Journal of Real-Time Image Processing (2021) 18:705–722
https://doi.org/10.1007/s11554-020-01012-8

ORIGINAL RESEARCH PAPER

A high-speed feature matching method of high-resolution aerial images

Zhiyong Peng[1,2] · Jun Wu[1] · Yongjun Zhang[3] · Xianhua Lin[1]

Received: 22 July 2019 / Accepted: 21 August 2020 / Published online: 2 September 2020

Abstract
This paper presents a novel corner detection and scale estimation algorithm for image feature description and matching. Inspired by Adaboost's weak classifier, a series of sub-detectors is elaborately designed to obtain reliable corner pixels. The new corner detection algorithm is more robust than the FAST and HARRIS algorithm, and it is especially suitable for the implementation in FPGA. The new scale estimation method can be directly implemented in the original image without building Gaussian pyramid and searching max response value in each level, which not only increase computational efficiency but also greatly reduces memory requirement. Based on the proposed algorithm, a CPU-FPGA cooperative parallel processing architecture is presented. The architecture overcomes the memory space limitation of FPGA and achieves high-speed feature matching for massive high-resolution aerial images. The speed of the CPU-FPGA cooperative process is hundred times faster than SIFT algorithm running on CPU, and dozens of times faster than SIFT running in CPU + GPU system.

Keywords Image matching · Corner detection · Scale estimation · Parallel computing · CPU-FPGA cooperative processing

1 Introduction

Nowadays, it is a big challenge to effectively and efficiently handle massive high-resolution images in the field of photogrammetric and remote sensing. Massive aerial images can be captured from ongoing satellite platforms or various Unmanned Aerial Vehicles (UAVs). The traditional ways of feature matching undoubtedly are not suitable to process massive aerial images, because they cannot meet the requirements of the image data throughput, time cost and power consumption. Image matching is the fundamental step of aerial images processing, such as pose estimation, image geo-registration, object detection and tracking, geometrical calibration, 3D reconstruction and so on. So high-speed (near) real-time image matching of massive high-resolution images is the key in emergency remote sensing applications [1].

The existing image matching methods can be roughly classified into two categories [2]: Local Matching and Global Matching. So far, the local matching methods are represented by SIFT [3], ASIFT [4], Hessain affine [5], etc., and they have been able to obtain robust and reliable matching results under obvious illumination, geometric transformation (rotation, scaling, affine [6]. The global matching methods are represented by SGM [7], Graph Cut [8], Belief Propagation [9], etc., and they can output high-quality dense disparity images even in difficult image areas, e.g. texture deficiency, repetition and gray discontinuity boundaries. However, along with the improvement of matching robustness and disparity quality, the computational complexity and time cost of matching algorithms increase dramatically [10]. Because of high complexity, these algorithms cannot meet the demand for high-speed processing on CPU. For this reason, parallel processing is an effective strategy to improve computational efficiency for meeting the real-time processing requirements of massive image data.

Basically, the classical CPU processors based on the Von Neumann architecture are not suitable for large amounts of duplicated data calculation, but the acceleration devices GPU and FPGA are suitable because of powerful parallel

✉ Jun Wu
wujun93161@163.com

1 School of Electrical Engineering and Automation, Guilin University of Electronic Technology, Guilin, China

2 Guangxi College's Emphasis Laboratory Foster Base for Optoelectronics Information, Guilin, China

3 School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China

computing capacity [11]. Due to the limit of the Single Instruction Multiple Data calculation model and corresponding Pipeline Stage division, the algorithm parallelization requires all calculation units to have the same pace in GPU when different data is processed; so many time-delays are inevitable in the calculation. The algorithm parallelization in FPGA is with extremely low latency [30], and there is a lower power consumption and competitive market price than GPU. Because it is quite different between Von Neumann architecture processor and Harvard architecture processor, the complex of FPGA development is closely related to the parallelism, computational complexity and storage requirements of implemented models [31]. At present, the implementations of high-performance algorithm in FPGA, e.g., SIFT and SURF, are limited, which is just suitable for low-resolution images, decreased accuracy and the number of feature points. It can be implemented that using corner to do feature-point matching in the FPGA. But if the image has an obvious change of luminance or shooting angle, detected corners usually cannot be tracked repeatedly [32]. A reliable corner detection and scale estimation algorithm is proposed in this paper and further implemented in a way of "CPU + FPGA" cooperative process, which can meet the requirement of the high-speed feature detection and matching for aerial images with large pixel arrays.

To summarize, we make the following contributions. First, we proposed a robust parallel corner detection algorithm which is suitable for FPGA implementation. Second, a direct scale estimation algorithm is proposed in the original image, which avoids to build Gaussian pyramid. Third, we implement the proposed algorithm in the CPU + FPGA cooperative processing architecture for high- resolution aerial images.

2 Related work

Feature matching can be roughly divided into two types of algorithm: matching and feature point-based matching. Classical region-based feature matching algorithms have Medial features (MFD), edge-based region detector (EBR) and maximally stable extremal regions (MSER). Medial features (MFD) [12] are detected based on shape. In MFD, although more information is used, gradient strength is sensitive to lighting and scale variations. The EBR starts from corner points and exploits nearby edges by measuring photometric quantities across them. EBR is suitable for man-made structures like buildings, but not for generic matching, as shown in [5]. The MSER of [13] is a similar watershed segmentation and matching algorithm, it can detect regions of stable intensity and, therefore, avoids common problems of gradient-based methods like localization accuracy and noise. MSER can only detect bright or dark extremal regions, and it cannot detect gray regions that are adjacent to both brighter and darker ones. Classical feature matching algorithms based on feature-point have high-performance local matching methods SIFT, SURF, and corner detection algorithm Hessain-affine, Harris-affine, etc. The proposed algorithm belongs to the feature matching algorithm based on feature-point. In the practice, for the parallel implementation of high-speed feature matching, there are two parallel methods which can be used to accelerate algorithm, by GPU and FPGA. Some researchers have got many valuable research achievements in the past.

Acharya et al. [14] presented a parallel implementation of SIFT on a GPU by combined a kernel optimization, and obtained a speed of around 55 fps for 640 × 480 images. At the same time, there are some researchers have also implemented SIFT and SURF algorithm on GPU, e.g., [15–17], but compared to CPU implementation, they all are no more than 20 times the acceleration. Because of necessary many time-delays in the calculation and the limit of the pipeline number, the acceleration performance is limited using GPU.

The implementation of feature-point matching on FPGA has two types of algorithm: classical high-performance feature-point matching algorithms (SIFT, SURF) and corner pixel matching algorithm (Harris, FAST, and so on). The implementation of classical high-performance feature-point matching algorithms on FPGA: Huang et al. [18] and Zhong et al. [19] partly implement the SIFT algorithm in FPGA board, and feature detection and matching can be finished in about 80 ms for a frame video image. For the purpose of real-time traffic sign detection, Zhao et al. [20] implement SURF in FPGA board and handle with 800*600 resolution video streams with the speed of 60 fps. [21] detected and matched the feature points by combined SURF detector and Binary Robust Independent Elementary Features (BRIEF) descriptor by FPGA, and the speed is about 27 times of CPU-based implementation. [22] has present multiple hardware implementations of the semi-global matching (SGM) algorithm on the FPGA by simplified SGM, and achieved the real-time performance of 30 frames/s with 128 matching pairs per frame for the image resolution of 640_480. When the high-performance feature-point matching algorithm is implemented on FPGA, because of high computational complexity, the algorithm usually need to be simplified, the number of feature-point is limited for several hundreds and the resolution of image usually is less than one million pixels.

The corner detector, e.g., FAST [27] and Harris [23] are widely used for image matching for the high detection efficiency. [24] implemented the Harris corner detection algorithm on the FPGA using a sliding window, which can avoid the demand of large buffer memory. [25] proposed an efficient hardware approach that offloads the repetitive feature extraction procedures into logic gates, and the result shown that the speed and accuracy of the feature detector are good enough for many real-world applications. [26] implemented FAST algorithm on the FPGA, which can real-timely detects corner. FAST has been incorporated into ORB [28] and further implemented with FPGA [29], which offer higher frame rates than CPU. At now, corner algorithm can be implemented on the FPGA, and finish high-speed feature matching. But the corner detection algorithm is not robust, if the image has an obvious change of luminance or shooting angle, detected corners usually can't be track repeatedly.

3 Corner detection and matching algorithm

The aim of this paper is to develop one high-speed corner detection and matching algorithm, which can handle with massive aerial images with large pixel arrays using FPGA hardware acceleration. To this end, proposed corner detection and matching algorithm in this paper is not only robust but also suitable for FPGA implementation. Commonly, there are two big challenges for handling image in FPGA board: At first, because the on-board Static Random-Access Memory (SRAM) of FPGA is limited, usually 3 ~ 4 M Byte, image pixels have to be loaded into the FPGA locally and dynamically so as to occupy less buffer space, meanwhile, the image operation in the big size window or variable-size window should be avoided. The second, the implementation of floating-point arithmetic, e.g., division, square root, logarithm and complex arithmetic operation, is very difficult and inefficiency on FPGA. In this paper, corner detection and matching algorithm are proposed with a concise arithmetic operation, as well as the structure of algorithm is suitable for timing convergence and clock optimization management on FPGA.

3.1 Corner detection

Due to high information entropy in image, corner points are widely used in the field of computer vision. Several definitions of the corner are proposed as [33]: the pixel with the biggest gradient of gray in the local area; the cross-point of two or more edge lines; the pixel whose value and direction of gradient all change rapidly, etc. Based on those definitions and inspired by the Adaboost weak classifier [34], a series of sub-detector is elaborately designed to get reliable corner pixels in the paper.

3.1.1 Sub-detector C_1

The purpose of sub-detector C_1 is to remove background pixels in the image, according to the gradient function $\nabla(x,y) = \left|\nabla_x\right| + \left|\nabla_y\right|$. If a pixel's gradient value is smaller than given threshold $D_{1,}$ this pixel will be discarded as a background pixel. The D_1 is defined as: $D_1 = K_1 * \overline{I_{xy,}}$ where $\overline{I_{xy}}$ is the average gray in the local area and k_1 is an empirical constant ranged between 1.0 and 1.5.

3.1.2 Sub-detector C_2

The aim of this sub-detector is to obtain candidate corners by means of the symmetry of gray distribution and non-maximum suppression of gray change on both sides of corners. As seen in Fig. 1, a polar coordinate system is set up for each candidate corner, of which pixel P is the origin point and the direction of the gradient δ is set as the direction of polar coordinate. Two points $[p_4(x_a, y_a)$ and $p'_4(x_b, y_b)]$ are got on the perpendicular line ($\theta = 0$ and π) of gradient direction. The ruler of symmetry is set to require that the local gray value of $p_4(x_a, y_a)$ and $p'_4(x_b, y_b)$ is close to each other $\left|\overline{I_w(x_a, y_a)} - \overline{I_w(x_b, y_b)}\right| < D_2$. The D_2 is a given threshold and defined as $D_2 = K_2 * \overline{I_{xy}}$, where $\overline{I_{xy}}$ is the average gray in the local area, and k_2 is an empirical constant ranged between 0.05 and 0.2. Finally, the candidate corner is got by means of non-maximum suppression on the gray change of the pixel under the symmetry of gray distribution in the local area (5*5 window). The gray value change function $\overline{\nabla_m(x, y)}$ is expressed as formula (1)

$$\overline{\nabla_m(x, y)} = \frac{\left|\overline{I_w(x, y)} - \overline{I_w(x_a, y_a)}\right| + \left|\overline{I_w(x, y)} - \overline{I_w(x_b, y_b)}\right|}{2} \quad (1)$$

3.1.3 Sub-detector C_3

The purpose of sub-detector C_3 is to remove non-corner edge pixels using the ruler of gradient direction change. The ruler

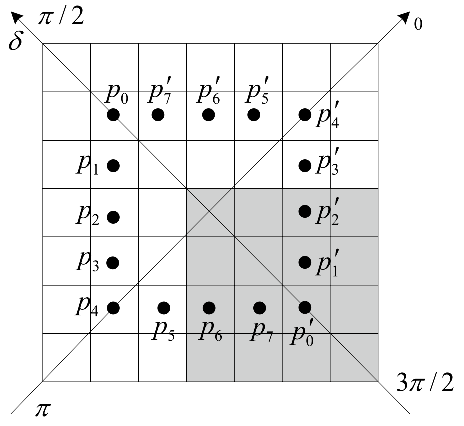


Fig. 1 The polar coordinate of the candidate corner and distribution of detection pixels

of gradient direction change is that the corner is asked to have a different gradient direction between center pixel and adjacent pixels on the vertical line of gradient direction. As shown in Fig. 1, two points P_4 and P_4' are got in the direction of $\theta = 0$ and π. The gradient direction $\nabla\theta_4$ and $\nabla\theta_4'$ of P_4 and P_4' are calculated, respectively. If $\nabla\theta_4$ and $\nabla\theta_4'$ meet the condition $C_3 : \left|\nabla\theta_4\right| > D_3 \cap \left|\nabla\theta_4'\right| > D_3$, the center pixel (x_0, y_0) is a candidate corner, or it is discarded. Usually, the angle range of a corner ranges from 0^o to 160^o, and the value of threshold D_3 set to 20 which is a good choice in practice. Because the sub-detector C_3 excludes the non-corner pixels by gradient direction change instead of gradient value, it can effectively avoid the influence of different angle light changes on the detection results.

3.1.4 Sub-detector C_4

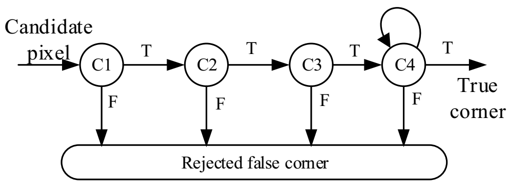
The purpose of sub-detector C_4 is to remove random non-corner noise pixels based on the shape of corner. The pixels in the neighborhood of corner candidate are divided into two connected sets. As shown in Fig. 1, the 16 pixels are determined by computing its distance to the center point (x_0, y_0). The shape ruler of corner is that there is a connected set of “1” around of P_0 and a connected set of “0” around of P_0'.

Real corners can be got by synthesizing the output of 4 sub-detectors as C = C1 ∩ C2 ∩ C3 ∩ C4. Because the calculation of pixel gradient and average gray level in C_1 and non-maximum suppression in C_2 only involve small neighbourhood (5*5 or 3*3 windows), the division calculation in C_2 can be implemented by shift operation, the calculation and comparison of the angle between gradient direction in C_3 can be accomplished directly by coordinate vector subtraction, proposed corner detector is easy to be implemented by FPGA. Considering the pixel gradient and neighbourhood average gray calculation (pre-processing), the corner detection process in this paper can be roughly designed as two ways: serial and parallel, as seen in Fig. 2. The former has the advantage of avoiding repeated processing of non-corner pixels. The latter has the advantage of large parallel granularity and is adopted for the realization of the functions on FPGA.

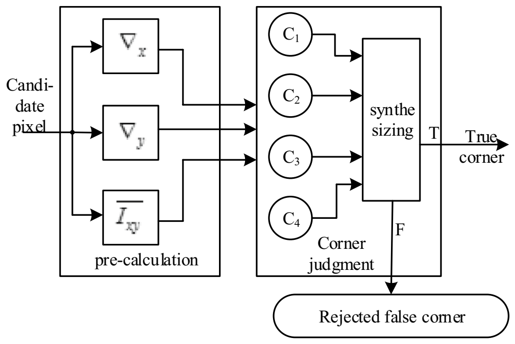
3.2 Corner scale estimation

In the classical feature-point matching algorithm, the DoG convolution function is commonly used to get feature scale. There are two problems which are often involved: large RAM requirement when building Gaussian pyramid; and massive calculation when doing convolution on each level of pyramid. In this paper, we propose to directly estimate corner scale in original image without building Gaussian pyramid and searching max response value in each level of pyramid. This is achieved by replacing the DoG convolution function with the square wave function, as seen in Fig. 3. Figure 3a shows the DoG convolution functions (1D) at 5 continuous scales (k = 1.26, σ = 1.6) in SIFT Gaussian pyramid Octave 1; and Fig. 3b is the approximation of Fig. 3a using square wave function.

It can be seen from Fig. 3 that the DoG convolution functions of different scale is corresponding to the square wave functions with different “width” and “height”. The difference of “width” means that the size of convolution template is different; and the difference of “height” means that the convolution coefficient is different. Considering two square wave functions on continuous scales, we can observe their convolution difference on the image will be caused by two kinds of pixel: (1) Non-overlapped pixels belonging to the large-scale convolution template and not covered by small-scale convolution template; (2) Overlapped pixels covered both by the small-scale and large-scale convolution template but have different template coefficient. The observed image convolution difference can be expressed as formula (2):



(a) The serial-structure.



(b) The parallel-structure.

Fig. 2 The structure of the corner detection procedure

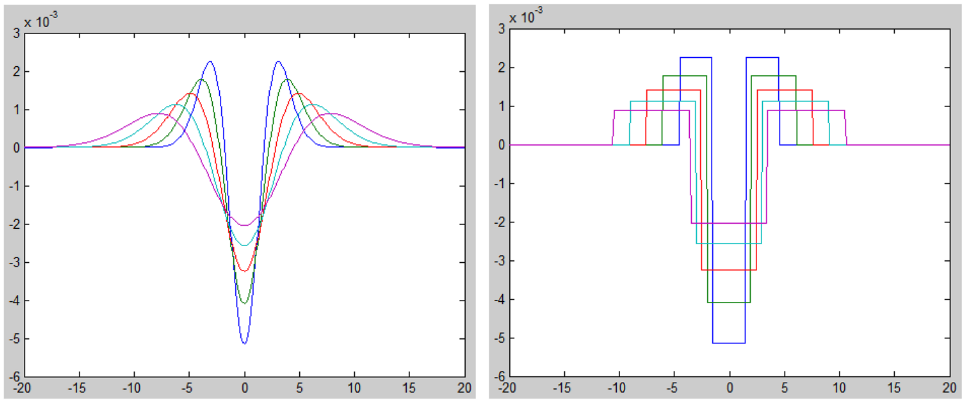


Fig. 3 Depict the square wave function and DoG convolution function in SIFT Gaussian pyramid

(a) The 1D DoG convolution function. **(b)** The square wave function.

$$\nabla DoG' = DoG'_{(1,outer)} + DoG'_{(1,inner)} - DoG'_{(0,inner)}$$

where, $\nabla DoG'$ is the observed image convolution difference for a certain feature pixel at different scale spaces estimated with square wave function; '$0'$' and '$1'$' represent adjacent two scale spaces and the lager number corresponds to the large scale space; 'inner' and 'outer' represent the overlapped and non-overlapped pixels, respectively; $\mathrm{DOG}'_{(1,outer)}$ is the result of image convolution contributed by the non-overlapped pixels; $DOG'_{(1,inner)} - DOG'_{(0,inner)}$ is the result of image convolution contributed by the overlapped pixels.

We impose a constraint on the adjacent two scale spaces in formula (2) that, the size of large-scale convolution template is expanded a circle than that of small-scale convolution template. It is easily to understand that the non-overlapped pixels in formula (2) will belong to the boundary of large-scale convolution template (called outer circle). In addition, though the overlapping pixels in formula (2) have two coefficients of large-scale convolution template and small-scale convolution template at the same time, but most coefficients are the same or very close except at the boundary of small-scale convolution template. So the result of $DOG'_{(1,inner)} - DOG'_{(0,inner)}$ in formula (2) will be mainly contributed by the overlapped pixels belonging to the boundary of small-scale convolution template (called outer circle). When using square wave functions as convolution template with imposed constraint, we can make a comparison of adjacent scale spaces for a feature pixel by directly inspecting pixels belonging to the inner and outer circle aforementioned, thus eliminating the need to construct a differential Gauss pyramid.

With the change of image scale, it is not difficult to understand that significant difference will happened between a given feature point and its neighbourhood pixels at the specific direction. So we confine the comparison of adjacent scale spaces to the pixels in a particular direction (equivalent to introducing direction constraint) on the two circles aforementioned. r_{ij}, r'_{ij} be the radius of inner circle and outer circle around a detected corner pixel, where 'i' indicates the Octave in predefined scale space, similar to SIFT algorithm, and 'j' indicates the level in each Octave. Figure 4 shows the method, which we code the pixels belonging to the inner and outer circle to estimate the scale for a certain feature point.

As seen in Fig. 4a, the pixels on the inner circle and outer circle are compared with each other based on predefined specific directions. Here the "specific directions" are determined by the overlapped pixels with local maximum template coefficient difference and belonging to the inner

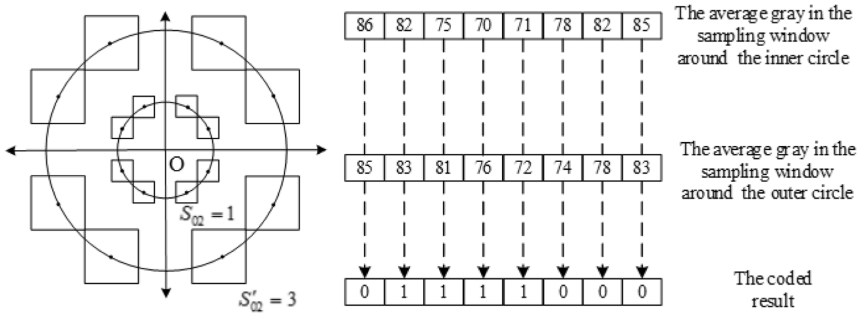
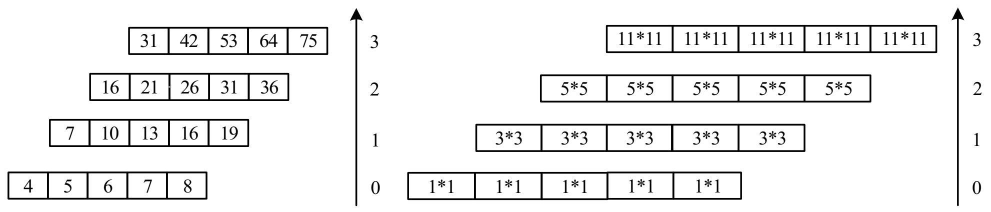

O
$S_{02} = 1$
$S'_{02} = 3$
86 82 75 70 71 78 82 85
The average gray in the sampling window around the inner circle
85 83 81 76 72 74 78 83
The average gray in the sampling window around the outer circle
0 1 1 1 1 0 0 0
The coded result

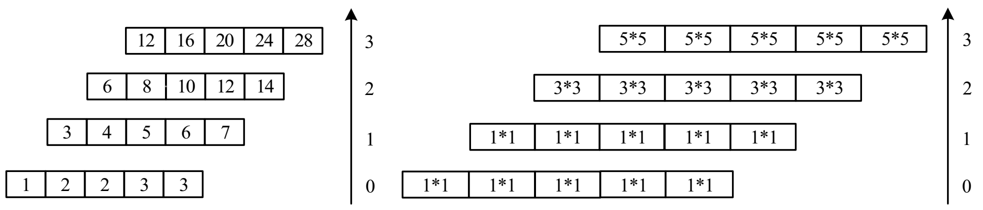
Fig. 4 The areas are selected and coded on the inner circle and outer circle when i equal 0, j equal 2

(a) The distribution of areas on specific direction. **(b)** The procedure of comparison coded.



(a) The radius value of outer circle. **(b)** The size of sampling window S_{ij} around outer circle.

Fig. 5 The radius value of the outer circle and the size of the sampling window S_{ij} with a different scale



(a) The radius value of inner circle. **(b)** The size of sampling window S_{ij} around inner circle.

Fig. 6 The radius value of the inner circle and the size of the sampling window S_{ij} with a different scale

circle. In addition, for the better anti-noise performance, the gray level of pixels to be replaced with the average gray of the sampling window S_{ij} where the size of sampling window is connected with the radius of circle at specific scale space to be estimated. Figures 5 and 6 show the size of sampling window, S_{ij} 02, respectively, corresponding to the radius of outer/ inner circle at different scale space (Octave). Finally, as seen in Fig. 4b, we quantize the image convolution difference at adjacent scale spaces for a given corner pixel by coding its specific neighbour pixels as formula (3):

$$b_{n,i,j} = \begin{cases} 1 & \overline{x_{n,i,j}} > \overline{x'_{n,i,j}} \\ 0 & \overline{x_{n,i,j}} \le \overline{x'_{n,i,j}} \end{cases}$$

where $\overline{x_{n,i,j}}$ is the average gray in the sampling window, which the centre is at the n-th neighbour pixel belong to outer circle; and $\overline{x'_{n,i,j}}$ is the average gray in the sampling window, which the centre is at the n-th neighbour pixel belong to the inner circle. The number of encoded "1" will be the quantized result $M_{i,j}$ Since the number of sampling pixels in formula (3) is varied with the radius of the inner circle, we normalize $M_{i,j}$ using formula (4).

$$M'_{i,j} = \frac{M_{i,j}}{N_{i,j}}, \quad i = 0,1,2,3 \quad j = 1,2,3,4,5$$

where $N_{i,j}$ is the number of pixels to be inspected on the inner circle.

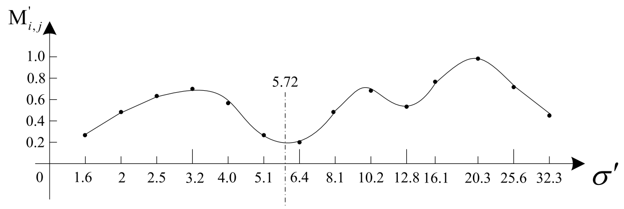


Fig. 7 Depicts the search of optimal feature scale and its precision position

As shown in Fig. 7, the optimal feature scale of the corner is obtained by searching for the minimum value $M'_{i,j}$ in continuous scale space. For any one corner pixel, if its reliable scale cannot be found, this corner will be discarded, otherwise, the scale will be used to get feature descriptor for the corner. Usually, the precise scale value is got using two order polynomial fit method.

3.3 Corner description and matching

In the paper, BRISK feature description algorithm [35] is used to get the corner feature, and the result is 512-bits binary feature vector for each corner. The range of neighborhood is delimited using the method in the corner scale estimation section to estimate scale. Consequently, the matching of any two corner feature vector is a simple computation of Hamming Distance (HD) [36].

From the aforementioned algorithm of corner detection and matching, we can find the HD computation for massive corner feature vectors will be very efficient in FPGA through a bitwise XOR operation. Meanwhile, proposed corner detector is quite concise in numerical (symbol) operation due to low computational complexity, e.g., simple division operation in C2 can be accomplished by one shift, the calculation and comparison of gradient orientation angles in C3 can be accomplished directly by coordinate vector subtraction, the calculation of pixel gradient, average gray level and non-maximum suppression in C1 only involve small neighbourhood operations (5*5 or 3*3 windows), and thus, it is very suitable for FPGA implementation. However, for the corner scale estimation and feature description in this paper, either large neighbourhood (the size of the window is up to hundreds pixel) or complex division/trigonometric function are required, and thus, it is difficult for FPGA implementation. To this end, a CPU-FPGA cooperative processing framework is proposed in the next section for high-speed image corner detection and matching.

4 CPU-FPGA cooperative processing

4.1 Computing framework

Proposed algorithm is implemented in a CPU-FPGA cooperative processing way. The assignment is assigned by (1) feature detection and HD calculation of feature vectors in the proposed algorithm are implemented on FPGA board because they have intensive computation and are easy to be processed in parallel. (2) Feature scale estimation and description in proposed algorithm involve complex function operations (e.g., division, triangular function) in larger neighborhoods, as well as huge memory space is required to store a large number of feature descriptors, so we implement them on CPU. CPU is also responsible for overall task scheduling, data transmission and matching distance comparison. Figure 8 shows the CPU-FPGA collaborative computing framework. The processes as following:

(1) Image data is transmitted from CPU to FPGA through PCIE interface and stored in FPGA's static random access memory (SRAM). FPGA detects corners and return them to CPU in real time. Synchronously, CPU completes the corner scale estimation and feature description, and save all the feature descriptors (vectors). Once feature descriptors are available for the image matching, their HD distance computation will be triggered in FPGA.
(2) Feature vectors of the reference image are at first transferred from CPU to FPGA board and stored in Double-Data-Rate Three Synchronous Dynamic Random Access Memory (DDR3 SDRAM). Then, the feature vectors of the image are transferred to FPGA one by one, and their HD to all the feature vectors of the reference image are computed parallelly. Finally, the optimal and sub-optimal HD of the matching points are returned to the CPU, and CPU outputs the tie points by judging their ratio of optimal HD to sub-optimal one according to a given threshold value.

Usually, a computer CPU can cooperate with several FPGA boards using PCIE as a transmission interface. Thus, it's easily understood that the processing capacity of the CPU-FPGA collaborative computing framework in this paper is proportional to the number of FPGA boards configured on the computer motherboard. Because the frequency of CPU is higher than FPGA, we transfer data using Direct Memory Access (DMA) technology based on the PCI-E hardcore. In the paper, 64 bits PCIE $\times$ 8 data bus is used and 4 pixels are transferred each time. The max speed at 4 GB/S can be achieved. Using FPGA to realize PCI-E function

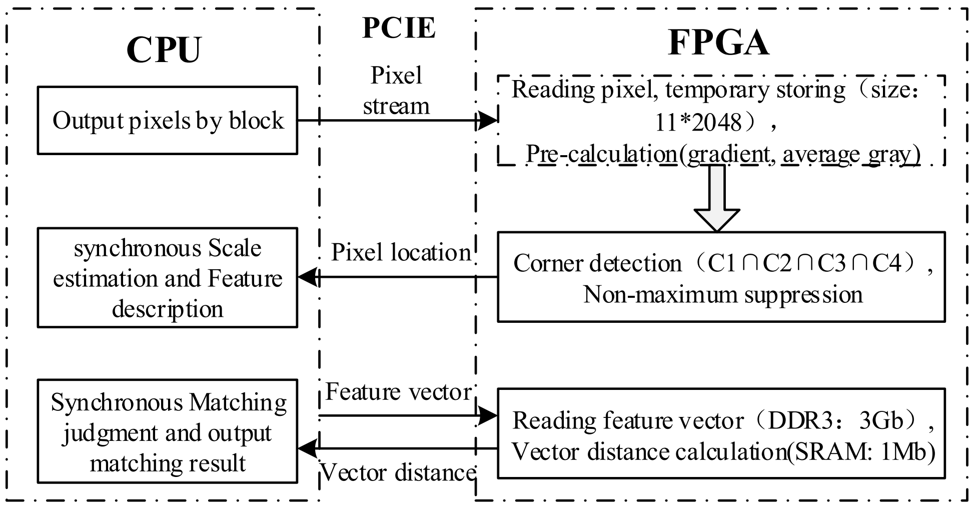

CPU
PCIE
FPGA
Output pixels by block
Pixel stream
Reading pixel, temporary storing（size: 11*2048）,
Pre-calculation(gradient, average gray)
synchronous Scale estimation and Feature description
Pixel location
Corner detection（C1∩C2∩C3∩C4）, Non-maximum suppression
Synchronous Matching judgment and output matching result
Feature vector
Vector distance
Reading feature vector（DDR3: 3Gb）, Vector distance calculation(SRAM: 1Mb)

Fig. 8 The CPU-FPGA cooperative processing framework for high-speed corner detection and matching

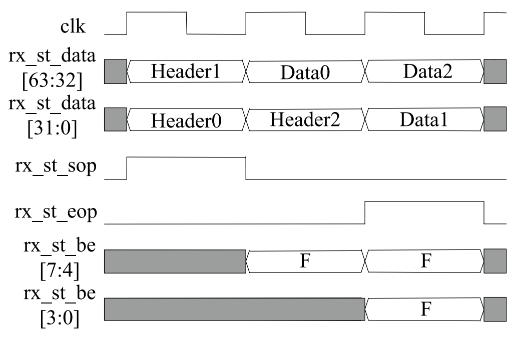
is simply to complete the processing of TLP (Transaction Layer Packet). Avalon-ST (Avalon Streaming Interface) is defined by Altera Company, which is used to transmit TLP. Avalon-ST includes receiving signals and sending signals, and the time sequence between receiving and sending is similar. Figure 9 depicts the transmitting timing diagram of Avalon-ST interface in PCI-E.

4.2 Algorithm Implementation on FPGA

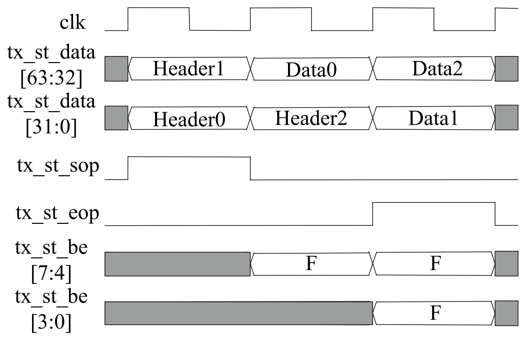
As seen in Fig. 8, implemented an algorithm module on FPGA including image pre-processing, corner detecting and HD computation in feature matching.

4.2.1 Image pre-processing

The implementation of image pre-calculation in FPGA is showed in Fig. 10. The width of image is inputted to FPGA and set as 2048. At first, we need to implement a parallel sliding window function. We use a Shift_RAM with 11 lines, 2048 data in each line and 8bits for each data, which is a shift register chain, to get the pixel matrix of 11*11 in FPGA, as seen in Fig. 10a. At the same time, the output of each Shift-Register is saved in cache and gets the pixel matrix of 11*11 size, as seen in Fig. 10b. Then we synchronously get the gradient value ∇_y and ∇_x with the convolution between the pixel matrix and convolution kernel, as seen in Fig. 10c and Fig. 10d. In the implementation of gradient value ∇_y, at first, we calculate the sum of 3 pixels in the first line and the sum of 3 pixels in the third line; then calculate the difference value between the first line and the third line, the top digit is the sign bit according to the comparison of data. In the implementation of gradient value ∇_x, at first, we calculate the sum of 3 pixels in the first column and the sum of 3 pixels in the third column; then calculate the difference value between the first column and the third column, the top digit is the sign bit according to the comparison of data. The implementation of gradient value ∇_y and ∇_x is showed in Fig. 11a. The average gray $\overline{I_{xy}}$ of 8-neighbourhood is got by calculating the convolution between the pixel matrix and convolution kernel, as seen in Fig. 10e. In the implementation of average gray $\overline{I_{xy}}$, at first, we calculate the sum of 3 pixels in the line direction, and 3



(a) The receiving time sequence.



(b) The sending time sequence.

Fig. 9 The transmitting timing diagram of 3-D Word header TLP in PCI-E Avalon-ST interface

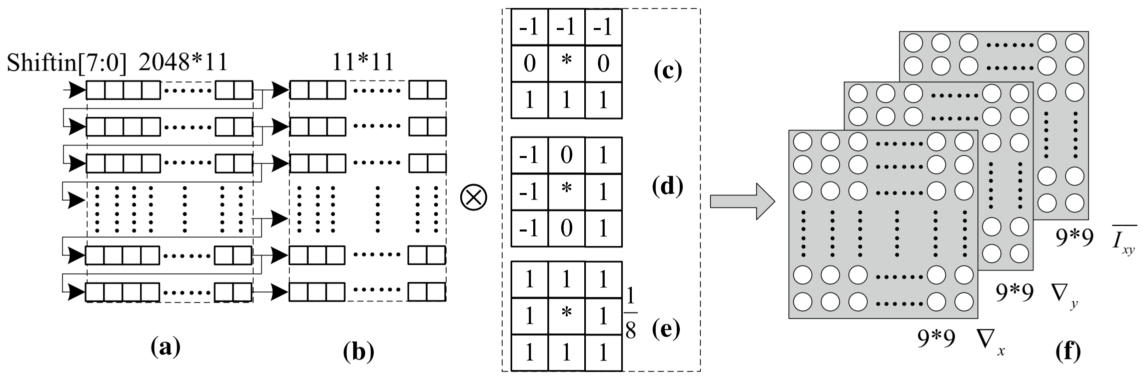
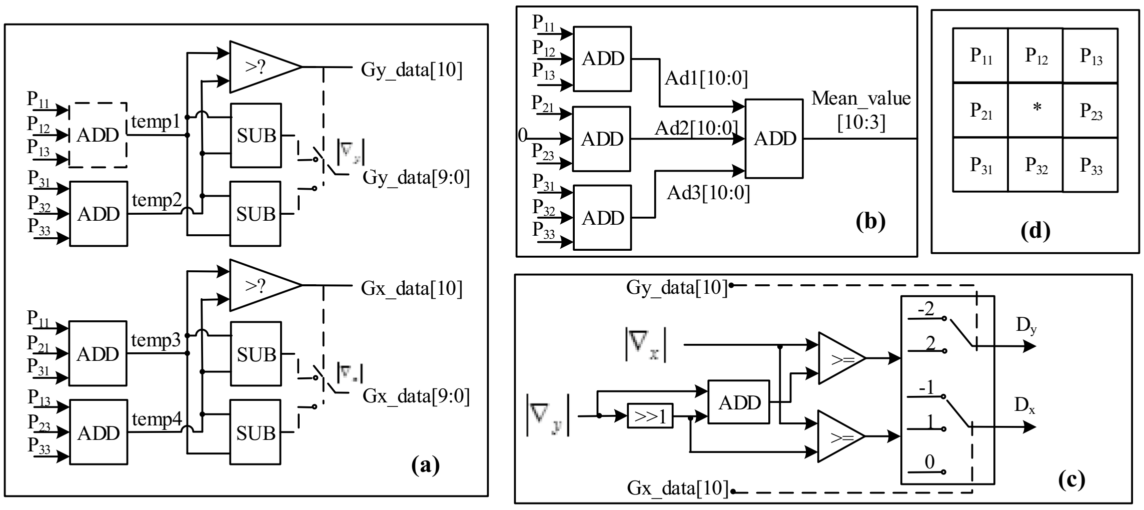


Fig. 10 Depict of image pre-calculation on FPGA. **a** 11 Shift-Registers; **b** the pixel matrix; **c** the convolution kernel of Y direction gradient; **d** the convolution kernel of X direction gradient; **e** the convolution kernel of average gray; **f** the result of pre-calculation



P11 P12 P13 ADD temp1
P31 P32 P33 ADD temp2
SUB SUB >? Gy_data[10] |∇y| Gy_data[9:0]
P11 P21 P31 ADD temp3
P13 P23 P33 ADD temp4
SUB SUB >? Gx_data[10] |∇x| Gx_data[9:0]
(a)
P11 P12 P13 ADD Ad1[10:0]
P21 0 P23 ADD Ad2[10:0]
P31 P32 P33 ADD Ad3[10:0]
ADD Mean_value [10:3]
(b)
P11 P12 P13 P21 * P23 P31 P32 P33
(d)
Gy_data[10] |∇x| |∇y| >>1 ADD >= >= -2 2 -1 1 0 Dy Dx Gx_data[10]
(c)

Fig. 11 The implementation of corner detection on FPGA. **a** The gradient calculation. **b** The average calculation. **c** The 16 detection pixels location. **d** The position distribution of pixels P_{11}, P_{12}, ...

lines are parallelly calculated; then the final sum value is got by calculating the sum of 3 sums in each line; the end, the calculation of divide 8 is replaced by drop lowest 3 bits (LSB). The implementation of average gray $\overline{I_{xy}}$ is showed in Fig. 11b. The implementation of image pre-calculation will consume 22 Kbit SRAM of FPGA.

4.2.2 Corner detecting

The core problem of corner detection implementation on FPGA is how to obtain the pixel gradient $\nabla(x, y)$ and locate 16 pixels in the neighbourhood of candidate corner. The implementation of absolute value $\left|\nabla_x\right|$ and $\left|\nabla_y\right|$ in sub-detector C_1 can be got when calculation gradient value, as shown in Fig. 11a. When gradient values are calculated, the position distribution of neighboring pixels of the current pixel is shown in Fig. 11d. The implementation of locating 16 pixels in the neighbourhood of corner candidate by three sub-detector C_2, C_3, C_4 is shown in Fig. 11c, in which the gradient ∇_x and ∇_y are mapped to get their integer step Dx/Dy, and the comparer and selectors are used to replace division computation involved in the mapping. Because the possible value of integer step Dx /Dy is 0, − 1, 1, − 2, 2 and the sum of $\left|D_x\right|$ and $\left|D_y\right|$ is 2, we set that the rule of mapping is shown in the formula (5). The sign bit of D_x and D_y is same as the sign bit of gradient ∇_x and ∇_y. We can easily get 16 pixels in the neighbourhood of corner candidate.

$$\begin{cases} \left|D_x\right| = 0, \left|D_y\right| = 2 & if \left|\nabla_x\right| < \left|\nabla_y\right|/2 \\ \left|D_x\right| = 1, \left|D_y\right| = 1 & if\ 3 * \left|\nabla_y\right|/2 > \left|\nabla_x\right| \geq \left|\nabla_y\right|/2 \\ \left|D_x\right| = 2, \left|D_y\right| = 0 & if \left|\nabla_x\right| \geq 3 * \left|\nabla_y\right|/2 \end{cases}$$

4.2.3 HD computation

The implementation of HD computation on FPGA is presented in Fig. 12. At first, all feature vectors of the reference image are transferred from CPU to FPGA board and stored in external DDR3 SDRAM. During the matching, feature vectors of the reference image are imported from the DDR3 SDRAM to FPGA SRAM using data block mode. The data block's size is 65536 bits (the depth is 16 bits; the width is 4096 bits). DDR3 SDRAM is composed of 8 chips and the data bus is 128 bits. Data are transferred from DDR3 SDRAM to FPGA SRAM using 400 MHz two-channel internal storage strategy. 2048 feature vectors can be prestored in the FPGA SRAM. When FPGA is receiving a new feature vector to be matched, FPGA timely computer its HD to all feature vectors pre-stored in the SRAM. With PCIE × 8 data bus, the transfer of a feature vector with 512 bits from CPU to FPGA can be done in 16 clock cycles of FPGA in this paper. As seen in Fig. 12, 128 HD calculators, each one is composed of 512 bits XOR operation and bits accumulator, are used to parallelly calculate the HD of two feature vectors. Hence, within a clock cycle of FPGA, the HD of one feature vector can be computed with 128 feature vectors pre-stored in SRAM. Finally, the matching points associated with the best and second-best HD will be got with two-level comparer.

5 Experiments and discussion

We implement the proposed algorithm and architecture on a normal PC and FPGA board. The PC is configured with a 2.0 GHz Intel E5-2650 CPU, 32G RAM, Win10 operating

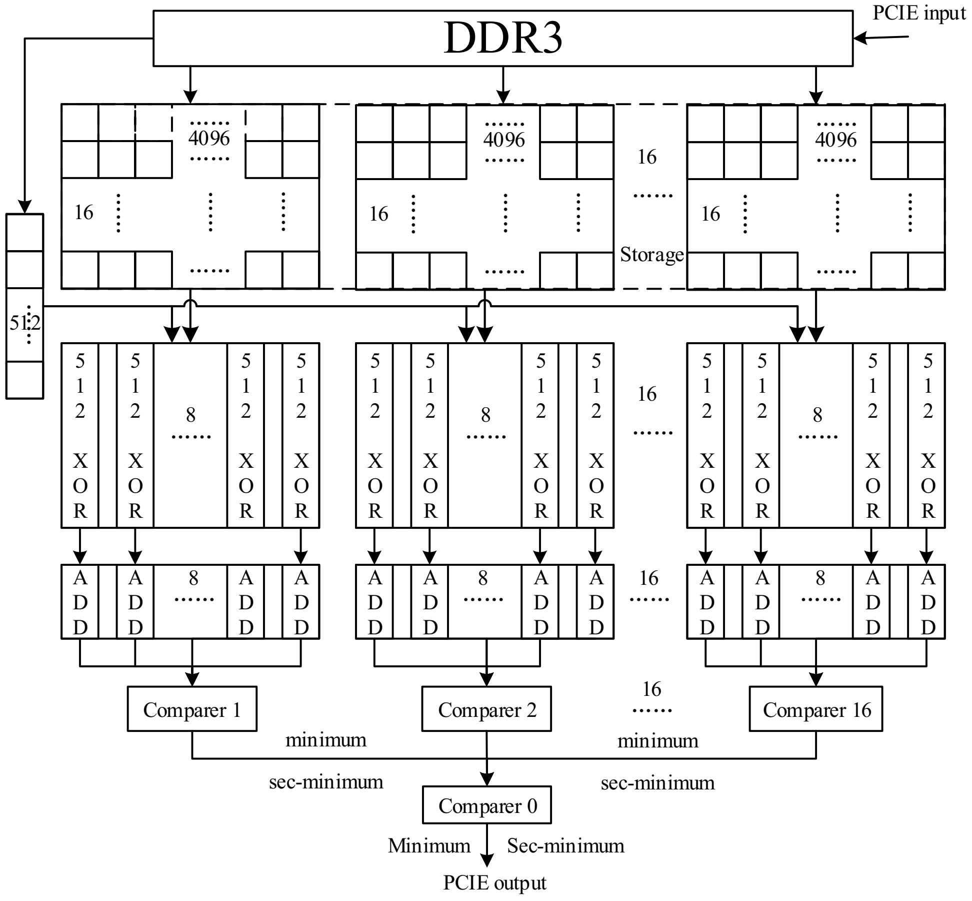


Fig. 12 Depict of feature vector matching on FPGA

system and VS2010 Code compiler. The selected FPGA board is the Altera 5CGXFC5C6F27C7N FPGA, which has 77 K Logic Cells, 4884 Kb SRAM, and runs on 100 MHz clock frequency. The FPGA board also contains 3 GB DDR3 SDRAM with 128 bits bus-width. The test items include the performance of corner detection and the performance of the CPU-FPGA cooperation processing.

5.1 Performance of corner detection

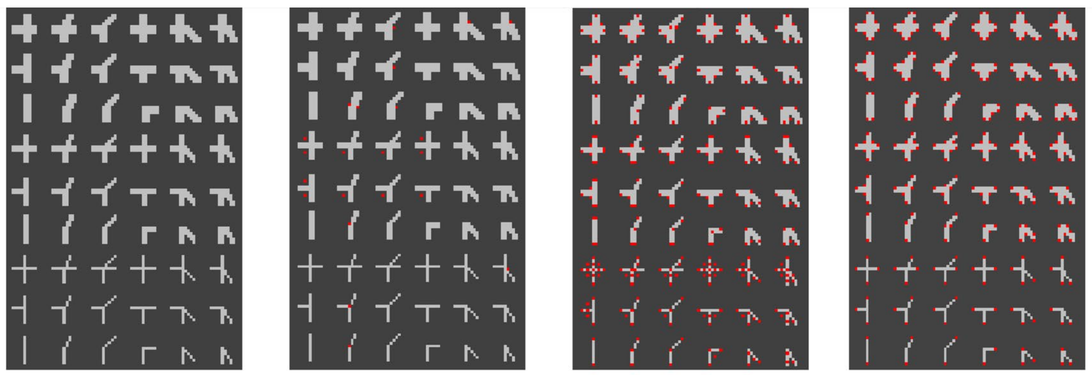
Proposed corner detector with serial-structure is implemented on the PC and we compare its results to that of FAST, HARRIS algorithm. The performance test includes the accuracy and repeatability of the algorithm.

5.1.1 Accuracy test

We use Scott Krig's synthetic corner image [37] to test the robustness of the corner detection algorithm. The synthetic corner image, in the size of 2048*2048 pixels, includes 16*24 image units of 54 unique patterns. Each pattern is arranged on a grid of 14*14 pixel rectangles. Gray values are 0×40 and $0\times C0$. The original noise-free image (Fig. 13), Salt and Pepper noise image (the noise density is $D=0.05$, Fig. 14) and Gaussian noise image (the mean is $M=0$; the variance is $V=0.01$, Fig. 15) are used to test the accuracy and robustness of corner detector. If the Grid Distance between the detected corner and true corner is more than 1 pixel, the detected corner is set as an imprecise corner. The accuracy of tested corner detectors is illustrated in Table 1. Results show that the proposed detector in the paper has the highest accuracy at 98.7% for the original noise-free image. In a test of noise image, FAST almost is a failure to detect corner. The performances of HARRIS and proposed detector all have declined for noise images, but the accuracy of the proposed detector is better than HARRIS. The accuracy of the proposed detector is about 21% higher than HARRIS for Salt & Pepper noise image, and about 10% higher than HARRIS for Gaussian noise image.

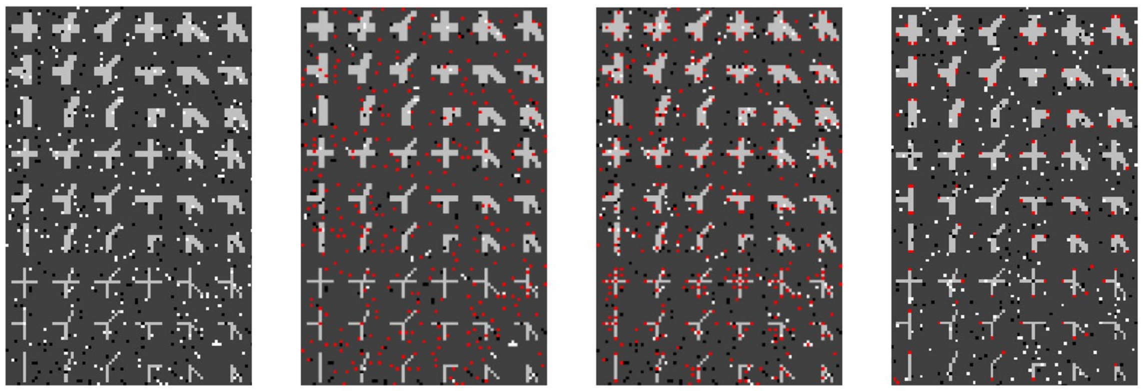
5.1.2 Repeatability test

Corner repeatability test is implemented on Edward Rosten's Maze image set and Bas-relief repeatability test image set [38]. As seen in Fig. 16, Maze dataset is got by taking a prop with abundant textural features and geometric features, but it has serious projective warps. Bas-relief dataset is got by taking a flat plane with significant relief, but it has



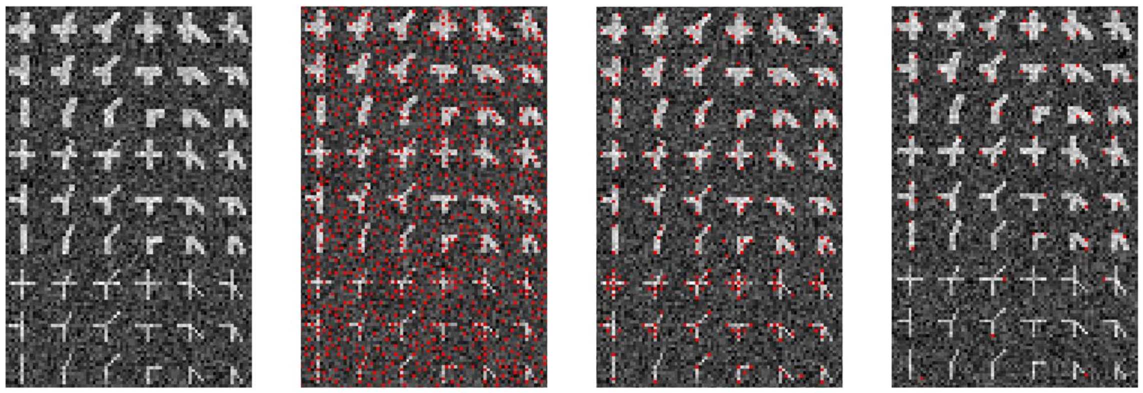
(a) The Original image. **(b)** The result of FAST. **(c)** The result of HARRIS. **(d)** The result of the new algorithm.

Fig. 13 The test results from noise-free synthetic images



(a) Salt & Pepper noise. **(b)** The result of FAST. **(c)** The result of HARRIS. **(d)** The result of the new algorithm.

Fig. 14 The test results from synthetic images with Salt and Pepper noise



(a) Gaussian noise. **(b)** The result of FAST. **(c)** The result of HARRIS. **(d)** The result of the new algorithm.

Fig. 15 The test results from synthetic images with Gaussian noise

feature changes from different viewpoints. The resolution of the test image in the dataset is 768*576. The repeating distance is set to 3 pixels and we compare our results to that of FAST, HARRIS, SIFT algorithm, respectively, as shown in Fig. 17a, b. It can be found that repeatability of the proposed detector in the paper is best for a given number of feature point, especially; the new algorithm has a significant advantage to Maze dataset because there are abundant clear corner points in image.

5.2 Performance of feature-point matching

We use the publicly available datasets of [5] to test our detector against the state-of-the-art on CPU. In the dataset, scale/rotation images (boat) is used to test of our algorithm when the scale has changed of image. Figure 18 shows repeatability and matching score plots for all the detectors included in the classical feature matching algorithm. The result shows that the new algorithm is quite invariant to scale. In the test, the parameters of the classical algorithms are set with default parameters. Figure 18b shows that the number of feature-point pairs is basically constant when the image matches with different scale images. Figure 18a shows that the accuracy rate of the new algorithm is slightly lower than SIFT, but it is better than other classical algorithms.

5.3 CPU-FPGA cooperation processing test of the new algorithm

Two kinds of aerial image, medium resolution (5616*3744) images and high resolution (11320*17310) images, as shown in Figs. 19 and 20, are used in the CPU-FPGA cooperation processing test of the new algorithm. At first, we comparison test the new algorithm of CPU-FPGA cooperation implementation and its CPU implementation. Then, we comparison test the new algorithm and SIFT algorithm. SIFT algorithm is the most widely used to do feature-point matching in the field of photogrammetric and remote sensing. But

Table 1 The data report of detected corners on synthetic image

Algorithm	Noise-free image			Salt and Pepper noise image			Gaussian noise image		
	FAST	HARRIS	The paper	FAST	HARRIS	The paper	FAST	HARRIS	The paper
Total corner	8064	119040	119002	139236	155916	42624	376064	109700	29568
Real corner	3840	102912	117455	13440	111756	39556	54528	84356	25222
Accuracy- rate	47.62%	86.45%	98.7%	9.7%	71.7%	92.8%	14.5%	76.9%	85.3%

(a) Maze dataset.

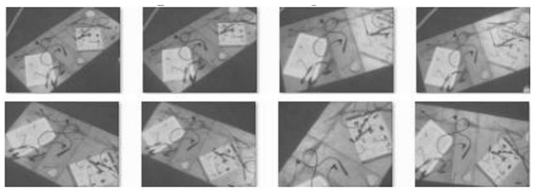
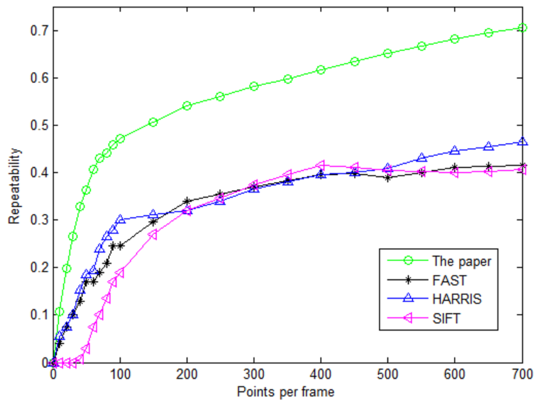
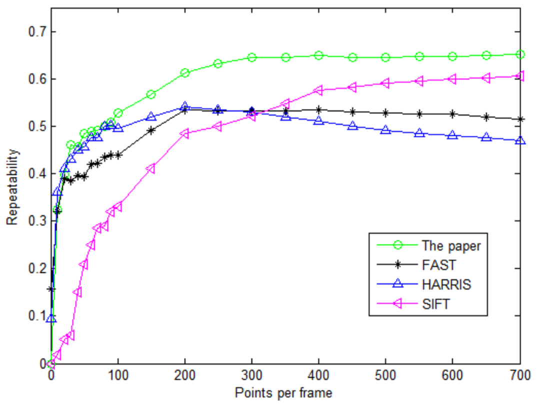

(b) Bas-relief dataset.

Fig. 16 Depict of image dataset for repeatability test

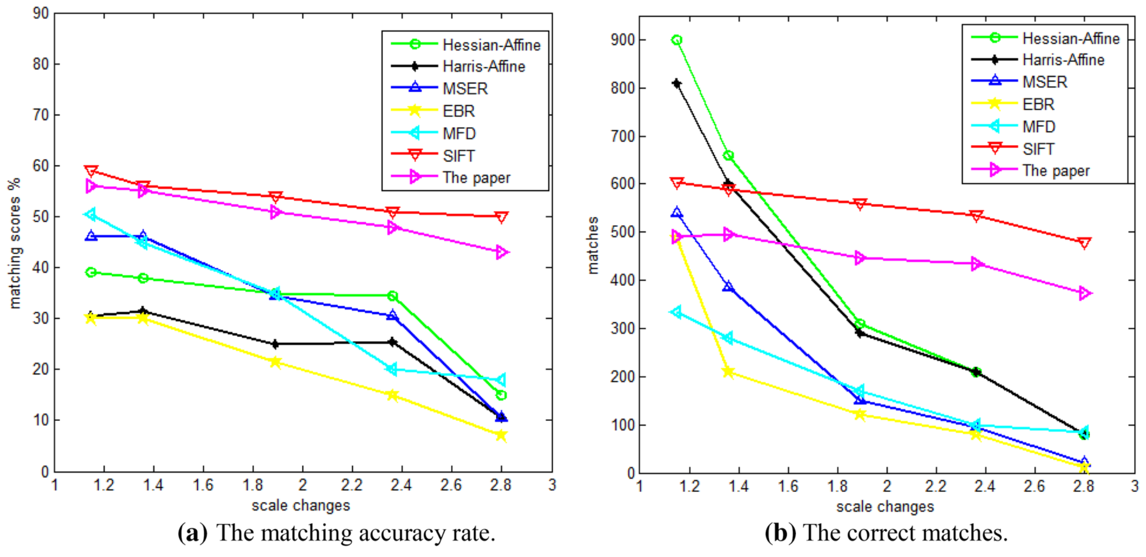


(a) Repeatability result of Maze dataset.



(b) Repeatability result of Bas-relief.

Fig. 17 Depict of the repeatability test results from two image dataset



(a) The matching accuracy rate. **(b)** The correct matches.

Fig. 18 The performance of feature-point matching for different scale images

the hardware implementations of SIFT is limited on high-resolution aerial images because the SIFT algorithm has a high computational cost and memory requirement. So SIFT is implemented by two types in the comparison test: the one is by CPU, another one is by CPU + GPU cooperation. In the CPU-FPGA implementation of new algorithm, because the preset width of the image is 2048 in FPGA, the aerial image has to be divided into small images with constant width 2048 pixels before it is transferred to FPGA, as seen the red line in Figs. 19 and 20. If the image width is less than 2048 pixels, "0" is added to the end.

The speed data of a new algorithm by CPU-FPGA cooperation implementation and its CPU implementation can be found in Table 2. When the high-resolution aerial images are used to test new algorithm, the feature-point detection of CPU-FPGA cooperation is about 16 times faster than alone CPU; the matching of CPU-FPGA cooperation is about 451 times faster than alone CPU. When the medium resolution aerial images are used to test new algorithm, the feature-point detection of CPU-FPGA cooperation is about 32 times faster than alone CPU; the matching of CPU-FPGA cooperation is about 443 times faster than alone CPU. The result shows that the new algorithm by FPGA paralleled has a great acceleration.

The speed data of the new algorithm and SIFT algorithm can be found in Table 2, too. The new algorithm is implemented by CPU-FPGA cooperation. SIFT algorithm is tested on the CPU and CPU + GPU. When the high-resolution aerial images are used to test algorithm, the number of feature-point pairs by new algorithm is about two times SIFT; the feature-point detection of new algorithm is about 300 times faster than SIFT algorithm which is on a CPU; the matching of new algorithm is about 340 times faster than SIFT algorithm which is on a CPU; the feature-point detection of new algorithm is about 15 times faster than SIFT algorithm which is on CPU + GPU; the matching of new algorithm is about 13 times faster than SIFT algorithm which is on CPU + GPU; The speed of scale estimation and feature coding in the new algorithm is about more 40–30% than SIFT. When the medium resolution aerial images are

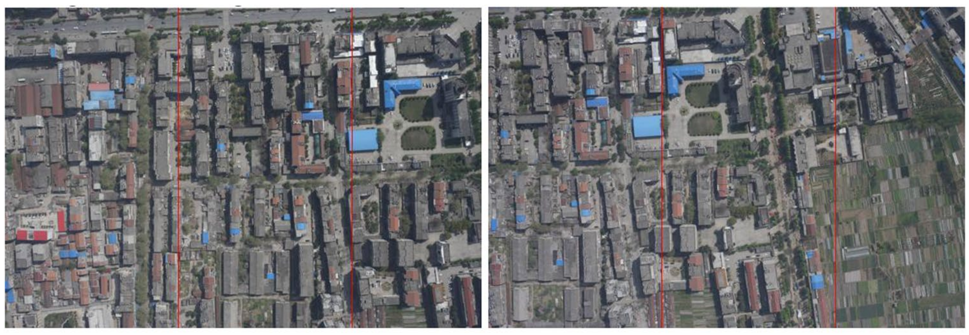


Fig. 19 The medium resolution aerial image pair

(a) Left image. **(b)** Right image.

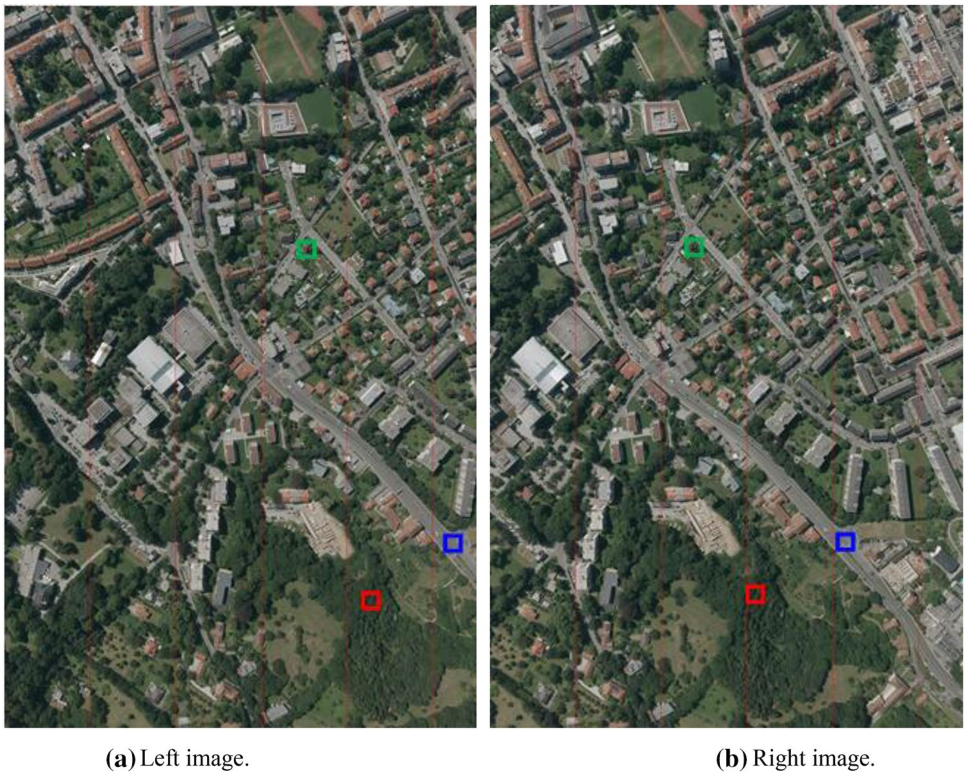


Fig. 20 The high-resolution aerial image pair

(a) Left image. **(b)** Right image.

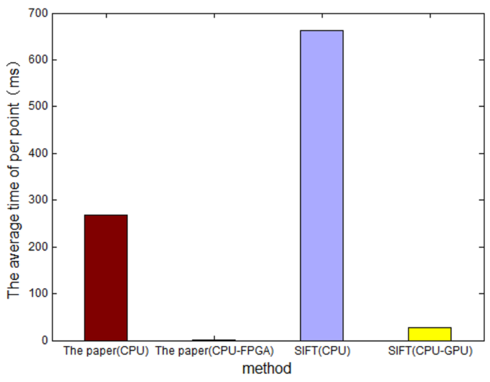
Table 2 The results of corner detecting and matching on two kind of aerial image stereo

Method	Aerial images	The number of point		Feature-point pair	Time (ms)			Processor
		Left image	Right image		feature-point detection	Feature coding	Matching	
The paper	High resolution	1,217,688	1,169,177	187,768	143,348	32,606	50,314,518	CPU
					8555	32,571	111,427	CPU + FPGA
	Medium resolution	80,038	85,498	4588	15,382	2267	243,080	CPU
					462	2259	548	CPU + FPGA
SIFT	High resolution	820,632	867,658	61430	2,528,471	52,745	38,067,826	CPU
					128,364	45,947	1,487,940	CPU + GPU
	Medium resolution	92,024	89,038	4636	266,370	5586	438,064	CPU
					13,160	4927	17516	CPU + GPU

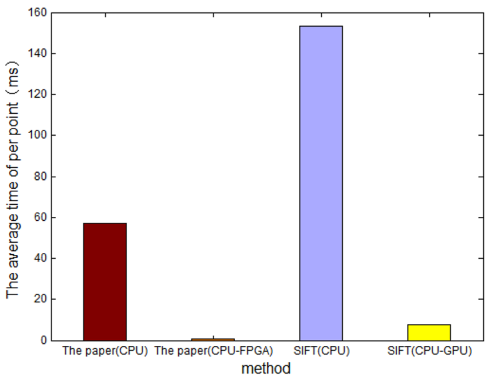
used to test algorithm, the number of feature-point pairs by new algorithm is very close SIFT; the feature detection of new algorithm is about 576 times faster than SIFT algorithm which is on a CPU; the matching of new algorithm is about 799 times faster than SIFT algorithm which is on a CPU; the feature detection of new algorithm is about 28 times faster than SIFT algorithm which is on CPU + GPU; the matching of new algorithm is about 32 times faster than SIFT algorithm which is on CPU + GPU; The speed of scale estimation and feature coding in the new algorithm is about 2 times SIFT.

As seen in Fig. 8, the feature-point detection and feature coding are synchronously processed in the CPU-FPGA cooperative processing framework of the new algorithm. The speed of feature coding by CPU is slower than feature-point detection by FPGA, so the total time of the new algorithm is the sum of feature coding and matching. According to Table 2, the total time of new algorithm by CPU-FPGA cooperative processing is 143998 ms for the high-resolution aerial image pair, and it is 2807 ms for the medium resolution aerial image pair. Because the time of feature coding and matching is related to the amount of feature-points, we calculate the consumed average time of each feature-point pair. According to Table 2, the average time of per feature-point pair of the new algorithm in CPU-FPGA cooperative processing is about 0.8 ms for the high-resolution aerial images, and it is about 0.6 ms for the medium resolution aerial images. Based on the average time of per feature-point pair, we compare the new algorithm with the SIFT algorithm, and get the result, as shown in Fig. 21. From Fig. 21, it can be found that the speed of the new algorithm is hundred times SIFT which is on a CPU, and dozens times SIFT which is on CPU + GPU. Hence, very high-speed corner detecting and matching performance is achieved with the proposed algorithm on CPU-FPGA cooperative process.

Figures 22, 23 and 24 shows the matched feature-point pairs of the new algorithm and SIFT on test images. For the high-resolution aerial images with large pixel array, part matching result on local small areas are denoted by a rectangle with different colour, corresponding to three kinds of different scenes: building, vegetation and road, as seen in Fig. 22 and Fig. 23. It can be found from the result that the proposed algorithm can get more feature-point pairs than SIFT, especially in vegetation and road area. For medium resolution aerial images, our matching results are close to SIFT, and both of them have no obvious error feature-point pairs, as seen from Fig. 24.

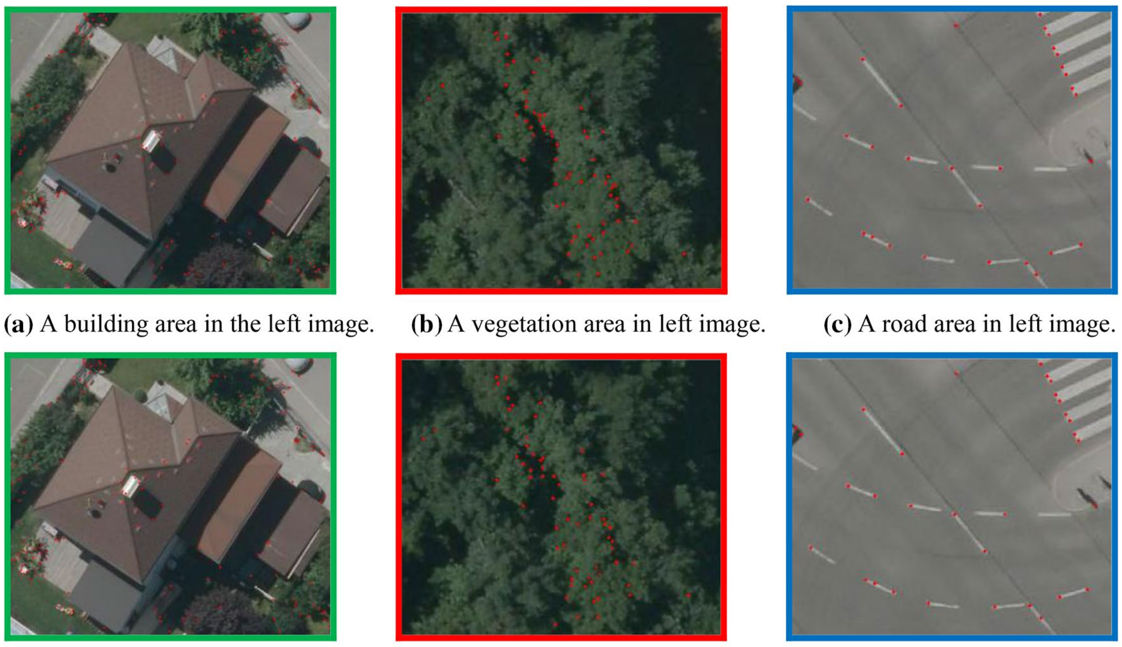


(a) The result of medium resolution aerial images.



(b) The result of medium resolution images.

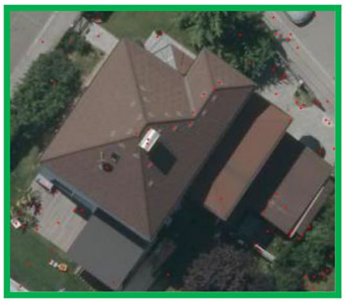
Fig. 21 Comparison of average time cost on per feature-point pair



(a) A building area in the left image. **(b)** A vegetation area in left image. **(c)** A road area in left image.

(d) A building area in right image. **(e)** A vegetation area in right image. **(f)** A road area in right image.

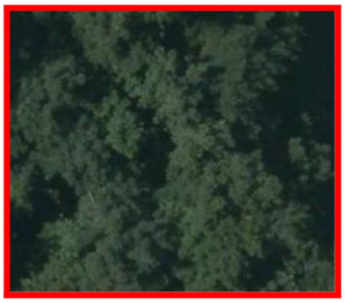
Fig. 22 Matched local feature-point pairs by proposed algorithm on the high-resolution aerial images


(a) A building area in left image.

(b) A vegetation area in left image.

(c) A road area in left image.

(d) A building area in right image.


(e) A vegetation area in right image.

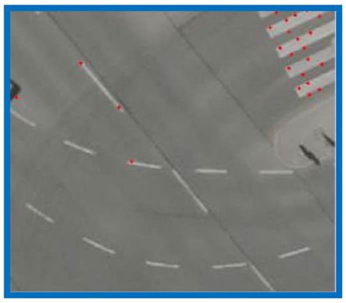
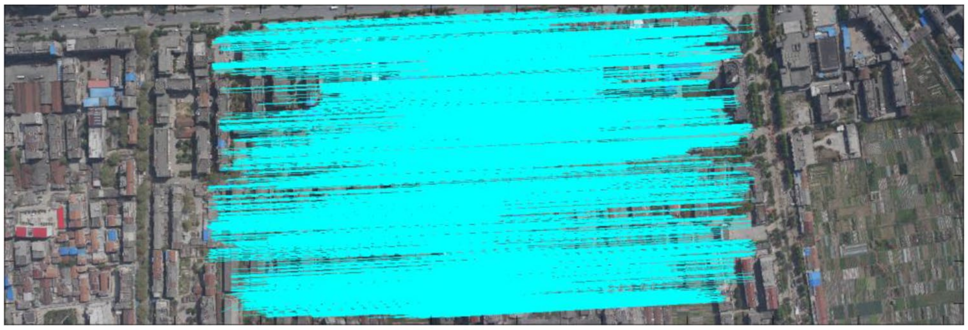
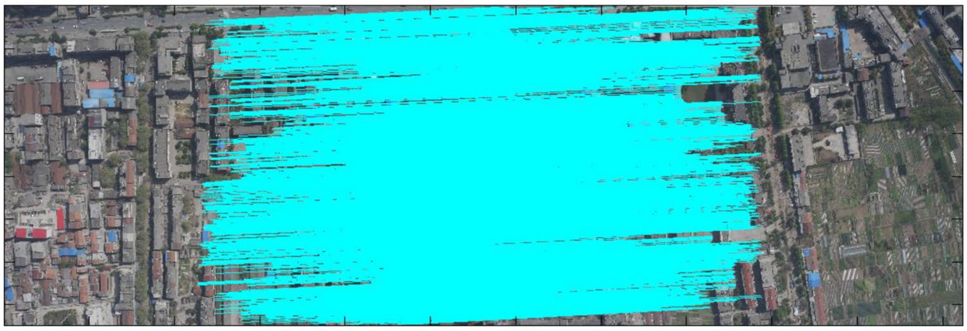

(f) A road area in right image.

Fig. 23 Matched local feature-point pairs by SIFT on the high-resolution aerial images

Fig. 24 Matched feature-point pairs on the medium resolution aerial images


(a) The results by proposed algorithm.


(b) The results by SIFT.

6 Conclusions

This paper presents a novel corner detection and scale estimation algorithm. The new corner detector has a higher accuracy rate and robustness than FAST and HARRIS. The accuracy of the new corner detector is 98.7% for noise-free simulation image, about 21% higher than HARRIS for Salt and Pepper noise image, and 10% higher than HARRIS for Gaussian noise image. The repeatability of new corner detector is best than several famous local feature detection algorithm including FAST, HARRIS and SIFT. The new scale estimation method can get the feature scale value in the original image without building Gaussian pyramid and searching max response value in each level, which greatly increase computational efficiency and reduces memory cost.

A CPU-FPGA cooperative processing architecture is established along with the proposed algorithm. In the architecture, the corner detection and the HD computation are parallelly implemented in FPGA; the scale estimation, feature coding and storage of corner location are implemented in CPU. The architecture has very high computation efficiency: the average time cost of per feature-point pairs is about 0.8 ms for the 11320*17310 high-resolution aerial images and 0.6 ms for the 5616*3744 medium resolution aerial images. The speed of new algorithm in the CPU-FPGA cooperative process is hundred times SIFT which is on a CPU, and dozens times SIFT which is on CPU + GPU. Furthermore, the computing efficiency of the CPU-FPGA architecture can be further increased, and it is proportional to the number of configured FPGA circuit board which depend on the capacity of PCIE slots on the computer motherboard. The architecture also works effectively on the aerial images with huge pixel arrays, since the image can be divided into some small images with constant width 2048. If the FPGA board with 3 GB DDR3 SDRAM and the number of detected corner in an image is less than one thousandth of the number of pixels, proposed architecture can handle the aerial image up to 600 million pixels. The new algorithm and CPU-FPGA cooperative architecture are valuable for the application of massive aerial images.

Acknowledgements This work was partly supported by National Natural Science Foundation of China (41761087), Natural Science Foundation of Guangxi Province (2017GXNSFAA198162, 2020GXNSFAA159091), Guangxi emphasis laboratory for optoelectronics information Project (GD18108), and Innovation Project of Guangxi Graduate Education (YCBZ2017051, 2018YJCX64), at the same time, thank for the study abroad program for graduate student of Guilin University of Electronic Technology.

References

1. Zhang, B.: Intelligent remote sensing satellite system. Remote. Sens. **15**(3), 415–431 (2011)
2. Zitová, B., Flusser, J.: Image registration methods: a survey. Image. Vis. Comput. **21**(11), 977–1000 (2003)
3. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
4. Morel, J.M., Yu, G.: ASIFT: a new framework for fully affine invariant image comparison. SIAM. J. Imag. Sci. **2**(2), 438–469 (2009)
5. Mikolajczyk, K., Tuyt Elaars, T., Schm Id, C.: A comparison of affine region detectors. Int. J. Comput. Vis. **65**(1–2), 163–186 (2005)
6. Kehl, C., Buckley, S.J., Viseur, S., et al.: Automatic illumination-invariant image-to-geometry registration in outdoor environments. Photogramm. Rec. **32**(158), 93–118 (2017)
7. Hirschmüller, H.: Stereo processing by semiglobal matching and mutual information. IEEE. Trans. Pattern. Anal. Mach. Intell. **30**(2), 328–341 (2008)
8. Boykov, Y., Veksler, O., Zabin, R.: Fast approximate energy minimization via graph cuts. IEEE. Trans. Pattern. Anal. Mach. Intell. **23**(11), 1222–1239 (2001)
9. Sun, J., Shun, H.Y., Zheng, N.N.: Stereo matching using belief propagation. IEEE. Trans. Pattern. Anal. Mach. Intell. **25**(7), 787–800 (2003)
10. Scharstein, D., Szeliski, R.: A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. Int. J. Comput. Vis. **47**(1–3), 7–42 (2002)
11. Lee, C.A., Gasster, S.D., Plaza, A.: Recent developments in high performance computing for remote sensing: a review. IEEE. J. Selec. Top. Appl. Earth. Observ. Remote. Sens. **4**(3), 508–527 (2011)
12. Avrithis, Y., Rapantzikos, K.: "The medial feature detector: Stable regions from image boundaries." In: international conference on computer vision (ICCV) (2011)
13. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide-baseline stereo from maximally stable extremal regions. Image. Vis. Comput. **22**(10), 761–767 (2004)
14. Acharya, K.A., Venkatesh, B.R., Vadhiyar, S.S.: A real-time implementation of SIFT using GPU. J. Real. Time. Image. Proc. **14**(2), 267–277 (2018)
15. Li, Z., Jia, H., Zhang, Y.: HartSift: A high-accuracy and real-time SIFT based on GPU. 2017 IEEE 23rd international conference on parallel and distributed systems (ICPADS). IEEE computer society (2017)
16. Jiang, C., Geng, Z.X., Wei, X.F., et al.: SIFT implementation based on GPU. ISPDI 2013-fifth international symposium on photoelectronic detection and imaging. International society for optics and photonics (2013)
17. Patel, V., Patel, B.: Indexing SURF features by SVD based basis on GPU with multi-query support[C]// international conference on intelligent computing. Springer, Cham (2014)
18. Huang, F.C., Huang, S.Y., Ker, J.W.: High-performance SIFT hardware accelerator for real-time image feature extraction. IEEE. Trans. Circuits. Syst. Video. Technol. **22**(3), 340–351 (2012)
19. Zhong, S., Wang, J., Yan, L.: A real-time embedded architecture for SIFT. J. Syst. Architect. **59**(1), 16–29 (2013)
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: 1–6 (2013)
21. Huang, J.J., Guoqing, Z.: On-Board Detection and Matching of Feature Points. J. Remote Sens **9**(6), 1–17 (2017)
22. Qamar, A., Muslim, F., Gregoretti, F., et al.: High-level synthesis for semi-global matching: is the juice worth the squeeze? IEEE. Access. **99**, 1–1 (2016)
23. Harris, C., Stephens, M.: A combined corner and edge detector. Proc. Alvey. Vis. Conf. Citeseer. **3**, 147–151 (1988)

24. Amaricai, A., Gavriliu, C.E., Boncalo, O.: An fpga sliding window-based architecture harris corner detector. 2014 24th International conference on field programmable logic and applications (FPL), IEEE: 1–4 (2014)
25. Chao, T.L., Wong, K.H.: An efficient FPGA implementation of the harris corner feature detector. 2015 14th IAPR International Conference on Machine Vision Applications (MVA), IEEE: 89–93 (2015)
26. Brenot, F., Fillatreau P., Piat, J.: FPGA based accelerator for visual features detection. In: 2015 IEEE International Workshop of Electronics Control Measurement, Signals and their Application to Mechatronics (ECMSM) IEEE: 1–6 (2015)
27. Rosten, E., Drummond, T.: FAST machine learning for high-speed corner detection. Eur. Conf. Comput. Vis. **2006**, 1–14 (2006)
28. Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: an efficient alternative to SIFT or SURF. IEEE 2012 international conference on computer vision, 58 (11), 2564–2571 (2012)
29. 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; 1–10. (2015)
30. Nurvitadhi, E., Sheffield. D., Sim. J.: Accelerating binarized neural networks: comparison of FPGA, CPU, GPU, and ASIC. International conference on field-programmable technology (2017)
31. Puglia, L., Vigliar, M., Raiconi, G.: Real-time low-power FPGA architecture for stereo vision. IEEE. Trans. Circuits Syst. II Exp. Briefs. **64**(1), 1307–1311 (2017)
32. Gil, A., Mozos, O.M., Ballesta, M., Reinoso, O.: A comparative evaluation of interest point detectors and local descriptors for visual SLAM. Mach. Vis. Appl. **21**(6), 905–920 (2010)
33. Rosten, E., Porter, R., Drummond, T.: Faster and better: a machine learning approach to corner detection. IEEE. Trans. Pattern. Anal. Mach. Intell. **32**(1), 105–119 (2008)
34. Zhu, J., Arbor, A., Hastie, T.: Multi-class AdaBoost. Stat. Inter. **2**, 349–360 (2006)
35. Leutenegger, S., Chli, M., Siegwart, R. Y.: BRISK: binary robust invariant scalable keypoints. IEEE 2011 international conference on computer vision, 58 (11), 2548–2555 (2011)
36. Calonder, M., Lepetit, V., Strecha, C., Fua, P.: BRIEF: binary robust independent elementary features. In proceedings of the European conference on computer vision (ECCV). (2010)
37. Krig, S.: Computer Vision Metrics: Survey, Taxonomy, and Analysis. Apress, Berkely, CA (2014)
38. Rosten, E., Drummond, T.: Machine learning for high-speed corner detection. 9th European conference on computer vision, Graz: 430–443 (2006)

Zhiyong Peng is a senior experimentalist in the College of Electronic Engineering and Automation at Guilin University of Electronic Technology (GUET),Guilin, China. And worked in Oklahoma State University as a visiting scholar in 2017-2018. He received his M.S. in Signal and Information Processing from GUET in 2006. He is currently pursuing the Ph.D. His research interests include computer vision, photogrammetry and high-performance parallel image processing.

Jun Wu is a Professor and Ph.D. supervisor in the College of Electronic Engineering and Automation at Guilin University of Electronic Technology, Guilin, China. He holds B.S. degrees and M.S. in Engineering both from Wuhan University, a Ph.D. in Photogrammetry and Remote Sensing from Wuhan University, and Postdoctoral research in Old Dominion University, USA. His research interests include robot vision, photogrammetry, in which areas he has published over 60 papers in both qualified journals and conferences.

Yongjun Zhang was born in 1975. He received the B.S. degree in geodesy, the M.S. in geodesy, and the Ph.D. degree in geomatics from Wuhan University, Wuhan, China, in 1997, 2000, and 2002, respectively. He is currently a Professor of photogrammetry and remote sensing with the School of Remote Sensing and Information Engineering, Wuhan University. His research interests include computer vision, space, aerial, and low-attitude photogrammetry, image matching, combined bundle adjustment with multi-source data sets, 3-D city reconstruction, and industrial inspection.

Xianhua Lin is a graduate student. He is studying in Guilin University of Electronic Technology at now, and currently pursuing about embedded system and image processing.