An Optimized Structure on FPGA of Key Point Detection in SIFT Algorithm

Chenyu Xu[1], En Zhu[1], Tiange Su[1] and Xiaoran Li[2]

[1]*School of Information Science and Engineering, Southeast University, Nanjing, China*
[2]*School of Electronic Science and Engineering, Southeast University, Nanjing, China*

Abstract. SIFT algorithm is the most efficient and powerful algorithm to describe the features of images and it has been applied in many fields. In this paper, we propose an optimized method to realize the hardware implementation of the SIFT algorithm. We mainly discuss the structure of Data Generation here. A pipeline architecture is introduced to accelerate this optimized system. Parameters' setting and approximation's controlling in different image qualities and hardware resources are the focus of this paper. The results of experiments fully prove that this structure is real-time and effective, and provide consultative opinion to meet the different situations.
Keywords. SIFT; pipeline architecture; Gaussian window function; approximating errors

1 Introduction

Image matching is the process to identify the features of pictures to confirm the similarity and establish relevance between these similar pictures. The most essential parts of image matching is feature obtaining. The definition of features and the detection of them is the key to the whole process. There are many features in a picture, including edges and corners, and many algorithms are proposed to detect a specific type of features, like Canny operator and Harris corner. In 1999, David G. Lowe proposed a new algorithm [1], which is known as Scale-Invariant Feature Transform (SIFT), and revised it latter in 2004. This algorithm is established on the invariance of features no matter how environment changes. These changes can be described as multi-scales, and only those points which can show their invariance in each scale could be defined as key points. When all key points have been detected, corresponding descriptors will be generated and matching is the final step. With its distinctive process, SIFT algorithm has shown its advantages in many aspects, and been adopted widely.

Although SIFT algorithm has been realized on software platform consummately and proficiently nowadays, including a program in OpenCV patented by Lowe himself, it is another matter when SIFT algorithm is implemented to hardware platform. Time and resources consumption is the prime consideration; thus huge computations need to be transformed and simplified. Consequently, many scholars propose their designs for hardware implementation of SIFT algorithm. All these designs require the introduction of pipeline structure. This structure can utilize time most profitably. Those particular designs mainly differs in several specific steps of SIFT algorithm. The first step is the Generation of Gaussian Pyramid. ROI is introduced to generate suitable initial images in [2] and initial images are divided into smaller partitions in [3]. The size of Gaussian window function varies in [3]-[6], even the number of scales and octaves [7]. The value of σ also differs in [3]. Some IP kernels are introduced to calculate the filtering such as MAC and MUX in [2]. The second step is the Detection of Key Points. ADI and DER are introduced to obtain key points more accurately [7]. There are also some scholars using other methods like Harris corner to detect features [4]. The third step is the Calculation of Gradient and there is no difference in this part. Other steps include Generation of Descriptor and Matching. Some scholars complete these parts in combination with DSP [7], or other method like BRIEF descriptor [8], and we do not discuss these steps in details here.

After referring to these multiple designs, we propose our own optimized SIFT algorithm and implement it to hardware platform with Verilog. We mainly emphasize and discuss the setting of parameters and controlling of approximating errors in different conditions. The whole system is verified to be real-time and effective.

The following content of this paper is divided into 5 sections. Section II briefly discusses the algorithm proposed by David G. Lowe and its optimization to suit the hardware environment. Section III is the design of whole structure on hardware. Section IV is the results of experiments. Section V is the final conclusion of this paper.

2 Theory of SIFT Algorithm and Its Optimization

To be frank, when we implement SIFT algorithm on hardware platform, there are a couple of problems we need to handle, especially the time and resources

consumption. Therefore, we need to figure out its principle first, and consider how to adjust and optimize SIFT algorithm to meet the hardware conditions without the introduction of DSP. This significant algorithm can be divided into two main stages: Key Point Detection and Key Point Description. In this paper, we mainly discuss the first stage, and we will briefly introduce the theory of this stage and its optimization in this section.

2.1 Theory of SIFT Algorithm

It has been verified that the only possible scale-space kernel is the Gaussian function [1]. The scale space of an image is defined in (1):

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

The Gaussian filters are defined in (2):

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

The complete realization of Gaussian scale space has a couple of steps: first, each scale is generated from the previous scale, and σ_i are different in each scales; second, after generating some scales of image, we can assemble them and call them an octave, then the last third scale is down-sampled by a factor of 2 to generate the first scale of next octave; third, the generating processes are the same in different octaves and the σ_i series are the same, too; fourth, the initial image should be up-sampled by a factor of 2 to generate the zero octave, and smoothed latterly by a Gaussian filter to create the first scale of zero octave.

In this process, σ_i series are crucial to keep continuity in scale space. In Lowe's work, the initial image is supposed to be incrementally convolved with Gaussians by a constant factor k [1], and the σ_i is defined as $k^{i-1}\sigma_0$. Here Lowe recommends using $\sigma_0 = 1.6$ [1]. If there are $s+3$ scales in one octave, k is defined as $2^{1/s}$.

After generating Gaussian Pyramid, we need to generate difference-of-Gaussian function, which is defined in (3):

$$D(x,y,\sigma) = L(x,y,k\sigma) - L(x,y,\sigma) \tag{3}$$

Then we can detect the local maxima and minima of the middle scales, by comparing each sample point in these scales to its eight neighbors in the current image and nine neighbors in the scale above and below [1]. Only the point which is larger than all of its neighbors or smaller than all of them is the extremum.

However, not all of the extrema are key points. Some of them have low contrast or are poorly localized along an edge [1]. The most effective way to remove or revise these points is to use Hessian matrix, which is defined in (4):

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{4}$$

All valid and reasonable key points should meet the restriction of (5) and here $r = 10$:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r} \tag{5}$$

The final step is to compute the gradient magnitude and orientation. When the calculation is finished, we can store the results into RAMs and they can be easily called by Generation of Descriptor.

2.2 Optimization of SIFT Algorithm

The main optimization is in the step of Generation of Gaussian Pyramid. There are a couple of consensus in all implementations: first, the generation of all scale of an octave should be completed at one time, the new σ_i is defined in (6); second, the size of Gaussian filter should be fixed and medium-sized; third, approximating errors exist in any computation; fourth, the generation of the zero octave should be abandoned.

$$\sigma_{i+1}' = \sqrt{\sigma_i'^2 + \sigma_{i+1}^2} \tag{6}$$

In this paper, we mainly discuss the realization of second and third point of consensus.

To the second point, on one hand, if the size of Gaussian filter is limited, differences compared with the standard values will be created. The differences will become more severe if the size of template is much smaller than the standard of $(6\sigma_i + 1) \times (6\sigma_i + 1)$. Besides, these differences will be accumulated when (6) is used repeatedly. On the other hand, the size of template should not be too enormous, considering the restriction of hardware resources. However, if we assume $\sigma_0 = 1.6$, the shortest width of window to avoid the differences is 41, which is too difficult to achieve. Only [7] and [9] emphasize this issue and announce that they manage to adopt this size in their work. Therefore, it is criticized to set a medium-sized template. Yet some scholars prefer to choose much smaller size of template. It seems that much smaller size of template could also obtain satisfying results. Whether the restriction of setting size of template is meaningful will be examined and elaborated in Section IV.

Related experiments show that the larger of the template size, the more accurate of the final results. If we could not enlarge the size of template, we could reduce the value of σ_0, while the results may not as effective as the size broadening.

To the third point, approximating errors will introduce some redundant key points. If the number of these extra key points is too large, the speed of the whole system will be degraded. The only method to solve these problems is to scale up the coefficients first, rather than remove the redundant points in Key Point Detection. One is the

enlargement factor of each pixel; the other is the enlargement factor of Gaussian template.

The calculation of gradient magnitude and orientation also need to be revised. The formula for gradient magnitude is defined in (7):

$$m(x,y)=|L(x+1,y)-L(x-1,y)|+|L(x,y+1)-L(x,y-1)| \quad (7)$$

The orientation of gradient can be divided into some equal parts. In Lowe's work, it is 36 parts, and here we consider that 16 parts is enough.

3 FPGA Implementation of Optimized Algorithm

When the preparation has been finished, it is the time to implement this optimized algorithm. In this section, we briefly introduce the design of our pipeline architecture, and the main structure of each step of Data Generation.

3.1 Pipeline Architecture

The pipeline architecture is a kind of structure analogous to assembly line. There are some basic aspects: first, the data are sent in one by one in each data line; second, when the first datum is being carried on the second step of operation, the second datum will be carried on its first step of operation; third, the first octave of Gaussian Pyramid is the first part of the pipeline; fourth, Extrema Detection, Hessian Matrix Rejection, and Gradient Calculation could be operated at the same time.

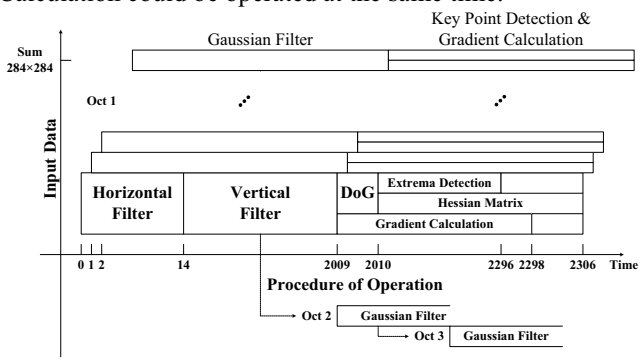


Figure 1. Structure of Pipeline

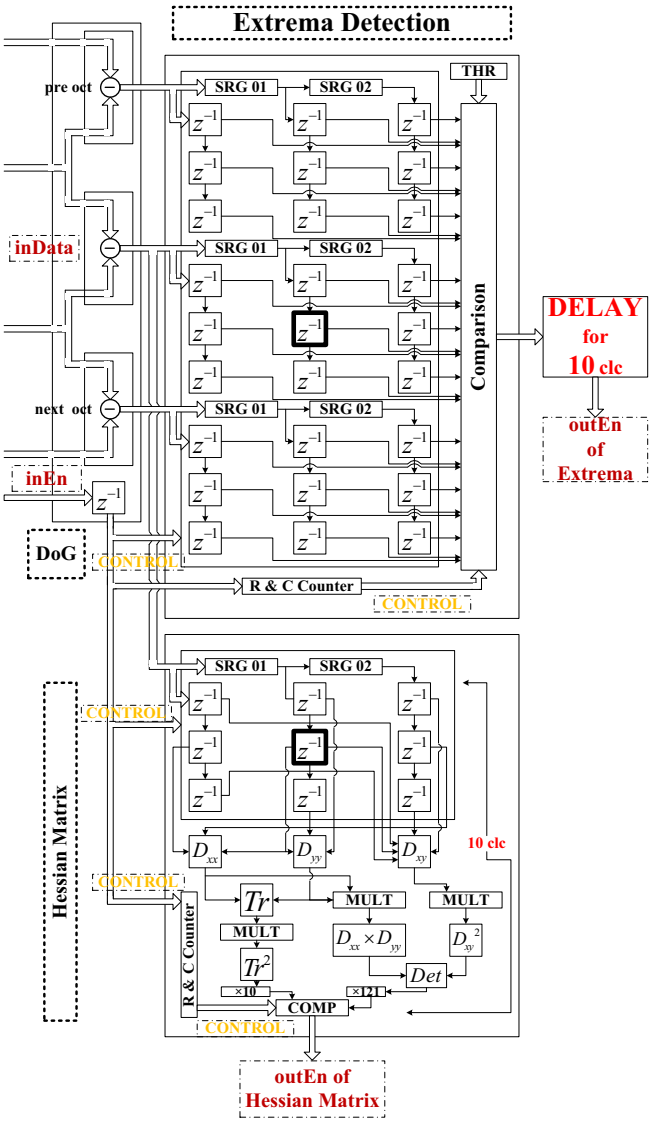


Figure 2. Structure of Key Point Detection

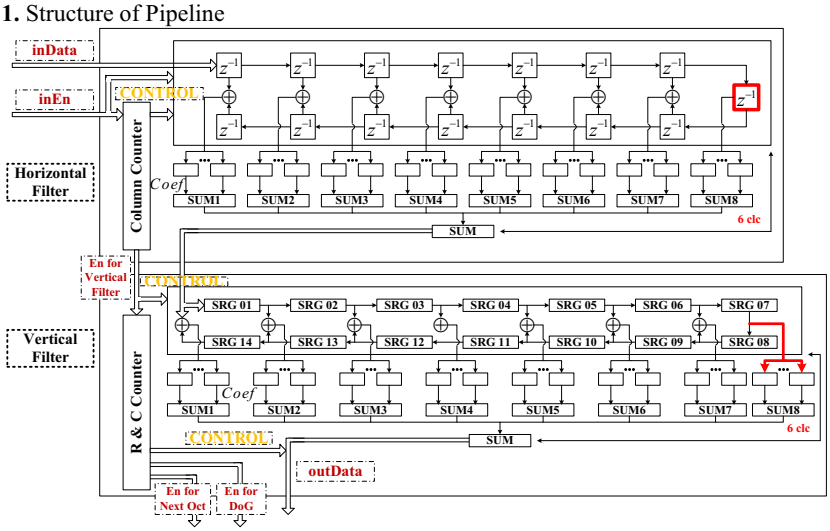


Figure 3. Structure of Gaussian Pyramid

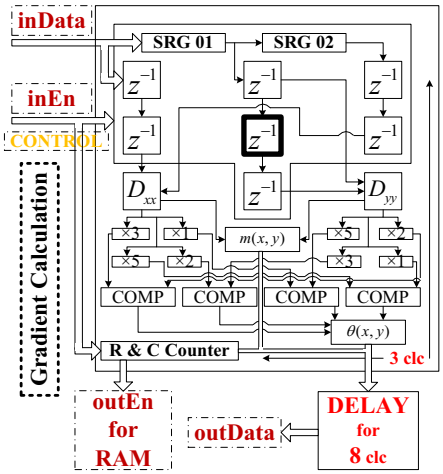
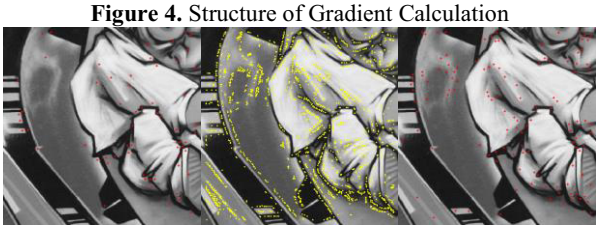


Figure 4. Structure of Gradient Calculation



(a) (b) (c)

Figure 5. Results of Key Point Detection

Table 1. Numbers of Detected Key Point

Sca & Oct	o1 s2	o1 s3	o2 s2	o2 s3	o3 s2	o3 s3
OpenCV	61	55	32	13	7	3
No Approx	123	1	51	1	12	0
Miss No.	37	55	13	13	1	3
Mistake No.	99	1	32	1	6	0
Approx	1916	944	417	128	75	18
Miss No.	28	53	14	13	4	3
Mistake No.	1883	942	399	128	72	18

Apart from the common views above, we make some adjustment in our system: first, the size of the initial image is 284×284 ; second, Gaussian filtering is separated into horizontal filtering and vertical filtering, with advantages analyzed in 2; third, the output of Extrema Detection, Hessian Matrix Rejection, and Gradient Calculation should be sent out at the same time.

A crude description of the relations of each steps and direction of data flows is shown in Figure 1. The operating process of the first datum costs 2306 clocks. And from this moment, the computing results of the points in first octave will be sent out continuously. Whereas the outputs of the second and third octave will not be continuous.

3.2 Structure of Optimized System

Due to the constraints of resources, we set the number of s as 2, the size of Gaussian filter as 15×15 , the enlargement factor of each pixel as 1 and that of Gaussian template as 1024. The whole diagram of Gaussian Pyramid, Key Point Detection and Gradient Calculation is shown in Figure 2-4 respectively.

The most important parts in these structures are the shift registers and the enable signals. A shift register can store a whole row of image data, and the outputs of these registers are continuous in a line. These output data are for the following computations. The enable signals are the key to control the identification of the valid data.

Buffers could be introduced to reduce the consumption of resources and the order of filters should be reversed.

4 Experimental Results

In order to evaluate the performances of the proposed architecture and examine the effects of setting parameters in different conditions, we divide the experiment into two parts:

The first part has three steps: first, we test the processing time and observe the final results of Key Point Detection; second, we adjust the system in a further optimization and observe the final matching result; third, we change the size of window and observe the matching performances.

In the second part, noise will be introduced and the effects of parameters will be analyzed systematically.

The whole hardware implementation is realized on Xilinx ZC702, and clock frequency is 100MHz. The reference standards are the results in OpenCV. The results of tests are shown through MATLAB.

4.1 Experimental Results without Noise

When $\sigma_0 = 1.6$, and other parameters are the same as those in Part B, Section III, the result of the algorithm in OpenCV is shown in Figure 5(a), and the result of our hardware implementation is shown in Figure 5(b). Only the results of second scale in first octave are shown here. We can discover that too many key points detected because of the unreasonable setting of enlargement factors. However, if we examine the ideal result without approximating errors, shown in Figure 5(c), we could conclude that to set window width as 15 seems inappropriate. Related data is presented in Table 1.

Yet the time of the whole process is 966990 clocks, less than 1ms. Suppose that Descriptor Generation and Matching cost 3ms per 1000 key points. Whole SIFT algorithm along with matching costs no more than 5ms. It is a considerable processing speed and is enough for continuous real-time matching. However, the consumption of resources is much higher, for those shift registers employ too much LUTs. In this case, buffers are necessary.

Therefore, we adjust and revise the structure of our system. The enlargement factor of pixel and template are switched into 64 and 128 respectively. The result of Key Point Detection is presented in Table 2. We could find out that the result is much better though still unsatisfying. However, when we observe the final matching results, shown in Figure 6, we could find out that the matching

result is accurate. Here the size of larger image is 800×640 and that of smaller one is 284×284, with an angle of rotation existing between two images.

Furthermore, if we set the width of window as 3 and enlargement factors are adaptive, the results are shown in Figure 7. We would discover that this extreme case could also achieve the same accurate result while this can never be required on software platform. Besides, all key points detected here are generated because of the approximating errors introduced by simplification of calculations.

Therefore, we could conclude that on one hand, SIFT algorithm has the best ability of robustness. On the other hand, the setting of parameters does not need to be so strict when we do hardware implementation. In addition, matching results are what we should pay attention to, rather than intermediate results.

4.2 Experimental Results with Noise

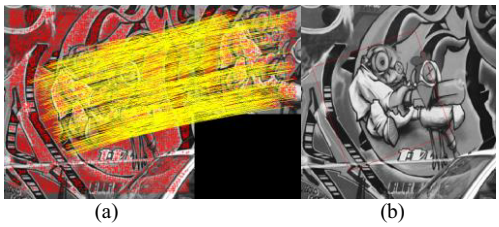
4.1 is based on the condition without noise, the results could be different if noise is introduced. Thus, we introduce four kinds of noise respectively. The maximum extents of noises when matching results are still correct are presented in Table 3. We can reveal that the smaller value of window width, the poorer ability of noise immunity. In this case, the restriction of window width is significant. Various factors into consideration, we could conclude that a medium size of Gaussian filter is the best choice for hardware realization and 15×15 is such one.

Table 2. Numbers of Key Point after Adjustment

Sca & Oct	o1 s2	o1 s3	o2 s2	o2 s3	o3 s2	o3 s3
Approx	408	125	88	39	16	7
Miss No.	42	62	10	20	2	3
Mistake No.	372	121	67	38	11	6

(a) (b)

Figure 6. Matching Results with $w=15$



(a) (b)

Figure 7. Matching Results with $w=3$

Table 3. Limited Extent of Noise

w	3	15	31
average	[4,4]	[6,6]	[7,7]
disk	2	4	4
Gaussian ($\sigma=5$)	[4,4]	[7,7]	[8,8]
Motion ($\theta=45^{\circ}$)	5	9	11

5 Conclusion

In this paper, we have proposed an optimized structure of Data Generation in SIFT algorithm, and its implementation on hardware platform. The whole system is based on a pipeline structure and verified to be real-time and effective. The results of experiments fully prove that the setting of parameters and controlling of approximations are crucial in hardware implementation, and this process will be affected by different conditions. Here we recommend using a medium-sized Gaussian window function.

References

1. D. G. Lowe, "Dinstinctive Image Features from Scale-Invariant Keypoints," Int. J. Comput. Vision, vol. 60, no. 2, pp. 91 - 110, 2004.
2. K. Mizuno, H. Noguchi, Guangji He, Y. Terachi, T. Kamino, H. Kawaguchi and M. Yoshimoto "Fast and Low-Memory-Bandwidth Architecture of SIFT Descriptor Generation with Scalability on Speed and Accuracy for VGA Video," in Proc. Int. Conf. Field Programmable Logic and Applications (FPL), 2010, pp. 608 - 611.
3. L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, "An Architecture of Optimised SIFT Feature Detection for an FPGA Implementation of an Image Matcher," in Proc. Int. Conf. Field Program. Technol., 2009, pp. 30 - 37.
4. Takahiro Suzuki and Takeshi Ikenaga, "SIFT-Based Low Complexity Keypoint Extraction and Its Real-Time Hardware Implementation for Full-HD Video", in Proc. Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific.
5. Jie Jiang, Xiaoyang Li, Guangjun Zhang, "SIFT Hardware Implementation for Real-Time Image Feature Extraction," IEEE Trans. Circuits Syst. Video Technol., vol. 24, no. 7, pp. 1209 - 1220, Jul. 2014.
6. S. Zhong, J. Wang, L. Yan, L. Kang, and Z. Cao, "A real-time embedded architecture for SIFT," J. Syst. Arch., vol. 59, no. 1, pp. 16 - 29, Jan. 2013.
7. Han Xiao, Wenhao He, Kui Yuan, Feng Wen, "Real-time Scene Recognition on Embedded System with SIFT Keypoints and a New Descriptor," in Proc. 2013 IEEE int. Conf. Mechatronics and

Automation (ICMA), Aug. 4-7 Takamatsu, Japan, pp. 1317 – 1324.

8. J. Wang, S. Zhong, L. Yan, and Z. Cao, "An Embedded System-on-Chip Architecture for Real-time Visual Detection and Matching," IEEE Trans. Circuits Syst. Video Technol., vol. 24, no. 3, pp. 525 - 538, Mar. 2014.
9. M.Qasaimeh, A.Sagahyroon, T.Shanableh, "A Parallel Hardware Architecture for Scale Invariant Feature Transform (SIFT)," in Proc. Int. Conf. Multimedia Computing and Systems (ICMCS), 2014, pp. 295 - 300.