SIFT Hardware Implementation for Real-Time Image Feature Extraction

Jie Jiang, Xiaoyang Li, and Guangjun Zhang

***Abstract*— This paper introduces a high-speed all-hardware scale-invariant feature transform (SIFT) architecture with parallel and pipeline technology for real-time extraction of image features. The task-level parallel and pipeline structure are exploited between the hardware blocks, and the data-level parallel and pipeline architecture are exploited inside each block. Two identical random access memories are adopted with ping–pong operation to execute the key point detection module and the descriptor generation module in task-level parallelism. With speeding up the key point detection module of SIFT, the descriptor generation module has become the bottleneck of the system's performance; therefore, this paper proposes an optimized descriptor generation algorithm. A novel window-dividing method is proposed with square subregions arranged in 16 directions, and the descriptors are generated by reordering the histogram instead of window rotation. Therefore, the main orientation detection block and descriptor generation block run in parallel instead of interactively. With the optimized algorithm cooperating with pipeline structure inside each block, we not only improve the parallelism of the algorithm, but also avoid floating data calculation to save hardware consumption. Thus, the descriptor generation module leads the speed almost 15 times faster than a recent solution. The proposed system was implemented on field programmable gate array and the overall time to extract SIFT features for an image having 512 × 512 pixels is only 6.55 ms (sufficient for real-time applications), and the number of feature points can reach up to 2900.**

***Index Terms*— Feature extraction, field programmable gate array (FPGA), parallel and pipeline architecture, real time, scale-invariant feature transform (SIFT).**

I. Introduction

FEATURE detection is widely used in the computer vision field. One of the first widely used feature-detection algorithm is the Harris and Stephens [1] detector, which is derived from the Moravec [2] corner detection algorithm. It is capable of immunity and invariance to intensity and rotation changes. However, the Harris corner detector is susceptible to changes in image scale, which is subjected to sharp degeneration of performance when dealing with multiscale images. Several feature detectors have also been investigated. Among them, the scale-invariant feature transform (SIFT) algorithm proposed by Lowe [3] is one of the most robust approaches. Because the SIFT features are invariant to image transition, scaling, rotation, and partially invariant to changes of illumination and 3-D camera viewpoint, it is widely used in many applications, such as robot navigation [5], panoramic photographs [6], stereo matching [7], [8], and object detection and recognition [4]. Although many other affine-invariant algorithms have been proposed recently [9], [31], the SIFT feature is proved to be the most robust feature by Mikolajczyk and Schmid [10].

The SIFT algorithm mainly contains two parts, i.e., feature point detection and descriptor generation. It has to be mentioned that the nature computational complexity and huge demand of memory of SIFT prohibits its real-time applications (30 frames/s) in terms of pure software. To achieve real-time processing ability, strategies for speeding up are usually classified to the following categories: 1) parallel algorithm based on general symmetry multicore processors; 2) parallel algorithm based on customized multicore processors; and 3) parallel algorithm based on field programmable gate array (FPGA).

Zhang and Chen [11] introduced two parallel SIFT feature-extraction algorithms using general multicore processors, as well as some techniques to optimize the performance on multicore. The proposed architecture led to a 6.7× faster speed on a dual socket, quad-core system, which facilitated an average of 45 frames/s for a video graphic array (VGA) (640 × 480) video; some implementations and accelerations of SIFT feature extraction on graphics processing units were introduced by the papers of [12]–[14]. With the parallelism and powerful computational ability, they achieved high processing speed. However, the power consumption of these processors is enormous; consequently, they are not suitable for real-time embedded system applications.

Kim *et al.* [15] introduced an object recognition processor, which was integrated with ten processing units for task-level parallelism; and it contained a single instruction multiple data instruction to exploit the data-level parallelism. However, the system can only detect the feature point without generating the descriptor, and the processing time for a quarterVGA (QVGA) (320 × 240) video was 42 ms when the operation frequency was 200 MHz, which was not sufficient for real-time applications. Huang [24] proposed an all-hardware SIFT accelerator, which had two interactive hardware components, one for key point identification, and the other for feature descriptor generation. With a parallel architecture, including a three-staged pipeline, the processing time of the key point

Manuscript received June 26, 2013; revised September 16, 2013 and November 4, 2013; accepted January 17, 2014. Date of publication January 28, 2014; date of current version June 27, 2014. This work was supported in part by the National Natural Science Foundation of China under Grant 61222304 and in part by the Specialized Research Fund for the Doctoral Program of Higher Education under Grant 20121102110032. This paper was recommended by Associate Editor T.-S. Chang.

The authors are with the Key Laboratory for Precision Opto-Mechatronic Technology, Ministry of Education, Beihang University, Beijing 100191, China (e-mail: jiangjie@buaa.edu.cn; lxy890123@gmail.com; guangjunzhang@buaa.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSVT.2014.2302535

identification was only 3.4 ms for one VGA image. The feature descriptor generation part was implemented by hardware, which significantly improved the processing speed, in comparison with software solutions. However, a finite state machine (FSM) was used to regulate the descriptor generation procedure in hardware, and it was a serial processing schedule that approximately consumed 0.0331 ms for each feature. For a VGA image, the number of feature points is about 1800 (two scales per octave) [3], and thus the overall processing time is about 63 ms, which is not sufficient for real time (33 ms) applications.

Large number of researchers have successfully applied FPGA to speed up the SIFT algorithm. Bonato [21] proposed a hardware architecture of SIFT, which incorporated both hardware and software in a FPGA system for robot application of simultaneous localization and mapping. The system was able to detect features up to 30 frames/s for a QVGA video. However, minima points were skipped, which might not be suitable for other applications, in addition, the descriptor generation part took 11.7 ms for each feature point, to generate the descriptor, and this became the bottleneck of the overall system. Yao [22] proposed an architecture of optimized SIFT feature detection for an FPGA implementation of an image matcher. The feature-detection module took 31 ms to process a typical VGA image. However, the descriptor was generated by a soft-core MicroBlaze CPU, which was still the bottleneck of the entire system. Many other studies for speeding up SIFT hardware architectures [16], [17], based on FPGA tend to focus on the feature-detection part of SIFT. Although they were able to achieve high frame rate, they were not sufficient for real-time applications, if we take the descriptor generation part into consideration. Kim and Lee [20] proposed a new hardware organization to implement SIFT with less memory and hardware cost. Only a single Gaussian filter bank is shared by all octaves, and on average, it takes about 3017 Hz to generate a key point with its descriptor, so it takes much longer for hardware implementation and only 553 feature points/frame can be process for VGA (640×480) image at 30 frames/s. Chiu *et al.* [23] and Suzuki and Ikenaga [18], [19] used integral image in key point detection part, which not only decreasing computation complexity but also reducing DSP resources. However, the key point detection part and the descriptor generation part are running serially in these papers. Thus, with the increasing size of image, the key point detection part consumes more time, to reach 30 frames/s, the number of key points processed is decreased. Wang *et al.* [25] proposes a new FPGA based embedded system architecture for feature detection and matching. It consists of SIFT feature detection, BRIEF feature description and matching. It is able to establish accurate correspondences between consecutive frames for 720 p (1280×720) video, but the robustness to rotation and scale change of the proposed method is weak. Zhong *et al.* [26] presents a low-cost embedded system based on a new architecture that integrates FPGA and DSP.

Huang [24] analyzed the time consumption of each part of SIFT by running the SIFT algorithm on a 2.1 GHz Intel CPU and a soft-core of 100 MHz 32-bit NIOS II CPU, the convolution of Gaussian pyramid was identified to be the most time consuming part, using 72.85% and 85.63% of the total time on Intel CPU and NIOS II CPU, respectively. The second-time consuming part was the descriptor generation, which used 18.57% of time on an Intel CPU and 11.69% on a NOIS II CPU. As a result, most of the previous studies focused on improving the parallelism and reducing the running time of the feature-detection part of SIFT. If speeding up the first part, the descriptor generation, which is the second time consuming part of SIFT consumes much more time than the first part, and becomes the bottleneck of system performance, and thus, prohibits real-time applications.

Considering above, this paper proposes parallel–pipeline hardware architecture, which not only speeds up the key point detection part of SIFT, but also improves the parallelism, and thus, accelerates the descriptor generation part with proposed algorithm. With the in-depth understanding of SIFT algorithm and FPGA, task-level pipeline structure and task-level parallelism are exploited between the hardware blocks, in addition, the data-level pipeline architecture and the data-level parallelism are exploited inside each block.

The structure of this paper is as follows. Section II reviews the SIFT algorithm proposed by Lowe [3]. Section III presents the architecture of our SIFT hardware implementation. In Section IV, experimental results are used to demonstrate the performance of the system. The conclusion is given in Section V.

II. Description of the SIFT Algorithm

SIFT algorithm can be divided into two parts, one for key point detection including three stages, and the other for descriptor generation including two stages. Details can be seen as below.

A. Construction of the Difference of Gaussian Pyramid

The difference of Gaussian (DoG) detector is used to detect invariant features, and the first stage of the SIFT algorithm is used to construct the Gaussian pyramid. First, we convolve the given source image, denoted as $I(x, y)$, with a Gaussian function $G(x, y, \sigma)$, the result is a Gaussian-blurred image, denoted as $L(x, y, \sigma)$, as shown in (1). The DoG image, denoted as $D(x, y, \sigma)$, is defined as the difference between two adjacent Gaussian-blurred images, as shown in (2) as

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2}$$

where the Gaussian kernel with a scale of σ is shown as

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{x^2 + y^2}{2\sigma^2}\right], \sigma \geq 0. \tag{3}$$

The process to construct the DoG images is shown in Fig. 1. The Gaussian pyramid contains two octaves and four scales per octave. The input image has a resolution of 512×512 pixels; the second octave is desampled with reduced 256×256 pixels. The DoG space is built up, by simply applying the subtraction to two adjacent Gaussian scale images, pixel by pixel.

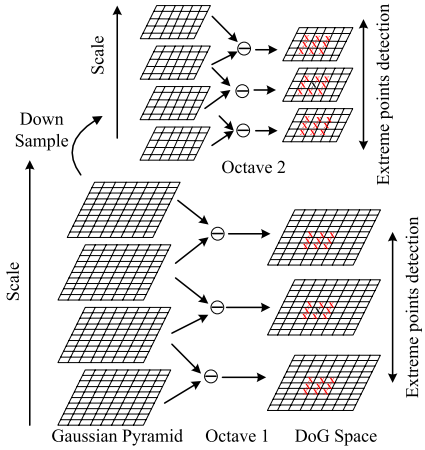


Fig. 1. Gaussian pyramid and the DoG pyramid, there are two octaves and four scales in each octave.

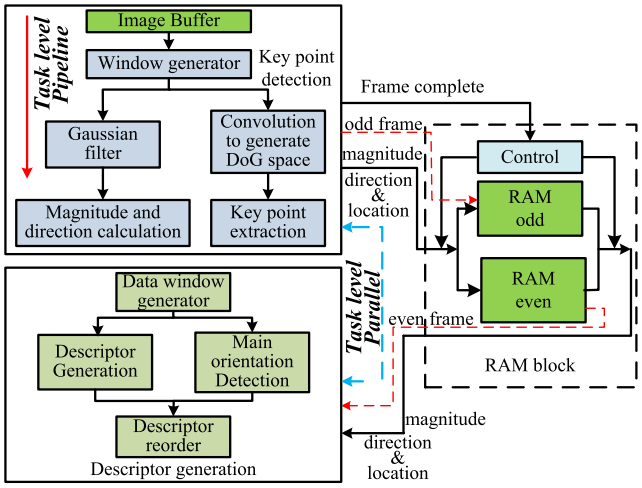


Fig. 2. Overall hardware architecture of the proposed SIFT algorithm.

B. Key Point Detection

The feature points are chosen from the local maxima or minima in the DoG space. Each pixel in the middle scale of DoG space is compared with its 26 neighbor pixels, where there are eight pixels in current scale image, and nine neighbors in the scale above and below, as shown in Fig. 1.

C. Calculation of Gradient Magnitude and Orientation

The magnitude-orientation histogram is used to describe a feature point, which is computed from the gradient magnitude and orientation of the neighbor pixels around the candidate feature point. For a given pixel (x, y) the gradient magnitude and orientation are computed as

$$m(x,y) = \sqrt{\big(L(x+1,y)-L(x-1,y)\big)^2+\big(L(x,y+1)-L(x,y-1)\big)^2} \tag{4}$$

$$\theta(x,y) = \tan^{-1}\left(\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y)}\right). \tag{5}$$

The orientation value ranging from 0° to 360° is normalized to be integer number ranging from 0 to 35.

D. Main Orientation of Feature Point

The gradient orientation of sample point around the key point region is used to create the orientation histogram. The orientation histogram has 36 bins, which cover the 360° range of orientations. Gradient magnitude of each sample, added to the histogram, is scaled by a Gaussian weighted kernel that is 1.5 times of the scale of the key point. Peaks in the orientation histogram correspond to dominant directions of local gradients.

E. SIFT Descriptor Generation

In this stage, a vector with rotation and illumination invariance is generated to describe the feature point. The SIFT descriptor generation part falls to following stages: First, a 16×16 window region around feature point is selected and rotated to the main orientation. Then, the direction of each pixel in this region is transferred to eight bins, in the main orientation, and the gradient magnitude is weighted by Gaussian kernel with a scale equal to one half of the width of the descriptor window. After that, the 16×16 window is divided into sixteen 4×4 subregions. Finally, an orientation histogram of eight bins is generated for each 4×4 subregions. Overall, the SIFT descriptor will be represented by a vector having $16 \times 8 = 128$ elements.

III. Proposed Hardware Architecture of SIFT

A. Overall System Architecture

The overall architecture of our proposed SIFT hardware implementation is introduced in the sections below in details. As mentioned in Section II, the SIFT algorithm mainly consists of two modules: the key point detection module and the descriptor generation module.

In this paper, two symmetric RAMs are adopted with ping–pong operation to run the key point detection module and descriptor generation module in task-level parallelism, as shown in Fig. 2. The RAM block consists of an even RAM and an odd RAM to buffer locations of the key point, the gradient magnitudes and the directions. The key point detection module processes the image in data stream manner continuously and the results of even frame and odd frame are saved to the even RAM and odd RAM, respectively. On the other hand, the descriptor generation module reads the result of the previous frame and generates a descriptor for each key point. With the parallel architecture, the processing speed of our system is determined by the longest time consumed by the two modules. In this paper, the RAM has an ability to save descriptors for up to 2900 key points, and the time consumed by descriptor module is less than the key point detection module. The procedures in key point detection module are processed with pipeline structure, and the descriptor generation module consists four blocks, running in pipeline–parallel structure, and they will be introduced in the following sections in detail.

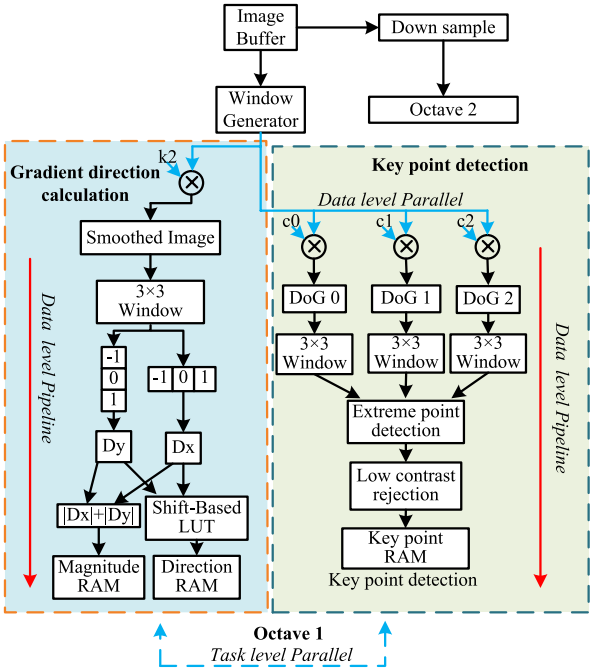


Fig. 3. Block diagram of key point detection and gradient and direction calculation module; k2 is the Gaussian kernel with scale of $\sigma 2$; c0, c1, and c2 are difference of adjacent Gaussian kernels $G(\sigma_{i+1}) - G(\sigma_i)$.

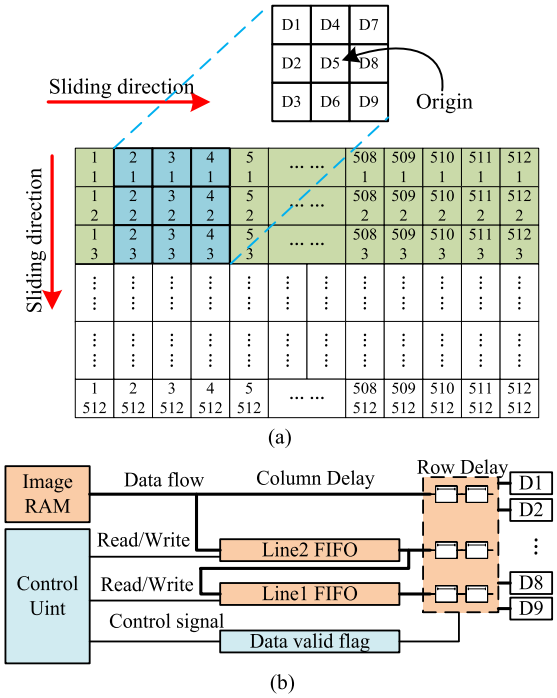


Fig. 4. (a) Window operation. (b) Hardware architecture of window generator with a size of 3 × 3.

B. Key Point Detection Module

Since (2) is equal to (6) shown below according to paper [3], the key point detection module can be divided into two main stages: 1) construction of DoG pyramid and key point extraction and 2) gradient magnitude and direction calculation. As shown in Fig. 3, these two procedures run in parallel because the data being processed between them is independent. In data-level parallelism, four convolutions (one for Gaussian smooth, and the other three for constructing DoG pyramid) within one octave, the gradient magnitude calculation, and direction calculation also run in parallel. The data-level pipeline is adopted in the progress of each block with data stream manner as

$$D(x, y, \sigma) = (G(x, y, \kappa\sigma) - G(x, y, \sigma)) * I(x, y). \quad (6)$$

1) Sliding Data Window: Windowing operators use a window, or neighborhood of pixels, to calculate their output, and it is widely used in image filtering, convolution and so on. This paper uses a 15 × 15 window for Gaussian filter, a 3 × 3 window for gradient magnitude and orientation calculation, Furthermore, a 3 × 3 × 3 window for key point detection. Fig. 4(a) shows a 3 × 3 window, which is scanned from left to right and top to bottom. Typically, the image data from a camera is shifted into sliding window serially in data stream manner. With a $n \times n$ sliding window operation, the throughput is significantly increased from one pixel per clock to $n \times n$ pixels per clock.

A shift register was suggested for implementing sliding window in [29], [30]. However, shift registers is normally realized by flip-flops, which consumes much more transistors than static random access memory (SRAM). Thus, Chiu *et al.* [23] use rotating SRAM banks to support sliding window operation. However, this structure is too complicated to operate with switch network and multibank SRAM.

In this paper, first in first out (FIFO) buffers are employed to perform sliding window operation. The FIFO performs like shifting registers from outside, where the data is shifted into the first FIFO buffer, in a data stream manner, while inside FIFO, it consists of a SRAM that can save chip area as mentioned in [23]. The hardware architecture of a 3 × 3 window generator is shown in Fig. 4(b). The image data stream from image sensor is one pixel per clock. Two FIFO buffers with length of image width are used and the data valid flag is used to tell other blocks that whether the window data is valid or not.

2) Magnitude and Orientation Calculation: To reduce the computational complexity, the square root operation of (4) is replaced by the absolute operation, and so we have

$$\begin{aligned} \text{grad} = &|L(x+1, y) - L(x-1, y)| \\ &+ |L(x, y+1) - L(x, y-1)|. \end{aligned} \quad (7)$$

We use matching accuracy to compare the performances of these two operations. The matching accuracy is defined as

$$P(\text{Matching\_Accuracy}) = \frac{N(\text{Correct\_matches})}{N(\text{Correct\_matches}) + N(\text{False\_matches})}. \quad (8)$$

Twenty-five sets of image samples with different scenes have been used for experiments, one set of the image samples include 16 samples, as shown in Fig. 5, which has five degree rotation transform within each sample. The matching results are shown in Fig. 6. The experimental results show that the

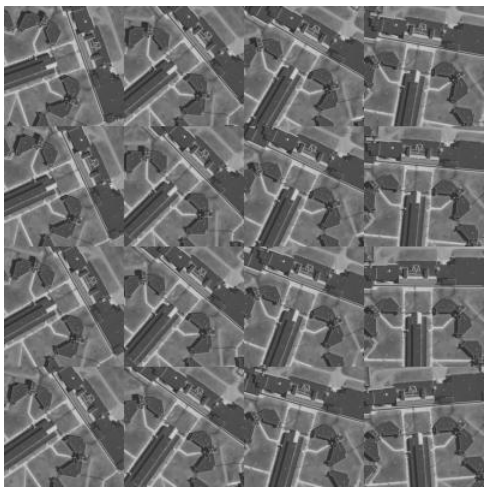


Fig. 5. Image samples with rotation transform to test the matching accuracy of absolute operation versus to square root operation.

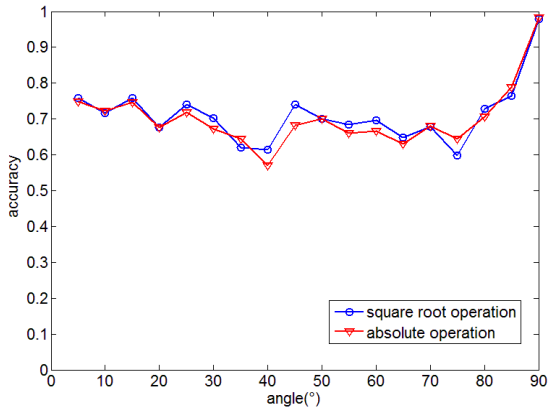


Fig. 6. Comparison of matching accuracy between square root result and the absolute operation result.

absolute operation leads to almost the same matching accuracy with the square root operation. As a result, in this paper, we use the absolute operation instead of square root operation to calculate the gradient magnitude with negligible performance degeneration.

A simplified look-up table (LUT) is used to calculate the 16 bins direction based on shifting algorithm instead of (5), and an internal memory unit is built up to store the gradient and orientation. The gradient magnitude is normalized to 8 bits, and the direction is represented in 4 bits. Thus, a 12-bit RAM is used to represent gradient and direction of each pixel (with gradient in high 8 bits).

C. Descriptor Generation Module

The descriptor generation module is the second time consuming part of the entire system, as mentioned in Section I. after speeding up the key point detection part; this module would become the bottleneck of the system performance. Therefore, in this section, a hardware architecture is designed to speed up the descriptor generation part of SIFT.

Typically, a descriptor with 128 elements is generated for each key point region in original algorithm proposed by Lowe.

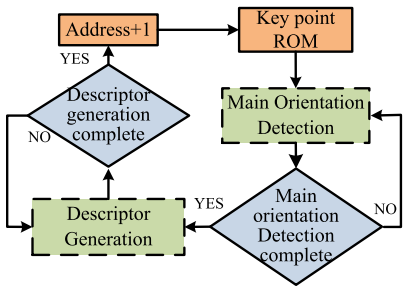


Fig. 7. State transition diagram of traditional descriptor generation algorithm.

As shown in Fig. 7, the descriptor generation module is divided into two blocks: 1) main orientation detection block and 2) descriptor generation block. The second block works only after the first block finishes main orientation detection, thus the two blocks are configured to work interactively. Every time when the main orientation is calculated, it evokes the second block to start generating the descriptor, and after the second block has completed descriptor generation for one key point, the first block resumes and calculates the main orientation of next key point. The interaction continues until the descriptors of all key points have been generated.

However, the algorithm described above is heavily time consuming, because of the cascade structure. Chiu *et al.* [23] propose a hardware solution and the processing speed is faster than any existing software solutions. However, the procedures in this module are regulated by FSM, and the two main blocks also run interactively, therefore, the parallelism is limited. To improve the parallelism of the system, we proposed a new algorithm to generate the SIFT descriptor with parallel structures, which are introduced as follows.

1) Algorithm: The optimized algorithm to generate the descriptor is shown in Fig. 8. In the main orientation generation block, firstly, a region with 15×15 pixels is selected (instead of 16×16 window in the original algorithm). Then, the gradient is weighted by a Gaussian kernel, and a 16-bin (corresponding to the 16 directions) magnitude-orientation histogram is built according to the direction of each pixel. Finally, the maximum bin of the histogram is chosen as the main orientation of the key point.

In the descriptor generation block, to facilitate rotation-invariant, Lowe rotated the window to the main orientation. Coordinates of the descriptor region are calculated by (9) and (10), which require floating data calculation. However, the floating calculation is difficult for FPGA to be implemented at the expense of lots of hardware resource as

$$x' = x\cos\theta - y\sin\theta \tag{9}$$

$$y' = x\sin\theta + x\cos\theta. \tag{10}$$

To achieve rotation invariance and avoid window rotation operation, Yao [22] divided the descriptor region into nine circular subregions with a polar arrangement. However, the circular subregion is inconvenient for FPGA to be implemented, because the coordinates in circular with polar arrangement are also floating data calculation. In addition, the algorithm only applies eight directions and the matching accuracy is much

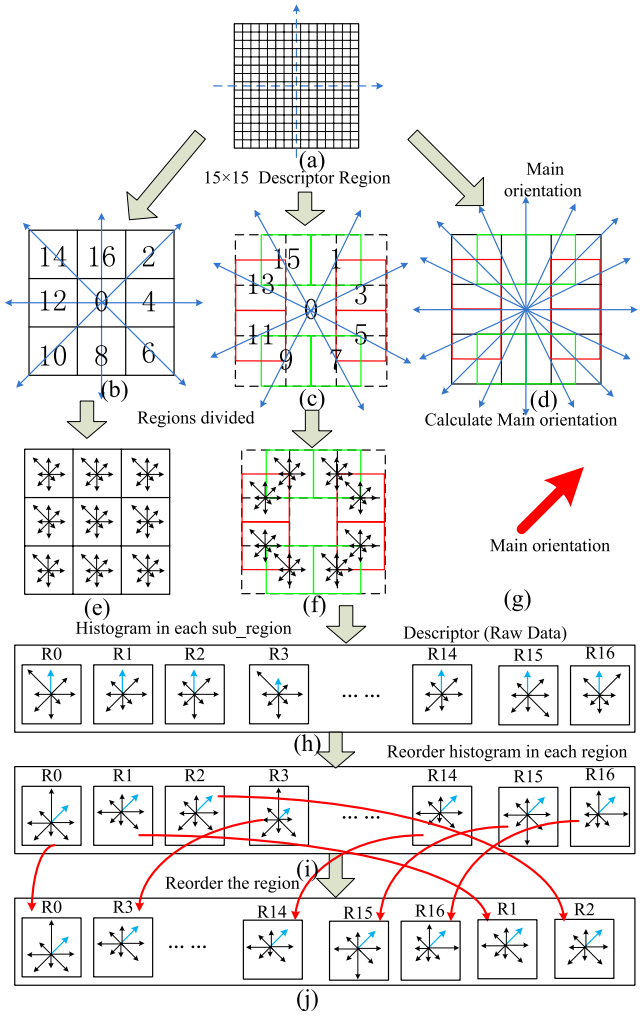


Fig. 8. Optimized algorithm to generate descriptor. (a) Descriptor region. (b) Subregions in black are in the direction of 22.5° ×n (even). (c) Subregions in red and green are in the direction of 22.5° ×n (odd). (d) 16 directions. (e) and (f) Generating the histogram in each subregions. (g) Main orientation detection. (h) Raw data of descriptor. (i) Reorder histogram in each subregion according to the main orientation. (j) Reorder the regions according to the main orientation.

worse than the original algorithm. From our experiments, this method failed to match the images with rotation angle bigger than 30°, or scale transform ratio bigger than 1.2.

Therefore, a novel window-dividing method is proposed to generate the descriptor, as shown in Fig. 8(a)–(d). The descriptor region is the same as the window, to calculate the main orientation. The data window is divided into 17 square subregions. One of those lies in the center of the window without any overlap, and the other 16 subregions lie in 16 directions. Each subregion covers 45° of the window, and the overlap is designed to improve the robustness of the algorithm.

In our design, since 17 subregions are divided into 16 directions, the descriptors are generated by reordering the subregions and the histograms of each subregion, instead of the window rotation. The histogram of each subregion can be built without main orientation, thus, the main orientation detection block and the descriptor generation block run in parallel. Finally, the histogram in each subregion and the regions are

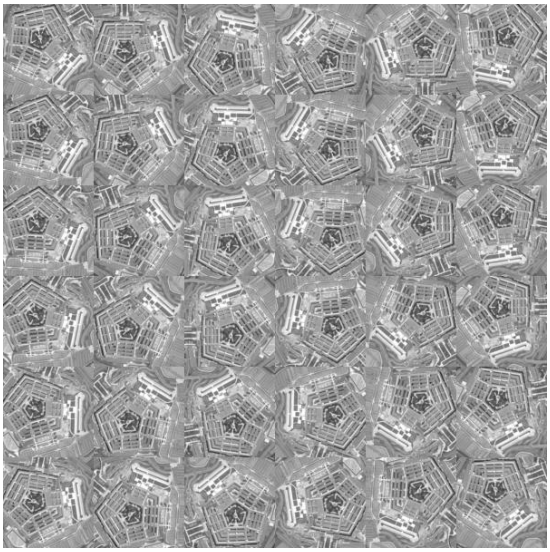


Fig. 9. Image samples with rotation transform to test matching accuracy of optimized algorithm versus original algorithm.

reordered in the main orientation, as shown in Fig. 8(i) and (j). In our design, the descriptor has 17 subregions that generate 136 elements.

As introduced above, the algorithm can be summarized as follows.

1) Main orientation detection:
   a) smoothing the gradient with Gaussian kernel (δ_1);
   b) building the histogram with 16 bins;
   c) detecting the main orientation, this is the maximum bin of the histogram.
2) Generating the descriptors in each subregion:
   a) smoothing the gradient with Gaussian kernel (δ_2);
   b) dividing the window into 17 subregions (5 × 5 pixels in each subregion);
   c) calculating the histogram in each subregion.
3) Reordering the histogram:
   a) reordering the histograms in each subregion to the main orientation;
   b) reordering the regions in the main orientation, and then assembling them into the final descriptor.

Fifteen sets of image samples with different scenes have been used to verify the proposed algorithm. One set of test images include 36 samples with rotation transform, as shown in Fig. 9. The matching accuracy is shown in Fig. 10. In addition, the results show that our algorithm has achieved similar matching accuracy with the original algorithm.

2) Overall Hardware Architecture: In the following sections, we will introduce the hardware architecture of the optimized algorithm in details. First, the overall architecture is shown in Fig. 11. The descriptor generation module is divided into four blocks: 1) data window generator block; 2) main orientation detection; 3) generation of the histogram in each subregion; and 4) histograms and regions reordering. The main orientation block and the histogram generation block run in parallel, therefore, the parallelism of this algorithm is greatly improved. The task-level pipeline is adopted to execute

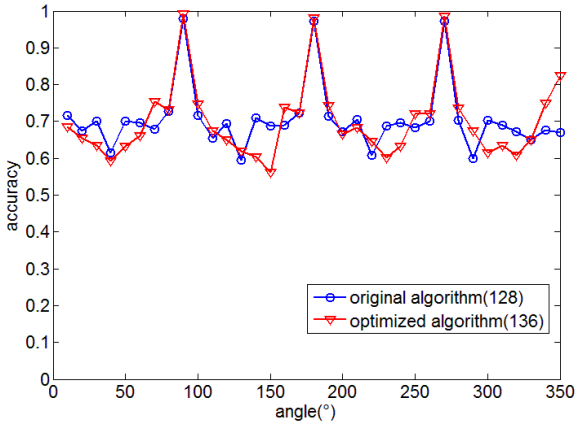


Fig. 10. Comparison of matching accuracy between our optimized algorithm versus original algorithm.

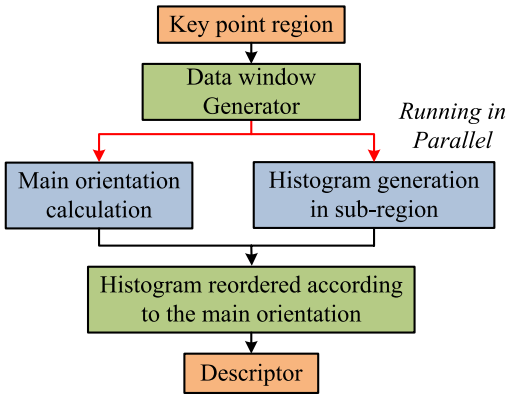


Fig. 11. Structure of our optimized algorithm, with main orientation detection block and histogram generation block running in parallel.

the substages inside these four blocks; in addition, the data-level pipeline is used to process the data stream inside each substage, and the data stream through these four blocks. With the pipeline structure, the data generated by each operation is transferred to the next operation without storage, thus we greatly reduce the RAM consumption.

3) Hardware Architecture of the Data Window Generator: The descriptors are generated by the 15 × 15 data window around the key point. The hardware architecture of this block is shown in Fig. 12. Window address agent works as a counter and generates the address of the window. Then, the address is used to calculate the RAM address of the data window, according to the location of the key point. After that, the gradient magnitude and the direction is obtained and sent to the main orientation detection block and histogram generation block, simultaneously to generate the magnitude-orientation histogram. The window address is also sent to these two blocks to provide the Gaussian weight; and in the histogram generation block, the window address is also used for dividing subregions.

4) Hardware Architecture of the Main Orientation Detection: The main orientation detection block is divided into three substages by the different functions that they perform as follows: 1) Gaussian weighted magnitude calculation, defined as WM1; 2) building the 16-bin magnitude-orientation

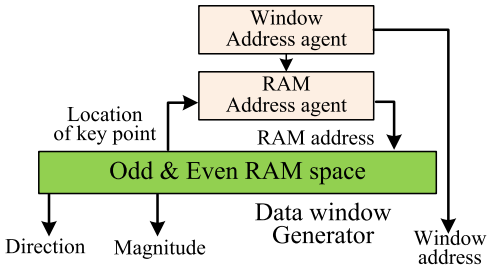


Fig. 12. Data window generator, where the gradient magnitude and direction are read out, the region is with 15 × 15 pixels around the key point.

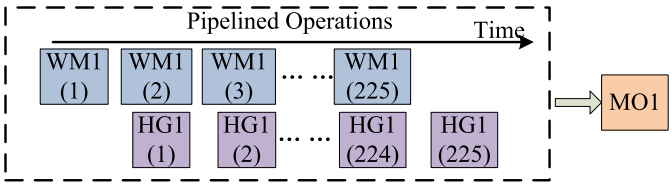


Fig. 13. Processing architecture of main orientation detection block.

histogram, defined as HG1; and 3) detecting the main orientation, defined as MO1. Since there is not a parallelism between these substages, they cannot be split or processed in parallel. However, the data processed in first two substages is independent to the data in the previous clock or the data in the next clock, so, it has no need to wait until all the data inside the window of previous stage are processed, thus they can be done in stream processing manner, instead of block processing via adopting task-level pipeline structure. Fig. 13 shows the two stages of the task-level pipeline processing architecture. The main orientation detection block starts to process after the construction of magnitude-orientation histogram is completed, thus it cannot be implemented in pipeline structure. In this substage, a serial of comparators are configured to detect the main orientation with data-level parallel and pipeline structure, which lead to a magnificent improvement of processing speed. With the 16-bin of histogram, this process can be done in four clock cycles. The main orientation is detected and transferred to the reordering block of the histograms and regions to achieve rotation invariance.

The implementation hardware architecture of main orientation detection block is shown in Fig. 14. The data stream indicates that the data is transferred and updated in every clock cycle. The magnitude and direction are obtained from the RAM and the corresponding weight is accessed from the Gaussian ROM, simultaneously controlled by the window address. Then, they are sent to the multiplier to generate the weighted magnitude. After that, the magnitude-orientation histogram is built up via accumulating the weighted magnitude to the particular bin in the direction. With the pipeline structure, these procedures run simultaneously in each clock cycle and the data is transferred to the next stage. Finally, parallel and pipeline structures are adopted to speed up the detection of the main orientation, which appears to be the maximum bin in the histogram.

5) Hardware Architecture of the Histogram Generation: The histogram generation block is divided into three substages as follows: 1) Gaussian weighted magnitude calculation,

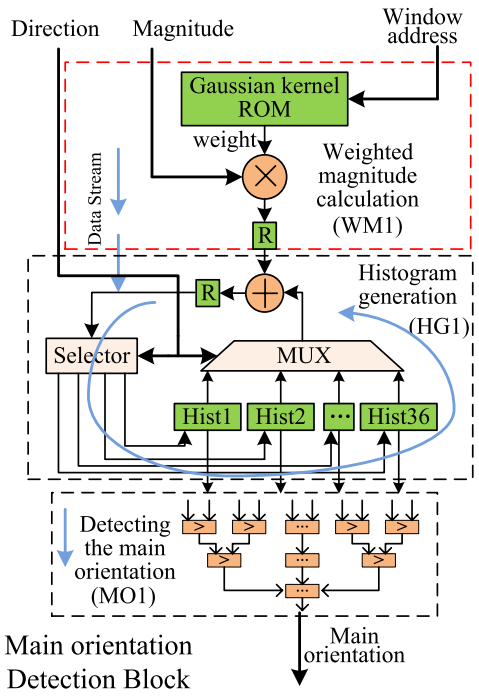


Fig. 14. Hardware implementation architecture of the main orientation detection block.

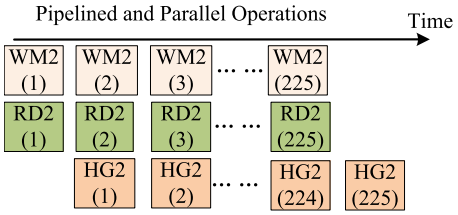


Fig. 15. Processing architecture of the descriptor generation block.

defined as WM2; 2) region dividing, defined as RD2; and 3) generating the histogram in each subregion, defined as HG2. These procedures are almost the same as those on the main orientation detection block, except the RD2. In RD2, the data window is divided into 17 square subregions according to the window address. Then, the label of sub-region generates 8-bit address (Sel [7:0]) to control the MUX and the Selector in cooperation with 8-bin of direction. Since there is no data dependence between WM2 and RD2, they run in parallel, and the task-level pipeline and parallel structures are adopted to regulate these three procedures, as shown in Fig. 15.

The implementation hardware architecture of a histogram generation block is shown in Fig. 16. With in comparison with the main orientation detection block, homologous architecture is adopted to regulate the procedures. However, there are some more procedures to be performed in this block. To achieve rotation invariance, the data window is divided into 17 subregions in 16 directions with overlap. The region dividing procedure controls the weighted magnitude to be accumulated to the particular bin in the histogram in cooperation with the 8-bin directions. Because of the overlap between subregions, most of the data belongs to two subregions, thus there are two adders performing parallel operations in this block. Within the task-level and data-level pipeline structure, the result of each substage is updated at each clock cycle.

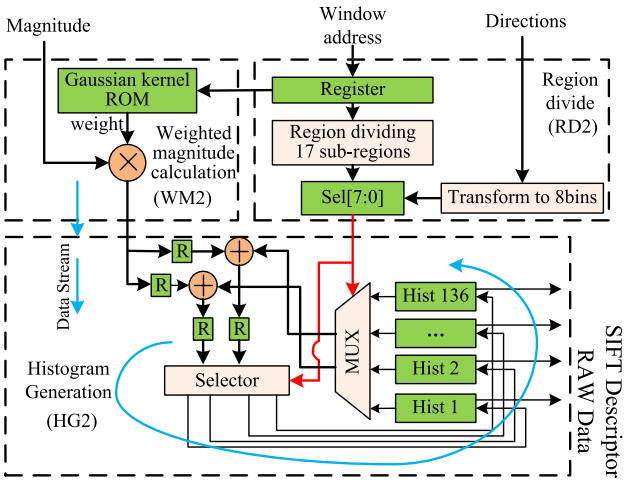


Fig. 16. Hardware implementation architecture of the descriptor generation block.

6) Generation of the Final Descriptor: In this substage, main orientation and the histogram of each subregion are generated. There are still two more stages to be performed for achieving rotation invariance. The first stage is to reorder the histogram in each subregion in the main orientation, as shown in Fig. 8(i). Then, in second stage, the sub-regions are reordered according to 16 bins of direction, as shown in Fig. 8(j). In this block, the descriptors are only reordered according to the main orientation without any other mathematic operation, thus the procedures in these two stages fit the hardware very well.

In our system, the gradients and directions in the 15×15 window are only read out once, and they are both sent to the main orientation detection block and histogram generation block simultaneously, as they run in parallel. With the optimized architecture as proposed, the overall processing time to generate descriptor for each key point is 225 clock cycles plus a few cycles of pipeline delay. We have achieved almost $15\times$ faster speed compared with recent work in [24], with the overall processing time of 0.0023 ms for each key point to generate descriptor, when the operating frequency is 100 MHz.

IV. Test and Verification

In this section, experimental results are analyzed in three aspects. First, a serial of matching experiments are conducted to verify the reliability of the SIFT feature in our design. Then, consumption of hardware resources is evaluated. Finally, the overall time consumption, which is the most important character of our design, is given in details.

A. Reliability

The SIFT features mainly lie in the second level of DoG space as mentioned in [27]. Thus, the SIFT algorithm is configured with two octaves and four scales for each octave in this paper. The initial scale of the Gaussian kernel is $\sigma = 1.1$ for both octaves, and σ of 1.3, 1.6, and 2.0 are the other three scales of both octaves [22]. Assuming the pixel value is in the range of [0, 1], threshold of the low contrast rejection is 0.03. Size of the input image is 512×512.

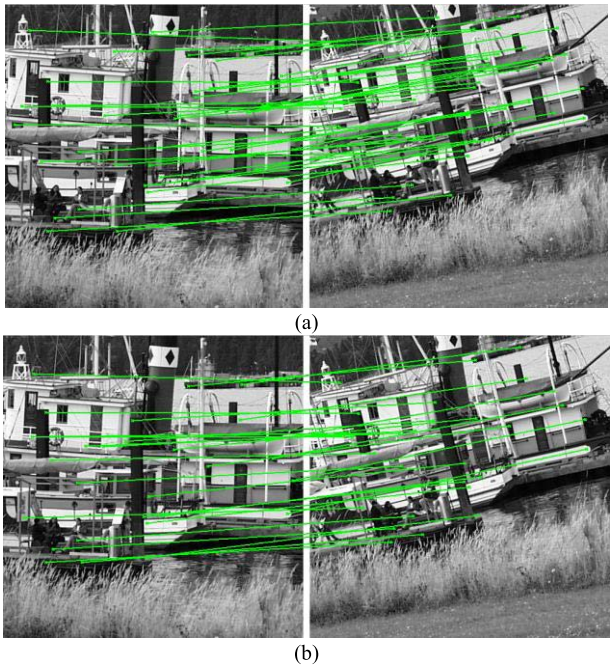


Fig. 17. Matching results of the affine transform. (a) Matching result of original algorithm. (b) Matching result of our proposed algorithm.

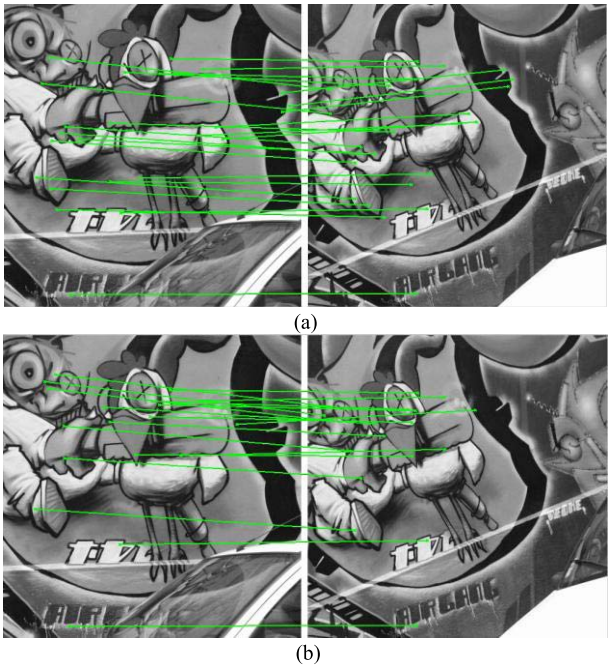


Fig. 18. Matching results of the camera transition. (a) Matching result of the original algorithm. (b) Matching result of our proposed algorithm.

In this paper, different bit widths of registers are used to represent a pixel in different processes to reduce the consumption of redundant registers. Considering the result of convolution module (consist of one Gaussian filter and three DoG filter) is the source of other processes, 18-bit width registers (with eight integer bits and ten fractional bits) are used to represent each pixel in this procedure, and it has an error of 0.04%, as compared to software implementation. In DoG space, eight integer bit registers are used to eliminate the low contrast point [22]. In addition, we use eight integer bits and six integer bits to represent the magnitude and direction to reduce the RAM consumption with an error of only 0.19% and 0.85%, respectively.

Matching accuracy is used to verify the system reliability, SIFT features are calculated by original algorithm and our proposed algorithm, respectively. The key points are matched based on a Euclidean distance between their descriptors as

$$d_{\text{euc}} = \sqrt{\sum_{i=1}^{136} (\text{desc}_1(i) - \text{desc}_2(i))} \tag{11}$$

$$k = \frac{d_{\text{euc}}(\min)}{d_{\text{euc}}(\max)}. \tag{12}$$

d_{euc} is the distance of Euclidean, desc_1, desc_2 are two descriptors for matching, and k is the ratio between minimum distance and maximum distance among all points. If the k is less than threshold (in our paper, $k = 0.7$), then we define this set of descriptors are matched.

Then, the correct matching points are detected based on the random sample consensus (RANSAC) algorithm. The images under test are from benchmark data set of Mikolajczyk and Schmid [31], the matching results are shown in Figs. 17–21. Each line between two images indicates a pair of corresponding feature points. The matching accuracies are shown in Table I. Although the correct match points and match points of the proposed algorithm are less than the original algorithm for some images, as shown in Table I, but the proposed algorithm has higher matching accuracies comparing with the original algorithm.

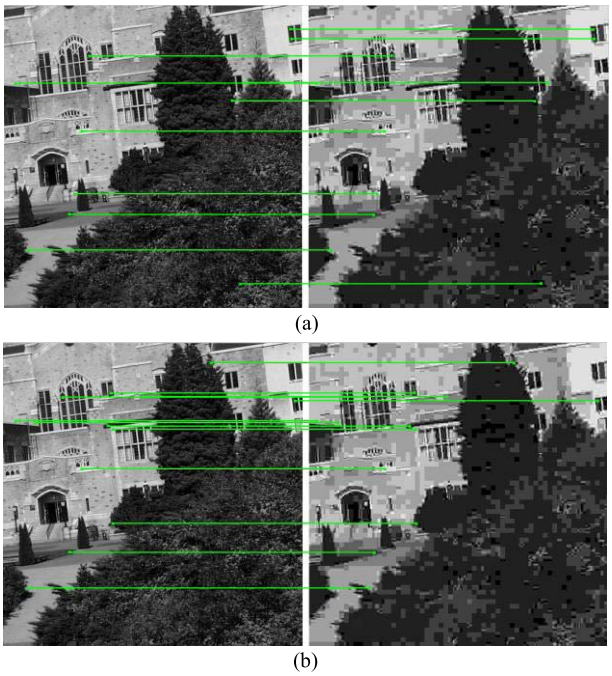


Fig. 19. Matching results of JPEG compression. (a) Matching result of original algorithm. (b) Matching result of our proposed algorithm.

B. Hardware Consumption

Our proposed architecture is implemented on Xilinx Virtex-5 LX330, the prototype is shown in the Fig. 22.

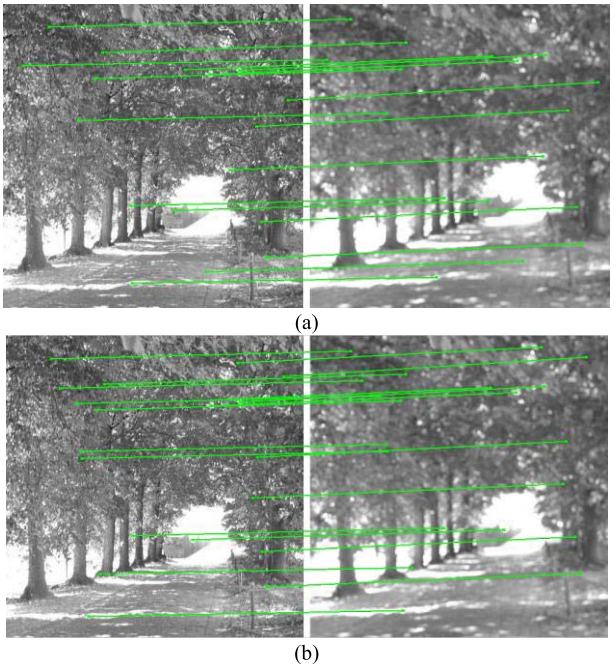


Fig. 20. Matching results of image blur. (a) Matching result of the original algorithm. (b) Matching result of our proposed algorithm.

Fig. 21. Matching results of illumination changes. (a) Matching result of the original algorithm. (b) Matching result of our proposed algorithm.

The whole system is implemented in Verilog HDL. Considering our design is more complete and the development platform is different, in comparison with [21]–[24], it is difficult to compare hardware resource usages between them. Some important aspects are listed in Table II. Although our system is more complete than in [21] and [22], it is obvious from the table that our system reduces 46% of registers consumption and 35.9% of LUTs consumption. Because of using the integral image, the usage of DSP block in [18] and [23] are reduced, but the consumption of the registers and LUTs are increased. In our proposed architecture, power consumption is 3.9 W.

TABLE I
COMPARISON OF MATCHING ACCURACY BETWEEN OUR PROPOSED ALGORITHM AND THE ORIGINAL ALGORITHM

			Correct Match Point	Match point	Accuracy
Affine transform	Boat	Original	46	121	0.38
		Ours	33	82	0.40
Camera transition	Graf	Original	29	94	0.31
		Ours	25	67	0.37
JPEG compression	UBC	Original	14	54	0.26
		Ours	17	41	0.41
Image Blur	Trees	Original	23	71	0.32
		Ours	25	59	0.42
Illumination change	Leuven	Original	55	90	0.61
		Ours	66	96	0.69

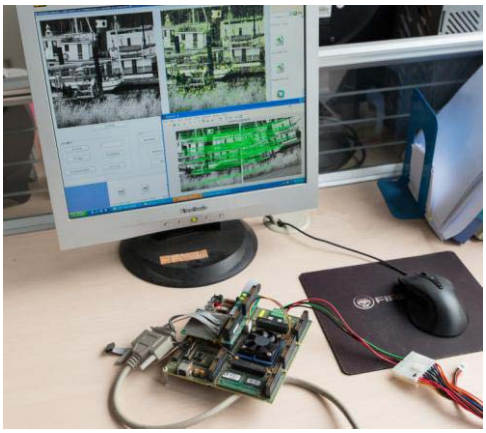


Fig. 22. Picture of prototype.

The memory usage is described in Table III. The on-chip memory is mainly divided into two parts corresponding to the two different modules: 1) buffers for key point detection (i.e., the FIFO for window generator and memory to buffer the gradient magnitude, orientation, and location of the key point) and 2) buffers for descriptor generation.

As can be seen from Table III, the memory usage of our system is less than the investigations in [21] and [22]. If our implementation is configured with ping–pong structure, it uses more memory than the work in [24], but achieves much higher frame rate than the previous studies in [21]–[24], because two identical memories (odd RAM and even RAM to buffer the magnitude, orientation and location of key point of odd frame and even frame, respectively) are adopted to execute the key point detection module and descriptor generation module in parallel. Our proposed hardware structure can also be configured to run in cascade. In this case, our design uses less memory than the work in [24], but achieves three times faster processing speed. Due to the layer parallel scheme and related hardware implementation techniques, the memory usage of Chiu *et al.* [23] is only 0.5 Mbit.

TABLE II
USAGE OF HARDWARE RESOURCES (COMPARED WITH TWO RECENT WORKS BASED ON FPGA)

	Bonato [21]	Yao [22]	T.Suzuki [18]	Chiu[23]	Ours [1]
DSP blocks	64	97	3	8	89 [2]
LUTs	43,366	35,889	108322	57598	26,398 [3]
Registers	19,100	19,529	55470	24988	10,310

[1] Our design contains both the key point detection module part and the descriptor generation module;

[2] 87 DSP blocks are used in key point detection module and 2 in descriptor generation module.

[3] Number of Slice LUTS 12821; Number of Slice LUT-Flip Flop pairs 13577

TABLE III
MEMORY USAGE

		Bonato	Yao	Huang	Ours
Image size		320×240	640×480	640×480	512×512
Memory Usage	DoG	1.35 Mb	3.24 Mb	0.224 Mb	57.3 Kb
	Key point Detection			0.404 Mb	51.6 Kb
	Magnitude & orientation			0.241 Mb	7.7 (3.85) Mb [1]
	Descriptor Generation	16 Mb	8.19 Mb	4.86 Mb	4.5 Kb
Total memory usage		17.35 Mb	11.43 Mb	5.73 Mb	7.8 [2] (3.95)[3] Mb

[1] Including memory used to store the magnitude and orientation

[2] Ping-Pong structure, with key point detection and feature calculation in parallel.

[3] Without Ping-Pong structure, key point detection and feature calculation are running in cascade manner.

C. Time Consumption

The timing constraint after place and route for the input clock frequency is 79.4 MHz, in this paper, we adopt a 50 MHz clock, and the 100 MHz clock for the descriptor generation part is generated from digital clock managers inside the chip [28] provided by Xilinx. As mentioned in Section III, the processing time is divided into two parts corresponding to the two main modules: 1) key point detection module and 2) descriptor generation module. The processing time of first module T_1 is 6.55 ms for an input image with 512×512 pixels under an operating frequency of 50 MHz, and it is directly proportional to the size of the input image. The processing time of second module T_2 is expressed as

$$T_2 = \frac{N_{kp} \times (225 + \text{pipeline\_delay})}{100000} = N_{kp} \times 0.0023 \text{ ms} \quad (13)$$

where N_{kp} represents the number of key points. The time, used to generate descriptor for each key point, is under 0.0023 ms with an operating frequency of 100 MHz, and the overall time of the second module is under 6.5 ms when the number of key points is less than 2900. With the parallel architecture of the two modules introduced in Section III, the processing time of the whole system is determined by the longest time consumed by these two blocks of 6.55 ms. Because of the limitation of hardware resources (block RAM) in Virtex-5 LX330, the Full HD images are simulated with ModelSim under 70 MHz. The result shows that the processing time of our proposed architecture is 29.6 ms and the number of key points can reach to 15 000. Since the systems proposed in [21]–[24] are more complete and recently developed; we compare our work with them, as shown in Table IV. We list several important aspects of a SIFT hardware system. In key point detection part, our system and the work in [24] achieve much faster processing speed than the work in [21] and [22]. In the descriptor generation part, our design has a 15× faster speed than the work in [24] and 2× faster than the work in [23] for the descriptor generation of each key point. As can be seen from the overall time consumption, our system has a great improvement in processing speed than others.

TABLE IV
PERFORMANCE COMPARISON

	Bonato	Yao	Huang	Chiu	Ours
Image size	320×240	640×480	640×480	640×480	512×512
Structure(Octave, Scale)	3,6	2,4	3,6	2,4	2,4
Key point detection time(ms)	33	31	3.4	No present	6.55
Descriptor implementation method	NOIS II	Micro Blaze	Hardware	Hardware	Hardware
Descriptor generation time(ms/key point)	11.7	No present	0.0331	0.0052	0.00223
Frequency of key point detection	50MHz	100MHz	100MHz	100MHz	50MHz
Frequency of descriptor part	100MHz	No present	100MHz	100MHz	100MHz
Implementation platform	STRATIX II	Virtex-5	TSMC CMOS	90nm ASIC	Virtex-5
Overall time consumption (ms)	No present	No present	33ms 890 key points	33ms ~6000 key points	6.55ms [1] (11.15ms) [2] ~2900 key points

[1] System is configured with Ping-Pong structure, two modules are running in parallel.

[2] System without Ping-Pong structure, two modules are running in cascade manner.

V. CONCLUSION

The SIFT algorithm gains its reputation in computer vision because of its scale and rotation-invariant characteristics. In this paper, we have implemented an optimized all-hardware SIFT algorithm on FPGA with parallel and pipeline architecture. The system mainly contains two individual processing

modules that work in task-level parallel by adopting two identical RAMs with ping–pong operation. A new algorithm for descriptor generation is proposed with square subregions arranged in 16 directions to achieve rotation invariance, thus we can not only improve the parallelism of the algorithm, but also avoid floating calculation to save hardware resource consumption. With pipeline architecture to implement the descriptor generation module, the system achieves nearly 15× higher processing speed than a recently developed solution; as a result, it magnificently reduces processing time of the entire system. With the optimized hardware architecture implemented on FPGA, We cannot only reduce nearly 46% of hardware resources consumption, but also achieve a very high frame rate of 152 frames/s (6.55 ms per frame), which is sufficient for real-time processing for video with 512 × 512 pixels. The number of feature points can reach up to 2900. The accuracy is almost the same as a PC-based implementation.

REFERENCES

[1] C. Harris and M. J. Stephens, "A combined corner and edge detector," in *Proc. Avley Vis. Conf.*, Manchester, U.K., 1988, pp. 147–152.

[2] H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proc. 5th Int. Joint Conf. Artif. Intell.*, vol. 2. 1977, p. 584.

[3] D. G. Lowe, "Distinctive image features from scale-invariant key points," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Jan. 2004.

[4] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Int. Conf. Comput. Vis.*, vol. 2. Sep. 1999, pp. 1150–1157.

[5] S. Se, D. G. Lowe, and J. J. Little, "Vision-based global localization and mapping for mobile robots," *IEEE Trans. Robot.*, vol. 3, no. 21, pp. 364–375, Jun. 2005.

[6] M. Brown and D. G. Lowe, "Recognizing panoramas," in *Proc. 9th IEEE Int. Conf. Comput. Vis.*, vol. 2. Oct. 2003, pp. 1218–1225.

[7] D. G. Lowe, "Local feature view clustering for 3D object recognition," in *Proc. IEEE Comput. Soc. Conf. CVPR*, vol. 1. Dec. 2001, pp. 682–688.

[8] N. Pettersson, "Online stereo calibration using FPGAs," in *Proc. IEEE Intell. Veh. Symp.*, Jun. 2005, pp. 55–60.

[9] T. Lindeberg and J. Gårding, "Shape-adapted smoothing in estimation of 3-D shape cues from affine deformations of local 2-D brightness structure," *Image Vis. Comput.*, vol. 15, no. 6, pp. 415–434, Jun. 1997.

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

[11] Q. Zhang and Y. R. Chen, "SIFT implementation and optimization for multi-core systems," in *Proc. IEEE IPDPS*, Apr. 2008, pp. 1–8.

[12] S. N. Sinha and J. M. Frahm, "Feature tracking and matching in video using programmable graphics hardware," *Mach. Vis. Appl.*, vol. 22, no. 1, pp. 207–217, Jan. 2011.

[13] K. Heymann and K. Muller, "SIFT implementation and optimization for general-purpose GPU," in *Proc. WSCG*, 2007, pp. 317–322.

[14] N. Cornells and L. V. Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *Proc. IEEE Comput. Soc. Conf. CVPRW*, Jun. 2008, pp. 1–8.

[15] D. Kim, K. Kim, J. Y. Kim, S. Lee, S. J. Lee, and H. J. Yoo, "81.6 GOPS object recognition processor based on a memory-centric NoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 370–383, Mar. 2009.

[16] L. Chang and J. Hernández-Palancar, "A hardware architecture for SIFT candidate keypoints detection," in *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. New York, NY, USA: Springer-Verlag, 2009, pp. 95–102.

[17] K. Mizuno, H. Noguchi, and G. L. He, "Fast and low-memory-bandwidth architecture of SIFT descriptor generation with scalability on speed and accuracy for VGA video," in *Proc. Int. Conf. FPLA*, Sep. 2010, pp. 608–611.

[18] T. Suzuki and T. Ikenaga, "SIFT-based low complexity keypoint extraction and its real-time hardware implementation for full-HD video," in *Proc. APSIPA ASC*, Dec. 2012, pp. 1–6.

[19] T. Suzuki and T. Ikenaga, "Low complexity keypoint extraction based on SIFT descriptor and its hardware implementation for full-HD 60 fps video," *IEICE Trans. Electron.*, vol. 96, no. 6, pp. 1376–1383, Jun. 2013.

[20] E. S. Kim and H. J. Lee, "A novel hardware design for SIFT generation with reduced memory requirement," *J. Semicond. Technol. Sci.*, vol. 13, no. 2, pp. 157–169, Apr. 2013.

[21] V. Bonato, "A parallel hardware architecture for scale and rotation invariant feature detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 12, pp. 1703–1712, Dec. 2008.

[22] L. F. Yao, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," in *Proc. Int. Conf. FPT*, Dec. 2009, pp. 30–37.

[23] L.-C. Chiu, T.-S. Chang, J.-Y. Chen, and N. Y.-C. Chang, "Fast SIFT design for real-time visual feature extraction," *IEEE Trans. Image Process.*, vol. 22, no. 8, pp. 3158–3166, Aug. 2013.

[24] F. C. Huang, "High-performance SIFT hardware accelerator for real-time image feature extraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 3, pp. 340–351, Mar. 2012.

[25] J. Wang, Z. Sheng, L. Yan, and Z. Cao, "An embedded system-on-a-chip architecture for real-time visual detection and matching," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 3, pp. 525–538, Mar. 2014.

[26] S. Zhong, J. Wang, L. Yan, L. Kang, and Z. Cao, "A real-time embedded architecture for SIFT," *J. Syst. Archit.*, vol. 59, no. 1, pp. 16–29, Jan. 2013.

[27] S. X. Cao and J. Jiang, "Multi-scale image mosaic using features from edge," *J. Comput. Res. Develop.*, vol. 48, no. 9, pp. 1788–1793, Sep. 2011.

[28] *Virtex-5 FPGA User Guide, V5.3.*, Xilinx, Inc., San Jose, CA, USA, 2009, pp. 47–88.

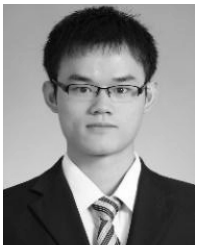
[29] F. M. Alzahrani and T. Chen, "A real-time edge detector: Algorithm and VLSI architecture," *Real-Time Imag.*, vol. 3, no. 5, pp. 363–378, Oct. 1997.

[30] P. Y. Hsiao, C. H. Chen, S. S. Chou, L. T. Li, and S. J. Chen, "A parameterizable digital-approximated 2D Gaussian smoothing filter for edge detection in noisy image," in *Proc. IEEE ISCS*, May 2006, pp. 3189–3192.

[31] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *Proc. 7th ECCV*, May 2002, pp. 128–142.

Jie Jiang received the bachelor's, master's, and Ph.D. degrees from Tianjin University, Tianjin, China, from 1991 to 2000.

She is a Professor with the School of Instrumentation Science and Opto-Electronics Engineering, Beihang University, Beijing, China. She has authored more than 50 articles and more than 30 inventions. Her research interests include real-time image processing, machine vision, and optical sensing.



Xiaoyang Li received the B.S. degree from the College of Precision Instrument and Opto-Electronic Engineering, Tianjin University, Tianjin, China, and the M.S. degree from the School of Instrumentation Science and Opto-Electronics Engineering, Beihang Unversity, Beijing, China, in 2010 and 2013, respectively.

His research interests include the very-large-scale integration design for image acquisition and real-time image processing.

Guangjun Zhang received the Ph.D. degree from the Department of Precision Instrumentations Engineering, Tianjin University, Tianjin, China, in 1991.

He was a Visiting Professor with North Dakota State University, Fargo, ND, USA, from 1997 to 1998. He was a Yangtze River Scholar Award Program Professor in 2000. He is currently a Professor with the School of Instrumentation Science and Opto-Electronics Engineering, Beihang University, Beijing, China. His research interests include laser precision measurement, machine vision, and optical sensing.