An Architecture of Optimised SIFT Feature Detection for an FPGA Implementation of an Image Matcher

Lifan Yao [#1], Hao Feng [*2], Yiqun Zhu [#3], Zhiguo Jiang [*4], Danpei Zhao [*5], Wenquan Feng [+6]

[#] *Department of Electrical and Electronic Engineering, The University of Nottingham*

University Park, Nottingham, NG7 2RD, United Kingdom

[1] eexly6@nottingham.ac.uk

[3] yiqun.zhu@nottingham.ac.uk

[*] *School of Astronautics, Beihang University, Beijing, 100191, China*

[2] fenghao@sa.buaa.edu.cn

[4] jiangzg@buaa.edu.cn

[5] zhaodanpei@buaa.edu.cn

[+] *School of Electronic and Information Engineering, Beihang University, Beijing, 100191, China*

[6] buaafwq@buaa.edu.cn

Abstract -- **This paper has proposed an architecture of optimised SIFT (Scale Invariant Feature Transform) feature detection for an FPGA implementation of an image matcher. In order for SIFT based image matcher to be implemented on an FPGA efficiently, in terms of speed and hardware resource usage, the original SIFT algorithm has been significantly optimised in the following aspects: 1) Upsampling has been replaced with downsampling to save the interpolation operation. 2) Only four scales with two octaves are needed for our image matcher with moderate degradation of matching performance. 3) The total dimension of the feature descriptor has been reduced to 72 from 128 of the original SIFT, which leads to significantly simplify the image matching operation. With the optimisation above, the proposed FPGA implementation is able to detect the features of a typical image of 640x480 pixels within 31 milliseconds. Therefore, compared with the existing SIFT FPGA implementation, which requires 33 milliseconds for an image of 320x240 pixels, a significant improvement has been achieved for our proposed architecture.**

I. Introduction

The image matching, which is a process to identify the relationship between a reference image and an image under investigation, has been widely used for a variety of applications, such as object localization, object recognition, motion estimation, 3D reconstruction and etc. A great number of image matching algorithms have been proposed during the past decades, which falls into two categories: pixel based and feature based. Generally speaking, pixel based algorithms can estimate simple transition motion robustly, but they may fail when dealing with either images with serious transformation or highly degraded images. Optical flow and phase correlation are two of the most popular pixel based methods [1][2]. On the other hand, feature based algorithms have been widely investigated recently [3][4]. They are able to extract those image features, which are invariant to image transition, scaling, rotation, illumination and limited ranges of viewpoint changes, so they can offer a robust image matching capability when tackling dramatically changed images or degraded images. Therefore, feature based algorithms can have higher image matching performance than pixel based ones, in terms of reliability and precision. However, the feature based algorithms are not practical to some real-time applications due to the nature of computational complexity and the huge demanding of memory consumption. [5] proposed a fast feature based image matching algorithm, but it is still considered to be too slow to be adopted for real-time applications though it has significantly improved the processing speed for feature extraction and generation of the feature descriptors.

A parallel hardware architecture for the SIFT (Scale Invariant Feature Transform) [4] feature detection has been proposed and implemented on an FPGA in [6]. In this architecture, the number of features detected has been greatly reduced by ignoring local minima for the feature detection to increase the processing speed. The authors claimed the reduction of the number of features has not degraded the performance for their robot application of Simultaneous Localization and Mapping (SLAM). However, the behaviour of feature detection in their architecture is not the same as that of the original SIFT algorithm anymore because it has been found from our own test based on similar test images that there should be a great number of features in some particular areas if the original SIFT algorithm were used, but they have been optimised away in their implementation for the purpose of their application. Therefore, it could be a problem for their optimised SIFT architecture to be applied to other image matching applications. In [7], a SIFT based object recognition processor has been developed by using NoC (Network-on-Chip) and Visual Image Processing (VIP) memory. Although the entire SIFT algorithm has been implemented using 0.18um CMOS process, the time required for the feature detection for an image of 320x240 pixels is 62 milliseconds with the clock frequency of 200 MHz, which is considered to be too slow for a typical real-time application. Therefore, the existing hardware implementations of the SIFT feature detection either have the low reliability of feature detection for conventional image matching applications or suffer from the low processing speed, which is one of the key hurdles for real-time applications.

In order to improve the processing speed of SIFT feature detection to the level required by conventional real-time application, we have taken an unconventional approach in our proposed FPGA implementation of SIFT feature detection for image matching applications. Specifically, before going straight to implement the SIFT algorithm on a state-of-the-art FPGA device, we have done a great deal of optimisation on the original SIFT algorithm for FPGA implementation without significant performance degradation towards image matching applications. Therefore, the key contributions we have made in this paper are as follows:

1) With the in-depth understanding of the SIFT algorithm and our FPGA design expertise, substantial optimisation has been done on the original SIFT algorithm, which not only facilitates the FPGA processing nature of parallel computing, but also greatly reduces the hardware resource usage, such as memory consumption. Having tested the C model of the optimised SIFT feature detection with a large population of images for the purpose of image matching, it has been shown that the matching performance is almost identical to that from [5].
2) The optimised SIFT algorithm from 1) has been implemented on a Xilinx Virtex-5 FPGA device based on a novel image partition techniques, which will be described in detail in Section III. It is shown that the proposed FPGA implementation is able to detect the features of a typical image of 640x480 pixels within 31 milliseconds with the clock frequency of 100MHz. Therefore, compared with the existing SIFT FPGA implementation, which requires 62 milliseconds for an image of 320x240 pixels with the clock frequency of 200MHz, our FPGA implementation not only has a huge improvement of processing speed, which is equivalent to four times faster if running by the same clock frequency, but also can deal with a larger image of 640x480 pixels. In other words, we can process 32 frames of 640x480 pixels per second, which meets the requirement of conventional real-time applications.

The paper is organised as follows: Section II describes the optimised SIFT algorithm for image matching applications. Section III presents the FPGA implementation of the optimised SIFT feature detection algorithm on a Xilinx Virtex FPGA device. Both the verification of the optimised SIFT algorithm and the FPGA implementation results are given in Section IV. Finally, conclusions and further work are in Sections V and VI, respectively.

II. Description Of Optimised SIFT Algorithm For An Image Matcher

The optimised SIFT algorithm for an image matcher is presented in this section. The fundamental idea of the original SIFT algorithm [4] based image matching is: Those features invariant to image transition, scaling, rotation, illumination and limited ranges of viewpoint changes are detected, extracted from the image under the consideration, and then transformed to descriptive vectors named as feature descriptors. The corresponding pairs of features can be identified by matching feature descriptors between the image under matching and the reference image and then the transformation relationship between the features matched can be obtained based on the least squares method [8].

The optimised SIFT based image matching algorithm can be broken down to five stages, which will be described individually in the following.

A. Gaussian Pyramid Construction

The original SIFT algorithm has been proposed in [4], in which the DoG (Difference-of-Gaussian) detector is adopted to detect those invariant features, so the first stage of the SIFT algorithm is to construct the Gaussian Pyramid.

According to the original SIFT algorithm, a Gaussian image can be obtained from the convolution operation of a Gaussian function with the input image as follows:

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

Where * is the convolution operation, *I* is the input image and *G* represents the Gaussian kernel with the size of σ as shown below:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2}$$

The corresponding scale space can be built up by increasing the size of Gaussian kernels for the convolution operations. A Gaussian pyramid is composed of a number of octaves with different image size; within each octave, there are several scales with the same image size.

The numbers of both the octaves and the scales need to be carefully selected because these two parameters affect the robustness and the complexity. Apparently, the larger number of either scales or octaves offers higher robustness on matching scale changes, but higher computational complexity will be introduced. On the contrary, the smaller number of either scales or octaves leads to lower robustness with lower complexity. A Gaussian pyramid with two octaves and four scales per octave will be constructed in our optimised SIFT algorithm because of the targeted images with a variation of within 2.5 times in resolution for our image matching application. Another key simplification is that upsampling operation has been avoided to reduce the high computational complexity of interpolation operation, in other words, the convolution operations start off with the input image rather than the upsampled image to produce the scale space for the first octave and go further for the downsampled image. In our case, therefore, considering the input image is of 640x480 pixels, the image size of these two octaves are 640×480 pixels and 320×240 pixels, respectively. The test has shown that there is only moderate performance degradation for such simplification, but with a significant reduction of hardware resource.

It can be seen from the original SIFT algorithm that the construction of the Gaussian pyramid is a typical sequential process: Firstly, all the scale images of the first octave are generated from the same input image directly. Secondly, all the scale images of the following octaves are generated from the last scale image of the previous octave. Therefore, it is not quite straightforward to use parallel computing to speed up the scale space construction in the original SIFT algorithm. In order to address this, another simplification for the scale space construction is proposed in our optimised SIFT algorithm, in which all the scale images of all the octaves are generated from the same input image directly. In our case, an initial size of the Gaussian kernel, $\sigma = 1.1$, is chosen for the first scale of both octaves and an σ of 1.3, 1.6 and 2.0 has been used for the other 3 scales of both octaves, respectively.

One more point needs to be mentioned is that the Gaussian kernel has been scaled up by 1000 times for the purpose of the high precision for Gaussian pyramid construction.

B. DoG Space Construction and Feature Point Identification

With the Gaussian pyramid constructed above, the DoG (Difference-of-Gaussian) space can be built up by applying the subtraction to two adjacent scale images pixel by pixel, as shown in (3) and Fig. 1. For the purpose of simplification and the improvement of robustness explained later, the accuracy of DoG is reduced in the proposed optimised SIFT algorithm by scaling down 1024 times with only integer values preserved. The key reason on this is that it is found from our investigation that scale-down of 1024 times is a good compromise between memory consumption and the detection performance. Another obvious reason is that it can be easily implemented in hardware by using shift operation.

$$
\begin{aligned}
D(x,y,\sigma) &= \frac{(G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y)}{1024} \\
&= \frac{(L(x,y,k\sigma) - L(x,y,\sigma))}{1024}
\end{aligned}
\tag{3}
$$

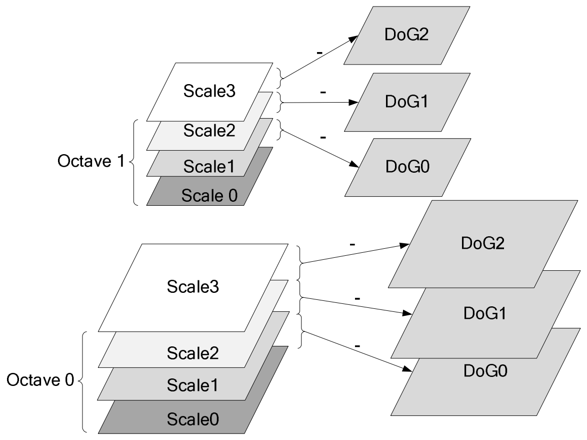


Fig. 1. The Construction of DoG

The feature points are chosen from the local maxima or minima in the DoG space. Each pixel in D(x, y, σ) will be compared with its 26 neighbour pixels, of which 8 pixels located in current scale image and others located in the scale above and below. As shown in Fig. 2, the candidate pixel in black is compared with those other 26 pixels in white. The candidate pixel will be considered to be a feature point and its coordinate and scale will be recorded if the candidate pixel value is larger than all those 26 pixel values or smaller than them.

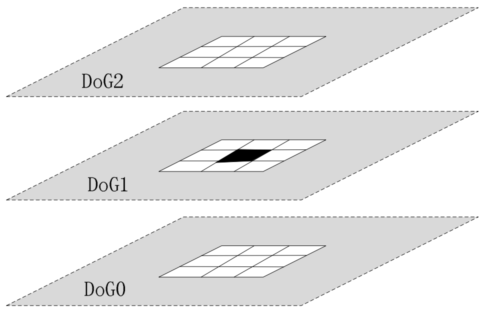


Fig. 2 The Detection of the Local Maxima and Minima

In our optimised SIFT algorithm, the Gaussian kernels have been re-presented in fixed-point values rather than floating-point for the purpose of hardware implementation efficiency. As a consequence, the inaccuracy introduced from this simplification enables a number of unstable feature points to be extracted, which lead to unnecessary computation burden and also degrade the matching performance in the end. In order to tackle the problem, the refining process is introduced in the original SIFT algorithm to eliminate the feature points located on the edge or with lower contrast. Although the same technique could be adopted in our optimised SIFT algorithm, we eliminate these unstable feature points by reducing the accuracy of DoG space as shown in equation (3), which further reduces the computational complexity. More specifically, the feature points with low contrast have similar value to non-feature points around the local region, and the feature points located on the edge have the similar value to non-feature points along the edge, so reducing accuracy is equivalent to eliminating the feature points with low contrast and higher edge response by truncation. We have tested such optimization, which supports our arguments above.

C. Gradient-Orientation Histogram Generation

The gradient-orientation histogram [4] is adopted to describe a feature point, which is computed from the gradient magnitude and orientation of the neighbour pixels around the candidate feature point. For a given pixel (x, y), the gradient magnitude and orientation are computed from equation (4).

$$
\begin{aligned}
m(x,y) &= \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \\
\theta(x,y) &= \tan^{-1}((L(x,y+1) - L(x,y-1)) / (L(x+1,y) - L(x-1,y)))
\end{aligned}
\tag{4}
$$

The gradient orientation of a pixel will be normalized to one of the eight directions named as 8 bins, so for the gradient-orientation histogram of a circular local region as shown in Fig. 3 (a), the direction of a particular arrow as shown in Fig. 3 (b) represents the orientation value and the length of the arrow represents the accumulation of the gradient magnitude of all the pixels with the same orientation. The histogram is usually represented in 2D coordinate as shown in Fig. 3 (c).

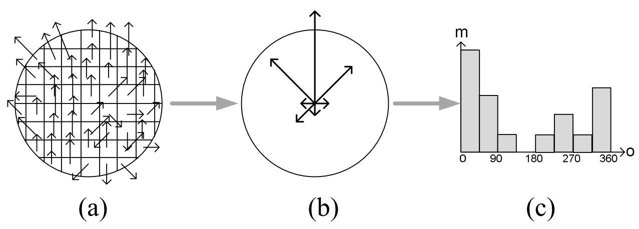


Fig. 3 Gradient-Orientation Histogram

The square-root operation in (4) has been replaced with the absolute operation to reduce computational complexity, so we have (5).

$$\widetilde{m}(x,y)=\frac{\left|L(x+1,y)-L(x-1,y)\right|+\left|L(x,y+1)-L(x,y-1)\right|}{1024} \quad (5)$$

In our optimised algorithm, all gradient magnitudes are divided by 1024 whereas the orientation value of range 0^o to 360^o is normalized to an integral number from 0 to 35.

The gradient magnitudes and orientations of all pixels can be computed in advance before generating feature descriptors. There are two benefits from this approach. Firstly, the magnitude and orientation of each pixel are computed only once. Secondly, the gradient magnitude and orientation can be computed in parallel with the DoG construction because these processes have no data dependency from each other.

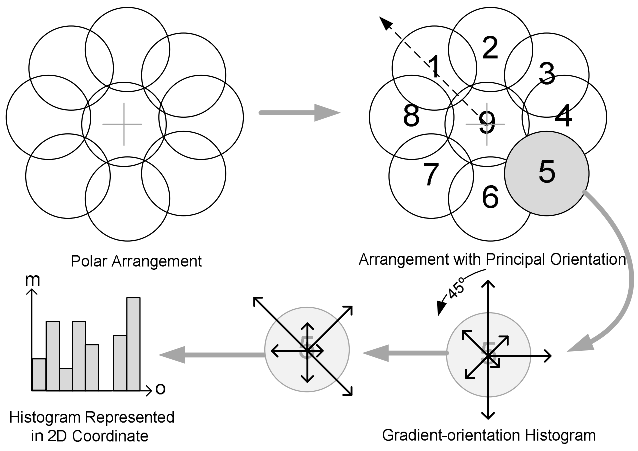
D. Feature Descriptor Generation

In the stage of feature descriptor generation, the pixels within the local region of a feature point are transferred to a descriptor.

A principal orientation needs to be identified first. The principal orientation is the gradient orientation corresponding to the maximum gradient magnitude in the gradient-orientation histogram for the entire local region of the feature point. With the principal orientation, the raw pixels or the spatial arrangement is rotated to achieve rotation invariance.

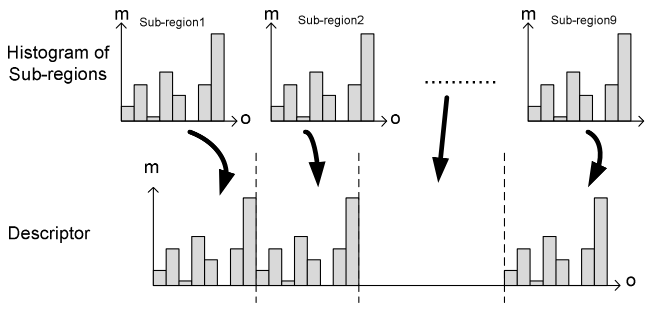
It can be seen from [4] that the local region of the involved feature point is segmented to a number of square sub-regions, such as 2x2, 4x4 or etc. The gradient-orientation histogram of each sub-region are computed independently, which are finally linked together to generate a descriptor for the feature point. However, the arrangement of segmentation affects the performance of the descriptor, we take the advantage of the Simon's polar sampled spatial arrangement [9], which improves the robustness on rotation with relatively lower computational complexity and lower descriptor dimension. The improvement on rotation is very attractive, which enables us to rotate the arrangement with only a small number of arithmetic operations without much loss in accuracy due to its isotropy characteristic. Finally, the final feature descriptor can be built up only by re-ordering both the sub-region and the bins of each sub-region rather than rotating all pixels in the entire local region of the feature point, which is required for the original SIFT algorithm. Therefore, a great reduction in computational complexity has been achieved.

Fig. 4 shows the process of generating a feature descriptor with our approach. Firstly, with polar arrangement, total 9 circular sub-regions of a feature point as shown in Fig. 4 (a) are identified, in which the cross in the central sub-region indicates the location of the feature point. Each circular sub-region has a diameter of 20 pixels and the overlapping of sub-regions is designed for robustness. Secondly, a gradient-orientation histogram of all 9 circular sub-regions is built up and then the principle orientation can be identified. Assume that the principle orientation is of the up-left direction pointed by the arrow with the broken line in Fig. 4 (a). Each of the 9 circular sub-regions is then assigned a number from 1 to 9 starting from the circular sub-region pointed by the principle orientation, following up with others in a clockwise fashion and ending up with the sub-region in the centre. Thirdly, the gradient-orientation histogram of each circular sub-region is generated and then it is re-ordered with the bin pointed by the principle orientation in the first place in the clockwise fashion. Finally, all re-ordered histograms are packed together with the order of the circular sub-region number as shown in Fig. 4 (b).



1
2
3
4
5
6
7
8
9
Polar Arrangement
Arrangement with Principal Orientation
45°
m
o
Histogram Represented in 2D Coordinate
Gradient-orientation Histogram

(a) Computation of the Histogram of Circular Sub-region 5



(b) Packing the Histogram of Circular Sub-regions to Descriptor

Fig. 4 The Process of Generating a Feature Descriptor

In our implementation, the gradient-orientation histogram of each circular sub-region has 8 bins covering the 360 degrees of orientation and there are 9 sub-regions in total, so the dimension of a feature descriptor is 72, which is significantly less than 128 from the original SIFT algorithm. In order to further improve the performance, in terms of robustness, the gradient magnitude of a pixel is weighted with a Gaussian function before accumulated to the corresponding bin. In other words, the farther from the centre of the circular sub-region has the lower contribution to the histogram.

E. Image Matching

The matching of the descriptor is based on the nearest neighbour, which is defined as the minimum Euclidean distance in terms of feature descriptor vectors. Given two descriptors, if the Euclidean distance between descriptor vectors is less than the predefined threshold, we assume the two feature points are matched.

III. The FPGA Implementation of The Optimised SIFT Feature Detection for an Image Matcher

It can be seen from Section II that the optimised SIFT algorithm for an image matcher consists of five stages: 1) Gaussian pyramid construction. 2) DoG space construction and feature identification. 3) Gradient and orientation histogram generation. 4) Feature descriptor generation. 5) Image matching. Considering the nature of Xilinx FPGA embedded system, a top level system partition which is similar to [6] has been adopted for the FPGA implementation. More specifically, the first three stages are implemented as a hardware core named as SIFT feature detection module, whereas the last two stages are considered to be implemented as a software module named as SIFT feature generation and image matching module using Xilinx MicroBlaze software processor [10]. In this section, we are going to present the optimised SIFT feature detection module because the software module for SIFT feature generation and image matching is out of the scope in this paper.

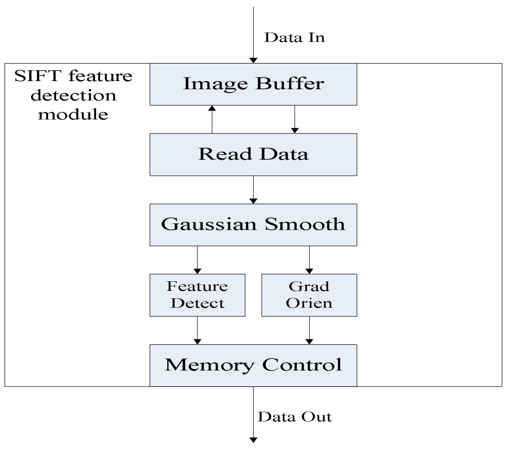


Fig. 5. A Block Diagram of the SIFT Feature Detection Module, In terms of Data Flow

As shown in Fig. 5, the SIFT feature detection module is composed of six components, of which Gaussian Smooth, Feature Detection and Gradient Orientation components correspond to the first three stages described in Section II, respectively.

Considering the clock frequency of 100MHz, in order for the real-time processing, which is considered to be 30 frames per second or 33 milliseconds per frame, to be achieved for the SIFT feature detection on the target images of 640x480 pixels, we have developed an image partition method, which enables parallel and pipeline processing and hence it maximises the throughput of the SIFT feature detection.

The parallel and pipeline processing is shown in Fig. 6. There are three-stage pipeline as follows: In stage one, *Read Data component* reads the first partition of the image. *Gaussian Smooth component* builds up the Gaussian pyramid in the second stage. In the final stage, *Feature Detect* and *GradOrien components* operate in parallel because there is no data dependency between them. The former works out the DoG space and identifies the feature points, whereas the latter works out the gradients and orientations for all the pixels. These three key modules are described in the following.

A. Read Data Component

We are going to present the image partition method here. The parallel processing can be easily achieved in the internal architecture of *Gaussian Smooth, Feature Detect* and *GradOrien components* to increase the processing throughput to the required level, so *Read Data component* is the bottleneck of the entire SIFT feature detection. In other words, the overall processing speed depends on how fast *Read Data component* can input the image data to *Gaussian Smooth component* partition by partition. We need to choose the partition size carefully to meet the throughput requirement, which is shown in (6).

$$\frac{ImageSize}{PartitionSize} \times TimeConsumedperPartition \le 33ms \tag{6}$$

Where *ImageSize* is the total valid size of the input images involved, which obviously includes the original image of 640x480 pixels and the downsampled image of 320x240 pixels, so.

$$ImageSize = (640-F)\times(480-F)+(320-F)\times(240-F) \tag{7}$$

F is the height and width of the boundary region, which has invalid data caused by the nature of the two dimensions Gaussian Filters. In our case, F is equal to 14 because the Gaussian kernel is of size 15*15.

PartitionSize is the size of the image partition, which is read into the SIFT feature detection component in a single stage with the pipeline fashion as shown in Fig. 6.

$$PartitionSize = x*y \tag{8}$$

x is the height of the partition, y is the width of the partition.

TimeConsumedperPartition is the number of clock cycles consumed to input a single image partition, which can be calculated as shown below.

$$TimeConsumedperPartition = (x+F+C1)\times(y+F+C2)+C3 \tag{9}$$

C1, C2 and C3, which are the numbers of the clock cycles required by relevant control signals, are 2, 5 and 2, respectively.

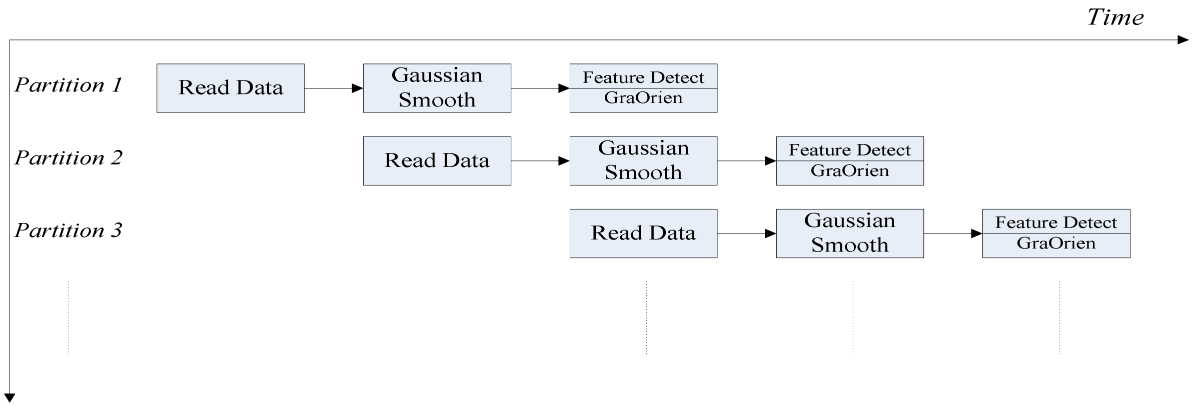


Fig. 6. The Pipeline and Parallel Architecture

The overall time requirement for the different partition sizes can be calculated from equation (6), which is shown in TABLE I.

TABLE I
DIFFERENT TIME CONSUMPTION ACCORDING TO DIFFERENT PARTITION SIZE

x	y	Time (ms)
7	11	32.43161
10	8	31.75674
8	10	31.48608
12	7	31.36150
6	14	31.27557
9	9	31.27557
7	12	30.71708
6	15	30.07267
13	7	29.98014
11	8	29.97698
8	11	29.60791
10	9	29.27073
7	13	29.26632
9	10	29.15044

The partition size of 7x12 pixels is chosen since some extra time budget is required for data interfacing for our implementation.

B. Gaussian Smooth Component

Considering the symmetrical nature of two-dimension Gaussian filter, a parallel architecture has been adopted in the *Gaussian Smooth Component* as shown in Fig. 7. Basically, an image partition of 7x12 pixels arrives from *Read Data Component* and then routed into seven *Gaussian Smooth units* via an *Image Buffer* under the control of *Address Control*. Therefore, seven rows of the image partition can be processed in parallel.

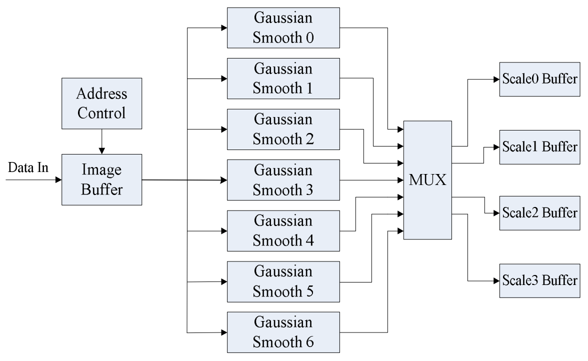


Fig. 7. The Block Diagram of Gaussian Smooth Component

It can be seen from Fig. 8 that the *Gaussian Smooth* unit mainly consists of filters and accumulators. Four scales are calculated in parallel to improve the system performance. Finally, scaled results are outputted to four FIFO buffers with a size of 128x18 bits each, which is large enough for one image partition of 7x12 pixels with a pixel value resolution of 18 bits. Moreover, with the re-use of arithmetic units and the conversion from floating-point to fixed-point calculation, the hardware resource usages are reduced significantly.

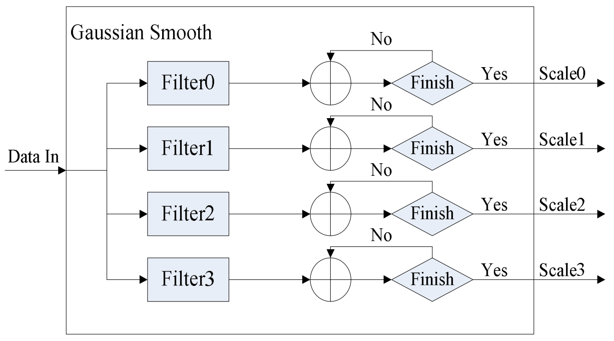


Fig. 8. Architecture of Gaussian Smooth Unit

C. Feature Detect and GradOrien Components

As shown in Fig. 9, the DoG scale images can be produced by simple subtraction pixel by pixel and then a feature point can be detected by comparing it with its 26 neighbours based on a high parallel structure, which is able to complete 28 pairs of comparison within one clock cycle. The coordinates of the feature points detected will be outputted to four FIFOs, which are created to store the x and y coordinates from octaves 0 and 1, respectively.

The gradient and orientation for each pixel in Scale 1image can be calculated in the *GradOrien Component* (Fig. 10) in parallel with the feature detect because there is no data dependency in between. A simplified look-up table is implemented for inverse tangent calculation. An external memory control unit is built up to store the gradient and orientation to an external SRAM of 256K×32 bits.

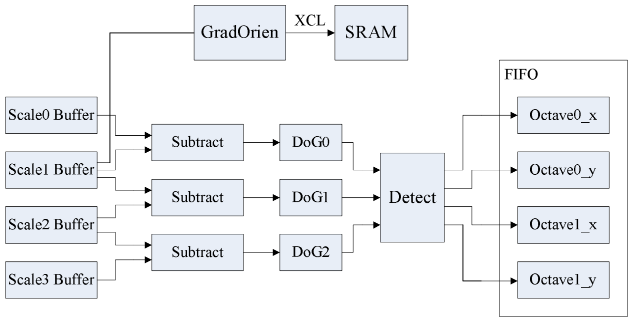


Fig. 9. Feature Detect and Gradient-orientation Calculation

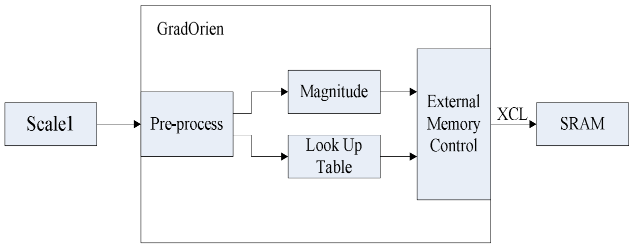


Fig. 10. Architecture of GradOrien Component

IV. Test and Verification

Test and verification has been conducted on both the software model of the optimised SIFT algorithm for an image matcher and the FPGA implementation of the optimised SIFT feature detection for an image matcher.

A. Software Model of the Image Matcher Based on the Optimised SIFT Algorithm

A software model of the entire image matcher based on the optimised SIFT algorithm has been built up on Microsoft Visual C++ 2005 Express Edition.

The standard evaluation [3] has been adopted to assess the matching performance, which is presented as a curve of recall versus 1-precision.

Given two images representing the same scene, the recall as shown in (10) is the ratio of the number of the correctly matched points to the number of corresponding matched points:

$$recall = \frac{\#\, correct\_matches}{\#\, correspondence} \tag{10}$$

The number of corresponding points is determined by the overlapping of the points in different images. The 1-precision as shown in (11) is the ratio of the total number of the falsely matched points to the sum of the number of the correctly matched points and the number of the falsely matched points:

$$1 - precision = \frac{\#\, false\_matches}{\#\, correct\_matches + \#\, false\_matches} \tag{11}$$

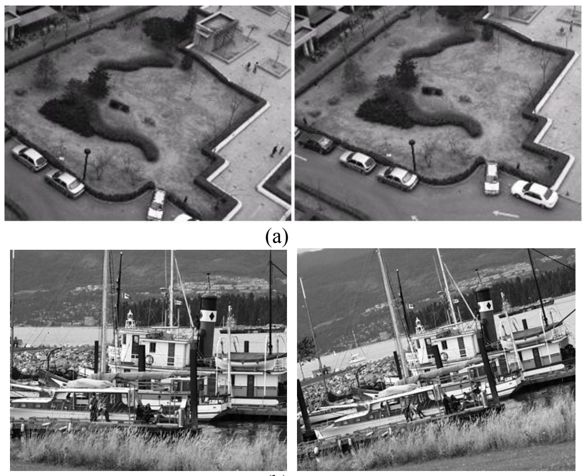


Fig. 11. Images Under Test

Two sets of images under test are shown in Fig. 11 (a) and (b), respectively. With these images, the image matching behaviour as shown in Fig. 12 has been obtained based on the software model of the optimised image matcher. In Fig. 12, each line between two images indicates a pair of corresponding feature points.

For the purpose of comparison, the corresponding software models of the original SIFT [4], SURF [5] and polar sampling descriptor [9] have also been implemented. The matching performance for each of these models are also obtained and presented with the curves of recall versus 1-precision as shown in Fig. 13 by varying the matching threshold. It can be seen that the performance based on our optimised SIFT is more or less the same as that from SURF [5] even though it is lower than the original SIFT.

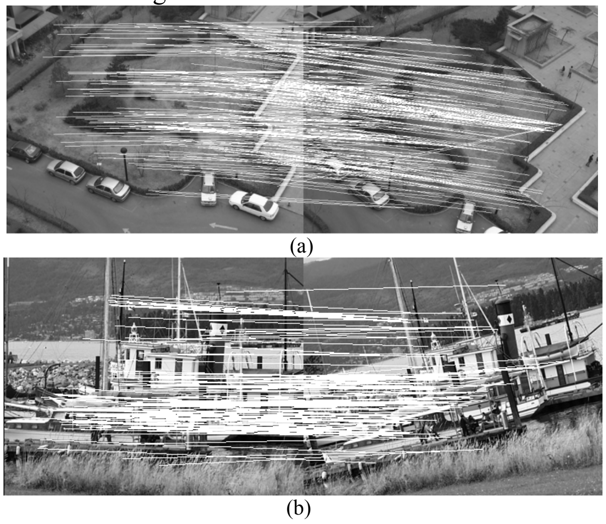


Fig. 12. Image Matching Behaviour

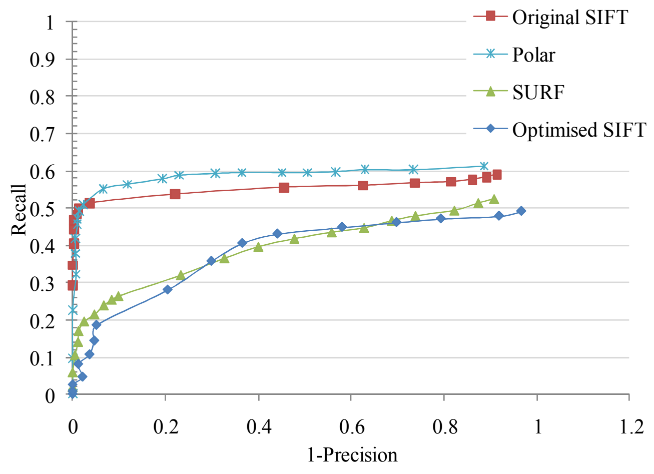


Fig. 13. Image Matching Performance Comparison

B. *FPGA implementation of the optimised SIFT Feature Detection for an Image Matcher*

The Feature detection module of the optimised SIFT algorithm has been implemented on Xilinx ML507 development board. The test on this module with the same images in Fig. 11 has shown that the matching performance based on this hardware module is identical to that from the corresponding software model and hence confirmed the correctness of the FPGA implementation. Furthermore, the test has also shown that the feature detection module only requires 31 ms to process a typical image of 640x480 pixels. However, 60 ms is required to detect features from an image of the same size in [11]. 33 ms is required for an image of 320x240 pixels in [6]. Therefore, our feature detection module offers the fastest processing speed.

Regarding FPGA resource usage, it is not easy to compare between our feature detection module and the module from [6] because of a different FPGA technology used. However, it can be seen from our FPGA implementation that our hardware module consumes 35,889 LUTs, 19,529 Registers, Block RAM of 3,240 Kbits and 97 DSP48E, but the module from [6] has used 43,366 LUTs, 19,100 Registers, Block RAM of 1,350 Kbits and 64 DSP blocks. Therefore, our feature detection module has used a similar level of hardware resources to that in [6] except that our module requires significantly more block RAMs because we need more block RAMs to buffer four times larger image than that from [6].

V. Conclusions

An entire optimised SIFT algorithm has been proposed for FPGA based real-time image matching applications in this paper. Not only has the corresponding software model been built up, but the key part of the optimized SIFT algorithm, which is the SIFT feature detection module, has also been implemented on a Xilinx Virtex-5 FPGA. It has been confirmed from the standard test that our SIFT feature detection module is able to extract features from a typical image of 640x480 pixels with 31 ms, which is faster than any existing systems. It can be seen from the matching performance comparison that our optimized SIFT algorithm has achieved a similar level to SURF, which is one of the well recognized image matching algorithms.

VI. Further Work

The key further work will be: 1) The software module named as SIFT feature generation and image matching module will be implemented based on Xilinx MicroBlaze soft processor with a custom parallel processor to achieve a similar processing speed the feature detection module has offered and hence the overall real-time processing speed can be achieved. 2) Further optimisation will be conducted to improve the matching performance close to that from the original SIFT algorithm without significantly increasing hardware complexity.

Acknowledgment

This work was supported by the partnership grant from Innovation China UK.

References

[1] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” *in Proc. IJCAI’81*, 1981, pp 674–679.

[2] E. De Castro and C. Morandi, “Registration of Translated and Rotated Images using Finite Fourier Transform,” *IEEE Trans on Pattern Analysis and Machine Intelligence*, vol.9 (5), pp 700-703, 1987.

[3] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.27, pp. 1615–1630, 2005.

[4] D.G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *in Proc. IJCV’04*, 2004, vol. 20(2), pp. 91–110.

[5] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded up robust features”. *In Proc ECCV’06*, 2006.

[6] V. Bonato, E. Marques, and G. A. Constantinides, “A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 18, NO. 12, Dec. 2008.

[7] D. Kim, K. Kim, J.-Y. Kim, S. Lee, and H.-J. Yoo, “An 81.6 gops object recognition processor based on noc and visual image processing memory,” in *Proc. IEEE Custom Integrated Circuits Conf.*, San Jose, CA, 2007, pp. 443–446.

[8] Alan J. Laub, “Matrix Analysis for Scientists and Engineers,” *ISBN-13: 978-0898715767*, pp 65-71.

[9] S.Winder and M. Brown, “Learning Local Image Descriptors,” *in Proc. CVPR’07*, 2007, pp.1-8.

[10] Xilinx, “Virtex-5 FXT PowerPC 440 and MicroBlaze Embedded Kit Reference Systems,” *Jan 26, 2009*.

[11] S. H.-K. Ng, P. Jasiobedzki, and T.-J. Moyung, “Vision based modelling and localization for planetary exploration rovers,” in *Proc. IAC*, Oct.2004, pp. 11–11.