IEIE Transactions on Smart Processing and Computing, vol. 4, no. 4, August 2015
http://dx.doi.org/10.5573/IEIESPC.2015.4.4.237

IEIE Transactions on Smart Processing and Computing

Real-time Speed Limit Traffic Sign Detection System for Robust Automotive Environments

Anh-Tuan Hoang, Tetsushi Koide, and Masaharu Yamamoto

Research Institute for Nanodevice and Bio Systems, Hiroshima University, 1-4-2, Kagamiyama, Higashi-Hiroshima, 739-8527, Japan {anhtuan, koide}@hiroshima-u.ac.jp

***** Corresponding Author: Anh-Tuan Hoang

Received July 15, 2015; Revised August 5, 2015; Accepted August 24, 2015; Published August 31, 2015

* *Regular Paper*

* *Extended from a Conference: Preliminary results of this paper were presented at the ITC-CSCC 2015, Summer 2015. This present paper has been accepted by the editorial board through the regular reviewing process that confirms the original contribution.*

Abstract: This paper describes a hardware-oriented algorithm and its conceptual implementation in a real-time speed limit traffic sign detection system on an automotive-oriented field-programmable gate array (FPGA). It solves the training and color dependence problems found in other research, which saw reduced recognition accuracy under unlearned conditions when color has changed. The algorithm is applicable to various platforms, such as color or grayscale cameras, high-resolution (4K) or low-resolution (VGA) cameras, and high-end or low-end FPGAs. It is also robust under various conditions, such as daytime, night time, and on rainy nights, and is adaptable to various countries' speed limit traffic sign systems. The speed limit traffic sign candidates on each grayscale video frame are detected through two simple computational stages using global luminosity and local pixel direction. Pipeline implementation using results-sharing on overlap, application of a RAM-based shift register, and optimization of scan window sizes results in a small but high-performance implementation. The proposed system matches the processing speed requirement for a 60 fps system. The speed limit traffic sign recognition system achieves better than 98% accuracy in detection and recognition, even under difficult conditions such as rainy nights, and is implementable on the low-end, low-cost Xilinx Zynq automotive Z7020 FPGA.

Keywords: Advanced driver assistance systems (ADAS), Speed limit traffic sign detection, Rectangle pattern matching, Circle detection, FPGA implementation

1. Introduction

Speed limit traffic sign recognition is very important for the fast-growing advanced driver assistance systems (ADAS). Under continual pressure for greater road safety from governments, traffic sign recognition and active speed limitation become urgent issues for an ADAS. Important traffic sign information is provided in the driver's field of vision via road signs, which are designed to assist drivers in terms of destination navigation and safety. Most important for a camera-based ADAS is to improve the driver's safety and comfort. Detecting a traffic sign can be used in warning drivers about current traffic situations, dangerous crossings, and children's paths, as shown in Fig. 1. Although the navigation system is available, it cannot be applied to new roads or places that the navigation signal cannot reach, or with electronic speed limit traffic signs where the sign changes depending on the traffic conditions. An assistant system with speed limitation recognition ability can inform drivers about the change in speed limit, as well as notify them if they drive over the speed limit. Hence, the driver's cognitive tasks can be reduced, and safe driving is supported.

Speed limit traffic (SLT) sign recognition systems face several problems in real-life usage, as shown in Fig. 2.

(1) Color: the color of an SLT sign will change depending on the light, weather conditions, and age of the sign.

(2) Sign construction: light-emitting diode (LED) SLT signs appear different in color, shape, and luminosity

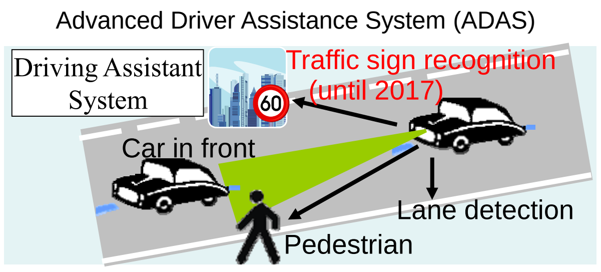

Advanced Driver Assistance System (ADAS)
Driving Assistant System
Traffic sign recognition (until 2017)
60
Car in front
Lane detection
Pedestrian

Fig. 1. Chalenges of ADAS system toward active safety.

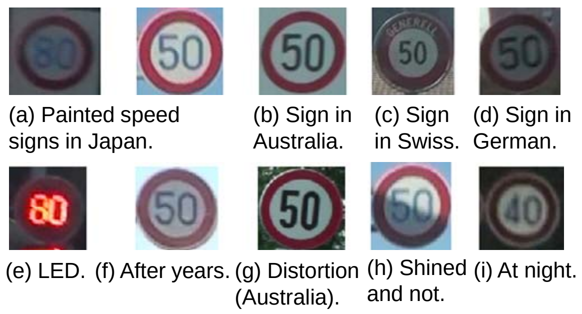


Fig. 2. The appearance differences of speed limit traffic sign in various conditions.

from printed signs, depending on the angle between the camera and the sign.

(3) Light conditions: the presence of the sun and of some types of lights, in daytime and at night, makes the sign look different.

(4) Font: fonts on traffic signs are decided by governments, and they are different in various countries. The difference can be seen in the thickness and shape of the number, as shown in Figs. 2 (a), (b), (c), and (d).

(5) Distortion: the image of an SLT sign has distortion along three axes (x, y, and z), which depend on the angle between camera and sign.

(6) Accuracy: high accuracy in recognition rate is required, which means that the same sign should be correctly recognized at least once in a sequence scene.

(7) Real-time processing: the automotive system must be able to process 30 to 15 fps under various platforms.

(8) Stability: under various platforms, the system must not need retraining when users change devices, such as the camera.

A lot of research on SLT sign recognition (SLTSR) for the ADAS has been done, but those SLTSR algorithms have difficulty recognizing when color has changed due to light conditions, such as the presence of sunshine (Fig. 2(h)), illumination at night (Fig. 2(i)), LED signs (Fig. 2(e)), and in recognizing signs in different countries. They also have difficulty with high-accuracy, real-time processing using few computational resources on available low-price devices.

In this study, we aim to solve these color and environment problems with a non-color–based recognition approach, in which grayscale images are used in both speed limit sign candidate detection and number recognition. SLT sign candidates are detected from each input frame before recognizing the limit speed in real time. Our system combines many simple and easy computation features of SLT signs, such as area luminosity, pixel direction, and block histogram, into a real-time, high-accuracy, and low-computational–cost design. Hence, it is implementable on a low-cost and resource-limited automotive-oriented field-programmable gate array (FPGA). The target platform is the Xilinx Zynq 7020, which has 85K logic cells (1.3 M application-specific integrated circuit gates), 53.2K lookup tables (LUTs), 106.4K registers, and 506KB block random access memory. Its price is about \$15 per unit [24].

Related works and an overview of our approach to SLT sign recognition is presented in Section 2. The available SLT sign recognition system architectures and related algorithms for SLT sign candidate detection are discussed in Section 3. Section 4 describes how to combine simple features of non-color–based SLT signs for a real-time recognition system. Section 5 offers an overview of the hardware implementation of the proposed architecture. Discussions on accuracy, hardware size, and throughput of the proposed algorithm for SLT sign detection are given in Section 6. Section 7 concludes this paper

2. Image Size and Scan Window Size Requirement for SLT Sign Detection

The right side of Fig. 3 shows the differences in image sizes of an SLT sign at different distances and angles in real life. We define a scene as a sequence of all frames in which the same sign appears in, and disappears from the observation field of the camera. In one scene, when a 640×360 pixel camera is located at more than 30 meters in front of the sign, the 60 cm diameter SLT sign in Japan will appear as small as 10×10 pixels, as shown on the left of Fig. 3. In that case, the size of the number in the sign is

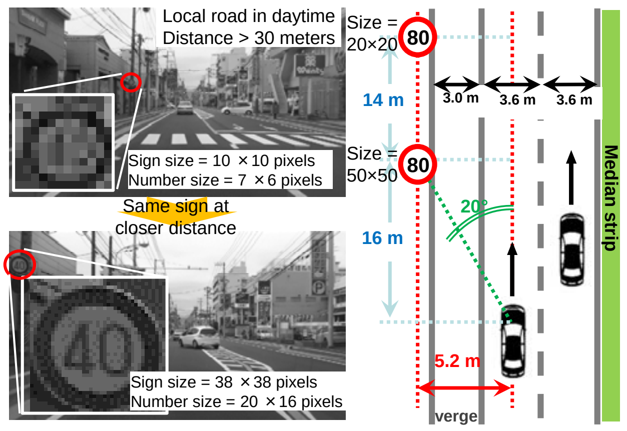

Local road in daytime
Distance > 30 meters
Sign size = 10 × 10 pixels
Number size = 7 × 6 pixels
Same sign at closer distance
Sign size = 38 × 38 pixels
Number size = 20 × 16 pixels
Size = 20×20
80
14 m
Size = 50×50
80
16 m
20°
5.2 m
3.0 m
3.6 m
3.6 m
verge
Median strip

Fig. 3.Difference of the image size of the SLT sign on different distance, speed, lane (angle) with 640×360 pixel camera [18].

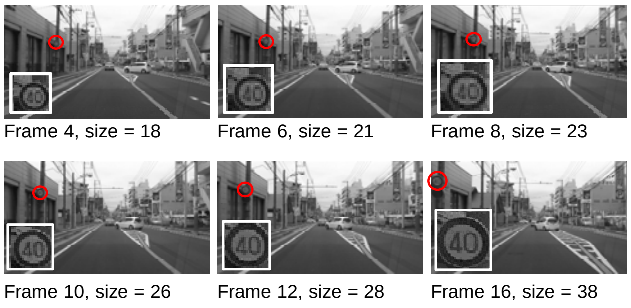


Fig. 4. A scene in real situation with image of SLT sign gradually increase.

as small as 7×6 pixels, which is really hard to recognize, even with the human eye. This size increases to 20×20 pixels when the camera gets closer to the sign at a distance of 30 meters (with a 10×10-pixel number); then, the size gradually increases to 50×50 pixels at a distance of 16 meters (on the highway) before disappearing from the observation field of the camera. At sizes bigger than 20×20, appearance of the speed sign becomes recognizable, as shown at the bottom of Fig. 3. When a 1920×1080-pixel (full HD) camera, or a camera with higher resolution, is used, the size of an SLT sign becomes bigger. Although the image size of an SLT sign and the processing size are a trade-off, the range of 20×20 to 50×50 is always available in a scene. An example of a scene in a real situation is shown in Fig. 4.

From that real situation, SLT sign detection should not use up too many computational resources to recognize the SLT sign at size smaller than 20×20 or bigger than 50×50. Since the SLT sign appears in a range of sizes, the system must process each input frame with scan windows in a range of sizes for SLT sign detection and recognition at the proposed distance. Our proposed SLT sign detection algorithm is designed to detect an SLT sign in the range of 20×20 to 50×50 pixels, which will appear in the observation field of cameras with resolution higher than 640×360.

If the vehicle moves at 200 km/h, a 60 fps camera can takes 15 frames for SLT sign detection during the 14 meter distance between 30 meters away and 16 meters away from the sign. If the SLT sign can be recognized from those frames, a detection and recognition system will work well, even if the vehicle moves at 200 km/h. Our system aims for this goal.

3. Related Works

3.1 Software-Oriented Implementations

3.1.1 Neural Network–Based Sign Recognition

The multi-column deep neural networks and the multi-scale convolutional network were introduced by Ciresan et al. [21] and Sermanet and LeCun [22] in the Neural Networks contest. They achieved as high as 99% recognition accuracy, but required a huge training data set, as well as huge computational resources (four 512-core GPUs) for traffic sign recognition. Recognition time for a full HD image will be significantly increased to an unacceptable level for real-life usage. In addition, they use color features in their recognition, and so face accuracy problems when recognizing signs under unknown light conditions.

3.1.2 Machine Learning–Based Recognition

Machine learning was used by Zaklouta and Stanciulescu [20] and Zaklouta et al. [23] for traffic sign recognition. They used support vector machine (SVM) with a histogram of oriented gradient (HOG) feature for traffic sign detection, and tree classifiers (K-d tree or random forest) to identify the content of the traffic signs. They achieved accuracy of 90% with a processing rate of 10-28 fps. Again, they faced the color problem in their implementation, and so it is difficult to apply to other situations (night, rain, and so on).

3.1.3 Color-Based, Shape-Based, and Template-Based Recognition

A general feature of traffic signs is color, which is predetermined to ensure they get the driver's attention. Hence, color is used as a feature in a lot of image segmentation research. Torresen et al. [16] detected the red circle of a traffic sign by utilizing a red–white–black filter before applying a detection algorithm. Miura et al. [7] detected traffic sign candidate regions by focusing on the white circular region within some thresholds. Zaklouta and Stanciulescu [20] also used color information within a threshold for traffic sign image segmentation. This detection method required a color camera and more computation resources, such as memory, for color image storage and detection. Similar to the neural network–based and machine learning–based recognition approaches, color-based segmentation relies on the red color, and so, has difficulty with recognition when the color of the traffic sign has changed due to age and lighting conditions.

Another method for traffic sign candidate detection is based on the shape of the signs. In this approach [5], a feature where a rectangular structure yields gradients with high magnitudes at its borders is used in traffic sign candidate detection. Moutarde et al. [8] used edge detection for rectangle detection and a Hough transform for circle detection. This method is robust to changes in illumination, but it requires complex computation, such as the Hough transforms. Processing the transformation and extracting matching peaks from big image are computationally complex for real-time processing systems.

Template matching [16] uses a prepared template for area comparison with various traffic sign sizes. The approach simply takes the specific color information of an area, and compares it with a prepared template for matches. Because the sizes of the traffic signs vary from 32x32 to 78x78 pixels, a lot of hardware resources and computation time are required.

3.2 Hardware Oriented Implementations

3.2.1 Hardware/Software Co-design Implemen-tations

Hardware/software co-design on a low-cost Xilinx Zynq FPGA system was presented by Han [1], in which an input color image of the traffic sign is processed by software on a PC before sending it to the Zynq 7000 system on chip (SoC). The traffic sign candidates are detected with hardware using color information before refining and performing template-based matching on an ARM core. This hardware processing requires a lot of memory access, as well as software processing on both the ARM core and PC, so latency is high and throughput is low. Big templates of 80×80 pixels are required for improvement of detection accuracy.

Muller et al. [9] applied software and hardware design flows on a Xilinx Virtex-4 FPGA to implement a traffic sign recognition application. It combines multiple embedded LEON3 processors for preprocessing, shape detection, segmentation, and extraction with hardware IPs for classification in parallel to achieve latency not longer than 600 ms to process one frame. However, this latency is not fast enough to apply to real-time detection and recognition.

Irmak [3] also utilized an embedded processor approach with minimal hardware acceleration on a Xilinx Virtex 5 FPGA. Color segmentation, shape extraction, morphological processing, and template matching are performed on a Power PC processor with software, and edge detection is performed on a dedicated hardware block.

Waite and Oruklu [17] used a Xilinx Virtex 5 FPGA device in a hardware implementation. Hardware IPs are used for hue calculation, morphological filtering, and scanning and labeling. The MicroBlaze embedded core is used for data communication, filtering, scaling, and traffic sign matching.

3.2.2 Neural Network on an FPGA

A neuron network implementation on a Xilinx Vitex 4 FPGA for traffic sign detection was presented by Souani et al. [14]. The system works with low-resolution images (640×480 pixels) with two predefined regions of interest (ROIs) of 200×280 pixels. The small ROI results in high recognition speed. However, the accuracy is as low as 82%, and the traffic signs must be well lit. Hence, this system has difficulty when applied at night or to back-lit signs.

3.3 A Proposed Approach for Robust Automotive Environments

All the related speed limit traffic sign recognition systems use color as an important feature for detection and recognition in their implementations, and so they have difficulty in recognizing long-term deterioration in traffic signs, traffic signs under different lighting conditions, and with LED traffic signs. They also use processing with high computational complexity, such as the Hough transform, for the traffic sign shape detection, and tree classifiers, SVM, and complex multi-layer neural networks for traffic sign identification. So the implementation resources become large, and the hardware costs are also high. The processing time of the available implementations also presents problems when applied in real life to low-end vehicles. In addition, the recognition approaches using neural networks and machine learning require learning processes with a huge dataset. For each user with a different platform, such as camera type, a specific dataset must be prepared and a relearning process is necessary to guarantee high accuracy.

Different from the above works, which rely on color features and high complexity processing in traffic sign recognition, our approach utilizes simple yet effective speed limit traffic sign features from grayscale images. The proposed approach utilizes multiple rough, simple, and easily computable features in three-step processing to achieve a robust speed limit traffic sign detection system. Using simple features makes the proposed system applicable to various platforms, such as the type of camera (color or grayscale, high- or low-resolution), and in the line-up of FPGAs (low-end, automotive, high-end). It is also robust under various conditions in Japan scenarios, such as illumination conditions (daytime, night time, and rainy nights), and types of road (local roads and highways). The simple yet effective features enable it to easily optimize a parameter set so as to meet features of the speed limit traffic sign systems in other countries. The proposed features include area luminosity difference, pixel direction, and an area histogram. The computation of these simple features only requires simple and low-cost hardware resources such as adders, comparators, and first-in, first-out (FIFO) stacks. So, it is possible to implement on any line-up of FPGAs. The proposed speed limit traffic sign recognition system can be also extended to other traffic sign recognition.

4. Rough and Simple Feature Combi-nation for Real-Time SLTSR Systems

4.1 Multiple Simple Features of SLT Signs

4.1.1 Shared Luminosity Feature Between Rectangle and Circle

Fig. 5 shows the area luminosity feature of a scan window with a circular speed limit traffic sign inside, in which the circle line of the sign is much darker than the adjacent white area in a grayscale image. It means that the luminosity of the areas that contain the circle is much lower than the adjacent ones. If the circle fits within the scan window, the location of the dark and the white areas can be predefined as B_1 to B_8 (say, for black) and W_1 to W_8 (say, for white) in Fig. 5. The luminosity differences between those corresponding black and white areas exceed a predefined threshold. Depending on the view angle, the brightness of the black and white areas is different, and so a variable threshold is used in our detection algorithm. This luminosity feature is simple but effective because it is extendable to the detection of other shape types, such as a

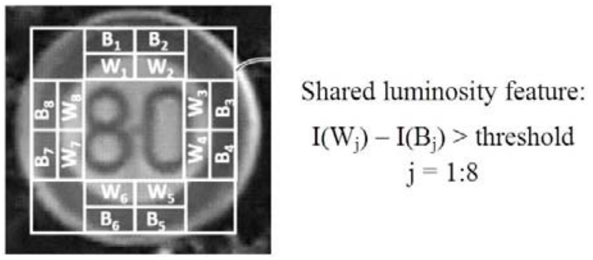


Fig. 5. Shared luminosity feature between rectangle and circle.

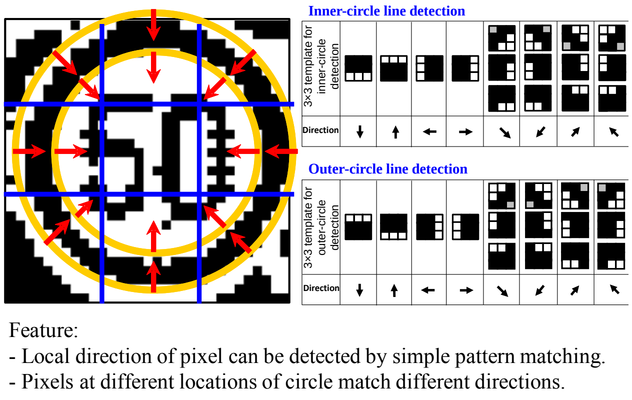

Inner-circle line detection
3×3 template for inner-circle detection
Direction
Outer-circle line detection
3×3 template for outer-circle detection
Direction
Feature:
- Local direction of pixel can be detected by simple pattern matching.
- Pixels at different locations of circle match different directions.

Fig. 6. Local pixel direction feature of circle.

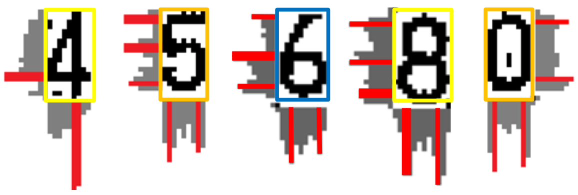


Fig. 7. Difference in histogram between numbers.

hexagon. In addition, this feature is applicable to any image size, from VGA to 4K. It is also stable if a high-resolution image is down-sampled to a lower resolution. Hence, down-sampling an image can be used in detection, instead of a high-resolution image to save computational resources without sacrificing accuracy.

4.1.2 Pixel Local Direction Feature of Circle

Fig. 6 shows a local feature of a circle in a binary image, in which pixels at the edge of a circle have different direction depending on its location. For a pre-determined size, those locations and directions are pre-determined. Direction of a pixel can be verified using simple patterns of 3×3 pixels. A circle will have the total number of pixels that match each specific direction to get into a predefined range.

4.1.3 Block Histogram Feature of Numbers

Fig. 7 shows examples of a histogram for binary images of the speed sign numbers “4”, “5”, “6”, “8” and “0” in Japan along two axes. Locations of the maximum and minimum in the histogram, as well as the ratio between those rows/columns and others, are different. The maximum and minimum in the histogram for each axis and area (the total number of pixels) are the features of the numbers and can be used to recognize the speed sign number [18].

4.2 Multiple Simple Feature-Based Speed Limit Sign Recognition System

4.2.1 System Overview

Fig. 8 shows the speed limit recognition system overview. The input grayscale image is raster scanned with a scan window (SW) in the rectangle pattern matching (RPM) [26] module. It computes the luminosity of rectangular and circular traffic signs, as shown in Fig. 5. The luminosity differences of those areas are then compared with a dynamic threshold to roughly determine if the SW contains a rectangle/circle shape as a traffic sign candidate with the same size. Since the RPM algorithm utilizes common features of the circle and rectangle, it can be used to recognize both circular and rectangular signs. The sign enhancement filter developed by our group includes a hardware-oriented convolution filter and image binarization, and is applied to the 8-bit grayscale pixels of traffic sign candidates, changing them to 1-bit black-and-white (binary) pixels for circle detection and speed number recognition. The sign enhancement process helps increase the features of the speed numbers and reduces the amount of data being processed. Circle detection uses a local direction feature at the circle’s edge to decide if the detected rectangle/circle candidates are really a circle mark or not, using a binary image. Direction of pixels in different areas in the SW and patterns for pixel local direction confirmation are shown in Fig. 6. Finally, the speed number recognition (NR) module analyzes block histogram features of the regions of interest and compares them with the predefined features of speed numbers for the NR module [18].

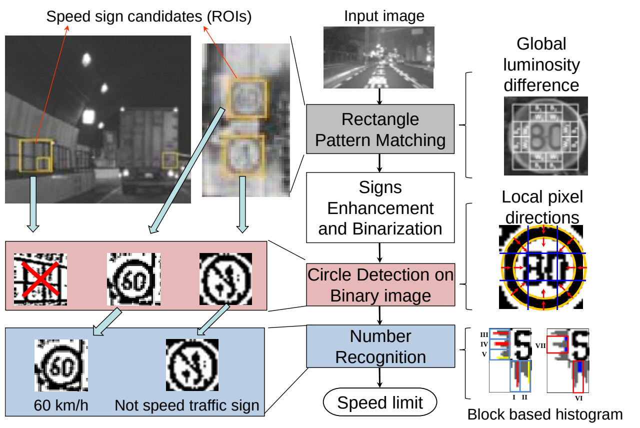

Speed sign candidates (ROIs)
Input image
Global luminosity difference
Rectangle Pattern Matching
Signs Enhancement and Binarization
Local pixel directions
Circle Detection on Binary image
Number Recognition
Speed limit
60 km/h
Not speed traffic sign
Block based histogram

Fig. 8. Speed limit traffic sign recognition system based on multiple simple features.

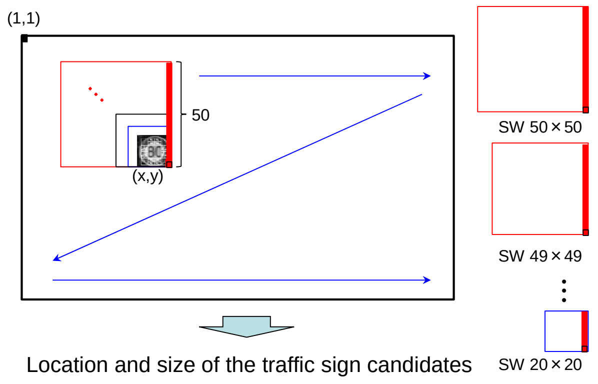


Fig. 9. Multi-scan window size raster scan in parallel by last column pixels processing.

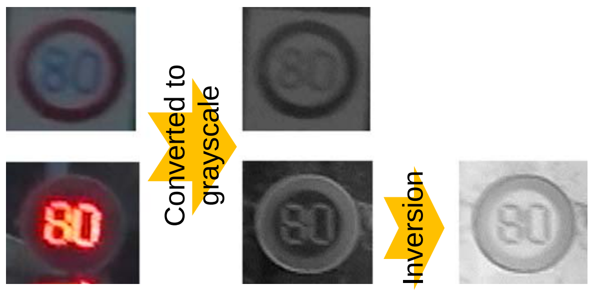


Fig. 10. Extendibility to LED sign recognition.

4.2.2 Parallel Raster Scanning of Multiple Scan Window Sizes

Although the image size of the system is variable, and our system is applicable to 4K, full HD, and VGA sizes, in the following section, for simplification of explanation, we assume the input image size is 640×360. The detectable speed limit traffic sign is in a range from 20×20 to 50×50 pixels. We use a raster scanning method with the above scan window sizes, as shown in Fig. 9, to keep the processing time constant. At any clock cycle, when a new pixel (*x*,*y*) gets into the system, a column of 50 pixels from (*x*,*y*) will be read from FIFO to SWs for detection, as shown on the right of Fig. 9. All those SW sizes (from 20×20 to 50×50 pixels) are processed in parallel in one clock cycle to find all candidates at different sign sizes at that position. In the circle detection module, the three last continuous columns of the scan window are buffered in registers for detection in the same manner.

4.2.3 Feature and Strategy for LED Speed Limit Sign Detection

Fig. 10 shows the difference between painted speed signs and LED type speed signs, in which the number in the LED speed sign is brighter than that in the painted sign, and the background of the sign is black in Japan. It makes the number in the LED sign became off-white in the grayscale image, and so the color of the circle line in the grayscale image becomes white, while the adjacent area becomes dark, as shown in the middle of Fig. 10. Consequently, the luminosity feature in Fig. 5 is easily extended and applied to detect an LED speed sign. The detection of black and white luminosity is inverted to be applied to grayscale images of the LED speed sign to uniform features of painted and LED signs.

5. Hardware Implementation

5.1 Algorithm Modification for Efficient Hardware Implementation

Fig. 11 shows the data processing flow, which is optimized for hardware implementation. The hardware module, once implemented, will occupy hardware resources, even if it is used or not. Hence, in our hardware implementation, the sign enhancement and binarization (SEB) module will work in parallel with the rectangle pattern matching module to reduce the recognition latency by reducing random memory access and applying the raster scan. If a high-resolution camera is used, preprocessing, which only performs the down-sampling of the grayscale input image to one-third, is an option to reduce the computational resources occupied by RPM. The number recognition (NR) and the circle detection (CD) modules process each binary image candidate of a speed limit traffic sign in sequence, and the two modules can process the speed sign candidates in parallel to share the input. The circle detection result can be used to enhance the decision of speed number recognition. The hardware size is small, but processing time for NR is increased depending on the number of speed sign candidates. Another approach is making CD to process the speed sign candidate in parallel with RPM and SEB. Results of CD is used to reduce the number of speed limit traffic sign candidates detected by RPM. It reduces the processing time, but the penalty is an increase in hardware size. In our prototype design, we will introduce a pipeline design for the first approach.

The optional pre-processing module is used if high-resolution and/or an interlace camera is used. If an

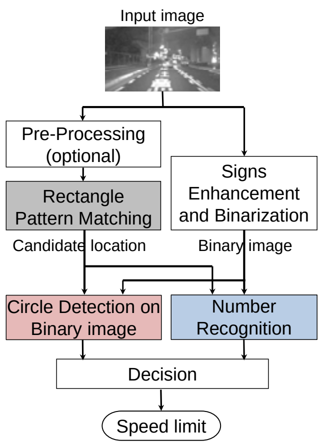

Input image
Pre-Processing (optional)
Rectangle Pattern Matching
Signs Enhancement and Binarization
Candidate location
Binary image
Circle Detection on Binary image
Number Recognition
Decision
Speed limit

Fig 11. Modification for hardware oriented speed limit sign recognition system.

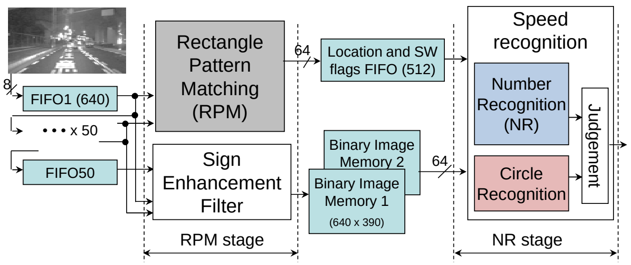


Fig. 12. Pipeline implementation of speed limit sign recognition system.

interlaced scan is used, the preprocessing module de-interlaces by taking odd or even lines and columns of the input image before sending data to other modules. If a high-resolution camera such as a full HD camera is used, the preprocessing module is used to down-sample the input image to the appropriate image size, such as 640×360 for RPM and CD. It helps to reduce the number of candidates that need to be processed in NR. Since the luminosity and the local pixel direction features used in RPM and CD are not affected by down-sampling, down-sampling is enough for hardware resource reduction without sacrificing detection accuracy.

5.2 Hardware-Oriented Pipeline Architecture

Fig. 12 shows the two-stage pipeline architecture of the speed limit recognition system. The system is able to scan for traffic signs up to 50×50 pixels in size. The proposed input image is 8-bit grayscale with a resolution of 640×360 = 230,400 pixels.

It contains two stages of RPM and NR with four main modules of RPM, SEB, CD and NR. A judgement module is included to decide which speed limit is shown in the traffic sign. Support for RPM is a number of 8-bit FIFOs.

The SEB and RPM processing are independent, and both of them work with grayscale images, and so they could be processed in parallel in the first pipeline stage as suggested by Fig. 11. Results from the CR module are used to strengthen the judgment of the speed limit recognition. CR and NR modules process binary images, and so are executed in parallel for input data sharing before the final judgment in the second pipeline stage. Those 8-bit processing modules and 1-bit binary processing modules are connected with the others through two memories. The first one, the location and scan window flags FIFO (LSW-FIFO), is used to store the position of the sign candidates in the input frame and the detected scan window sizes at that position. The second one, a general memory called binary image memory (BIM) with a size of 640×360 bits, is used to store the black-and-white bit value of each frame. Two independent memories and a memory-swapping mechanism are necessary to allow the 8-bit and 1-bit processing modules access the binary image memories to read and write in parallel.

Fig. 13 shows the timing of the two pipeline stages of the proposed SLT sign detection and recognition system. RPM and SEB occur at the RPM stage in parallel using the

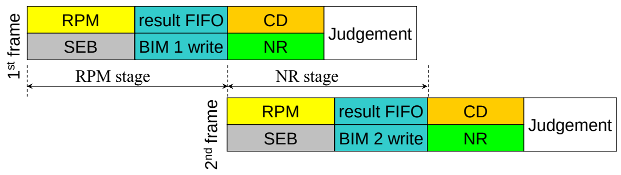


Fig. 13. Pipeline stages of speed limit sign recognition system.

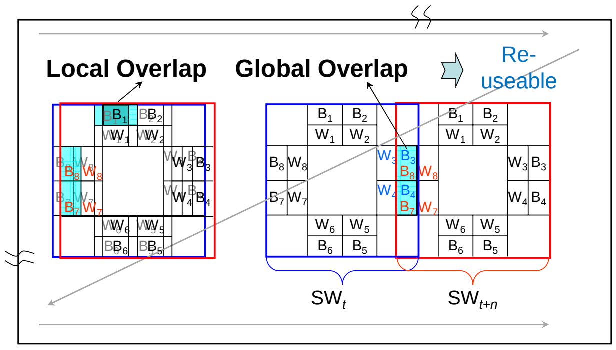


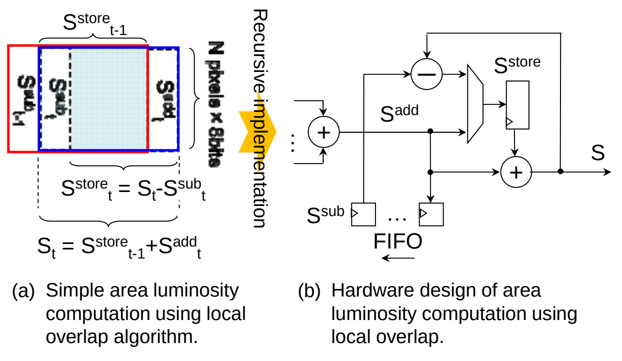
Fig. 14. Local overlap in adjacent scan windows and global overlap between with computational result reusable.

same pixel data input. An optional scaling-down module can be applied to input data for RPM if necessary for high-resolution camera usage. The scan windows in RPM and SEB are pipeline-processed with one input pixel in each clock cycle. Hence, about 640×360=230,400 clock cycles 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 binarization image result is written into BIM 1 for the second stage. At the second stage, the CD and the NR modules read data from the LSW-FIFO and BIM 1 for processing before handing the result to the judgment module. At the same time, the data of the second frame is processed in the RPM stage. The result is written into BIM 2. Then, the NR stage of the second frame occurs with the previously written data inside BIM 2. The same process occurs with other frames, and so the system processes all frames in the pipeline.

5.3 Implementation of Area Luminosity Computation for RPM using Computa-tional Result-Sharing on Overlap

Depending on the distance between the vehicle (camera) and the SLT sign, the size of the sign on the input image varies from 20×20 to 50×50 pixels. We need to scan the input frame with various scan window sizes, as shown in Fig. 9 for SLT sign detection at all distances. Inside each scan window, the shared luminosity feature between rectangle and circle in Fig. 5 is used. Two reusable computations generated by local and global overlaps, as shown in Fig. 14, are applied to the RPM implementation to reduce the hardware size.

The first re-usable computational result concerns local overlap between two adjacent scan windows, as shown in



S^store t-1
S^sub t-1
S^sub t
S^add t
N pixels x 8bits
S^store t = S_t - S^sub t
S_t = S^store t-1 + S^add t
Recursive implementation
S^store
S^add
S^sub
FIFO
S
(a) Simple area luminosity computation using local overlap algorithm.
(b) Hardware design of area luminosity computation using local overlap.

Fig. 15. Pipeline implementation of area luminosity computation using cmputational result sharing among local overlap.

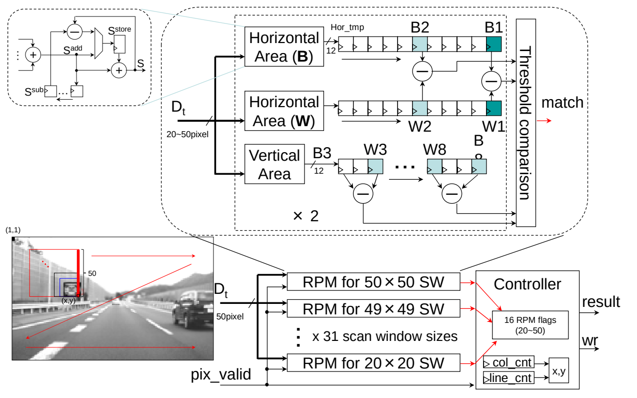


Fig. 16. Implementation of RPM using global and local overlap and FIFO.

the left side of Fig. 14. It is locally applied to the brightness computation of the same area in two adjacent scan windows (e.g. B_1 area) and is compatible with the raster scan method. The algorithm, which is compatible with the raster scan method, and hardware implementation for pipeline area luminosity computation, is shown in Fig. 15. The overlapped area between S_{t-1} and S_t (S^{store}_{t-1}) is reused without computation (Fig. 15(a)). The luminosity of the preferred area S_t is generated by that of the overlapped area S^{store}_{t-1} plus the luminosity of the new input area S^{add}_t. Luminosity of the overlapped area S^{store}_{t-1} is computed from the luminosity of area S_{t-1}, which is computed during the processing of the previous SW, and subtracts that of the subtraction area (S^{sub}_{t-1}). Hence, the computation is now for the luminosity of the addition area (S^{add}_t), and storing the result for later use. At the same time, the newly computed luminosity of the S^{add}_t area is added to that of the S_{t-1} area before subtracting the previously stored luminosity of the S^{sub}_{t-1} area for area S_t luminosity computation. Hardware design of one scan window size for the brightness computation for each area of B_1~W_8 is shown in Fig. 15(b).

The second overlap is globally applied to the scan windows inside a frame, as shown in the right side of Fig. 14. During the processing of scan window t (SW_t), the brightness of areas W_3, W_4, B_3, and B_4 are computed. During SW_{t+n} processing, these areas become B_7, B_8, W_7 and W_8, respectively. Hence, the brightness computation results of those areas in SW_t can be stored in FIFO for later reuse in SW_{t+n}. The upper right part of Fig. 16 shows how to use FIFO to design rectangle pattern matching for a single scan window size using global overlap.

Combination of the luminosity computation using local and global overlaps results in simple and compact hardware design of a single scan window, as shown in the upper part of Fig. 16. The computation for other scan window sizes at the same position can be done in parallel with the same input pixels in a column, as shown in Fig. 9. The final design with all the necessary scan window sizes operating in parallel is shown in the lower part of Fig. 16.

Due to the change in features of the LED speed sign shown in Fig. 10, that is, the black areas became white and vice versa, luminosity difference in the LED sign is also reversed. Instead of reversing the grayscale image for LED sign detection, which requires a lot of computational resources, inversion of the luminosity difference between black and white areas is enough for LED speed sign detection. It can be done by taking the absolute value of the luminosity difference before making a comparison with the threshold.

5.4 Local Pixel Direction Based Circle Detection Implementations

5.4.1 Straight-Forward Implementation

Fig. 17 shows the mechanism and design of the circle recognition module. A 3×3 pixel array is used to detect the direction of the input pixel using local border templates in Fig. 6. The direction is then compared with the expected direction of that pixel. The number of matches is voted on and stored in the register. The final number of matches is compared with a previously decided threshold. If the number of matches is in the predefined threshold range, the input SW is considered a circle.

5.4.2 Fast and Compact Implementation

The above direct circle detection design in Fig. 17 can be improved for faster operation using pipeline and computation reuse of local overlap, which are suitable for raster scanning. Since the templates are 3×3 pixels, data of the last three columns for an SW is buffered for direction confirmation and voting. The pipeline computation mechanism of matched pixel voting for each expected direction in a scan window is shown in Fig. 18. When a new pixel arrives, its corresponding column in the SW and two previous ones are buffered, making a 3-pixel column, as shown in the left side of Fig. 18. There is overlap in 3×3-pixel-patterns among different lines in the 3-pixel column; hence, a 1×3-pixel pattern of each line is checked separately before combining three adjacencies for the final direction confirmation. In a column, the upper part needs to be verified with three directions: down-right (↘), down (↓), and down-left (↙); the middle part needs to be verified with two directions: right (→), left (←); and the lower part needs to be verified with three directions: up-right (↗), up (↑), and up-left (↖). Dividing the input

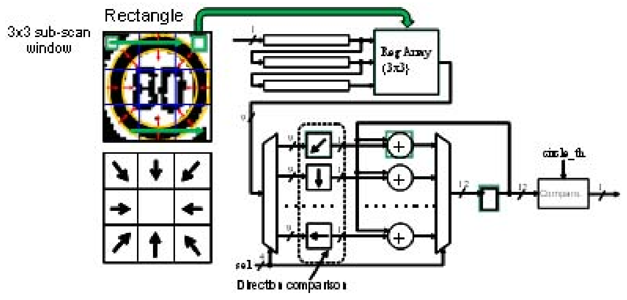


Fig. 17. Straight-forward implementation of circle detection using local pixel direction.

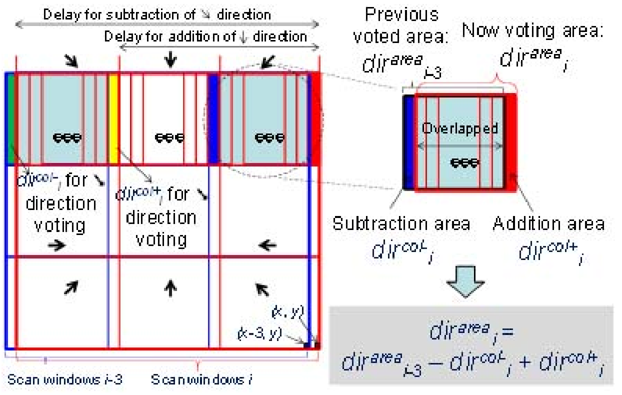

Delay for subtraction of ↘ direction
Delay for addition of ↓ direction
Previous voted area: dir^{area}_{i-3}
Now voting area: dir^{area}_{i}
Overlapped
dir^{col-}_{i} for ↘ direction voting
dir^{col+}_{i} for ↘ direction voting
Subtraction area dir^{col-}_{i}
Addition area dir^{col+}_{i}
(x, y)
(x-3, y)
$dir^{area}_{i} = dir^{area}_{i-3} - dir^{col-}_{i} + dir^{col+}_{i}$
Scan windows i-3
Scan windows i

Fig. 18. Fast and compact local pixel direction voting using overlap.

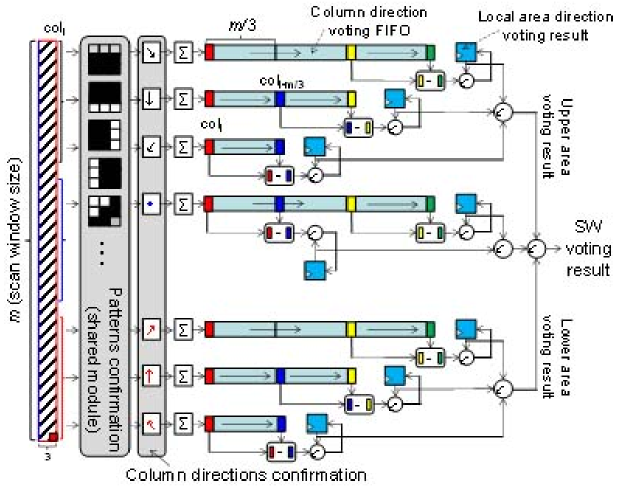

col_i
m/3
Column direction voting FIFO
Local area direction voting result
col_{i-m/3}
col_i
Upper area voting result
SW voting result
Lower area voting result
m (scan window size)
Patterns confirmation (shared module)
Column directions confirmation
3

Fig. 19. Fast and compact implementation of circle detection using local pixel direction and overlap.

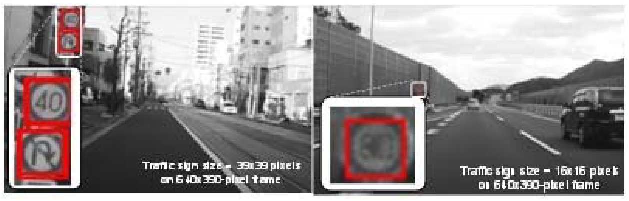

Traffic sign size = 39x39 pixels on 640x390-pixel frame
Traffic sign size = 16x16 pixels on 640x390-pixel frame

Fig. 20. Daytime scenes in local road and highway with sign distortion: recognizable with no difficulty.

column into three parts (upper, middle, and lower) helps to reduce the number of directions that need to be verified at each location to one-third. The right part of Fig. 18 shows the direction voting for one area among the eight, using reusable computational results, in which the direction voting results in the overlap in each area between SWi-3, and SWi is reused without revoting. The voting for an area in SWi is generated by the voting result of the overlapped area plus the voting result of the new input area (dircol+i). The voting result of the overlapped area is the result of the previously voted area (dirareai-3) without the subtraction area (dircol-i). The voting is now for the addition area (dircol+i) and storing the result for later usage as subtraction (dircol-i). At the same time, the newly voted column result dircol+i is added with dirareai-3 before subtracting the previously stored dircol-i voting result for that area's (areai) final voting result (dirareai). As shown in Fig. 18, the waiting time after the direction of the newly input pixels is verified and voted on, until the time it becomes addition dircol+i and subtraction dircol-i of different directions, is different. The hardware implementation of CD using voting for the local pixel direction at the edge is shown in Fig. 19. When a new binary pixel arrives in the system, the corresponding three columns (the last columns in Fig. 18) are given to the direction voting module. Pattern confirmation, a shared module among various SW sizes compares each of the three inputs in a line with the line patterns. Three adjacent results are then combined in a column-direction confirmation submodule before voting for the number of pixels that match a specific direction for each SW size. These results are pushed into column-direction voting FIFOs and become dircol+ and dircol- at a specific time, which depends on the pixel location and size of the scan window. The voting results of all columns in the corresponding area are accumulated, forming area voting result dirarea. Then, dirarea is stored in the local area direction voting register. All local area voting results are added together, making the SWi directions voting result for CD.

6. Evaluation Results and Discussion

6.1 Datasets in Various Conditions

6.1.1 Dataset Taken in Japan

The Japan dataset is used to verify speed sign recognition for moving images under various conditions, as shown in Fig. 20 and Fig. 21. It includes 125 daytime scenes on highways and local roads; 25 scenes on a clear night, and 44 scenes on rainy nights on local roads, as shown in Table 1. When a high-resolution camera (full HD) is used, images at both original and down-sampled sizes are tested. All of the frames are grayscale. Frames with full HD resolution are down-sized to 1/3 on each axis for simulation in our test. The algorithm is easily implemented with few hardware resources by reading the pixels after every three columns and rows. A scene is considered to be all the contiguous frames in which the

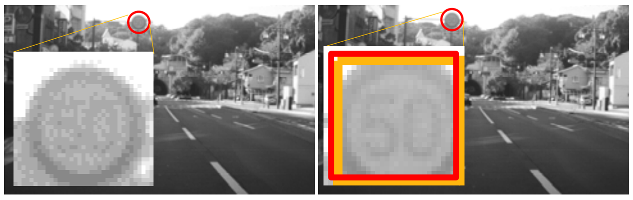


Fig. 21. Adjacent frames of the same sign under backlit condition: left: undetectable; right: detectable.

Table 1. Datasets for accuracy evaluation with various lighting, weather, and camera conditions taken in Japan.

Condition	Camera	Resolution [pixels]	No. of Frames	No. of Scenes
Daytime 1 (Japan)	Grayscale Cam [*1]	640×390 (Original)	41,120	60
Daytime 2 (Japan)	Video Cam [*2]	1920×1080 (Original)	40,000	65
		640×360 (down-sampling)	40,000	65
Clear Night (Japan)	Video Cam [*2]	1920×1080 (Original)	39,136	25
		640×360 (down-sampling)	39,136	25
Rainy Night (Japan)	Video Cam [*2]	1920×1080 (Original)	66,880	44
		640×360 (down-sampling)	66,880	44

[*1]: 60 fps gray scale camera (640 x 390 pixels).
[*2]: 60 fps interlace Full HD color camera.

same sign appears in the observable field of the camera until it disappears, as defined in Section 2 and Fig. 4. Depending on the speed of the car, the number of frames in a scene varies. The size of the sign also varies and gradually increases between adjacent frames as the vehicle gets closer to the sign. Lighting conditions for the same sign also change, depending on the distance and angle between the sign and the camera, as shown in Fig. 21. The dataset also includes signs with different recognition difficulties, depending on the weather and light conditions.

6.1.2 Dataset Taken in Germany

The German Traffic Sign Detection Benchmark (GTSDB) dataset [25], which includes 900 individual traffic images of 1360×800 pixels, is originally used for sign detection contest purposes with machine learning. Since we aim for two-digit speed traffic sign recognition, we created a sub-GTSDB dataset by taking all frames that contain two-digit speed traffic signs. The sub-GTSDB has 255 frames at 1360×800 pixels, as shown in Figs. 22 and 23. “Scene” is not applicable to this dataset because it contains individual frames only. Sizes of the speed limit signs range from 14×16 pixels to 120×120 pixels. Color

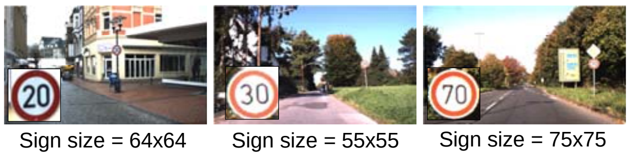


Fig. 22. Dataset (single frame) in German.

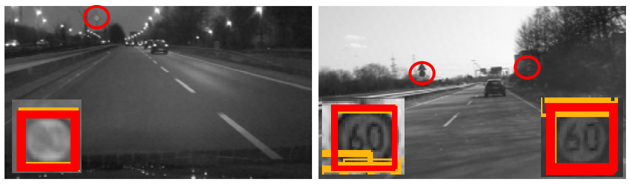


Fig. 23. German frames at day and night times in grayscale. Both are detectable and recognizable.

Table 2. Number of candidates and effectiveness of Circle Detection.

Conditions	No. of sign candidates		No. of speed sign candidates		Effectiveness of CD	
	Best	Avg.	Best	Avg.	Best	Avg.
Daytime (640×390)	200	118	168	109	32	9
Night time (640×360)	403	109	352	96	51	13

images are converted to grayscale images before testing.

6.2 Simulation Results and Discussion

6.2.1 Number of Speed Sign Candidates Detected in RPM and CD, and Processing Time for NR

Table 2 shows the number of traffic sign candidates detected by the RPM module and the number of speed traffic sign candidates detected by the CD module in one frame under various conditions. The full HD input image is down-sampled to match the designed 640×360 pixel resolution. On average, the number of candidates detected by the RPM module is 118, and the number of candidate detected after CD is 96 for night time. CD helps to reduce the maximum to 51 speed traffic sign candidates. This is not a big number, but it helps to remove complicated cases for NR, in which random noise in QR code form and Japanese kanji are taken as speed sign candidates by RPM.

The number of SLT sign candidates can also be reduced by applying region of interest, because the SLT sign appears in the input image in a known area, as shown in Fig. 24 (in Japan). We can concentrate on the SLT sign candidates detected in this area only to reduce processing time for NR.

The time needed to raster scan one frame is 640×360=230,400 clock cycles. In the worst case scenario, when the speed limit traffic sign candidate is as big as 50×50 pixels, the NR module needs two clock cycles for

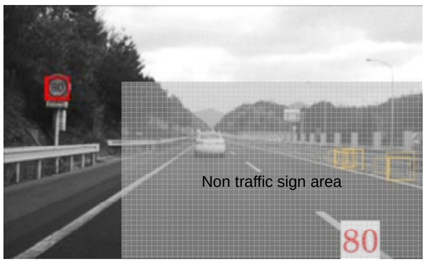


Fig. 24. Appearance area of SLT sign in Japan.

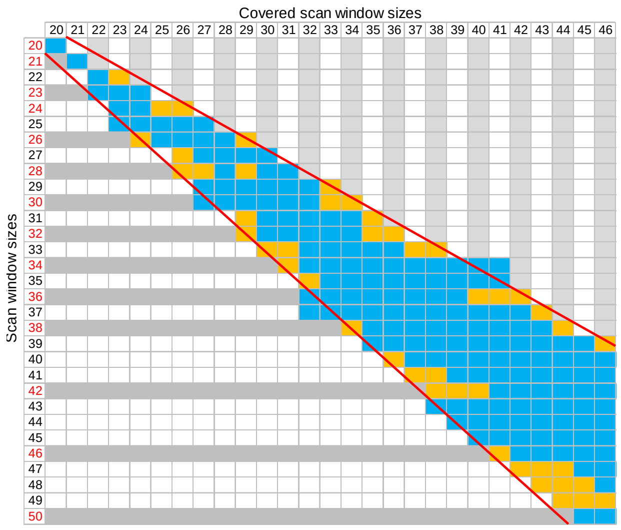
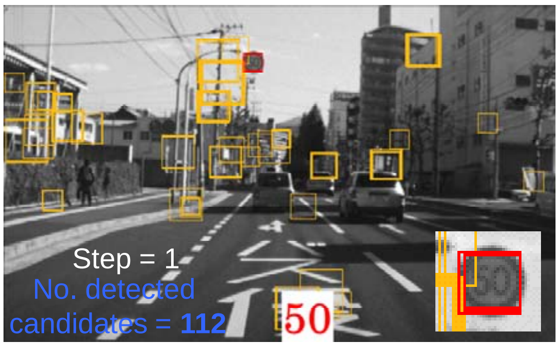


Fig. 25. Cover of each scan window size. Red marks picked up SW sizes for best hardware and accuracy trade-off.

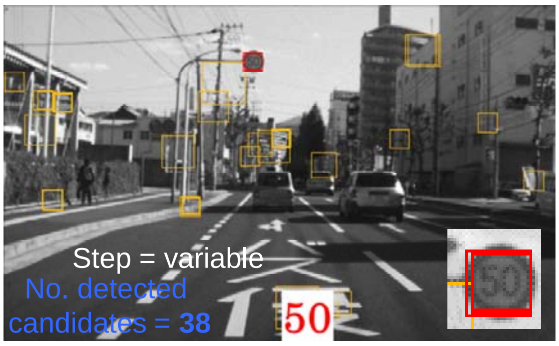
one line of data reading and processing [18], and so 100 clock cycles for number candidate processing. Hence, during 230,400 clock cycles used in RPM, NR can process 2,304 speed sign candidates, which is more than the number of speed sign candidates detected by RPM and CD. The available time (230,400 clock cycles) is enough for all SLT sign candidate recognition in the pipeline implementation in Fig. 12.

6.2.2 Optimization of SW Sizes and Scan Step

Fig. 25 shows the cover of one scan window size over the others. A small SW size, such as 20×20 pixels and 21×21 pixels cannot cover the others as well as be covered by the others. However, the cover size of a SW size is gradually increased, such that SW size of 23×23 pixels can cover SW size of 22×22 pixels, and a scan window size of 34×34 can cover a range from 32×32 to 41×41. The red lines show the coverable area for all SW sizes. Blue points show the SW sizes that are covered by other SW sizes, while the yellow points show the points in which data is not available in the dataset. It suggests that not all scan window sizes are necessary for implementation. Some scan window sizes with less detectable and recognizable areas of SLT sign can be removed. In our simulation, 14 scan window sizes (20, 21, 23, 24, 26, 28, 30, 32, 34, 36, 38, 42, 46, and 50) are enough for sign detection without reduction in accuracy.



(a) 112 speed sign candidates detected with scan step = 1



(b) 38 speed sign candidates detected with variable scan step

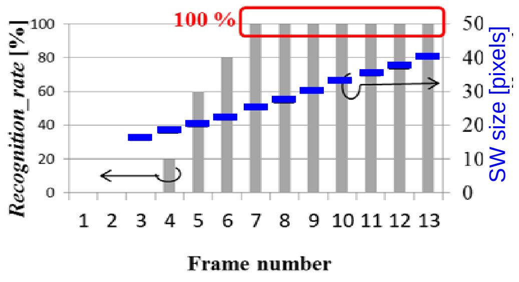
Fig. 26. Effectiveness of variable scan step.

When the scan window moves one pixel, change in the scan area is significant for small scan window sizes, such as 20×20 (a 5% change, because one row or one column of the SW is replaced), but this change is minor for big scan window sizes, such as 50×50 (a 2% change). Hence, scan step can be a variable depending on the SW size. Making the scan step variable reduces the number of candidates that need to be processed in the CD and NR stages, as shown in Fig. 26, in which the number of speed sign candidates is reduced from 112 to 38 with no impact on detection accuracy. SLT sign recognition accuracy in Fig. 27 shows that when the SLT sign size is bigger than 23×23 pixels, the recognition rate gets to 100%, and so, some SW sizes can be removed to save hardware resources with no reduction in accuracy.

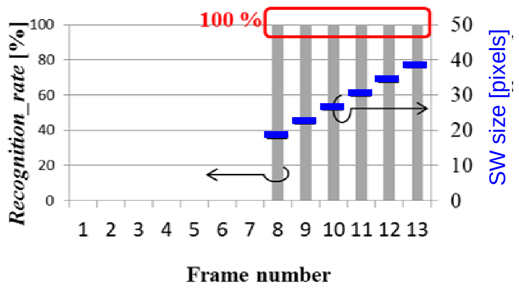
6.3 Hardware Implementation Results and Processing Ability

Table 3 shows the hardware implementation resources and frequency for RPM, CD and related modules in speed limit traffic sign candidate detection. Two implementations with different FIFO approaches are given.

The first implementation utilizes flip-flops available inside slices of the Xilinx FPGA to build FIFO stacks and shift registers that are required in Figs. 17 and 20. Due to the small number of registers available inside each slice, the implementation with all 31 SW sizes in the proposed range (20×20 to 50×50) utilizes 68,552 slice LUTs (128%) and cannot fit the target FPGA. Reducing the number of SW sizes to 14 (20, 21, 23, 24, 26, 28, 30, 32, 34, 36, 38, 42, 46, and 50) significantly reduces the required resources,



(a) Detection accuracy on local road by SW size



(b) Detection accuracy on highway by SW size

Fig. 27. Speed sign recognition accuracy by SW size.

Table 3. Hardware resources and latency.

		All SW sizes (20×20~ 50×50)			14 SW sizes*		
		# slice registers	# slice LUTs	Freq. (MHz)	# slice registers	# slice LUTs	Freq. (MHz)
Reg. based FIFO	RPM	19,763	34,189	202.2	8,289	14,394	202.2
	CD	26,669	34,363	390.3	11,061	14,437	390.3
	Total	46,432 (43.6%)	68,552 (128.9%)	202.2	19,350 (18.2%)	28,831 (54.2%)	202.2
RAM based shift reg. FIFO	RPM	13,951	28,391	321.9	57,704	11,808	321.9
	CD	4,232	14,885	390.3	1,755	6,346	390.3
	Total	18,183 (17.1%)	43,246 (81.3%)	321.9	7,459 (7%)	18,154 (34.1%)	321.9
Controller (RPM+CD)		19	350	-	19	350	-

14 SW sizes*: 20, 21, 23, 24, 26, 28, 30, 32, 34, 36, 38, 42, 46, and 50.
Target device: Xilinx Zynq Automotive Z7020 FPGA.
- 106,400 slice registers.
- 53,200 slice LUTs.

making it implementable on the target FPGA at a frequency of 202.2 MHz. This maximum frequency guarantees that over 60 full HD frames can be processed for sign detection in one second.

The second implementation utilizes the memory inside the LUT in SLICEM to generate a 32-bit shift register without using the flip-flops available in a slice. Since the size of the FIFO stack and shift registers required in Figs. 17 and 20 is not too big, the shift register generated by the memory inside the LUTs is applicable. Since the amount of memory inside the LUTs in a slice is much larger than the number of flip-flops inside a slice, this approach significantly reduces the required resources to 43,246 (81.3%) slice LUTs if all 31 SW sizes are implemented. This number reduces to 18,154 (34.1%) slice LUTs if 14 SW sizes are implemented. Fewer required computational resources also increases the maximum frequency for the detection module to 321.9 MHz, compared with 202.2 MHz achieved by the first one.

6.4 Comparison with Related Works

In terms of speed limit traffic sign candidate detection, our method achieves 100% accuracy. When number recognition is included, the accuracy of the proposed speed

Table 4. Detection rate and throughput of related works.

Dataset	Input image	Scene positive recognition rate (%)	Speed
Daytime 1 Daytime 2	**640×360** 8-bit grayscale	100($\frac{125}{125}$) scenes	**> 60 fps** (Xilinx Zynq 7020)
Night, Rainy night	**Full HD** 8-bit grayscale	100($\frac{69}{69}$) scenes	

Table 5. Detection rate and throughput of related works.

Method	Recognition rate (%)	Thpt. (fps)	Platform	Color / grayscale
Proposed method	**98 (100%*)**	**> 60 fps**	**Zynq***	**both**
SIFT [15]	90.4	0.7	Pentium D 3.2 GHz	Color
Hough [4]	91.4	6.7	Pentium 4 2.8 GHz	Color
Random forest [20]	97.2	18~28	-	Color
Neural network [14]	82.0	62.0	Virtex 4	Color
Fuzzy template [6]	93.3	66.0	Pentium 4 3.0 GHz	Color
Hardware/ Software [17]	-	1.3	Virtex 5	Color
Multi-Core SoC [9]	-	2.3	Virtex 4	Color
Hardware/ Software [1]	-	10.4	Zynq 4	Color
Hardware/ Software [3]	90.0	14.3	Virtex 5	Color
Multi-scale convolutional network [22]	98.6	NA		Color
Multi-column deep neural network [21]	99.5	NA	GPU 512 core	Color

100%* : achieved 100% accuracy with contrast adjustment.
NA : not applicable due to traffic sign recognition only.
Zynq* : Xilinx Automotive Zynq 7020 FPGA.
Virtex : Xilinx Virtex FPGA.

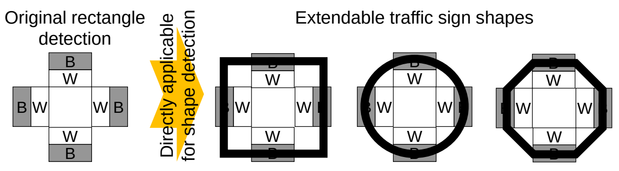

Original rectangle detection
Directly applicable for shape detection
Extendable traffic sign shapes
B
W

Fig. 28. Extendibility to other sign shapes.

limit traffic sign detection and recognition system is 98% if no contrast adjustment is applied to the rainy night scenes. With simple contrast adjustment, accuracy in speed limit sign detection and recognition increases to 100%, even under difficult conditions such as rainy nights, as shown in Table 4.

Table 5 shows the accuracy and throughput of related works. In comparison with other available research, our system gets a higher precision rate and higher throughput than all others. It also requires fewer hardware resources than the others, and so is implementable on a low-cost automotive-oriented Xilinx Zynq 7020 FPGA, whereas the others require a PC or high-end, and high-cost FPGAs, such as the Xilinx Virtex 4 or Virtex 5, in their implementations.

6.5 Function Extendibility

In terms of application, the rectangle pattern–matching algorithm can directly apply to other sign detections aside from the rectangle shape, such as the circle and octagon shapes, as shown in Fig. 28. These three shapes have the same global location and luminosity features, and so we can roughly recognize the ROI for those signs before dividing them into the correct shape based on their local features. The algorithm can also extend to other shapes, such as the triangle and hexagon. Depending on the shape of the target sign border, we need to change the location of black and white area computation.

7. Conclusion

This paper introduces our novel algorithm and implementation for speed limit traffic sign candidate detection using simple features of speed limit traffic signs in a grayscale-image (area luminosity, local pixel direction, area histogram) combination. By using grayscale images, our approach overcomes training and color dependence problems, which reduce recognition accuracy in unlearned conditions when color has changed, compared to other research. The proposed algorithm is robust under various conditions, such as during the day, at night, and on rainy nights, and is applicable to various platforms, such as color or grayscale cameras, high-resolution (4K) or low-resolution (VGA) cameras, and high-end or low-end FPGAs. The combination of coarse- and fine-grain pipeline architectures using results-sharing on overlap, application of a RAM-based shift register, and optimization of scan window sizes provides a small but high-performance implementation. Our proposed system achieves better than 98% recognition accuracy, even in difficult situations, such as rainy nights, is able to process more than 60 full HD fps, and is implementable on the low-cost Xilinx automotive-oriented Zynq 7020 FPGA. Therefore, it is applicable in real life. In the future, a full speed limit traffic sign recognition system will be implemented and verified on an FPGA. Extension to LED signs and other countries' speed limit sign recognition, as well as other traffic sign detection, will be done.

Acknowledgement

Part of this work was supported by Grant-in-Aid for Scientific Research (C) and Research (B) JSPS KAKENHI Grant Numbers 2459102 and 26280015, respectively.

References

[1] Han, Y.: Real-time traffic sign recognition based on Zynq FPGA and ARM SoCs, Proc. 2014 IEEE Intl. Conf. Electro/Information Technology (EIT), pp. 373-376, 2014. Article (CrossRef Link)

[2] Hoang, A.T., Yamamoto, M., Koide, T.: Low cost hardware implementation for traffic sign detection system, Proc. 2014 IEEE Asia Pacific Conf. Circuits and Systems (APCCAS2014), pp. 363-366, 2014. Article (CrossRef Link)

[3] Irmak, M.: Real time traffic sign recognition system on FPGA, Master Thesis, The Graduate School of Natural and Applied Sciences of Middle East Technical University, 2010. Article (CrossRef Link)

[4] Ishizuka, Y., and Hirai, Y.: Recognition system of road traffic signs using opponent-color filter, Technical report of IEICE, No. 103, pp. 13-18, 2004, (in Japanese).

[5] Keller, C. G., et al.: Real-time recognition of U.S. speed signs, Proc. Intl. IEEE Intelligent Vehicles Symposium, pp. 518-523, 2008. Article (CrossRef Link)

[6] Liu, W., et al.: Real-time speed limit sign detection and recognition from image sequences, Proc. 2014 IEEE Intl. Conf. Artificial Intelligence and Computational Intelligence (AICI), pp. 262-267, 2010. Article (CrossRef Link)

[7] Miura, J., et al.: An active vision system for real-time traffic sign recognition, Proc. Intl. IEEE Conf. Intelligent Transportation Systems, pp. 52-57, 2000. Article (CrossRef Link)

[8] Moutarde, F., et al.: Robust on-vehicle real-time visual detection of American and European speed limit signs, with a modular traffic signs recognition system, Proc. Intl. IEEE Intelligent Vehicles Symposium, pp. 1122-1126, 2007. Article (CrossRef Link)

[9] Muller, M., et al.: Design of an automotive traffic sign recognition system targeting a multi-core SoC implementation, Proc. Design, Automation & Test in Europe Conference & Exhibition 2010, pp. 532-537, 2010. Article (CrossRef Link)

[10] Ozcelik, P. M., et al.: A template-based approach for real-time speed limit sign recognition on an embedded

system using GPU computing, Proc. 32nd DAGM Conf. Pattern Recognition, pp. 162-171, 2010. Article (CrossRef Link)

[11] Raiyn, J.,and Toledo, T.: Real-time road traffic anomaly detection, Journal of Transportation Technologies, 2014, No. 4, pp. 256-266, 2014. Article (CrossRef Link)

[12] Schewior, G., et al.: A hardware accelerated configurable ASIP architecture for embedded real-time video-based driver assistance applications, Proc. 2011 IEEE Intl. Conf. Embedded Computer Systems (SAMOS), pp. 18-21, 2011. Article (CrossRef Link)

[13] Soendoro, D., and Supriana, I.: Traffic sign recognition with color based method, shape-arc estimation and SVM, Proc. 2014 IEEE Intl. Conf. Electrical Engi-neering and Informatics (ICEEI), pp. 1-6, 2011. Article (CrossRef Link)

[14] Souani, C., Faiedh, H., and Besbes, K.: 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. Article (CrossRef Link)

[15] Takagi, M., and Fujiyoshi, H.: Road sign recognition using SIFT feature, Proc. 18th Symposium on Sensing via Image Information, LD2-06, 2007, (in Japanese).

[16] Torresen, J., et al.: Efficient recognition of speed limit signs, Proc. 7th Intl. IEEE Conf. Intelligent Transportation Systems, pp. 652-656, 2004. Article (CrossRef Link)

[17] Waite, S. and Oruklu, E.: FPGA-based traffic sign recognition for Advanced Driver Assistance Systems, Journal of Transportation Technologies, Vol. 3, No. 1, pp. 1-16, 2013. Article (CrossRef Link)

[18] Yamamoto, M., Hoang, A-T., Koide, T.: Speed traffic sign recognition algorithm for real-time driving assistant system, Proc. 18th Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2013), pp. 195-200, 2013. Article (CrossRef Link)

[19] Zaklouta, F. and Stanciulescu, B.: Segmentation masks for real-time traffic sign recognition using weighted HOG-based trees, Proc. 14th Intl. IEEE Conf. Intelligent Transportation Systems (ITSC), pp. 1954-1959, 2011. Article (CrossRef Link)

[20] Zaklouta, F., Stanciulescu, B.: Real-time traffic sign recognition in three stages, Journal of Robotics and Autonomous Systems, Vol 62, Issue 1, pp. 16-24, 2014. Article (CrossRef Link)

[21] Ciresan, D., et al.: Multi-column deep neural network for traffic sign classification, Journal of Neural Networks, Vol 32, pp. 333-338, 2012. Article (CrossRef Link)

[22] Sermanet, P., LeCun, Y.: Traffic sign recognition with multi-scale convolutional networks, Proc. Intl. Joint IEEE Conf. on Neural Networks (IJCNN), pp. 2809-2813, 2011. Article (CrossRef Link)

[23] Zaklouta, F., Stanciulescu, B., and Hamdoun. O: Traffic sign classification using K-d trees and random forests, Proc. Intl. Joint IEEE Conf. on Neural Networks (IJCNN), pp. 2151-2155, 2011. Article (CrossRef Link)

[24] Article (CrossRef Link), access at March, 15th, 2015

[25] Article (CrossRef Link), access at June, 10th, 2015.

[26] Hoang, A-T., Yamamoto, M., Koide, T.: Simple yet effective two-stage speed traffic sign recognition for robust vehicle environments, Proc. 30th Intl. Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC 2015), pp.420-423, 2015.

Anh-Tuan Hoang was born in 1976. He received his Ph.D. from Ritsumeikan University in 2010. He was a postdoctoral researcher at Ritsumeikan University, Japan, from 2010 to 2012. Since 2012, he has been a postdoctoral researcher at the Research Institute for Nanodevice and Bio System, Hiroshima University, Japan. His research interest is side channel attack and tamper-resistant logic design, real-time image processing and recognition for vehicle and medical image type identification. He is a member of the IEEE and IEICE.

Tetsushi Koide (M'92) was born in Wakayama, Japan, in 1967. He received a BEng in Physical Electronics, an MEng and a PhD in Systems Engineering from Hiroshima University in 1990, 1992, and 1998, respectively. He was a Research Associate and an Associate Professor in the Faculty of Engineering at Hiroshima University from 1992-1999 and in 1999, respectively. After 1999, he was with the VLSI Design and Education Center (VDEC), The University of Tokyo, as an Associate Professor. Since 2001, he has been an Associate Professor in the Research Center for Nanodevices and Systems, Hiroshima University. His research interests include system design and architecture issues for functional memory-based systems, real-time image processing, healthcare medical image processing systems, VLSI CAD/DA, genetic algorithms, and combinatorial optimization. Dr. Koide is a member of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, the Institute of Electronics, Information and Communication Engineers of Japan, and the Information Processing Society of Japan.

Masaharu Yamamoto was born in 1990. He received a BEng in Electrical Engineering from Hiroshima University, Hiroshima, Japan in 2013. He is currently at TMEiC, Japan. Since 2012, he has been involved in the research and development of hardware-oriented image processing for advanced driver assistance systems (ADASs).