High Accuracy and Simple Real-Time Circle Detection on Low-Cost FPGA for Traffic-Sign Recognition on Advanced Driver Assistance System

Anh-Tuan Hoang [1)], Masaharu Yamamoto [2)], Tetsushi Koide [1)2)]

1) Research Institute for Nanodevice and Bio Systems (RNBS)
2) Graduate School of Advanced Sciences of Matter,
Hiroshima University
Higashi-Hiroshima, Hiroshima 739-8527
e-mail : {anhtuan, yamamoto-masaharu, koide}@hiroshima-u.ac.jp

Abstract - This paper describes a hardware oriented algorithm and its conceptual implementation for real-time traffic signs detection system on automotive oriented FPGA. The speed limit sign area on a grayscale video frame is detected through a two-stage simple computation process. Rectangle Pattern Matching roughly detects global luminosity sharing feature between rectangle and circle for Region of Interest (ROI). Then, Circle Detection roughly votes local pixel direction of circle inside the detected ROI in binary image for circle confirmation. The proposed system achieves 83 full HD fps and over 99% accuracy even in difficult situation such as rainy night. It occupies around 50% the hardware available on proposed Xilinx Zynq automotive FPGA, which has 85 K logic cells, 53.2 K LUTs, 106.4 K registers and 506 KB BRAM, and so be able to apply to Advanced Driver Assistance System on common vehicles.

I. Introduction

The traffic sign recognition would be very important in the future vehicle Advanced Driving Assistance System. The most important information is provided in the drivers' visual field by the road signs, which are designed to assist the drivers in terms of destination navigation and safe. The most important of a car assistant system is to improve the drivers' safety and comfort. Although the navigation system is available, it cannot apply to the new roads or the place that the navigation signal cannot reach to or in electronic speed limit signs (LED), where the sign changes depend on the traffic condition. Hence, detecting the traffic signs is important in warning the drivers about changes in traffic situation and potential dangerous. An assistant system with speed limitation recognition ability can inform the drivers about the change in speed limit as well as notify them if they drive at over speed. Hence, the drivers' cognitive tasks can be reduced and safe driving is supported. Even several researches on traffic sign detection system are available; they occupy large hardware size, low in throughput, and accuracy, and so not yet meet the requirement of real-time processing, high accuracy and low-cost for widely used on common vehicles.

In this study, we aim to solve this challenge and perform real-time speed limit signs recognition on a low resources embedded platform. The targeted platform is the Xilinx Zynq 7020, which has 85K logic cells, 53.2K LUTs, 106.4K registers and 506KB BRAM. In order to get the goal, our proposed recognition method combines simple and rough global luminosity difference and local pixel direction features together to find location of the traffic sign before finely recognize the number inside. The design uses coarse grain pipeline together with parallel processing architecture. The recognition processing of each frame is coarsely divided into two pipeline stages. One is used for traffic signs and size detection and the other is used for number recognition. The traffic signs and size detection raster scans each frame by various scan windows sizes in parallel, calculates the rough luminosity difference feature to find the sign candidate (Region Of Interest – ROI). Fine grain processing stage then processes the ROI for number recognition. The proposed method is able to recognize speed limit traffic sign (both fixed and LED signs) in day time, at night, and rainy night.

The related works in traffic signs recognition is shown in section II. The speed limit recognition system architecture and overview of related algorithms are shown in section III. Section IV emphasizes to the implementation of circle detection using simple pixel local direction voting, which is the main of this paper. Sections V and VI show the evaluation result and conclusion of the proposed architecture.

II. Related Works

In algorithm point of view, most traffic signs systems is structured into pre-processing the obtained frames with various segmentation techniques for signs detection before recognizing and classifying them. In performance point of view, several implementations on hardware are given to increase the processing time.

A. Algorithm

Detection algorithm is designed based on color, shape or template matching. Traffic signs are pre-decided in color to ensure the focus of the drivers, and so, color can be used in image segmentation. This trend can be found in [1], in which a red circle is examined as Region of Interest (ROI) or in [3], where white circular region is searching with some thresholds.

This detection method required colored camera and more computation resources such as memory for colored image storing and detection. Shape based detection approach can be seen in [4] and [5], in which complex Hough-transform is used for circle detection and SIFT feature computation is required. This method is robust to change in illumination but complex in computation for detection. Template matching in [1] uses the pre-prepared template for area comparison with various sizes from 32x32 to 78x78. Huge hardware resources and computation time are required for the comparison.

B. Hardware Implementations of Traffic Sign Recognition

Several hardware implementations or hardware accelerations for traffic sign recognition system have introduced on FPGA. Yan Han [13] introduces a hardware/ software implementation based on Xilinx Zynq SoC. Souani [10] implemented a neuron network for shape recognition on RGB image on Virtex 4 device. Imark [12] implemented on FPGA a system with edge detection, circle detection, triangle detection and rectangle detection based on angle before matching features of templates and candidate for recognition. Muller [11] introduced a recognition system, which aims to multi-core hardware/software on FPGA. However, they are implemented on big hardware such as Virtex 4 family, working with small image size, and so do not match requirement of real-time processing at low cost for really use in general vehicles.

III. Compact Hardware Oriented Speed Limit Recognition System with Circle Detection

In order to overcome the weakness in hardware size and computation time of the related work, we would like to introduce a novel algorithm and architecture for traffic sign candidate detection, which combines a simple global luminosity difference feature and simple local pixel direction together for circle shape sign detection. It can be applied to speed limit detection system as shown in Fig.1. Input data are 8-bit grayscale frames. Due to the simplicity of the algorithm, it supports for real-time system.

A. System Overview

Fig.1 shows the speed limit recognition system overviews with two special filters and three computation modules.

A simple noise reduction filter is applied to the gray-scale input frame for image correction and noise reduction. The sign enhancement filter, which is a special filter for sign enhancement and image binalization purposes, is then applied to the output of the noise reduction filter, changing the 8-bit value grayscale pixels to 1-bit value black-and-white pixel for number recognition. The sign enhancement process helps increase the features of the numbers and reduces the amount of data being processed.

Rectangle Pattern Matching step roughly detects the traffic sign candidates based on the common features of rectangle

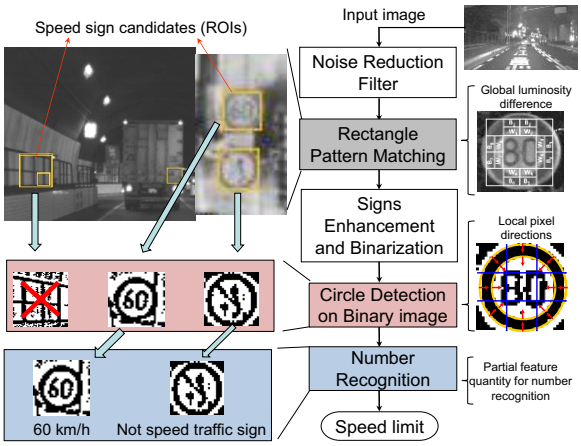


Fig. 1. Speed limit recognition system.

and circle, that is the difference in the brightness between the rectangle / circle line and its neighbors [8] as shown in Fig.2. The scan window corresponds with the traffic sign is processed in 16 areas, from B1 to B8 (says for Black) and W1 to W8 (says for White). The differences in brightness of corresponding Black and White areas (B1 and W2, B2 and W2, etc.) are computed. If they get over a threshold, the pattern is considered as a rectangle. At different view points, the brightness of the black and white areas is different. In our implementation, the threshold is considered as a variable, which relies on the different in brightness between W1 and B1.

The circle detection, which is the main subject of this paper, reduces the number of traffic sign candidates detected by rectangle pattern matching, decides if the detected regions of interest (ROIs) are really a circle mark or not using simple pixel local direction.

Finally, the number recognition module analyses simple features such as histogram and run length of black and white pixels of the ROIs and compares with those features of reference numbers for number recognition [7].

B. Raster Multi-Scan Windows

Our system uses input image size of 640x390. Input size of the traffic sign is in a range from 18x18 to 50x50 pixels. We use raster scanning method with the above scan window sizes as shown in Fig.3 to keep the processing time a constant. At any clock, when a new pixel (x,y) gets into the system, a

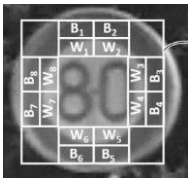


Fig. 2. Rectangle pattern matching for traffic sign detection.

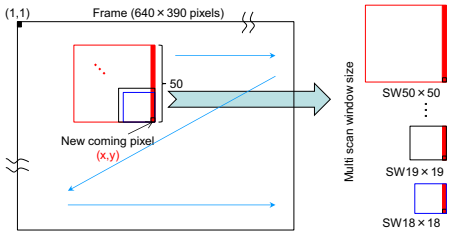


Fig. 3. Raster scan with multi-scan window.

column of 50 pixels from (x,y) will be read from FIFO for traffic sign candidate detection. In circle detection module, three last continuous columns of the scan window are buffered in registers for pixel local direction detection.

C. System Pipeline Architecture

Fig.4 shows the pipeline architecture of the speed limit recognition system. The system is able to scan for the traffic signs up to 50x50 pixels in size. The input image is 8-bit grayscale with the size of 640x390=249,600 pixels.

It contains three main modules of Rectangle Pattern Matching (RPM), Circle Recognition and the Number Recognition (NR). Other modules are Noise Reduction Filter, and Sign Enhancement Filter. Supported for the Noise Reduction Filter and the RPM are a number of 8-bit FIFOs.

The Sign Enhancement Filter is independent with the RPM processing, and so could be processed in parallel with the RPM. Different from the software implementation, in which the Circle Recognition is used to reduce the number of sign candidates, in the hardware implementation, the circle recognition result is used to strengthen the judgment of the speed limit recognition. Hence, the Circle Recognition and Number Recognition can be executed in parallel before the final judgment. The Noise Reduction Filter, the RPM, and the Sign Enhancement Filter work with 8-bit grayscale data while the NR and the Circle Recognition work with 1-bit black-and-white data. These 8-bit processing modules and 1 bit processing modules are connected with the other through two memories. The first one, the Location and Scan Windows flags FIFO, is used to store the position of the sign candidates in a frame and the detected scan window sizes at that position. The second one, a general memory called Binary image memory with the size of 249,600 bits, is used to store the black-and-white bit value of each frame. Two independent memories (Binary Image Memory 1 and 2) and a memory swapping mechanism are necessary. It allows the 8-bit processing modules in RPM stage and the 1-bit processing modules in NR stage in Fig.4 access the binary image memories for read and write in parallel.

Fig.4 also shows the two pipeline stages, named RPM and NR, of the speed signs recognition system. The Noise Reduction Filter, the Rectangle Pattern Matching, and the Sign Enhancement Filter occur at the RPM stage. The RPM and Sign Enhancement Filter works in parallel using data generated from the noise reduction filter. The scan windows (SW) in RPM and Sign Enhancement Filter are pipeline processed with one input pixel at each clock. Hence, about 390x640=249,600 clocks are necessary for the first stage. During the processing time of the first frame, the detection result is written into the result FIFO and the binalization image result is written into the Binary Image Memory 1 for the next stage. At the next stage, the Circle Detection and the Number Recognition modules read data from the FIFO and the Binary Image Memory 1 for processing before handling the results to the Judgment module. At the same time, the data of the second frame is processed in the RPM stage. Result is written into the Binary Image Memory 2. Then, the NR stage of the second frame occurs with data inside the Binary Image Memory 2. The same process occurs with other frames, and so the system processes all frames in pipeline.

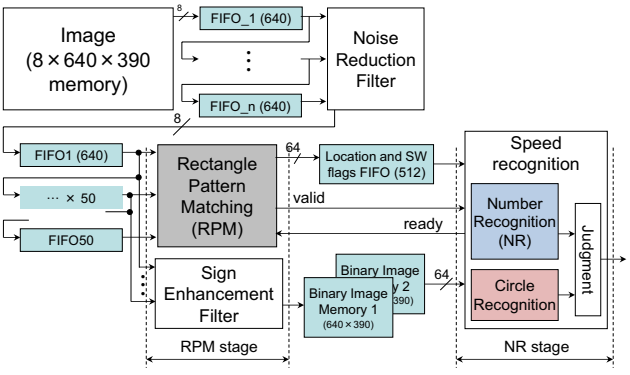


Fig. 4. Pipeline architecture of the traffic speed sign recognition system.

IV. Design of Circle Detection Using Local Direction Voting

A. Pixel Local Direction Feature of a Circle

In general, pixels at the edge of a circle will have pre-defined direction. A circle can be divided into eight areas with corresponding directions as shown in Fig.4. Our novelty is voting number of pixels that match its expected direction. If this number is bigger than the threshold, the SW is considered as circle traffic sign.

B. *Circle Detection Using Pixel Local Direction Voting*

Since the RPM roughly detects both rectangle and circle traffic sign candidates, we need to judge if the candidate has a circle shape or not by circle detection. Raster scan is used in our implementation, in which data of the last three columns for a SW is buffered for direction confirmation and voting.

Input data to the module is the binary image of each candidate at a specific position given by RPM. Depend on the relative location of the pixel inside the input SW; they should match with the corresponding direction among the eight ones shown in Fig.5. A 3x3 array is used to detect the direction of the input pixel using local border templates. Each group of 3x3 adjacent pixels, which locates in the eight areas in the

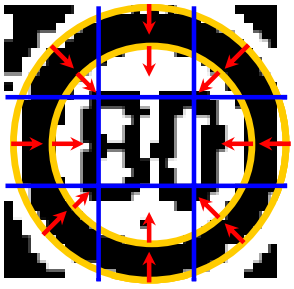


Fig. 5. Circle and local pixel directions at edge.

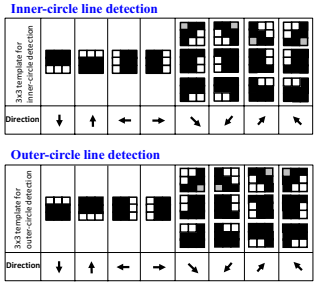


Fig. 6. Templates for pixel direction confirmation at inner and outer edge.

input candidate SW, is compared with the 32 patterns in Fig.6 for corresponding eight directions detection.

When a new pixel comes, its corresponding column in SW and two previous ones are buffered, making 3-pixel column as shown on the left of Fig.7. There is overlap in 3x3-pixel-pattern among different lines in the 3-pixel column; hence, 1x3-pixel-pattern of each line is checked separately before combining three adjacencies together for the final direction confirmation. In a column, the upper part needs to be verified with three directions of down-right (↘), down (↓), and down-left (↙); the middle part needs to be verified with two directions of right (→), left (←); and the lower part needs to be verified with three directions of up-right(↗), up (↑), and up-left (↖). Divide the input column into three helps to reduce the number of direction that need to verify.

The right of Fig.7 shows the direction voting for one area among the eight ones using reusable computational result, in which the direction voting result in the overlapped in each area between SW_{i-3} and SW_i is reused without re-voting. The voting for an area in SW_i is generated by the voting result of overlapped area plus the voting result of new input area (dir^{col+}_i). The voting result of overlapped area is the result of previously voting area (dir^{area}_{i-3}) without the subtraction area (dir^{col-}_i). The voting now became voting for the addition area (dir^{col+}_i) and storing the result for later uses as subtraction (dir^{col-}_i). At the same time, the newly voted column result dir^{col+}_i is added with dir^{area}_{i-3} before subtracting the previous stored dir^{col-}_i voting result for that area dir^{area}_i final result. As shown in Fig.7, the waiting time since direction of newly input pixels are verified and voted until the time it becomes dir^{col+}_i and dir^{col-}_i of different directions are different.

Hardware implementation of circle detection using voting the local pixel direction at edge is shown in Fig.8. When a new binary pixel gets into the system, the corresponding three columns (last columns in Fig.7) are given to the direction voting module. Pattern confirmation, a shared module among various SW sizes compares each three inputs in a line with the line patterns. Three adjacent results are then combined together in column directions confirmation sub-module before voting for the number of pixels that match a specific direction. These results are pushed into column direction voting FIFOs and become dir^{col+} and dir^{col-} at specific time, which depends on the pixel location. Voting results of all columns in the corresponding area are accumulated, forming area voting result dir^{area}. Then, dir^{arer} is

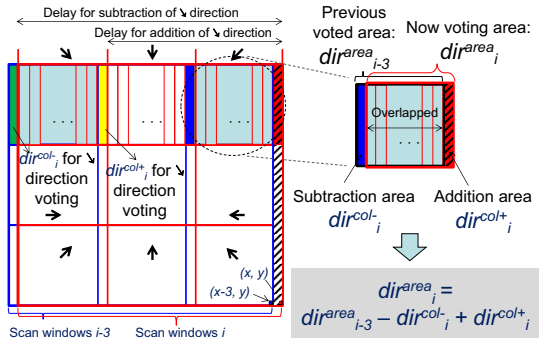


Fig. 7. Using local overlap between SWs for direction voting.

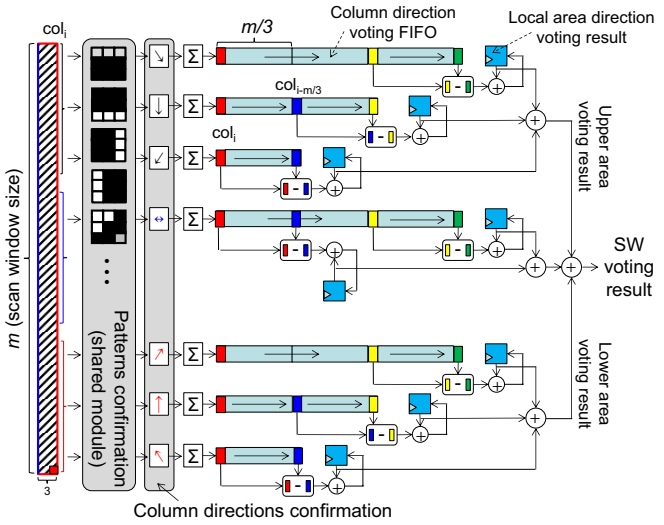


Fig. 8. Hardware design for the direction voting of a SW using direction voting reusable.

stored into local area direction voting register. All local area voting results are added together, making SW directions voting result for circle detection.

V. Evaluation Result and Discussion

A. Dataset

Our system is tested using a dataset with 194 scenes in various sign conditions, road conditions, light conditions, weather conditions, and image resolutions on real life as shown in Fig.9. It includes 60 scenes on daylight in highway and local roads at 640x390pixels resolution; 65 scenes on daylight in local road at 640x360 pixels resolution; 25 scenes on clear night in local road at 1920x1080 pixels (full HD) resolution; and 44 scenes on rainy night in local road at full HD resolution. Frames with full HD resolution are down-sized to 1/3 in each axis for simulation in our test. It is easily implemented with small hardware resources by reading the pixels after every three columns and rows. A scene is considered as all continuous frames that the same sign

Fig. 9. Traffic sign dataset in real life with various traffic condition, sign condition and light condition.

appears in the observable field of the camera until it disappears. Depending on the speed of the car, the number of frames in a scene is various. Size of the sign is also varied, which depends on the distance between the car and the sign.

B. Hardware Implementation Result and Discussion

Our design is implemented on Xilinx automotive Zynq 7020 device, which has 106,400 slice registers and 53,200 slice LUTs for estimation. Simulation on real situation shows that we need 14 scan window sizes of 20, 21, 23, 24, 26, 28, 30, 32, 34, 36, 38, 42, 46, and 50 for fully traffic sign recognition. The design in Fig.8 is implemented with biggest SW size of 50x50 pixels for evaluation and estimation. We implemented two versions, the first one uses registers as FIFO and the second one uses RAM based shift register, which utilizes memory inside LUT for FIFO. Table I shows the implementation results of RPM and CD for 50x50 SW size as well as total hardware estimation for all 14 SW sizes. The RPM and CD modules with all necessary SW sizes occupy nearly 19% of slice registers and 57% of slice LUTs on the proposed device. Small hardware size guarantees the implementable of the system on low-end low-cost automotive Xilinx Zynq family.

TABLE I. HARDWARE ESTIMATION OF TWO-STAGE CIRCLE DETECTION

SW size	Freq. (MHz)		No. of slice registers		No. of slice LUTs	
	RPM	*CD*	*RPM*	*CD*	*RPM*	*CD*
50x50 (register based)	202.2	390.3	880	1,524	1,229	1,885
50x50 (RAM based SRL)	-	390.3	-	195	-	986
All 14 SW sizes (register based)	202.2	390.3	20,138 (18.9%)		30,184 (56.7%)	

"All 14 SW sizes" shows the implementation of Rectangle Pattern Matching and Circle Detection modules for all 14 scan window sizes of 20, 21, 23, 24, 26, 28, 30, 32, 34, 36, 38, 42, 46, and 50 that necessary for fully trafic sign recognition, in which the bigest one is really implementd and the smaller ones are estimated from the bigest one.

High frequency as 200 MHz guaranties that our proposed system can process at as high as 83 fps as shown in Table 2, and is able to apply to real time processing.

C. Recognition Accuracy and Comparison with Related Works

We consider to two types of detection accuracy. The less important one is frame detection accuracy, which is low due to the individual frame quality such as small size of the sign. We define that in a frame, a traffic sign is incorrectly detected if the algorithm cannot shows the location and the size of the sign after RPM and CD. It is detectable if the algorithm can locate the position and size of the sign. When the camera gets closer to the sign, size of the sign increases, and the traffic sign becomes detectable with big size sign. The more important accuracy is the scene detection accuracy. In our experiment, we define that a traffic sign in a scene is detectable if the algorithm can locate it in at least one frame of that scene. Under that definition, since a scene always contains frames with big traffic sign, the signs in almost all the scenes in our dataset are able to be detected, achieving 98% of accuracy in terms of scene as shown in Table 2. This result is higher than all related works.

TABLE II. COMPARISON WITH RELATED WORK

Method	Scene detection rate (%)	Throughput (fps)	Platform
Hardware/Software [9]	-	1.28	Virtex 5
Neuron network [10]	82	62	Virtex 4
Multi-Core SoC [11]	-	2.3	Virtex 4
Hardware/Software [12]	90	14.3	Virtex 5
Hardware/Software [13]	-	10.4	Zynq
Fuzzy template [14]	93.3	66	Virtex 4
Random Forest [16]	97.2	18~28	-
Proposed research	98*	83*	Zynq 7020

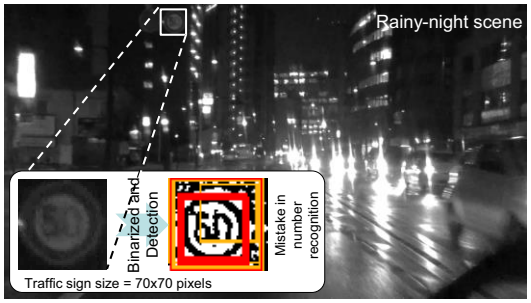
*: Number recognition is included in this estimation

D. Discussion on Difficulty Scenes at Night and Rain

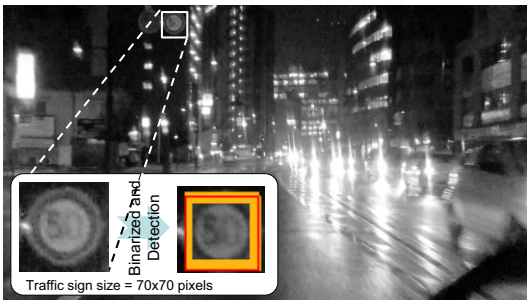
In some rainy night scenes, when the sign became very dark as effectiveness of lights from cars at other direction, the sign and the number inside are hardly detected as shown in Fig.10.a, where the traffic sign is detectable but the number inside is incorrectly recognized due to the bad quality input image. Simulation shows that our algorithm achieves 98% correct recognition rate even in such hard situation. However, the same scene is correctly recognized with some small change in the contrast as shown in Fig.10.b. Global luminosity of the previous frame can be used to decide if contrast enhancement is necessary or not.

E. Discussion About Tradeoff on Location of Circle Detection Module

The CD module can located either on the first pipeline stage



a. Undetectable speed limit sign at rainy night.



b. Undetectable sign at rainy night became detectable with change in contrast.

Fig. 10. Difficult situation with low frame accuracy (full HD scenes) and simple solution using contrast adjastment.

or at the second one. If it locates at the first stage, it reduces the number of ROI, and so reduces the number of speed traffic sign candidates that need fine analysis on number recognition, and so eases the task of the number recognition module. In that case, it directly receives binary data from the Sign Enhancement Filter in Fig.4. Penalty is the Sign Enhancement Filter needs to generate 50 binary data at the same time or 50 binary line buffers are required to store input column in Fig.8. If the CD works together with the number recognition module as shown in Fig.4, it shares input data with the NR, and so no more hardware is required but the penalty is in the worst case, time for number recognition for a frame increases depends on the scene. In our experiment, in average, 3.7 ROI is reduced from each frame for number recognition. In the worst case, it helps to reduce 72 ROIs among 454 ones detected after RPM.

VI. Conclusion

This paper introduces our novel algorithm and implementation for speed traffic sign candidate detection. The combination of coarse and fine grain pipeline architectures with parallel processing and simple pixel local direction feature allows our implementation meets the demand of a real time system. The computation re-use mechanism in scanning process significantly reduce the hardware size in our implementation, allows raster scan with small hardware size. Small hardware size and high accuracy results achieved on evaluation prove that our system is able to be implemented on the target low-cost automotive FPGA and applicable in real life.

In the future, the implementation will be combined with number recognition module, creating a speed traffic sign recognition system.

Acknowledgements

Part of this work was supported by Grant-in-Aid for Scientific Research(C) JSPS KAKENHI Grant Number 23560400.

References

[1] J. Torresen, et al. “Efficient Recognition of Speed Limit Signs”, *Proceeding of the 7th International IEEE Conference on Intelligent Transportation Systems*, pp. 652-656, 2004.

[2] P. M. Ozcelik, et al., “A Template-Based Approach for Real-Time Speed-Limit-Sign Recognition on an Embedded System Using GPU Computing”, *Proceedings of the 32nd DAGM conference on Pattern recognition*, pp. 162-171, 2010.

[3] J. Miura, et al., “An Active Vision System for Real-Time Traffic Sign Recognition”, *Proceeding of International IEEE Conference on Intelligent Transportation Systems*, pp. 52-57, 2000.

[4] F. Moutarde, et al., “Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular Traffic Signs Recognition system”, *Proceeding of International IEEE Intelligent Vehicles Symposium*, pp. 1122-1126, 2007.

[5] C. G. Keller, et al., “Real-time Recognition of U.S. Speed Signs”, *Proceeding of International IEEE Intelligent Vehicles Symposium*, pp. 518-523, 2008.

[6] F. Zaklouta and B. Stanciulescu, “Segmentation Masks for Real-time Traffic Sign Recognition using Weighted HOG-based Trees”, *Proceeding of the 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1954-1959, 2011.

[7] M. Yamamoto, et al., “Speed Trafic-Sign Recognition Algorithm for Real-Time Driving Assistant System”, *Proceeding of the 18th Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2013)*.

[8] A-T. Hoang, et al., “Low Cost Hardware Implementation for Traffic Sign Detection System”, *accepted at 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS2014)*, 2014.

[9] Sheldon Waite and Erdal Oruklu, “FPGA-Based Traffic Sign Recognition for Advanced Driver Assistance Systems”, *Journal of Transportation Technologies*, Vol. 3, No. 1, pp. 1-16, 2013.

[10] C. Souani, H. Faiedh, and K. Besbes, “Efficient algorithm for automatic road sign recognition and its hardware implementation”, *Journal of Real-Time Image Processing*, Vol. 9, Issue 1, pp. 79-93, 2014.

[11] M. Muller et al., “Design of an automotive traffic sign recognition system targeting a multi-core SoC implementation”, *Proceding in Design, Automation & Test in Europe Conference & Exhibition 2010*, pp.532-537, 2010.

[12] Hasan Irmak, “Real Time Traffic Sign Recognitio system on FPGA”, *Master Thesis, The Graduate School of Natural and Applied Sciences of Middle East Technical University*, 2010.

[13] Yan Han, “Real-time traffic sign recognition based on Zynq FPGA and ARM SoCs”, *Proceding of the 2014 IEEE International Conference on Electro/Information Technology (EIT)*, pp. 373-376, 2014.

[14] W. Liu et al., “Real-Time Speed Limit Sign Detection and Recognition from Image Sequences”, *Proceding of the 2014 IEEE International Conference on Artificial Intelligence and Computational Intelligence (AICI)*, pp. 262-267, 2010.

[15] D. Soendoro et al., “Traffic sign recognition with Color-based Method, shape-arc estimation and SVM”, *Proceding of the 2014 IEEE International Conference on Electrical Engineering and Informatics (ICEEI)*, pp. 1-6, 2011.

[16] F. Zaklouta, B. Stanciulescu, “Real-time traffic sign recognition in three stages”, *Journal of Robotics and Autonomous Systems*, Vol 62, Issue 1, pp. 16-24, 2014.