MVA2009 IAPR Conference on Machine Vision Applications, May 20-22, 2009, Yokohama, JAPAN

A Hardware Accelerator with Variable Pixel Representation & Skip Mode Prediction for Feature Point Detection Part of SIFT Algorithm

Jingbang QIU, Tianci HUANG, Yiqing HUANG, Takeshi IKENAGA

Graduate School of IPS, WASEDA Univ.
N355, 2-7, Hibikino, Wakamatsu, Kitakyushu, 808-0135, Japan
E-mail: megisgem0630@ruri.waseda.jp

Abstract

Scale Invariant Feature Transform (SIFT) is well accepted as a robust feature point detection algorithm, which is invariant to rotation, scaling, illumination and viewpoint changes. Though powerful, high computation complexity acts as a bottleneck of the real-time systems. It is not until recently that the only hardware implementation scheme is proposed to reach real-time processing. In this paper, we propose a hardware accelerator structure of the Feature Point Detection part in SIFT which is possible to implement on FPGA. We apply integer-based Variable Pixel Representation which represents a pixel with variable number of registers in different computational stages to reduce redundant register consumption. Also, we introduce Skip Mode Prediction into the system, eliminating redundant computation, so as to shorten averaged computation time per pixel. Our work proves to speed up Max Clock Frequency for 75.0%, lower Register Consumption for 13.6%, and achieve higher Accuracy for 10-20% and Efficiency for 10.4% over conventional work. The proposal is more suitable for real-time system design of SIFT.

1. Introduction

The SIFT algorithm proposed by David Lowe in the year of 1999 in [1] [4]. SIFT algorithm is widely accepted as a powerful method of feature point detection, which is invariant to Scale, Rotation, Viewpoint and Illumination Changes. Many modifications of the algorithm have been proposed and do help improve performance of SIFT. Nevertheless, bottleneck exists. Time consumption of the algorithm is relatively huge as a result of complex processes of the algorithm to achieve its robustness.

Due to its high complexity, hardly any real-time system exists. GPU-based system has been proposed in [3] [5] [6]. Although accelerated, this method greatly depends on the performance of the GPU chip and the PC environment, and the results vary much from computer to computer. Quite recently, the only hardware implementation architecture is proposed in [2], with focus in the Feature Point Detection Part. The result showed promising view of hardware-aided implementation of SIFT, but redundant register consumption and redundant computation exist.

Our aim is to further accelerate Feature Point Detection Part of SIFT algorithm. By introducing Variable Pixel Representation (VPR) Scheme and Skip Mode Prediction (SMP) Scheme, we successfully reduce register consumption, achieve higher Max Clock Frequency, and lower averaged computation time. Our proposal also shows about 10.4% higher efficiency, and 10-20% higher accuracy over conventional work.

This paper is arranged as follows. A brief introduction to SIFT and corresponding hardware-aided structure is given in SECTION 2, followed by problem statement of the existing hardware implementation proposal in SECTION 3. We will give out an advanced proposal of hardware structure of the Feature Point Detection part in SECTION 4. Results and Analysis will be given at the last part.

2. SIFT and Conventional HW-Aided Structure

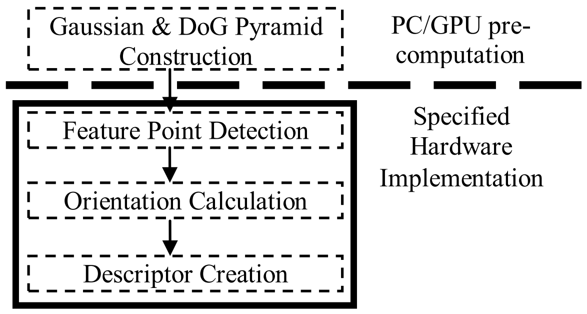

Figure 1 Hardware-Aided Structure of SIFT Implementation

The most significant advantage of SIFT over other algorithms is that the feature points detected are invariant to image scaling and rotation, while at the same time robust to changes in illumination, noise, occlusion and minor changes in viewpoint. In addition to these properties, those feature points are highly distinctive, relatively easy to extract, allow for correct object identification with low probability of mismatch and are easy to match against a database of local features. They are also robust to occlusion; as few as 3 SIFT features from an object are enough to compute its location and pose. In addition to object recognition, the SIFT features can be used for matching, which is useful for tracking and 3D scene reconstruction.

In [2], a hardware implementation idea is proposed, which is to break up SIFT into two parts (Fig. 1). The first part consists of only Guassian & DoG Pyramid Construction, which is implemented by PC or GP-GPU. The second part consists of Feature Point Detection, Orientation Calculation, and Descriptor Creation, which are most time consuming. The second part is proposed to be implemented on specified hardware. Research [2] majorly focus on implementation of Feature Point Detection Part because as a whole SIFT is too large a system to discuss at a time.

Research [2] also notice that, by introducing the hardware-aided proposal, time consumption of SIFT algorithm can be reduced greatly to less than 5% of what it was originally in PC implementation.

3. Problem Statement: Redundant Register & Redundant Computation

Research [2] though, has successfully implemented SIFT on FPGA and required similar result with original software implementation, that proposal, as a matter of fact, can be further improved by the means of resource consumption and averaged time consumption.

The two major problems from the paper are that,

1) Redundant registers used to represent one pixel. As not all of the processes need the same accuracy for computation, it is not wise to use the same number of registers to represent a pixel.
2) Redundant Computation. As analysis shown below, we can see that only very small part of pixels need full computation throughout the system, while in fact, most of the pixels would be eliminated by Contrast Pre-Elimination and Extrema Detection (Tab. 1).

Table 1 Pass-Through Rate Analysis (640x480)

Item	Computation Rounds	Pass-Through Pixels	Pass-Through Rate
Contrast Pre-Elimination	116415011	288403	0.25%
Extrema Detection	288403	1319	0.46%
Contrast Elimination	1319	727	55.04%
Too-Edge-Like Elimination	726	658	90.63%

4. Proposed Schemes

As shown in SECTION 3, redundant register consumption and redundant computation exist in conventional work. In order to solve these two problems, we propose 3 new schemes of Feature Point Detection Part implementation, that is, Variable Pixel Representation, Skip Mode Prediction and Parallel Hardware Architecture.

4.1 Variable pixel representation (VPR)

To solve the problem of redundant register consumption, we propose to use different numbers of registers to represent a pixel in different processes throughout the computation (Fig. 2).

The merit of doing this is that,

1) Fewer registers are consumed. In some part of the whole process, less accuracy may yield same result, e.g. the result is the same when we use 5 registers to represent a pixel in the Contrast Pre-Elimination Part as the 11-register case.

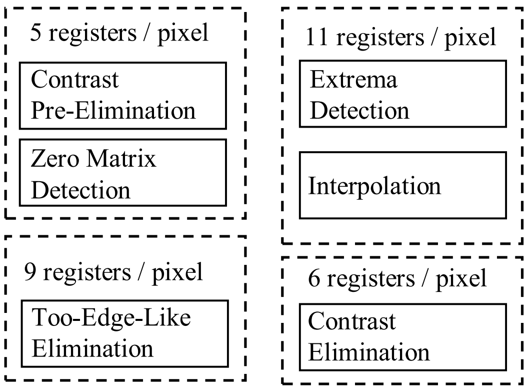

Figure 2 Variable Pixel Representation

2) Fewer gates are consumed. This is a consequence of reduced number of registers.
3) Time consumption slightly down-goes. Critical path can be shortened with smaller gate consumption.

4.2 Skip mode prediction (SMP) with zero matrix detection

As analyzed in SECTION 3, we can find out that by pre-computing the result of Contrast Pre-Elimination and Extrema Detection, we can skip a very large number of redundant computations.

Actually, we need not to compute to whole process to decide a certain pixel is useful or not. We can pre-compute the result and skip the computation of a certain point. This although does not bring advantage to shortening critical path, it does help to reduce average computation time (Fig. 3).

In our proposal we also combine the Zero Matrix Detection into the prediction. Zero Matrix Detection is to find out the potential Zero Matrix which is not able to create its Inversed Matrix.

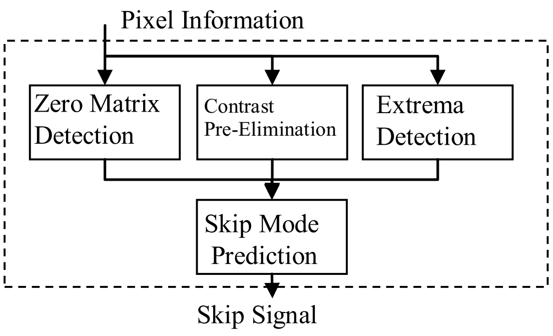

Figure 3 Block Diagram for Skip Mode Prediction

We use the Hessian Matrix to determine potential Zero Matrix as follow (Fig. 4),

$$|H| = \begin{vmatrix} Ixx & Ixy & Ixs \\ Ixy & Iyy & Iys \\ Ixs & Iys & Iss \end{vmatrix} = 0 \qquad (1)$$

Where *Ixx=(P15+P13)-(P14+P14); Iyy=(P16+P11)-(P14+P14); Iss=(P23+P5)-(P14+P14); Ixy=((P18-P16)-(P12-P10))>>2; Ixs=((P24-P22)-(P6-P4))>>2; Iys=((P26-P20)-(P8-P2))>>2;*

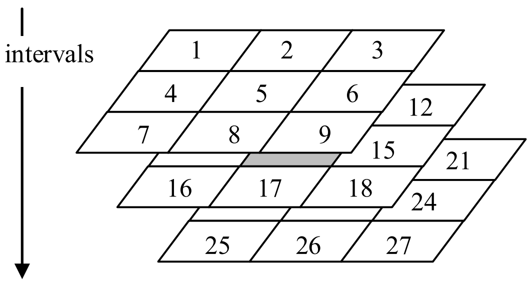

Figure 4 Pixel Arrangements for Hessian Matrix Computation

The function of Zero Matrix Detection is to eliminate error detection of feature point. The reason for using Hessian Matrix is that, Hessian Matrix shows the relationship among the 3D pixel area. When Hessian Matrix is a Zero Matrix, this would mean the current pixel area contains mostly pixels with similar values. However, feature point is not likely to be in these areas. So it is reasonable for us to apply Zero Matrix Detection.

4.3 Parallel HW Architecture

By introducing the former two schemes, we re-arrange the blocks in a parallel way as follows (Fig. 5),

We use the result from the Zero Matrix Detection, Contrast Pre-Elimination, and Extrema Detection to generate a Skip Signal which directly links to FP Memory Part as an asynchronized Control Signal, which directly generates a FINISH signal to the DoG Memory Part to indicate reloading of the next pixel.

This scheme shortens the critical path greatly because many computational stages are computed at a time instead of computing sequentially.

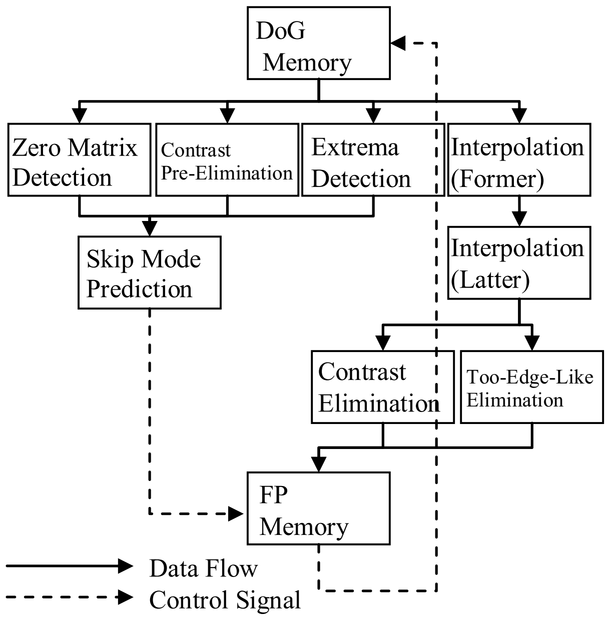

Figure 5 Parallel Architecture Modification

5. Result & Analysis

A software simulation using proposed modification is shown as below (Fig. 6 & Tab. 2).

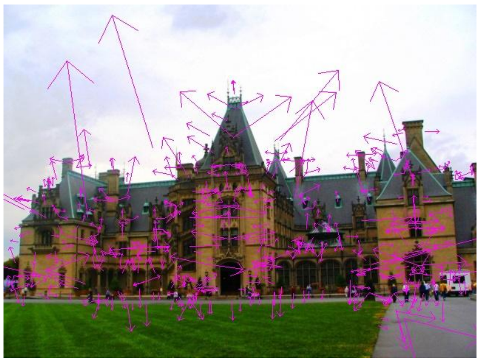

Figure 6 Software Simulation of Proposed Schemes

Table 2 Parameters & Data of Proposed Modification

Image Size	640x480		
Octaves	6	Intervals	3
Interpolation	1	#Feature Point	514
Efficiency*	98.44%	Cover Rate**	93.05%

* Efficiency is defined as division of number of effective feature points by number of total feature points. The higher efficiency is, the less time can be spent in redundant calculation.
** Cover Rate indicates how many percentages the detected feature points covers those feature points detected by original program.

By comparing our result with [2], we may find out that our advanced proposal does not only excel [2] by the means of less consumption of registers and gates, but also by the means of higher accuracy, higher efficiency and higher cover rate (Fig. 7 & Fig. 8 & Tab. 3).

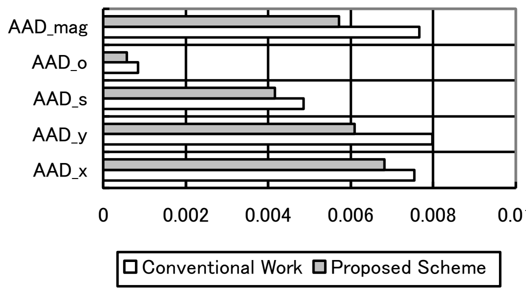

Figure 7 AAD (Averaged Absolute Difference) Comparison with Conventional Work*

*ADD is designed to compute how much the results from the proposed scheme are different from the original ones. Definition of ADD is as follows. Psw(i) is a certain value of the ith corresponding feature point by the original software solution; Phw(i) is a certain value of the ith corresponding feature point by the hardware solution; n is the total number of corresponding feature points.

$$AAD = \sqrt{\frac{\sum_{i=1}^{n} \left(P_{SW(i)} - P_{HW(i)}\right)^2}{n}} \qquad (2)$$

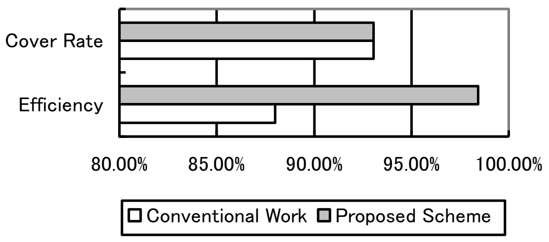

Figure 8 Cover Rate & Efficiency Comparison with Conventional Work

Table 3 Parameters of FPGA Implementation*

Image Size	640x480		
Item	Conventional Work	Proposed Scheme	Improved Ratio
Slice Flip Flop	2482	2134	13.9%
LUT	4896	4228	13.6%
Max Clock Frequency	68.0MHz	119.0MHz	75.0%

* Hardware design is based on Altera FPGA board and software environment of Quarters II.

6. Conclusion

As shown in the last SECTION, we can see that by introducing our proposal, the performance of Feature Point Detection Part has overall improvement over conventional work. We are able to achieve 10-20% higher accuracy, 10.4% higher efficiency, 13% of fewer registers and gates, and 75.0% of higher clock frequency.

Acknowledgement

This work was supported by fund from MEXT via Kitakyushu innovative cluster projects and CREST, JST.

References

[1] David G. Lowe, "Object recognition from local scale-invariant features" *International Conference on Computer Vision,* Corfu, Greece, pp. 1150-1157, Sep. 1999

[2] Jingbang QIU, Tianci HUANG, Takeshi IKENAGA, "Hardware Accelerator for Feature Point Detection Part in SIFT Algorithm & Corresponding Hardware-Friendly Modification", SASIMI 2009

[3] Konstantinos Dermitzakis, Supervisor: Eric McKenzie, "A GPU implementation of the SIFT algorithm: An MSc Project Proposal" www.inf.ed.ac.uk/events/jamboree/2007/Posters/k-dermitzakis.pdf

[4] David G. Lowe, "Distinctive image features from scale-invariant keypoints" *International Journal of Computer Vision,* 60, 2, pp. 91-110, 2004

[5] Sudipta N.Sinha, Jan-Michael Frahm, Marc Pollefeys, and Yakup Genc, "GPU-based Video Feature Tracking and Matching" Technical Report TR 06-012, Department of Computer Science, UNC Chapel Hill, May. 2006

[6] S. Heymann, K. Muller, A. Somlic, B. Frohlich, and T. Wiegand, "SIFT Implementation and Optimization for General-Purpose GPU", Journal of WSCG 15,Nr.1-3, 2007