International Journal of Control and Automation
Vol.7, No.3 (2014), pp.213-218
http://dx.doi.org/10.14257/ijca.2014.7.3.20

A Design of an Optimized ORB Accelerator for Real-Time Feature Detection

Kwang-yeob Lee

Dept. of Computer Engineering, Seokyeong University,
Jeongneung 4-dong, Seongbuk-gu, Seoul, Korea

kylee@skuniv.ac.kr

Abstract

In this paper, we propose an ORB hardware accelerator design. Feature-based SIFT and SURF hardware accelerators that detect features in real-time require high hardware costs. To overcome this disadvantage, an ORB algorithm consisting of simple calculations was designed as the hardware accelerator. A comparison between the proposed ORB hardware accelerator with SIFT and SURF accelerators revealed that the block RAM usage of the former decreased by approximately 25% and 83%, and its slice logic usage decreased by approximately 40% and 68%, respectively. As approximately 9 ms were required for processing, it was verified that the proposed hardware accelerator could perform real-time feature detection.

Keywords: *FAST corner detection, rBRIEF descriptor, hardware, computer vision*

1. Introduction

Many stuides on automatic detection and tracking of a certain object that exists in an image are being conducted in various fields including computer vision and pattern analysis. In particular, with the increased widespread use of mobile devices such as smart phones, which have built-in high-performance cameras, image recognition applications using the camera are being actively developed. Further, feature-based SIFT and SURF [1] are well-known algorithms for object recognition. However, SIFT and SURF algorithms require a large number of calculations in the process of feature detection and descriptor generation. To supplement this weakness, studies on designing SIFT and SURF algorithms using hardware accelerators are ongoing. However, such designs require high hardware costs. In this paper, we propose a method to design an optimized ORB algorithm [2] that fits the embedded environment instead of SIFT and SURF algorithms.

The feature-based recognition algorithm was designed by a simple calculation using the FAST corner detection algorithm and the rBRIEF algorithm to reduce the size of the hardware and use in an embedded system. Further, the zynq-7000 SoC platform was used for confirming the performance of the designed hardware. A comparison with the SIFT hardware and the SURF hardware accelerator revealed that the internal memory usage of the proposed hardware accelerator decreased by approximately 25% and 75%, respectively, and its slice logic usage decreased by approximately 40% and 26%, respectively.

ISSN: 2005-4297 IJCA

2. Feature-based Algorithm

The ORB algorithm is composed of the FAST algorithm and the rBRIEF algorithm. The FAST algorithm detects feature point and determines the main orientation on the basis of the intensity centroid. Then, the descriptor is generated using the feature point and the main orientation in the rBRIEF.

2.1. Feature detection

A feature point detector uses the FAST corner detection algorithm [3]. It consists of feature detection, feature score, and non-maximum suppression. In the FD stage, the candidate feature point is detected on the basis of the following order.

First, A 7 × 7 area is allocated around the center point *(P)*, as shown in Figure 1. Second, Four brightness values (I1, I5, I9, I13) located at No. 1, 5, 9, and 13 around the center point are compared using Formula (1). In Formula (1), I_i denotes the brightness value from No. 1 to 16, and I_p represents the brightness value of the center point. Third, if the sum of the compared values is more than 3, the brightness values from No. 1 to 16 are also compared using the same method. Finally, if the sum of the compared values is more than 12, it is detected as the candidate feature point.

$$\begin{cases} I_i \le I_p - t & (Darker) \\ I_p - t < I_i < I_p + t & (Similar) \\ I_p + t \le I_i & (Brighter) \end{cases} \qquad f(I_i) = \begin{cases} 1, I_i, if\ D\ or\ B \\ 0, I_i, if\ S \end{cases} \tag{1}$$

In the FS stage, a score for each feature point candidate is determined based on threshold values. In the NMS stage, the scores of neighboring feature points are compared to the reference feature point and the largest feature point set as the final feature point. [4]

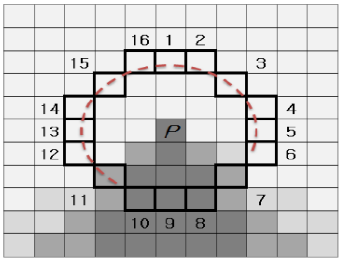

Figure 1. Area allocation around the center point

2.2. Orientation

The main orientation of each feature point is determined by the intensity centroid [5]. First, the 15 × 15 area, a half patch size, is set around the feature point. Then, as shown in Formula (2), the center point of brightness can be obtained using moment method.

$$m_{pq} = \sum_{x,y} x^p\, y^q I(x,y)$$
$$m_{00} = \sum_x \sum_y I(x,y), m_{10} = \sum_x \sum_y xI(x,y), m_{01} = \sum_x \sum_y yI(x,y) \tag{2}$$

In Formula (3), C refers to the center point of brightness. As indicated in Formula (3), the center point of brightness and its angle can set as the main orientation of each feature point by using the first moment function and arctan2 for each feature point.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right), \theta = atan2(m_{01}, m_{10}) \tag{3}$$

2.3. Descriptor generation

The rBRIEF algorithm is an improved version of the BRIEF algorithm and is used for descriptor generation. The BRIEF algorithm [6] creates a bit string descriptor consisting of 0 and 1.

$$\tau(p; x, y) := \begin{cases} 1 : p(x) < p(y) \\ 0 : p(x) \geq p(y) \end{cases}$$
$$f_n(p) := \sum_{1 \leq i \leq n} 2^{i-1} \tau(p; x_1, y_1) \tag{4}$$

In Formula (4), τ refers to a binary test, and n to vector length. $p(x)$ denotes the brightness value at point x, and $p(y)$ represents that at point y. if $p(x)$ is smaller than $p(y)$, the binary test value is 1; if not, it is 0. After repeating this method n times, we obtain n binary tests.

While conventional studies used 32 registers in a byte format, this study used a total of 8 registers in an integer format and saved information on the 256-bit string descriptor.

When the binary descriptor is generated using the main orientation of the feature point and rotating all comparison pixels, 0 and 1 are not equally distributed in the bit string descriptor, leading to a matching failure. To solve this problem, learned coordinates are created and used instead of the random coordinate. The learned coordinates are composed of 256 pixel pairs that can equally distribute 0 and 1 when the bit string descriptor is generated around the feature point.

3. Feature Detection Hardware Design

In this study, the hardware accelerator was designed on the basis of the ORB algorithm. Accordingly, image data appropriate for each scale are loaded from an external memory, and the coordinates of the feature point are detected in the FAST corner detection block. When the feature point is detected, the main orientation of the feature point is determined in the orientation block. Meanwhile, the descriptor suitable for the coordinates of the feature point is generated, and the data are saved in a buffer in the rBRIEF block.

3.1. Feature detection block

The process of detecting feature point is performed in the FAST corner detection block, as shown in Figure 2.

In the source loader stage, the brightness values of images are loaded one by one from the external memory. In the FD stage, the candidate feature point is examined, and the determined candidate feature point is saved in the FIFO.

In the FS stage, when the candidate feature point is detected, the coordinates of the feature point and the value of pixel brightness are loaded to set the score of the feature point. In the NMS stage, the coordinates and the score of the feature point are loaded from the buffer, and then its score is compared with that of the neighboring feature point. If the score of the feature point is higher than that of the neighboring feature point, it is examined in the process of

detecting the final feature point. To detect the feature point, the image data should be saved and the necessary pixel data should be loaded and processed using the internal memory.

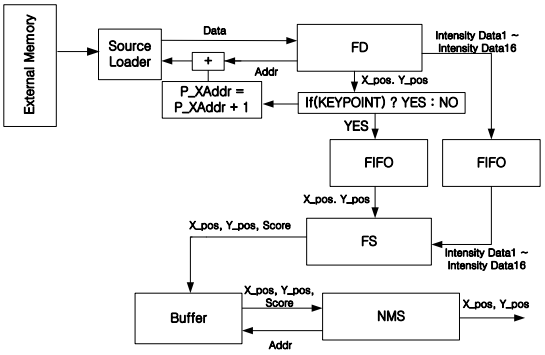

Figure 2. FAST Corner detection block

A dual-port BRAM method that loads the data from the 7-line memory and saves them in the 1-line memory was used. The memory usage was reduced by saving all the image data in the 8-line memory and then resaving the image data from the first storage place.

3.2. Orientation block

The process of determining the main orientation is performed in the orientation block. The coordinates of the detected feature point that are saved in the FIFO are used for calculating the first moment using the pixel data of brightness around the feature point detected in the orientation module. Cosine and sine, the main orientation, can be calculated through the first moment by using Arctan2 LUT. To calculate the first moment, 32 brightness pixels are loaded from the internal memory at once by using the 32-line memory. As Arctan2 LUT created a table using the necessary data to obtain the cosine and sine values in advance, the execution time was improved and the LUT usage increased.

3.3. Descriptor generation block

A descriptor generation block is a stage that generates the descriptor for each feature point; its performance process is illustrated in Figure 3.

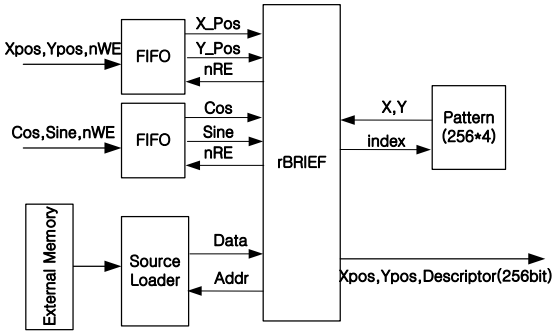

Figure 3. Descriptor generation block

First, the 31 × 31 area is allocated on the basis of the coordinates of the feature point. By calculating the generated orientation data and the learned coordinates loaded from the pattern

LUT, the comparison coordinates are obtained. Then, brightness pixels are compared by allocating the comparison coordinates on the basis of the coordinates of the feature point. On the basis of the comparison result, a descriptor consisting of 1 and 0 is generated. While the process should be repeatedly performed 256 times in total to create the 256-bit string descriptor, a total of 8 pixel pairs can be compared at once by adding three more internal memories. Thus, by repeating the process 32 times in total, a 256-bit string descriptor can be created.

4. Experiment and Result

The experimental environment was realized using the zynq-7000 SoC platform in which the ARM Cortex-A9 CPU and the Artix-7 FPGA are built. it detected by the ORB hardware accelerator using graffiti and sunflower images. Approximately 600 to 800 features were detected. Table 1 shows the result of the comparison between the proposed ORB hardware accelerator and the conventional SIFT hardware accelerator [7, 8].

Table 1. Performance comparison with the SIFT hardware accelerator

	[7]	[8]	Proposed hardware
Resolution	320x240	640x480	640x480
Block RAMs	168 KB	405 KB	125 KB
Register	19,100	19,529	6,411
LUTs	43,366	35,889	31,677
Clock cycle Per pixel	21 cycle	10 cycle	3 cycle

The block RAM usage of the proposed ORB hardware accelerator decreased by approximately 25% and 70% as compared to that of the SIFT hardware accelerator. the slice logic usage of the former decreased by approximately 39% and 31% as compared to that of the latter. As the operation frequency varied for each realized case, a clock cycle required for processing a single pixel was calculated by dividing (Frequency × Execution Time) by (Image Resolution). As the execution time of [7] is approximately 33 ms at the frequency of 50 MHz, 21 clock cycles/pixel. As the execution time of [8] is approximately 33 ms at 100 MHz, 11 clock cycles/pixel. As the execution time of the proposed hardware is approximately 9 ms at 100 MHz, 3 clock cycles/pixel. Thus, the latter shows three to seven times better performance improvement than the former.

Table 2 indicates the comparison result with the SURF hardware accelerator [9, 10]. The comparison results show that the block RAM usage of the proposed hardware accelerator decreased by approximately 83% and 75% more than that of the SURF hardware accelerator, while the slice logic usage of the former decreased by approximately 68% and 26%. At 100 MHz, the SURF hardware accelerator required approximately 18 ms and 33 ms for the execution time, and the proposed hardware accelerator required approximately 9 ms, thus exhibiting a higher execution speed.

Table 2. Performance comparison with the SURF hardware accelerator

	[9]	[10]	Proposed hardware
Block RAMs	752 KB	515 KB	125 KB
Slice Logic	101,348	43,115	31,677
Exec.Time	33 ms	18 ms	9 ms

5. Conclusions

In this study, the ORB algorithm was designed as the hardware to operate in the zynq SoC environment. In this study, we designed the ORB hardware accelerator to reduce the internal memory usage and hardware costs. A comparison with the SIFT and SURF hardware revealed that the block RAM usage of the proposed hardware decreased by approximately 25% and 75% while its slice logic usage decreased by approximately 40% and 26%, respectively. As an execution time of 3 ms was required at 100 MHz, the proposed hardware exhibited the performance more than approximately 100 frames per second.

Acknowledgements

This Research was supported by Seokyeong University in 2012.

References

[1] H. Bay, T. Tuytelaars and L. Van Gool, "Surf: Speeded up ro-bust features", In European Conference on Computer Vision, vol. 3951, (**2006**), pp. 404-417.
[2] E. Rublee, V. Rabaud and K Konolige, "ORB: An efficient alternative to SIFT or SURF", Computer Vision and 2011 IEEE International Conference, Barcelona, (**2011**) November 6-13.
[3] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection", In European Conference on Computer Vision, vol. 3951, (**2006)**, pp. 430-443.
[4] L. Kwang-yeob and B. Kyung-jin, "A hardware design of optimized ORB algorithm with reduced hardware Cost", Advanced Science and Technology Letters, vol. 43, (**2013**), pp. 58-62.
[5] P. L. Rosin, "Measuring corner properties", Computer Vision and Image Understanding, vol. 73, **(1999)**, pp. 291-307.
[6] M. Calonder and V. Lepetit, "Brief: Bi-nary robust independent elementary features", In European Conference on Computer Vision, vol. 6314, (**2010**), pp. 778-792.
[7] V. Bonato and E. Marques, "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection", IEEE Transactions on Circuits and Systems for Video Technology, vol. 18, **(2008)**, pp. 1703-1712.
[8] 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", IEEE Transactions on Circuits and Systems for Video Technology, Sydney, (**2009**) December 9-11.
[9] N. Eun-soo and J. Yong-jin, "FPGA Implementation of SURF-based Feature extraction and Descriptor Generation", Journal of Korea Multimedia Society, vol. 16, (**2013**) April, pp. 483-492.
[10] D. Bouris, A. Nikitakis and I. Papaefstathiou, "Fast and efficient FPGA-based feature detection employing the SURF algorithm", FCCM, Charlotte, (**2010**) May 2-4.

Authors

Kwang-yeob Lee

He received a B.S. degree in the Department of Electronics Engineering at Sogang University in 1985. He received M.S. and Ph.D. degrees in the Department of Electronics Engineering at Yonsei University in 1987 and 1994. He is a professor in the Department of Computer Engineering at Seokyeong University since 1995. His research interests include Microprocessor, Embedded System, Image recognition, and 3D Graphics System.

