified SURF algorithm is proposed. A sliding window buffer is used to store the shifted pixels of an integral image for each clock. The buffer is shared with the Hessian determinant. The SAS algorithm and parallel multi-scale space are used to implement the Hessian determinant. An additional 33 and 45 scales are used to replace the step of interpolation, without sacrificing the accuracy [74].

4) BRIEF Descriptor and Matching Module: A 256-bit BRIEF descriptor and matching require a low hardware cost. To reduce the complexity of the BRIEF descriptor, optimized parallel adder trees and parallel comparators are employed for the BRIEF descriptor and matching.

B. Integral Image Generator (IIG)

IIG module generates the integral image from the incoming 8-bit grayscale image via the MIG [40] to communicate with off-chip on-board DDR3 SDRAM [refer to Fig. 9(a)], which are stored by the row sequence.

The hardware architecture of the integral image [refer to Fig. 9(b)] consists of an address generator, row accumulator, multiplexer, and adder. The address generator module is designed to generate the read address (rd_addr) and write address (wr_addr) via the column and the row counter. The value of the row counter generates the selector (sel) signal of the multiplexer (MUX).

C. SURF Detector Implementation

1) Fast-Hessian Responses Implementation: After calculating the integral image in the IIG module, the integral image is sent to 52-line buffer [76] and 52 $\times$ 52 silence registers in the sliding window for the separable convolution. The detailed design of the multiscale-space Hessian detector is visualized in Fig. 10. This architecture includes the sliding window, parallel implementation second-order Gaussian derivatives, and parallel calculation Hessian determinants. To construct a sliding window for the Hessian response, an FIFO architecture includes 52-line buffers and 52 $\times$ 52 silence registers are used. The sliding window provides parallel access to the integral image that is required for the second-order Gaussian derivatives Dxx, Dyy, and Dxy using (18) thru (20). The 256 pixels (due to overlap, 220 pixels exist) enable access to the response of the 8 box filters.

After Dxx , Dyy, and Dxy are obtained, the Hessian response can be calculated in the three pipelines by using (17), as shown in Fig. 11. Eight Hessian determinants are concurrently calculated and then parallel output to the nonmaximal suppression module.

2) Nonmaximal Suppression Implementation: NMS module selects the local maximal Hessian determinant as a candidate point. A 5×5 sliding window is chosen in the NMS module, as shown in Fig. 12 [74]. Eight multiscales can be divided into six multiscale-spaces to concurrently obtain the local maximal of interest point. The 5×5 points of each matrix can be presented as $\mathrm{Top}\_m_{i,j}$, $\mathrm{Middle}\_m_{i,j}$, and $\mathrm{Bottom}\_m_{i,j}$ $(i = 1, 2, 3, 4, 5;\ j = 1, 2, 3, 4, 5)$. The center point $\mathrm{Middle}\_m_{3,3}$ is compared with its 74 neighbor points (24 neighbors on the same scale and 25 neighbors in the consecutive scales above and below) in parallel. If the result of the AND operation is true, the center point is regarded as a candidate point [50], as shown in Fig. 13. The candidate point is fed to the user-defined threshold module. Only the candidate point which is great than

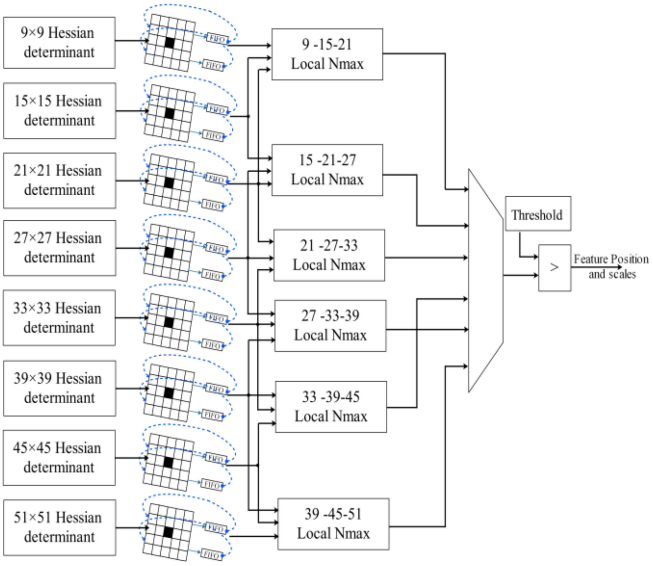

Fig. 12. Architecture of nonmaximal suppression module.

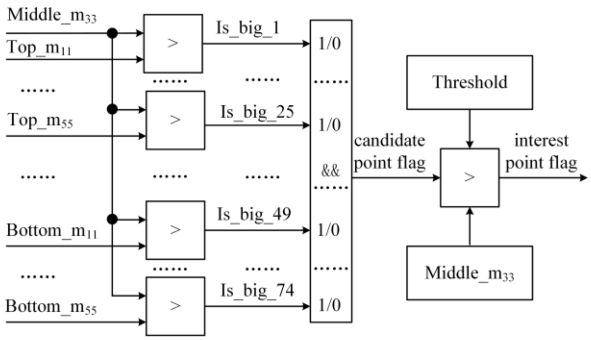

Fig. 13. Architecture of location maximum across the 74 neighbor points.

the threshold can be considered as a feature point, and the 1-bit interest point flag is set to *true* [72], [74].

The eight-scale Hessian determinants are parallelized to compute it synchronously. The search for a local maximum in the scale-space domain is also pipelined and parallelized using register FIFOs. Only the determinants that exceed a certain threshold are saved, while the other determinants, which are set to zeros points and not local maxima in the 3-D, scale-space domain, are suppressed (= 0). This method ensures that features do not overlap and are evenly spread over the input image [74]. Every local maximum is compared with a user-defined threshold value. The threshold controls the total sensitivity of the detector by fine-tuning the number of interest point that populate the image [58]. Considering that the features in the same octave will generate the same descriptor if they have the same coordinates but a different scale, we only store one of them if more than one feature simultaneously exists [77].

D. BRIEF Descriptor Implementation

The BRIEF-32 [26] algorithm is adopted to generate descriptors. As shown in Fig. 14, the BRIEF-32 structure includes two modules: image buffer and point pair comparator. In the

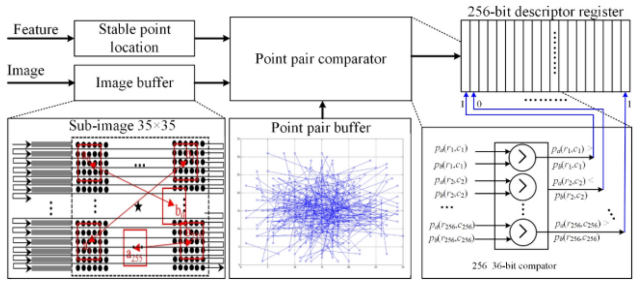

Fig. 14. Exemplary 256 patch-pairs for BRIEF descriptor.

image buffer module, a 35 × 35 subwindow was presented by Huang [50]. Calonder *et al*. [26] provided the time required for Gaussian smoothing, a simple box filter, and a box filter using the integral image, the latter was considerably faster. Furthermore, no matching performance loss occurred. In this article, a 5 × 5 box filter using integral images is used to smooth the original image. The point comparator module has 256-binary (32-byte) patch tests, which are related to blue lines that are sampled from an isotropic $(0, S^2/25)$ Gaussian distribution. A total of 256 patch-points $p(r_i, c_j)$ $(i = 1, \ldots, 256;\ j = 1, \ldots, 256)$ are parallel compared with the corresponding points using (8) in the same cycle, and the 1-bit result with a comparison of 1or 0 is stored in a 256-bit descriptor register in sequence.

E. BRIEF Matching Implementation

The Hamming distance is used to match the 256-bit descriptor in the BRIEF matching. The BRIEF descriptor is robust to illumination changes and small rotations [26], which renders it an excellent candidate point for the georeferencing. The BRIEF matching module consists of computing the Hamming distance module and finding the minimal Hamming distance module. The reference image descriptors are stored in FIFO1, and the sensed image descriptors are stored in FIFO2. To ensure a trade-off between the speed time and resource. The number of descriptor points determines the accuracy of the matching. However, the larger is the number of descriptor points, the larger is the number resources that are consumed. For example, 100 pair-descriptors are used to parallel implement the Hamming distance in Fig. 15 [50]. In the Hamming distance module [refer to Fig. 15(a)], the Hamming distance is computed using 256 XOR gates, and the results are stored in a 256-bit register. Improved parallel pipelined adder trees [77] are used to compute the number of "1s" in the 256-bit register. The 256-bit result of the XOR is divided into eight 32-bit registers, which are paralleled to calculate the Hamming distance by 5-level pipeline adder trees [refer to Fig. 15(c)]. However, 9-level pipeline adder trees were employed in [27].

The finding minimal Hamming distance module begins to work when the 100 Hamming distances are received. The minimum Hamming distance means the best matching. The corresponding matching feature point is defined as the GCP. The architecture of the finding minimal Hamming distance module is shown in Fig. 15(b), and an improved compactor module is used to compare the Hamming distance with seven levels of

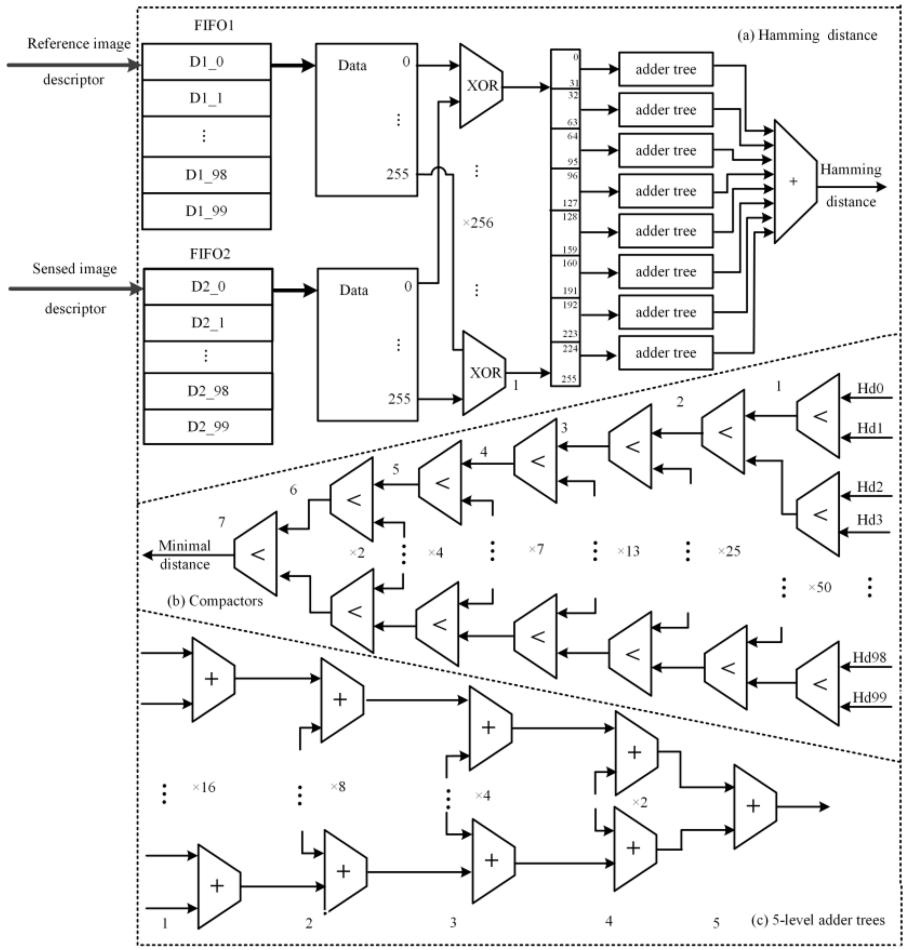

Fig. 15. Structure of matching coprocessor. (a) Computation Hamming distance. (b) Locating minimal Hamming distance. (c) 5-level adder trees.

pipeline. Three compactor modules are reduced compared with [50].

Based on the matching algorithms, the best matching feature points are the GCPs. The coordinates of GCPs are the scanning coordinates. However, in the georeferencing method, the geodetic coordinates are used in the projection transformation equation. Thus, the scanning coordinates must be transformed into geodetic coordinates [1].

V. Experiment

A. Hardware Environment and Dataset

The proposed system is implemented in a signal Xilinx XC7VX980T FPGA that has 612 000 logic cells, 1 224 000 Flip-Flops, 1500 kB Block RAM, and 3600 DSP slices. The development kit Vivado (14.2 version) is used to design the hardware of the system in Verilog HDL, and the simulation tool is Vivado simulator. The first data sets are obtained from [50] [see Fig. 17(a) and (b)], the second datasets are downloaded from the BIGMAP software [see Fig. 17(c) and (d)]. The image resolution of 512 $\times$ 512, and a 100 MHz working frequency are assumed. Additionally, the results that are generated by the implemented FPGA are compared with those of the OpenCV library, which is written by Chris Evans in the MATLAB. As expected, the results are identical.

B. Interest Point Analysis

The number of interest points is affected by some parameters such as octave, scale, resampling, size of the nonmaximal matrix, and the threshold [24], [50], [80]. Table III shows the number of interest point with the different thresholds. As seen from Table III, the variation in the threshold has a significant effect on the interest point distribution, which is consistent with [72], [74], and [80].

Fig. 16 shows the results of the different textures pair-points matching. In Fig. 16(a), the number of matching and mismatching points is 79 and 21, respectively. When the image pairs are covered with the high-rise building in Fig. 16(b), the matching points and mismatching points are 83 and 17, respectively. There are 700 matching points and 100 mismatching points in Fig. 16(c). And there are 90 matching points and 130 mismatching points in Fig. 16(d). The results indicated that the uniform distribution of the matching points and the matching rate are affected by the number of matching points and the textures of the object. Additionally, some errors occur in the matched point pairs; however, these points can be eliminated by

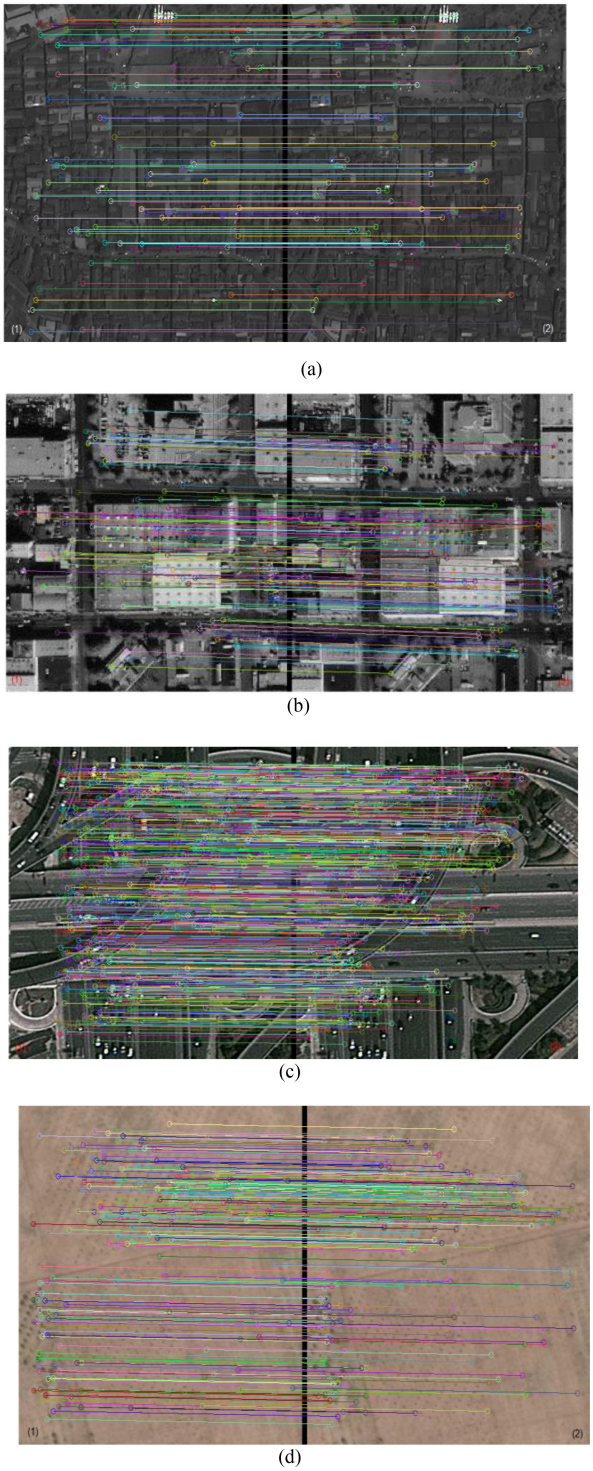

Fig. 16. Matched remote sensing image pairs. (a) Matched bungalows image pairs with the best 100 matching pair-points. (b) Matched high-rise building pairs with the best 100 matching pair-points. (c) Matched expressway pairs with the best 800 matching pair-points. (d) Matched bare soil pairs with the best 220 matching pair-points.

using robust fitting methods, such as RANSAC [15], [27], [42], [81] or a combined algorithm of slope-based rejection (SR) and correlation-coefficient-based rejection [48].

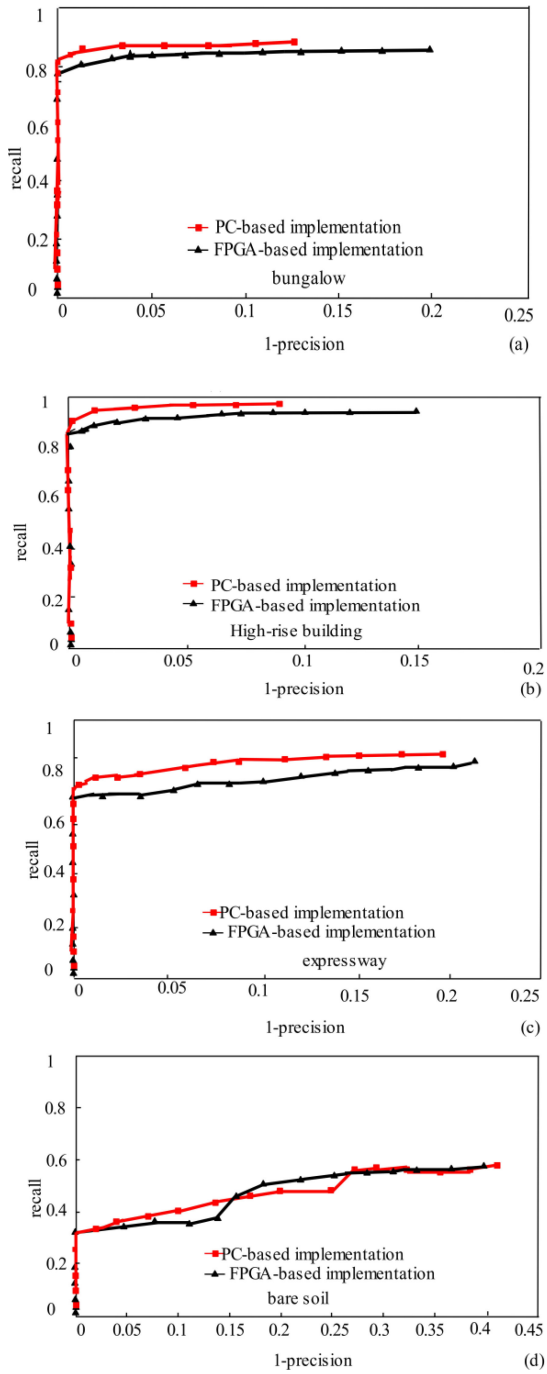

Fig. 17. Recall versus 1−precision of four RS image pairs. (a) Bungalows. (b) Rise-high building. (c) Expressway. (d) Bare soil.

TABLE III
NUMBER OF FEATURE POINTS WITH DIFFERENT THRESHOLDS (THS)

Image pairs	No TH	TH=1000	TH=8000	TH=15400
Bungalows 1/2	6666/6526	1467/1452	125/118	68/63
High-rise building 1/2	10110/10548	2309/2277	1169/1125	703/576
expressway 1/2	3898/3928	2931/2918	12701220	718/683
Bare soil 1/2	3349/3349	491/491	32/32	26/26

C. Accuracy Analysis

The *recall* versus *1−precision* curves [16] describe useful characteristics of a feature's performance, and are widely used as standard criterion, which is defined as follows:

$$\text{recall} = \frac{\#\text{correctmatches}}{\#\text{correspondences}} \tag{21}$$

$$1 - \text{precision} = \frac{\#\text{falsematches}}{\#\text{correctmatches} + \#\text{falsematches}} \tag{22}$$

where *recall* is the ratio of the number of correctly matched points relative over the number of corresponding matched points. $1-precision$ is defined as the ratio of the number of false matches to the total number of matches, which includes the false matches and the correct matches. The curves are generated below a threshold *t*, which determines whether two descriptors are matched, if *recall* is increasing and $1-precision$ is equal to 0, it means that the point pairs are all correctly matched without any mismatching; if *recall* is static and $1-precision$ is increasing, it means that the number of falsely point pairs is increasing, while the correctly point pairs remain unchanged [40].

SURF+BRIEF method has a better matching performance in different land coverages than that of SURF in the software (OpenCV2.4.9 + Microsoft Visual Studio 2015) [50]. In this article, the performance evaluation of the FPGA-based implementation is also conducted and compared with that of OpenSURF in MATLAB [60]. The relationship between the two images can be described by the Homography matrix [40]. The Homography matrices of the four image pairs are calculated in advance by MATLAB on a PC. The results are listed as (23)–(26) shown at the bottom of this page.

Inspired by [40], the number of detected points in each image is also defined as approximate 100 and there are 100 point-pairs output by the PC. The higher value of *recall* and the lower value of $1-precision$ mean better performance of matching. When $1-precision = 0$ (i.e., the number of falsely point pairs is 0), the value of *recall* is equal to "*m*," which means "$100 \times m$" correctly point pairs (the maximum number of point pairs is 100). While $1-precision$ stops at value "*n*," the number of falsely point pairs is "$100 \times n$". The curves of *recall* versus $1-precision$ of FPGA and PC implemented are shown in Fig. 17.

In the bungalow image pairs [see Fig. 17(a)], the curve of FPGA implemented is slightly lower than that of PC-based. In other words, the performance of the PC-based is slightly better than that of the FPGA's. In addition, when $1-precision = 0$, the *recall* of FPGA-based and PC-based is approximately 0.79 and 0.81, respectively. Finally, the $1-precision$ of FPGA-based and PC-based stops at approximately 0.20 and 0.14, respectively.

In the high-rise building image pairs [see Fig. 17(b)], the curve of PC-based is slightly higher than that of FPGA-based. When $1-precision = 0$, the *recall* of FPGA-based and PC-based is approximately 0.82 and 0.85, respectively. Finally, the $1-precision$ of FPGA-based and PC-based stops at approximately 0.15 and 0.10, respectively.

The expressway texture in Fig. 17(c), the PC-based curve is slightly higher than that of FPGA-based. When $1-precision = 0$, the *recall* of FPGA-based and PC-based is approximately 0.70 and 0.71, respectively. Finally, the $1-precision$ of FPGA-based and PC-based stops at approximately 0.23 and 0.20, respectively.

In the bare soil texture image pairs [see Fig. 17(d)], the FPGA-based curve is slightly higher than that of PC-based. When $1-precision = 0$, the *recall* of FPGA-based and PC-based is approximately 0.30 and 0.29, respectively. Finally, the $1-precision$ of FPGA-based and PC-based stops at approximately 0.41 and 0.44, respectively.

The combined SURF+BRIEF method was also presented in [50] and [72], and the same criterion *recall* versus *1−precision* curves is used to evaluate the matching performance. Experiments indicated that the combined SURF+BRIEF method has a similar performance with [40], [50], [72]. Hence, the final computation accuracy is acceptable.

As shown in Fig. 17, the performance of the FPGA-based implementation is slightly worse than that of the PC-based implementation. The reasons are listed as follows [50], [73].

1) Error is inevitable when the floating-point data are approximated with fixed-point data, and an error analysis is conducted [72]. To save logic resource, several fixed-point approximations are implemented to save resources and fit the entire architecture on a single Xilinx chip. The Hessian determinant is approximated using only shifting and the adding operation [35], [48], [50], [69], [70].

$$H_{\text{bungalows}} = \begin{bmatrix} 1.024682517497177 & 0.018725015152393 & -5.404435483561459 \\ 0.015206213894699 & 1.014273691140137 & -2.552171349203126 \\ 0.000024222830514 & 0.000023299587958 & 1.000000000000000 \end{bmatrix} \tag{23}$$

$$H_{\text{high-risebuilding}} = \begin{bmatrix} 1.099260397052538 & 0.002259608451860 & -7.957861186118295 \\ 0.090326101809412 & 0.955409181508876 & -4.421034816054379 \\ 0.000271661969942 & -0.000085315081328 & 1.000000000000000 \end{bmatrix} \tag{24}$$

$$H_{\text{expressway}} = \begin{bmatrix} 1.000297662301777 & 0.025695798331601 & -66.220856181085963 \\ -0.032698271963169 & 1.023565673743905 & 8.462326102920549 \\ -0.000064630077908 & 0.000065894730718 & 1.000000000000000 \end{bmatrix} \tag{25}$$

$$H_{\text{baresoil}} = \begin{bmatrix} 1.061661003420602 & -0.064774241169707 & -22.139327368389228 \\ 0.030845167584049 & 0.861666704860821 & 6.043495802072064 \\ 0.000065697594479 & -0.000382940095883 & 1.000000000000000 \end{bmatrix}. \tag{26}$$

TABLE IV
COMPARISON OF FPGA UTILIZATION

Method	FFs	LUTs	BRAMs(Kb)	DSPs
SURF+BRIEF [50]	122000	88462	1	0
OpenSURF [73]	/	52844	142	116
SURF [74]	42267	47255	128	136
SURF [33]	29165	37592	68	178
SURF [32]	108581	179559	80	244
SURF [78]	35804	37662	105	80
SIFT+BRIEF [27]	30002	21119	4672	80
SURF [53]	/	107873	1185	295
SURF+BRIEF [72]	29541	25463	116.5	160
Proposed	298864	267095	11	144

2) Only two octaves are implemented to detect feature points, which will inevitably cause performance degradation.

D. Performance of FPGA Analysis

1) FPGA Resources Utilization: A total of 298 864 (48.8%) slice LUTs, 267 095 (21.82%) slice FFs, 144 DSP (4%), and 11 Kb (0.73%) memories that are used to implement the proposed architecture. Two octaves and 8 scales [6 O-SURF scales and 2 extra scales (33 and 45)] are used to build the Hessian response. The SAS, sliding window, parallel multiscale-space, and parallel pipelined add-trees are used to optimize the SURF detector and BRIEF descriptor. Compared with [27], [53], and [72], the BRAMs resource is significantly reduced but the slice logical resource substantially increases compared with [27], [32], [53], [72], [74], and [78]. In [50] and [72], the same algorithm was proposed, but only six scales were used to compute the Hessian detection, and the interpolation step was omitted. This method consumes fewer resources but reduces the performance of subpixel precision [73]. Cai *et al.* [33] have designed a parallel and pipeline architecture for SURF algorithm. The SURF image feature point detecting system is implemented with hardware and software co-design. There are four modules in the FPGA architecture, which are integral image module, integral image buffer module, Hessian calculation module, and nonmaximal suppression module. However, we proposed an FPGA architecture including SURF detector, BRIEF descriptors, and BRIEF matching. Table IV lists the comparison results between the proposed method and the method that have been published based on the FPGA.

2) Speed Comparison: Speed is one of the most import factors of on-board detection and matching. In the proposed method, the run time of the integral image, SURF detection, BRIEF descriptor, and BRIEF matching is 2.62 μs, 2.6 ms, 2.48 μs, and 15.56 μs, respectively. The total run time is about 2.62 ms, which guarantees a frame rate of 380 fps with a resolution of 512 $\times$ 512 pixels at 100 MHz. An efficient image matching system based on an FPGA chip was proposed in [27] which can be implemented with the SIFT feature detector, BRIEF descriptor, and BRIEF matching for two 1280 $\times$ 720 images within 33 ms, at 30 fps. The hardware and software co-design system was proposed in [76], only the Fast-Hessian detector part of SURF has been chosen for hardware-only implementation. Generation of the SURF descriptor is handled entirely by software. The FPGA-SURF implementation achieves about 10 fps at HD (1024 $\times$ 768 pixels) resolution, which is a necessity for real-time operation. A model which combines the modified SURF detector and BRIEF descriptor was presented and implemented in FPGA [50], which supports a throughput of 304 fps for 512 $\times$ 512 pixels under a 100 MHz. An FPGA design that combines SURF detector and FREAK descriptor for real-time object detection [49] can process video frames with 800 $\times$ 600 pixels resolution at 60 fps. In [78], the hardware-accelerated version using the FPGA obtained execution times was 0.047 s for the image sequences with a resolution of 640 $\times$ 480. In [74], an optimized FPGA-based SURF extractor was proposed, which achieved 131.36 fps for a video stream of VGA resolution at 40.355 MHz. The throughput of [27], [43], [50], and [78] is 28, 10, 80, and 60 Mbps respectively. The proposed system can achieve 100 Mbps throughput. The comparisons of the proposed method with other investigators are shown in Tables V and VI.

TABLE V
COMPARISON OF THE PERFORMANCE OF FPS (SW: SOFTWARE; HW: HARDWARE)

Method	Clock (MHz)	Resolution	fps	SW/HW	Descriptor
SURF [33]	100	640×480	270	SW+HW	Yes
SIFT+BRIEF [27]	100	1280×720	30	HW	Yes
SURF [76]	/	1024×768	10	SW+HW	Yes
SURF+BRIEF [50]	100	512×512	304	HW	Yes
SURF+ FREAK [49]	100	800×600	60	HW	Yes
SURF [54]	66.7	640×480	50	HW	Yes
SURF+BRIEF [72]	100	640×480	162	HW	Yes
SURF [74]	40.355	640×480	131.36	HW	Yes
SURF [79]	200	640×480	56	HW	Yes
Proposed	100	512×512	380	HW	Yes

TABLE VI
COMPARISON OF THE PERFORMANCE OF TIME

Method	Clock (MHz)	Resolut-ion	SW /HW	Descr-iptor	Match-i ng	Speed time
SURF+ BRIEF [50]	100	512×512	HW(F PGA)	Yes	Yes	3.29ms
SIFT [43]	100	640×480	HW(F PGA)	No	No	31ms
SIFT +BRIEF [27]	359	1280×720	HW(F PGA)	Yes	Yes	33ms
SURF [78]	66.7	640×480	HW	Yes	No	47ms
Proposed	100	512×512	HW	Yes	Yes	2.62ms

VI. CONCLUSION

An optimized FPGA-based architecture for SURF and BRIEF algorithm is proposed to select the robust ground control points for georeferencing with RS image. The pipeline and parallel structure on-chip system includes a modified SURF detector and BRIEF descriptor. In the SURF detector module, the WLR method is used to reduce the word length of the integral image

without a loss of accuracy. MEPA method is used to compute the output of the FIFO in parallel. SAS method is adopted to simplify the response of the Hessian determinant, and sliding widow is used to compute the Hessian determinant in parallel. In the BRIEF descriptor module, only a 256-bit vector memory footprint is used to store a feature point descriptor. Enhanced adder trees are employed to reduce the complexity of the BRIEF matching step.

Four pairs of remotely sensed images with different textures are applied to evaluate the performance of FPGA-based implementation. The results of experiment indicated that the proposed architecture can achieve a real-time performance with 380 fps at 100 MHz. The proposed algorithm combines the accuracy of SURF detectors and the rapidity of the BRIEF descriptor to obtain a quick and accurate way of matching. Hence, the combined SURF-BRIEF system has the advantages of real-time, low-power, and high portability.

Acknowledgment

The authors would like to thank the reviewers and the associate editor for their insightful comments and suggestions.

References

[1] D. Liu *et al.*, "On-board georeferencing using FPGA-based optimized second-order polynomial equation," *Remote Sens.*, vol. 11, no. 2, Jan. 2019, Art. no. 124, doi: 10.3390/rs11020124.

[2] G. Zhou, R. Zhang, N. Liu, J. Huang, and X. Zhou, "On-board orthorectification for images based on an FPGA," *Remote Sens.*, vol. 9, no. 9, Aug. 2017, Art. no. 874, doi: 10.3390/rs9090874.

[3] C. C. Leng, H. Zhang, B. Li, G. R. Cai, Z. Pei, and L. He, "Local feature descriptor for image matching: A survey," *IEEE Access*, vol. 7, pp. 6424–6434, 2018.

[4] X. Deng, Y. Huang, S. Feng, and C. Wand, "Ground control point extraction algorithm for remote sensing image based on adaptive curvature threshold," in *Proc. IEEE Int. Geosci. Remote Sens. Symp.*, 2008, pp. 137–140.

[5] H. Zhang and Q. Hu, "Fast image matching based-on improved SURF algorithm," in *Proc. Int. Conf. Electron., Commun. Control*, Sep. 2011, pp. 1460–1463, doi: 10.1109/ICECC.2011.6066546

[6] F. Attneave, "Some informational aspects of visual perception," *Psychological Rev.*, vol. 61, no. 3, pp. 183–193, 1954, doi: 10.1037/h0054663.

[7] T. Tuytelaars and K. Mikolajczyk, "Local invariant feature detectors: A survey," *Found. Trends Comput. Graph. Vision*, vol. 3, no. 3, pp. 177–280, Jun. 2008, doi: 10.1561/0600000017.

[8] H. P. Moravec, "Rover visual obstacle avoidance," in *Proc. 7th Int. Joint Conf. Artif. Intell.*, Aug. 1981, pp. 785–790, doi:10.1007/s00427-011-0383-3.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vision Conf.*, vol. 15, no. 50, pp. 147–151, 1988, doi:10.5244/C.2.23.

[10] M. Y. I. Idris, N. B. A. Warif, N. M. Arof, A. W. A. Wahab, and Z. Razak, "Acceleration FPGA-SURF feature detection module by memory access reduction," *Malaysian J. Comput. Sci.*, vol. 32, no. 1, pp. 47–61, Jan. 2019, doi:10.22452/mjcs.vol32no1.4.

[11] S. M. Smith and J. M. Brady, "SUSAN—A new approach to low level image processing," *Int. J. Comput. Vision*, vol. 23, no. 1, pp. 45–78, May 1997, doi:10.1023/A:1007963824710.

[12] T. Lindeberg, "Scale-space theory: A basic tool for analyzing structures at different scales," *J. Appl. Statist.*, vol. 21, nos. 1/2, pp. 225–270, Jun. 1994, doi:10.1080/757582976.

[13] T. Lindeberg, "Scale-space: A framework for handling image structures at multiple scales," in *Proc. CERN School Comput.*, Sep. 1996, pp. 1–12.

[14] T. Lindeberg, "Feature detection with automatic scale selection," *Int. J. Comput. Vision*, vol. 30, no. 2, pp. 79–116, 1998, doi:10.1023/A:1008045108935.

[15] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vision*, vol. 60, no. 1, pp. 63–86, Oct. 2004, doi: 10.1023/b:visi.0000027790. 02288.f2.

[16] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors", *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 10, pp. 1615–1630, Feb. 2005.

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vision*, vol. 60, no. 2, pp. 91–110, Jan. 2004, doi:10.1023/B:VISI.0000029664.99615.94.

[18] C. Evans, "Notes on the OpenSURF library," Univ. Bristol, Bristol, U.K., Tech. Rep. CSTR-09-001, Jan. 2009.

[19] Y. Ke and R. Sukthankar, "PCA-SIFT: A more distinctive representation for local image descriptors," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, Jul. 2004, pp. 506–513.

[20] E. N. Mortensen, H. Deng, and L. Shapiro, "A SIFT descriptor with global context," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 2005, pp. 184–190.

[21] A. E. Abdel-Hakim and A. A. Farag, "CSIFT: A SIFT descriptor with color invariant characteristics," in *Proc. Comput. Vision Pattern Recognit.*, 2006, pp. 1978–1983, doi: 10.1109/CVPR.2006.95.

[22] J. M. Morel and G. Yu, "ASIFT: A new framework for fully affine invariant image comparison," *SIAM J. Imag. Sci.*, vol. 2 no. 2, pp. 438–469, Apr. 2009, doi: 10.1137/080732730.

[23] H. Bay, T. Tuytelaars, and L.V. Goo, "SURF: Speeded up robust features," *in Proc. Eur. Conf. Comput. Vision*, 2006, pp. 404–417, doi: 10.1007/11744023_32.

[24] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (SURF)," *Comput. Vision Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008, doi: 10.1016/j.cviu.2007. 09.014

[25] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proc. IEEE Int. Conf. Comput. Vision*, Nov. 2011, pp. 2548–2555.

[26] M. Calonder, V. Lepetit, M. Oezuysal, T. Trzcinski, C. Strecha, and P. Fua, "BRIEF: Computing a local binary descriptor very fast," *IEEE Trans. Pattern Analy. Mach. Intell.*, vol. 34, no. 7, pp. 1281–1298, Jul. 2012.

[27] J. H. Wang, S. Zhong, W. H. Xu, W. J. Zhang, and Z. G. Cao, "A FPGA-based architecture for real-time image matching," *Proc. SPIE*, vol. 8920, Oct. 2013, Art. no. 892003, doi: 10.1117/12.2031050.

[28] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE Int. Conf. Comput. Vision*, Nov. 2011, pp. 2564–2571.

[29] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking," in *Proc. 10th IEEE Int. Conf. Comput. Vision*, Oct. 2005, pp. 1508–1515.

[30] R. D. Lima, J. Martinez-Carranza, A. Morales-Reyes, and R. Cumplido, "Accelerating the construction of BRIEF descriptors using an FPGA-based architecture," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs*, 2015, pp. 1–6, doi: 10.1109/ReConFig.2015.7393285.

[31] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast retina keypoint," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2012. pp. 510–517.

[32] J. Zhao, S. Zhu, and X. Huang, "Real-time traffic sign detection using SURF features on FPGA," in *Proc. IEEE High Perform. Extreme Comput. Conf.*, 2013, pp. 1–6.

[33] W. Cai, Z. Xu, and Z. Li, "A high performance SURF image feature detecting system based on ZYNQ," *DEStech Trans. Comput. Sci. Eng.*, vol. 12, pp. 256–261, 2017, doi:10.12783/dtcse/cii2017/17261.

[34] G. Zhou and R. Li. "Accuracy evaluation of ground points from IKONOS high-resolution satellite imagery," *Photogrammetry Eng. Remote Sens.*, vol. 66, no. 9, pp. 1103–1112, Sep. 2000, doi:10.1016/S0924-2716(00)00020-4.

[35] S. S. Cai, L. B. Liu, S. Y. Yin, R. Y. Zhou, W. L. Zhang, and S. J. Wei, "Optimization of speeded-up robust feature algorithm for hardware implementation," *Sci. China Inf. Sci.*, vol. 57, no. 4, pp. 1–15, Jan. 2014, doi: 10.1007/s11432-013-4946-y.

[36] G. Hua, M. Brown, and S. Winder, "Discriminant embedding for local image descriptors," in *Proc. IEEE 11th Int. Conf. Comput. Vision*, 2007, pp. 1–8.

[37] R. Rani, A. P. Singh, and R, Kumar, "Impact of reduction in descriptor size on object detection and classification," *Multimedia Tools Appl.*, vol. 78, no. 7, pp. 8965–8979, Nov. 2018, doi: 10.1007/s11042-018-6911-7.

[38] M. Calonder, V. Lepetit, P. Fua, K. Konolige, J. Bowman, and P. Mihelich, "Compact signatures for high-speed interest point description and matching," in *Proc. IEEE 12th Int. Conf. Comput. Vision*, 2009, pp. 357–364.

[39] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011

[40] J. Huang, G. Zhou, X. Zhou, and R. Zhang, "A new FPGA architecture of FAST and BRIEF algorithm for on-board corner detection and matching," *Sensors*, vol. 18, no. 4, Mar. 2018, Art. no. 1014, doi: 10.3390/s18041014.

[41] P. Čížek, F. Jan, and D. Masri, "Low-latency image processing for vision-based navigation systems," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 781–786.
[42] T. Krajník, J. Šváb, S. Pedre, P. Čížek, and L. Přeučil, "FPGA-based module for SURF extraction," *Mach. Vision Appl.*, vol. 25, no. 3, pp. 787–800, Feb. 2014, doi: 10.1007/s00138-014 -0599-0.
[43] L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," in *Proc. Int. Conf. Field-Programmable Technol.*, 2009, pp. 30–37.
[44] D. Kim, M. Kim, K. Kim, M. Sung, and W. W. Ro, "Dynamic load balancing of parallel SURF with vertical partitioning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3358–3370, Dec. 2014, doi: 10.1109/TPDS.2014.2372763.
[45] S. H. Cheon, I. K. Eom, and Y. H. Moon, "Fast descriptor extraction method for a SURF-based interest point," *Electron. Lett.*, vol. 52, no. 4, pp. 274–275, Feb. 2016, doi:10.1049/el.2015.3055.
[46] J. Kim, E. Park, X. Cui, and H. Kim, "A fast feature extraction in object recognition using parallel processing on CPU and GPU," in *Proc. IEEE Int. Conf. Syst.*, 2009, pp. 3842–3847.
[47] M. Schaeferling, U. Hornung, and G. Kiefer, "Object recognition and pose estimation on embedded hardware: SURF-based system designs accelerated by FPGA logic," *Int. J. Reconfigurable Comput.*, vol. 2012, no. 6, pp. 1–16, Jan. 2012, doi: 10.1155/2012/368351.
[48] J. Huang, G. Zhou, D. Zhang, G. Zhang, R. Zhang, and O. Baysal, "An FPGA-based implementation of corner detection and matching with outlier rejection," *Int. J. Remote Sens.*, vol. 39, no. 23, pp. 8905–8933, Aug. 2018, doi: 10.1080/01431161.2018.1500728.
[49] J. Zhao, X. Huang, and Y. Massoud, "An efficient real-time FPGA implementation for object detection," in *Proc. IEEE 12th Int. New Circuits Syst. Conf.*, 2014, pp. 313–316.
[50] J. Huang and G. Zhou, "On-board detection and matching of feature points," *Remote Sens.*, vol. 9, no. 6, Apr. 2017, Art. no. 601, doi: 10.3390/rs9060601.
[51] A. Li, W. Jiang, W. Yuan, D. Dai, S. Zhang, and Z. Wei, "An improved FAST + SURF fast matching algorithm," *Procedia Comput. Sci.*, vol. 107, pp. 306–312, Dec. 2017, doi: 10.1016/j.procs. 2017.03.110.
[52] T. Kasezawa, H. Tanaka, and H. Ito, "Integral image word length reduction using overlapping rectangular regions," in *Proc. IEEE Int. Conf. Ind. Technol.*, 2016, pp. 763–768.
[53] X. Fan, C. Wu, W. Cao, X. Zhou, S. Wang, and L. Wang, "Implementation of high performance hardware architecture of OpenSURF algorithm on FPGA," in *Proc. Int. Conf. Field-Programmable Technol.*, 2013, pp. 152–159.
[54] M. Pohl, M. Schaeferling, and G. Kiefer, "An efficient FPGA-based hardware framework for natural feature extraction and related computer vision tasks," in *Proc. 24th Int. Conf. Field Programmable Logic Appl*, 2014, pp. 1–8. doi: 10.1109/FPL.2014.6927463.
[55] M. Schaeferling and G. Kiefer, "Flex-SURF: A flexible architecture for FPGA-based robust feature extraction for optical tracking systems," in *Proc. IEEE Int. Conf. Reconfigurable Comput. FPGAs*, 2010, pp. 458–463. doi: 10.1109/ReConFig.2010.11.
[56] T. B. Terriberry, L. M. French, and J. Helmsen, "GPU accelerating speeded-up robust features," in *Proc. 4th Int. Symp. 3D Data Process., Visualisation Transmiss.*, Jun. 2008, vol. 8, pp. 355–362, doi: 10.1016/j.cviu.2007.09.014.
[57] E. Oyallon and J. Rabin, "An analysis of the SURF method," *Image Process. Online*, vol. 5, pp. 176–218, Jul. 2015, doi:10.5201/ipol. 2015.69.
[58] N. Zhang, "Computing optimised parallel speeded-up robust features (P-SURF) on multi-core processors," *Int. J. Parallel Program.*, vol. 38, no. 2, pp. 138–158, Apr. 2010, doi: 10.1007/s10766-009-0122-9.
[59] D. Gossow, P. Decker, and D. Paulus, "An evaluation of open source SURF implementations," in *Robot Soccer World Cup*. Berlin, Germany: Springer, 2010, pp. 169–179, [Online]. Available: https://link.springer.com/chapter/10.1007%2F978-3-642-20217-9_15#aboutcontent
[60] C. Evans, [Online]. Available: https://ww2.mathworks.cn/ matlabcentral/fileexchange/28300-opensurf-including-image-warp
[61] P. H. Hsu and S. Y. Chien, "Reconfigurable cache memory architecture for integral image and integral histogram applications," in *Proc. IEEE Workshop Signal Process. Syst.*, 2011, pp. 151–156.
[62] H. J. W. Belt, "Word length reduction for the integral image," in *Proc. 15th IEEE Int. Conf. Image Process.*, 2008, pp. 805–808.
[63] S. H. Lee and Y. J. Jeong, "A new integral image structure for memory size reduction," *IEICE Trans. Inf. Syst.*, vol. 97, no. 4, pp. 98–1000, Apr. 2014, doi: 10.1587/transinf.E97.D.998.
[64] S. Ehsan and K. D. McDonald-Maier, "Exploring integral image word length reduction techniques for SURF detector," in *Proc. IEEE 2nd Int. Conf. Comput. Elect. Eng.*, 2009, pp. 635–639.
[65] S. Ehsan, A. F. Clark, and K. D. McDonald-Maier, "Novel hardware algorithms for row-parallel integral image calculation," in *Proc. Digital Image Comput., Techn. Appl.*, 2009, pp. 61–65.
[66] S. Ehsan, A. F. Clark, N. Rehman, and K. McDonald-Maier, "Integral images: Efficient algorithms for their computation and storage in resource-constrained embedded vision systems," *Sensors*, vol. 15, no. 7, pp. 16804–16830, 2015.
[67] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 2001, pp. 511–518.
[68] B. Kisacanin, "Integral image optimizations for embedded vision applications," in *Proc. IEEE Southwest Symp. Image Anal. Interpretation*, 2008, pp. 181–184.
[69] T. Sledevič and A. Serackis, "SURF algorithm implementation on FPGA," in *Proc. 2012 13th Biennial Baltic Electron. Conf.*, 2012, pp. 291–294.
[70] M. Schaeferling and G. Kiefer, "Object recognition on a chip: A complete SURF-based system on a single FPGA," in *Proc. Int. Conf. Reconfigurable Comput. FPGAs*, 2011, pp. 49–54.
[71] M. Schmidt, M. Reichenbach, and D. Fey, "Generic VHDL template for 2D stencil code applications on FPGAs," in *Proc. IEEE 15th Int. Symp. Object/Compon. /Service-Oriented Real-Time Distrib. Comput. Workshops*, 2012, pp. 180–187.
[72] Q. Ni, F. Wang, Z. Zhao, and P. Gao, "FPGA-based Binocular image feature extraction and matching system," in *Proc. 4th Int. Conf. Multimedia Syst. Sig. Process.*, May 2019, pp. 182–187.
[73] C. Chen, H. Yong, S. Zhong, and L. Yan, "A real-time FPGA-based architecture for OpenSURF," *Proc. SPIE*, vol. 9813, Dec. 2015, Art. no. 98130k, doi: 10.1117/12.2205633.
[74] C. Wilson *et al.*, "A power-efficient real-time architecture for SURF feature extraction," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs*, 2014, pp. 1–8.
[75] P. Čížek and J. Faigl, "Real-time FPGA-based detection of speeded-up robust features using separable convolution," *IEEE Trans. Ind. Informat.*, vol. 14, no. 3, pp. 1155–1163, Mar. 2017.
[76] J. Svab, T. Krajnik, J. Faigl, and L. Preucil, "FPGA based speeded up robust features," in *Proc. IEEE Int. Conf. Technol. Practical Robot Appl.*, 2009, pp. 35–41.
[77] M. Fularz, M. Kraft, A. Schmidt, and A. Kasiński, "A high-performance FPGA-based image feature detector and matcher based on the FAST and BRIEF algorithms," *Int. J. Adv. Robotic Syst.*, vol. 12, no. 10, Oct. 2015, Art. no. 141, doi: 10.5772/61434.
[78] W. Chen *et al.*, "FPGA-based parallel implementation of SURF algorithm," in *Proc. IEEE 22nd Int. Conf. Parallel Distrib. Syst.*, 2016. pp. 308–315.
[79] D. Bouris, A. Nikitakis, and I. Papaefstathiou, "Fast and efficient FPGA-based feature detection employing the SURF algorithm," in *Proc. 18th IEEE Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, 2010, pp. 3–10.
[80] S. Ehsan, N. Kanwal, A. F. Clark, and K. D. McDonald-Maier, "An algorithm for the contextual adaption of surf octave selection with good matching performance: Best octaves," *IEEE Trans. Image Process.*, vol. 21, no. 1, pp. 297–304, Jan. 2012.
[81] J. Li, T. Xu, and K. Zhang, "Real-time feature-based video stabilization on FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 27, no. 4, pp. 907–919, Jan. 2016.

Dequan Liu received the B.S. degree in electronic science and technology from Northwest Normal University, Lanzhou, China, in 2004, and the M.E. degree in signal and information processing from Xidian University, Xi'an, China, in 2013. He is currently working toward the Ph.D. degree in circuits and systems in the School of Microelectronics, Tianjin University, Tianjin, China.

Guoqing Zhou (Senior Member, IEEE) received the Ph.D. degree from Wuhan University, Wuhan, China, in 1994.

He was a Visiting Scholar with the Department of Computer Science and Technology, Tsinghua University, Beijing, China, and a Post-doctoral Researcher with the Institute of Information Science, Beijing Jiaotong University, Beijing, China. He continued his research as an Alexander von Humboldt Fellow with the Technical University of Berlin, Berlin, Germany, from 1996 to 1998, and was a Post-doctoral Researcher with The Ohio State University, Columbus, OH, USA, from 1998 to 2000. He was Assistant Professor, Associate Professor, and Full Professor with Old Dominion University, Norfolk, VA, USA, in 2000, 2005, and 2010, respectively. He has authored five books and more than 380 refereed papers.

Dianjun Zhang received the B.S. and M.S. degrees in geographical information system from Ludong and Beijing Forestry University, Beijing, China, in 2008 and 2011, respectively, and the Ph.D. degree from the Institute of Geographic Sciences and Natural Resources Research, Chinese Academy of Sciences, Beijing, in 2015.

His research interests include thermal infrared remote sensing and retrieval of surface parameters from satellite data.

Xiang Zhou received the M.S. degree in the detection technology and automatic equipment from the Guilin University of Technology, Guilin, China, in 2014.

He is currently with the School of Microelectronics, Tianjin University, Tianjin, China. His research interests include LiDAR imaging.

Chenyang Li was born in Hebei, China, in 1990. He is currently working toward the Ph.D. degree in the School of Marine Science and Technology, Tianjin University, Tianjin, China. His research interests include airborne Lidar waveform data processing, the character of Blue-Green laser transmission through rough sea surface, and the methods and applications of airborne Lidar bathymetry.