Hardware Architecture of FAST Algorithm for Feature Point Detection

Chang-Sue Seo[1], Hoon-Ju Chung[1], Sung-Young Kim[1], Sangook Moon[2] and Yong-Hwan Lee[1]
Kumoh National Institute of Technology, Korea[1] and Mokwon University, Korea[2]
scs@kumoh.ac.kr, hjchung@kumoh.ac.kr, sykim@kumoh.ac.kr, smoon@mokwon.ac.kr and yhlee@kumoh.ac.kr

ABSTRACT

In this paper, we present method that detects useful feature points based on hardware architecture. We propose hardware architecture that uses the algorithm of FAST-n[1]. Feature point detection process needs extensive computing power and processing time. Therefore, we build hardware structure for real-time processing. The structure of the hardware is as follows. After loading the images in parallel, finding feature point candidates and selecting valid feature point modules operate simultaneously and independently using pipeline structure to reduce processing time. Proposed hardware architecture will operate in about 20,000 cycles in case of 320 x 240 resolution image. If our hardware structure is used for 1080p, the performance of processing will be about 70fps.

KEYWORDS

FAST, Corner detection, Edge detection,
Feature point, Hardware architecture

1 Introduction

Task of extracting feature points is the first step of many vision tasks such as object tracking, SLAM (simultaneous localization and mapping), localization, image matching and recognition. To extract feature points, a number of algorithms have been studied[1-7].

Most of the algorithms require extensive computational cost and time. These methods are not suitable for real-time processing of images.

Therefore, we propose hardware structure for real-time processing application. The hardware implementation is faster and requires fewer resources than software structure. The structure of the hardware is as follows. After loading the images in parallel, three modules which search the feature point candidates module, compute score of each feature point and select effective feature point in the feature point candidates module operate simultaneously and independently using pipeline structure. As a result, faster operation than software can be achieved.

2 FAST Algorithm

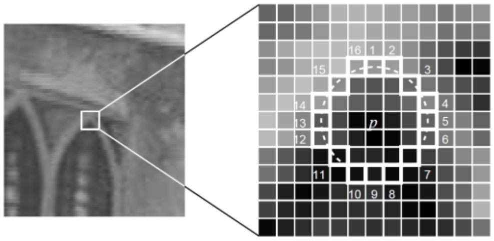

Figure 1. Segment test corner detection in an image patch

FAST algorithm is abbreviation of features from accelerated segment test. It loads 16 pixels around the circular for a single reference pixel P as shown Figure 1. At this time, as formula 1, each pixel is compared if it is greater than plus the threshold value to the reference pixel, or if it is smaller than the reference pixel minus the threshold value. The result of comparison will be divided into three states, point darker than the reference pixel, similar point and point brighter than the reference pixel.

$$S_{p\to x} = \begin{cases} d, & I_{p\to x} \le I_p - t \quad (dark) \\ s, & I_p - t < I_{p\to x} < I_p + t \quad (similar) \\ b, & I_p + t \le I_{p\to x} \quad (bright) \end{cases} \qquad (1)$$

The number of continuous dark or bright pixels determines the constant n of FAST-n algorithm. The FAST-n algorithm usually detects 7 to 12 consecutive pixels.

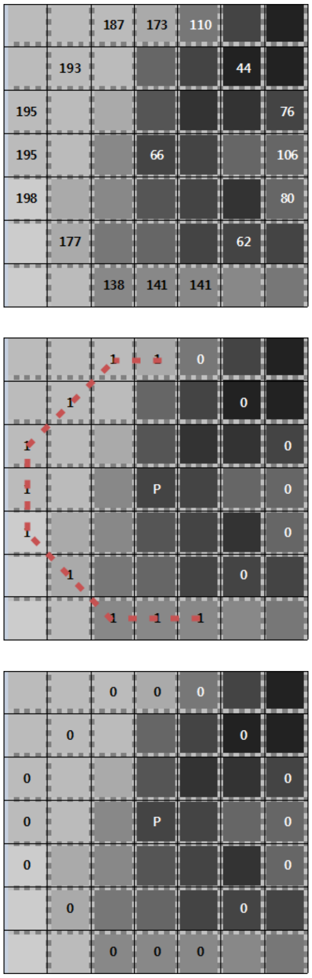

Figure 2. Center pixel and neighbor pixels (top: pixel value, middle: bright, bottom: dark)

Figure 2 illustrates how to classify pixel. The around pixels are compared to the value of center pixel plus threshold value 64. Then the around pixels have the value of 0 or 1 according to the differences.

Decision tree scheme is used for detecting continuous pixel, as shown in the Figure 3 [8].

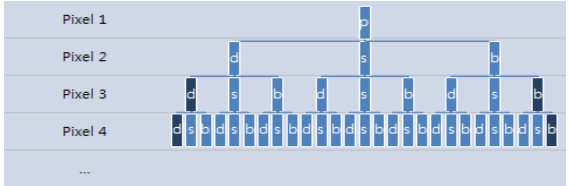

Figure 3. Decision tree of dark point, similar point, bright point

When this condition is satisfied, the reference pixel is selected as the feature point candidate. In this case, detected feature point candidates are often located around detected feature points because of characteristics of FAST algorithm. However FAST algorithm requires post-processing called NMS (non-maximal suppression). NMS selects valid feature points among the detected feature point candidates. Through formula 2, the maximum threshold that meets the conditions of the feature point candidate is calculated and stored. There are several intuitive definitions for V:

1. The maximum value of n for which p is still a corner.
2. The maximum value of t for which p is still a corner.
3. The sum of the absolute difference between the pixels in the contiguous arc and the center pixel.

Definitions 1 and 2 are very highly quantized measures, and many pixels share the same value of these. For speed of computation, a slightly modified version of 3 is used. V is given by : [1]

$$V = \max\Big(\sum_{x \in S_{bright}} |I_{p\to x} - I_p| - t,\ \sum_{x \in S_{dark}} |I_p - I_{p\to x}| - t \Big) \qquad (2)$$

To detect a valid feature points, only the feature point candidate which has the maximum value compared with others are left and the rests are removed.

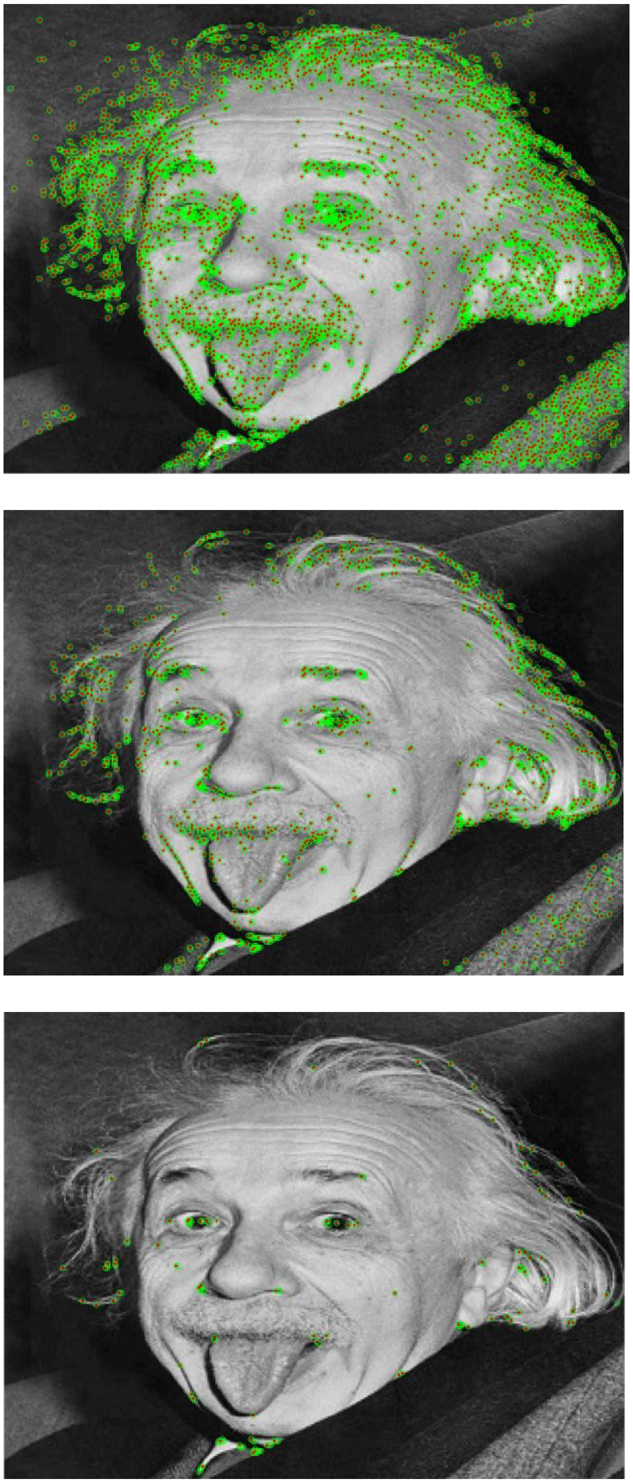

Figure 4. Result Einstein image of FAST-9 algorithm in case of different threshold (top 16, middle 32, bottom 64)

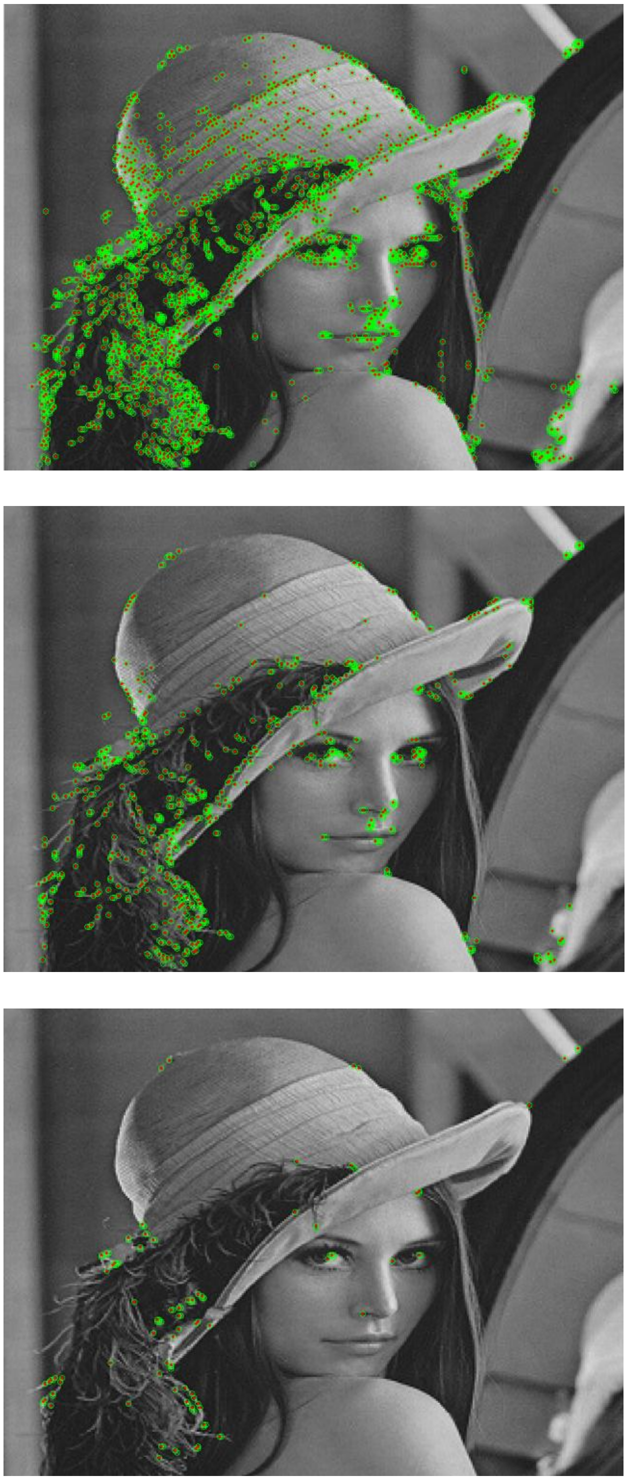

Figure 5. Result Lena image of FAST-9 algorithm in case of different threshold (top 16, middle 32, bottom 64)

Figure 4 and 5 are the result of FAST-9 algorithm simulation in Matlab. Size of the image is 320 x 240. Green circles are feature points candidate and red dots are NMS processed feature points.

Table 1. Result from Lena image of FAST-9 algorithm

Threshold	8	16	24	32
FD	6,456	2,732	1,418	777
NMS	2,654	1,239	704	419
per	41.11	45.35	49.65	53.93
Threshold	40	48	56	64
FD	431	267	171	120
NMS	249	160	105	73
per	57.77	59.93	61.40	60.83
Threshold	72	80	88	96
FD	70	49	27	19
NMS	45	33	22	17
per	64.29	67.35	81.48	89.47
Threshold	104	112	120	128
FD	12	8	6	3
NMS	11	7	5	3
per	91.67	87.50	83.33	100.00

Table 1 illustrates the result of FAST-9 algorithm applied to Lena image with varying threshold from 8 to 128. FD is number of feature point before NMS processing. NMS is number of feature point after NMS processing. Per is the percentage of NMS processed feature point number of feature point candidate number. It is shown that as number of feature point decrease as threshold increase.

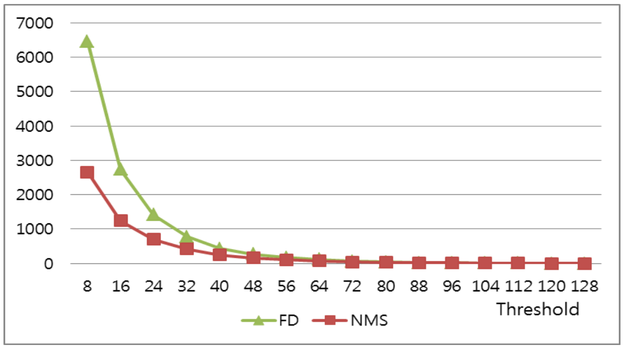

Figure 6. Result Lena image of FAST-9 algorithm in case of different threshold from 8 to 128

Figure 6 illustrates the count of feature point candidate and NMS processed feature point.

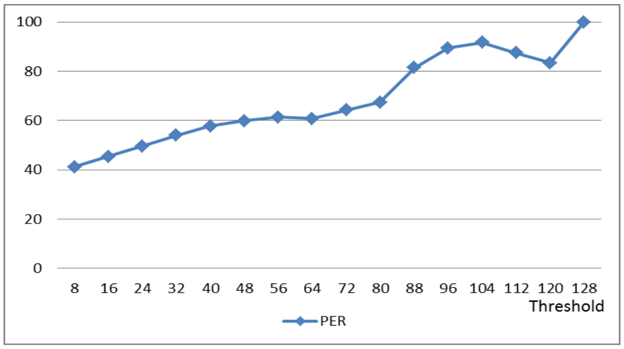

Figure 7. Result Lena image of FAST-9 algorithm in case of different threshold (top 16, middle 32, bottom 64)

Figure 7 illustrates the percentage of NMS processed feature point number of feature point candidate number. It is shown that as threshold increase, so do percentage.

3 Hardware Structure of FAST Algorithm

The Figure 8 illustrates the structure of software briefly. First, the image to gray scale is performed. Then FD function, FS function, NMS function are called in sequence.

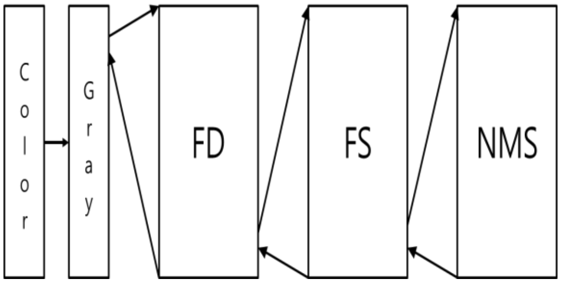

Figure 8. Software structure of FAST-n

Figure 9 illustrates overall hardware structure. Divided gray scale image is stored in block RAM. Then pipeline structure operates with the stages of FD, FS and NMS modules simultaneously and independently.

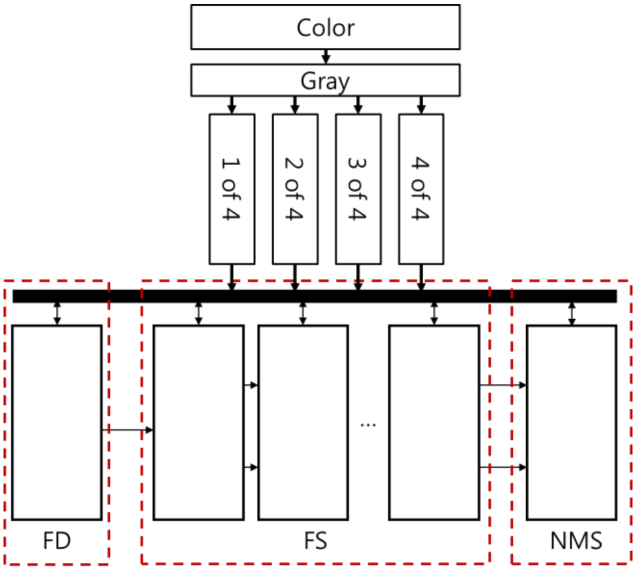

Figure 9. Hardware structure of FAST-n

When the center pixel and the neighbor pixels in the block RAM are loaded, neighbor pixels are made to vector as shown in Figure 10 so that vector is processed easily in hardware.

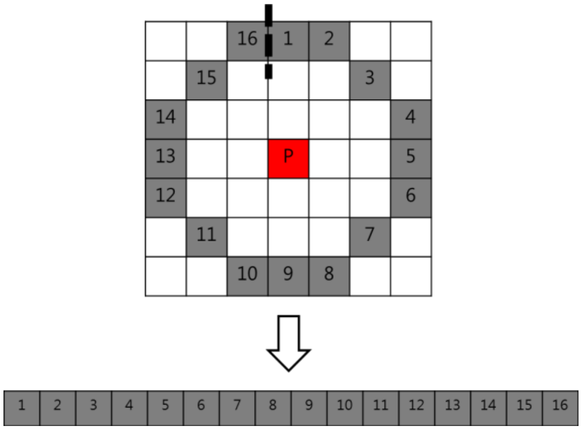

Figure 10. Aligned arc pixel

Figure 11 illustrates block diagram of FD (feature detection) module. Adder and subtractor are used to compute values from pixel and threshold. Values are compared with values in vector. If those results consist of continuous 1s of which the number is more 9, reference pixel would be a feature point candidate.

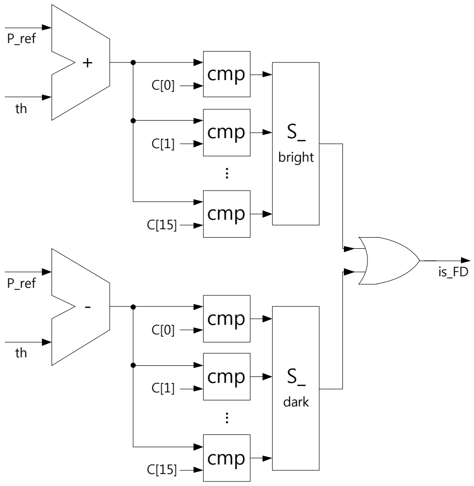

Figure 11. Block diagram of FD (feature detection) module

Figure 12 illustrates block diagram of FS (feature score) module. Score is calculated using bmin and bmax. That module requires from minimum 0 to maximum 8 cycles to find accurate score. However, because overall system takes much time due to this operation, we changed the architecture from repetitive FS to a series of FS through pipeline. Flip flop is added to series of FS in each FS module as shown Figure 13.

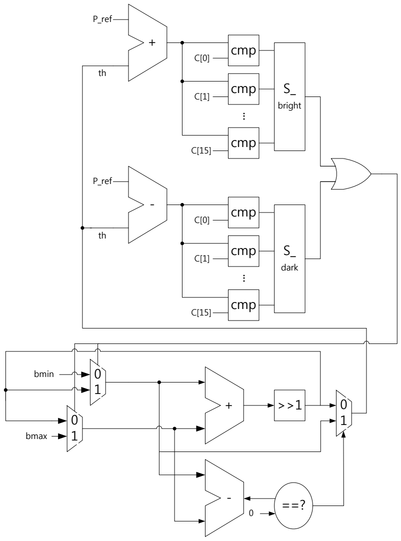

Figure 12. Block diagram of repetitive FS (feature score) module

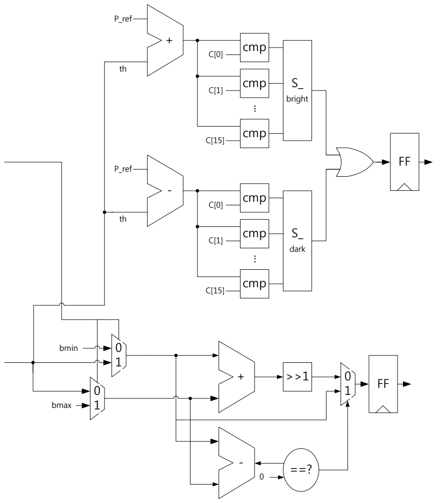

Figure 13. Block diagram of revised FS (feature score) module

Figure 14 illustrates the block diagram of NMS (non-maximal suppression) module. Calculated reference pixel score is stored to memory and compared with adjacent pixel score. If reference pixel score is the highest, reference pixel would be a valid feature point.

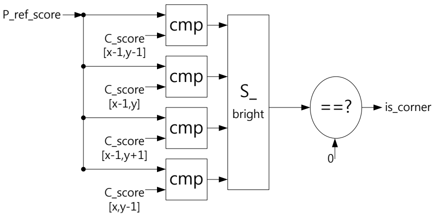

Figure 14. Block diagram of NMS (non-maximal suppression) module

4 Conclusions

We proposed hardware architecture of FAST algorithm for real-time processing. The pixel of gray scale image are divided and stored in block RAM. Pipeline structure is applied to FD (Feature Detection) module, FS (Feature Score) module and NMS (Non-Maximal Suppression) module in order to operate simultaneously and separately.

Proposed hardware architecture will operate in about 20,000 cycles in case of 320 x 240 resolution image. If our hardware structure is used for 1080p, the performance of processing will be about 70fps.

Object tracking is a key component in the system of caring companion animals. Proposed method would play a key role for efficient and fast tracking in the video module of the caring system.

ACKNOWLEDGEMENT

This work (Grants No. C0217661) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2014.

REFERENCES

[1] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” European Conference on Computer Vision, vol. 3951, pp. 430-443, May 2006.

[2] D. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” International Journal of Computer Vision, vol. 60, pp. 91-110, November 2004.

[3] C. Harris and M. Stephens, “A Combined Corner and Edge Detector,” Alvey Vision Conference, vol. 15, pp. 147-151, 1988.

[4] H. Bay, T. Tuytelaars, and L. Gool, “SURF: Speeded-Up Robust Features,” European Conference on Computer Vision, vol. 3951, pp. 404-417, May 2008.

[5] T. K. Kim, “An Embedded FAST Hardware Accelerator for Image Feature Detection,” Journal of The Institute of Electronics Engineers, vol. 49, pp. 28-34, March 2012.

[6] S. R. Kim, H. J. Yoo, and K. H. Sohn, “FAST and BRIEF based Real-Time Feature Matching Algorithms,” The Korean Society of Broadcast Engineers, vol. 2012, pp. 1-4, November 2012.

[7] H. Heo and K. Y. Lee, “FPGA based Implementation of FAST and BRIEF algorithm for Object Recognition,” Journal of IKEEE, vol. 17, pp. 202-207, June 2013.

[8] J.R. Quinlan, “Induction of decision trees,” Machine Learning 1, vol. 1, pp. 81-106, 1986.