RESOURCE EFFICIENT HARDWARE IMPLEMENTATION FOR REAL-TIME TRAFFIC SIGN RECOGNITION

Huai-Mao Weng, and Ching-Te Chiu

Department of Electrical Engineering, National Tsing-Hua University, Hsinchu, Taiwan
woodghost11235@gmail.com, ctchiu@cs.nthu.edu.tw

ABSTRACT

Traffic sign recognition (TSR) is one of the Advanced Driver Assistance System (ADAS) device in modern cars. We propose a high efficiency hardware implementation for TSR, which is divided into two stages. In the detection stage, we use Normalized RGB color transform and Single-Pass Connected Component Labeling (CCL) to find the potential traffic signs. In the recognition stage, the Histogram of Oriented Gradient (HOG) is used to generate the descriptor of the signs, and we classify the signs with the Support Vector Machine (SVM). The proposed method achieves 96.61% detection rate and 90.85% recognition rate while testing with the GTSDB dataset. Our hardware implementation reduces the storage of CCL and simplifies the HOG computation. By using TSMC 90nm technology, the proposed design operates at 105 MHz clock rate and processes in 135 fps with the image size of 1360×800. The chip size is about 1mm^2 and the power consumption is close to 8mW. Therefore, this work is resource efficient and achieves real-time requirement.

Index Terms— Traffic Sign Recognition, Advanced Driver Assistance System, Real-Time Processing, Connected Component Analysis, VLSI

1. INTRODUCTION

Advanced Driver Assistance System (ADAS) is a common device in modern cars. ADAS is designed to enhance car safety and driving comfort. The Traffic Sign Recognition (TSR, or Road Sign Recognition, RSR) can help driver notice the fast passed signs, thus avoiding traffic violations nor accidents. The TSR must be fast enough to react to the changing traffic conditions. The recognition rate should be high enough to detect most signs and prevent false information. The method should be fast as well as easy to implement with hardware, which is able to process in real-time. Besides, a well-designed hardware implementation should be low cost as well as energy saving, that means resource efficiency.

There are many researches for TSR proposed. Mammeri *et al.* [1] classify the most TSR methods into three categories. The image-processing based (traditional computer-vision) methods use various computer-vision technique to find traffic signs. It is fast, but the recognition rate is relatively low. The machine learning based method achieves high recognition rate, but the complexity makes it difficult to process in real-time. The hybrid method takes both advantages of fast computation and high recognition rate. It detects the potential signs in the source image by the image-processing based method, and recognizes the signs by the machine learning based method.

The hybrid method divides the TSR procedure into two stages, detection and recognition. In the detection stage, a specialized color transform with thresholding, followed by the Connected Component Labeling (CCL) procedure, is an effective way to extract traffic signs [2, 3, 4]. Some researches [5, 6, 7] use the maximally stable extremal region (MSER) to extract the stable regions. Liu *et al.* [8] propose the HCRE to extract the regions by a possibility map. In the recognition stage, the descriptor and classifier method is often used. The common descriptor choices are the Histogram of Oriented Gradient (HOG) [9, 5, 3, 4, 7] or modified HOG [10, 6]. The classifier can be support vector machine (SVM) [11, 5, 3, 4, 6], neural networks [7] or SRC-based classifier [8]. Most of the recently propose researches implement the hybrid method, so does this work.

The proposed TSR method is illustrated in Section 2. Section 3 shows the performance evaluations. Section 4 introduces the hardware architecture of the proposed method and the implementation results. A brief conclusion is given in Section 5.

2. PROPOSED TSR METHOD

2.1. Detection Procedure

The detection procedure is used to detect the potential traffic signs form the source image. It is the most time consumption part of the whole TSR procedure. Because we have to scan the whole source image to find the potential traffic signs (or Region-of-Interest, ROI) before recognizing every found ROIs. Figure 1 shows the results of each step in the detection procedure.

First, we reduce the searching space by removing the top 25% and the bottom 15% of the source images. In addition,

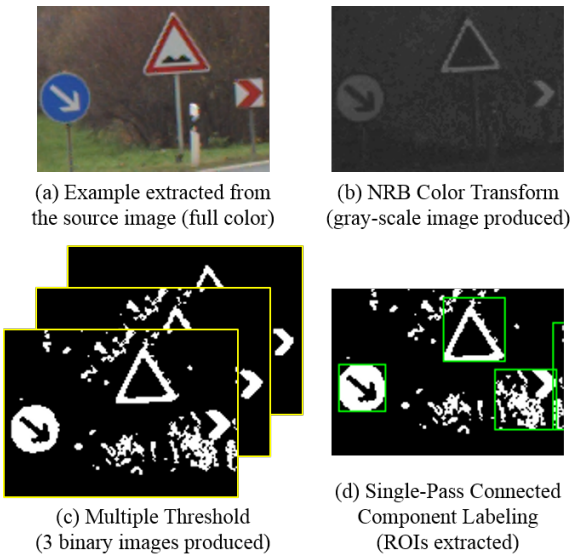

Fig. 1. Results of each step in the detection procedure

we split a source image into left-frame and right-frame, then process two frames independently. Our experiment shows that there are less than 4% of total signs are out of the searching space. Overall, the searching space is reduced by 40% and two independent frames generated.

Second, we use the color segmentation to extract the specified color of the traffic sign boundary, e.g. red and blue. The first part of color segmentation is the transform. The researches [11, 5, 6] claim that the Normalized-RGB Color Transform performs the most robust result under different lighting and contrast condition. A gray-scale image is generated by the color transform, and the value is higher for red or blue, lower for other colors. We want to emphasize the negative influence of green color. The modified normalized RGB color transform, called NRB color transform, is shown as following:

$$NRB(R,G,B) = \frac{max(R,B)}{R + G \times \beta + B + \alpha}, \qquad (1)$$

where R, G, B are the red, green and blue color channel, respectively. $max(R,B)$ is the maximum value between R and B. α is a small constant that set to 8. β is a green color magnification, we set to 4 in this work.

The second part of color segmentation is the thresholding. A threshold is used to determine if the pixel has the specified color or not. Thus, the binary image is produced. For the flexibility under different conditions, we use multiple thresholds instead of a single threshold. Hence, multiple binary images are produced by thresholding.

Third, we use the Connected Component Labeling (CCL) algorithm to group the white pixels in the binary image. The algorithm scans the binary image and assigns the label to each pixel using only local operations. A set of connected pixels is called a "region". Two pixels are "connected" if they are adjacent to each other. The single-pass CCL algorithm [12] eliminates the second pass of traditional two-pass algorithm by record the "features" of region during the first pass. Therefore, the theoretical computing time is reduced by half.

When scanning pixels in the image, the decision of the current label number is observed by only the neighbor labels in the previous row. The "row-buffer" records the labels of previous row instead of the whole labeled image. Because the single-pass CCL outputs the feature of regions, the "data-table" is introduced to store the extracted features. Overall, the memory of the CCL is greatly reduced by the elimination of the labeled image. We select the bounding box as the region feature in this work. The bounding box contains the boundaries of regions, which lets us extract the ROIs from the source image. Finally, we pass the detected ROIs to the next main procedure, recognition.

2.2. Recognition Procedure

The recognition procedure classifies the potential traffic signs found by the detection procedure. The schematic diagram of the recognition procedure is shown in Figure 2. We first check the region criteria by observing only the bounding box information. The width and height of the region should not be greater than 140*pixel* nor less than 20*pixel*. The width/height ratio should be within 0.75 and 1.25. According to the experiment, there are averagely 10.75 ROIs per image that meet the criteria. We extract the eligible regions from the source image. They are converted into gray-scale and resized to 32×32, before the HOG calculation.

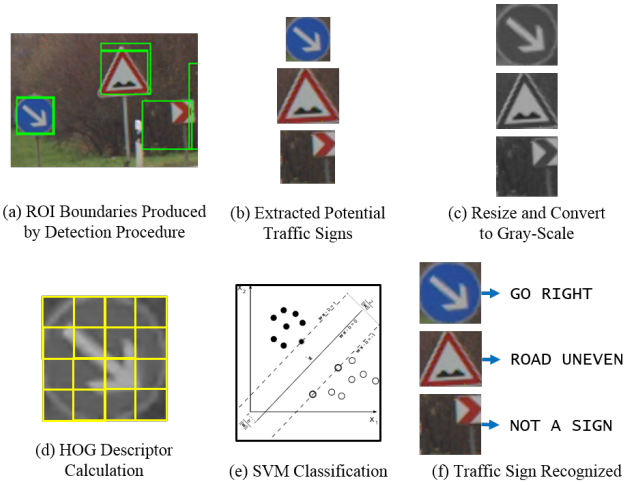

Fig. 2. Schematic diagram of the recognition procedure

Next, we calculate the HOG descriptor of the legal regions. We choose the HOG parameters that reference to Dalal

et al. [13]: 3x3 blocks per window, 2x2 cells per block, 8x8 pixles per cell, a 8-bin histogram spreads over 0 to 360 degrees, and the arithmetic mean is used for normalization. We use fixed-point calculation for simplification and faster computation. In other words, the quantization is performed for easier hardware implementation. The gradient calculation of X-axis and Y-axis are $[-1, 0, 1]$ and $[-1, 0, 1]^T$. The magnitude is approximate by the Manhattan distance:

$$|grad| = |grad_x| + |grad_y|\,, \tag{2}$$

where $|grad_x|$ and $|grad_y|$ are the absolute values of X-gradient and Y-gradient. When we need only 8 bins result that distributes over 0 to 360 degrees, the orientation calculation can be approximated as the following formula:

$$ori_{x,y} = (grad_y < 0) \cdot 4 + (grad_x < 0) \cdot 2 + (|grad_x| > |grad_y|) \cdot 1, \tag{3}$$

where the comparing calculation (e.g. $<$ and $>$) produces 1-bit result. The value of orientation result is an integer between 0 to 7, which represents a bin of the one-eighth in 360 degrees.

After the HOG descriptor of a region is generated, we classify the ROIs with the Support Vector Machine (SVM) [14]. The parameters for SVM are: one vs. one type, linear kernel, $\gamma = 0.1$, and $C = 10$. We use the 80-20 ratio for cross-validation. Finally, the traffic signs in the source image are recognized and the false positives are removed.

3. PERFORMANCE EVALUATION

3.1. Testing Dataset and Environment Settings

We use the German Traffic Sign Detection Benchmark (GTSDB) [15] to test our TSR method. The target sign size is between 24×24 and 128×128. Our experiment contains about 500 images with more than 700 traffic signs. Figure 3 shows the 35 target sign classes and the 3 major classes.

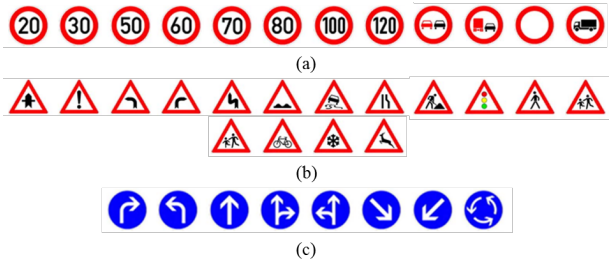

Fig. 3. Traffic sign classes and the major classes of GTSDB. (a) Prohibitory (b) Danger (c) Mandatory

If a detected region has more than 75% overlap area with the ground-truth, we decide that it is a detection hit. If there is a correct detected sign, it should be classified into the corresponding class. Else, the false detected regions should be classified as non-traffic-sign. The software implementation is written in C++ using the OpenCV 3.0 library. The SVM uses the LIBSVM [16] library. Our experiment executes on the PC with Intel Xeon E3-1230v2 (3.2GHz) CPU and 16GB RAM.

3.2. The Influence of Color Segmentation Thresholds

We use multiple thresholds to overcome the variation of weather and lighting conditions. Table 1 shows the detection rate under multiple thresholds. However, the improvement is negligible when using more than three thresholds. The disadvantage of multiple thresholds is that the CCL needs more time to process multiple images. Overall, we select three thresholds to reach the 96.61% detection rate.

Table 1. Color Segmentation Thresholds vs. Detection Rate

thresholds	56	52,56	52,60	52,64	56,60	56,64	52,56,60	52,56,64
detection rate	0.7791	0.9201	0.9327	0.9130	0.9046	0.9032	0.9661	0.9593

3.3. The Influence of Reducing HOG Calculation Bits

For easier hardware implementation and resource efficiency, we use fix-point calculation for the HOG descriptor. Thus, the bit width of HOG input and output should be decided. According to Table 2, the reducing of input bits does not have significant influence on recognition rate until 4-bit. Table 3 shows the influence of reducing HOG output bits. We find that the recognition rate is almost unchanged while reducing the descriptor bits, until the 4-bit fixed point is tested. In addition, if we truncate the descriptor values, there is one more output bit reduced. Even though, the recognition rate keeps unchanged, because there are too little overall affected values. In brief, the proposed TSR method reaches the 90.85% recognition rate when the data width is 4-bit for input and 3-bit for output.

Table 2. The influence of reducing HOG input bits vs. recognition rate

recognition rate (%)		input bits				
		6	5	4	3	2
output bits	10	91.40	90.29	90.43	87.79	77.66
	7	90.84	90.29	91.12	87.65	76.97
	4	90.70	90.15	90.29	86.13	75.86

Table 3. The influence of reducing HOG input bits vs. recognition rate

recognition rate (%)		output bits				
		6	5	4	3	2
input bits	8	90.70	91.12	91.40	88.76	80.85
	6	91.53	91.40	90.70	89.04	79.05
	4	90.70	90.29	90.29	87.79	79.88

4. HARDWARE IMPLEMENTATION

4.1. Overall Architecture

The hardware design should implement the most critical part of the TSR method. Therefore, we implement the detection procedure and the HOG calculation of recognition procedure in hardware. The source image first inputs to the hardware color segmentation unit, pixel by pixel as raster scanning. The pixels that determined as foreground or background are delivered to the single-pass CCL unit to group the connected pixels into a ROI. Next, the detected ROIs are checked with several conditions by the software procedure, and the ROIs that pass the conditions are resized to a constant size. The HOG unit generates the descriptors of resized ROIs. Finally, these descriptors are classified by software SVM, and the traffic sign recognition is done.

The top module for TSR contains two independent main modules, detection module and recognition module. The input data of detection module is the source image. The output is the region information of potential signs (or ROIs). In fact, the information is the feature vector of ROIs. The detection module contains two sub-modules, color segmentation (CS) and connected component labeling (CCL). After NRB transforming and thresholding in CS module, the color pixel reduces into 1-bit value and passes to CCL module. The input of recognition module is the resized region and the output is HOG descriptor. There is a HOG sub-module contained in the recognition module. Figure 4 shows the block diagram of TSR top module.

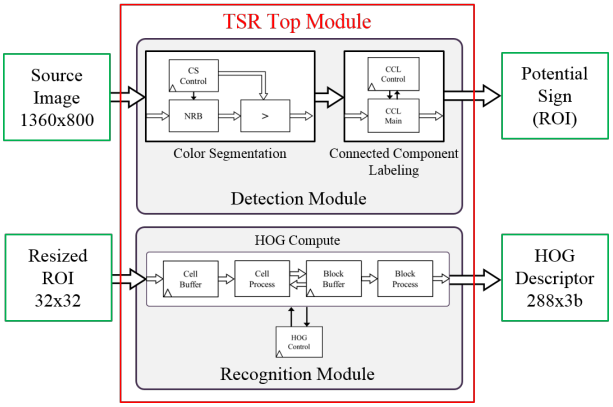

Fig. 4. Block diagram of TSR top module.

The detection module receives one pixel per cycle. We set the frame size to 680×480, and there are two frames in an image. Additional 128 cycles after scanning a row is needed. Thus, the detection procedure takes 387840 cycles to process a frame, which means 775680 cycles to process an image. For the HOG module, we need 64+64 cycles to fill and process the cell-buffer. The block-buffer contains 4 cells. There are another 32 cycles to output the descriptor in a block and total 9 blocks to be computed. Overall, the detection procedure takes 4896 cycles to process a ROI. There are 10.75 ROIs per image on average, which means it takes 52632 cycles to process all ROIs in an image. The detection module and the recognition module are independent. When the detection module produces a valid ROI, the ROI will be send to the recognition module. Therefore, the detection module is the bottleneck. Briefly, the proposed TSR design requires 775680 cycles to process an image.

4.2. Implementation Results and Comparisons

We synthesize and perform APR to the proposed TSR hardware architecture with TSMC 90nm CMOS technology. The core size is 0.26mm^2 and chip size is about 1mm^2. The design operates at 105 MHz clock frequency. It needs 775680 cycles to process an input image, which means the speed is up to 135fps, or 7.4ms per image.

We compare our TSR hardware design with other TSR hardware designs. The designs could be Field Programmable Gate Array (FPGA) [17, 18, 19, 20] or Application-Specific Integrated Circuit (ASIC) [20]. According to the comparison in Table 4, our design has the fastest processing speed with large input image size.

Table 4. Comparison of the TSR Hardware Designs

Design	Image Size	Technology	Area / Gate Count	Freq. (MHz)	Speed (ms/frame)
[17]	640x480	Xilinx Virtex 4	160K NAND	88	16
[18]	320x240	Xilinx Virtex 5	N/A	100	114
[19]	1280x720	Xilinx Spartan 6	N/A	75	16.6
[20]	320x240	130nm	10.61mm^2	200	33.3
This work	1360x800	90nm	0.26mm^2	105	7.4

5. CONCLUSIONS

In this paper, we propose a traffic sign recognition method and its resource efficient hardware implementation that processes in real-time. The proposed TSR method achieves 96.61% detection rate and 90.85% recognition rate while testing with GTSDB dataset. Our hardware implementation reduces the storage of CCL, and simplifies the HOG computation. The proposed hardware operates at 105 MHz clock frequency by using TSMC 90nm CMOS technology. With image size of 1360×800, the processing speed is up to 135 fps. The chip size is about 1mm^2 and the power consumption is close to 8mW. Therefore, this work is resource efficient and achieves real-time requirement.

6. REFERENCES

[1] A. Mammeri, A. Boukerche, and M. Almulla, "Design of traffic sign detection, recognition, and transmission systems for smart vehicles," *IEEE Wireless Communications*, vol. 20, no. 6, pp. 36–43, December 2013.

[2] A. Ruta, Y. Li, and X. Liu, "Real-time traffic sign recognition from video by class-specific discriminative features," *Pattern Recognition*, vol. 43, no. 1, pp. 416 – 430, 2010.

[3] Z. Chen, X. Huang, Z. Ni, and H. He, "A gpu-based real-time traffic sign detection and recognition system," in *2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems (CIVTS)*, Dec 2014, pp. 1–5.

[4] F. Zaklouta and B. Stanciulescu, "Real-time traffic sign recognition in three stages," *Robotics and Autonomous Systems*, vol. 62, no. 1, pp. 16 – 24, 2014, new Boundaries of Robotics.

[5] J. Greenhalgh and M. Mirmehdi, "Real-time detection and recognition of road traffic signs," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1498–1506, Dec 2012.

[6] X. Yuan, X. Hao, H. Chen, and X. Wei, "Robust traffic sign recognition based on color global and local oriented edge magnitude patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 4, pp. 1466–1477, Aug 2014.

[7] Y. Yang, H. Luo, H. Xu, and F. Wu, "Towards real-time traffic sign detection and classification," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 2022–2031, July 2016.

[8] C. Liu, F. Chang, Z. Chen, and D. Liu, "Fast traffic sign recognition via high-contrast region extraction and extended sparse representation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 79–92, Jan 2016.

[9] M. Liang, M. Yuan, X. Hu, J. Li, and H. Liu, "Traffic sign detection by roi extraction and histogram features-based recognition," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug 2013, pp. 1–8.

[10] G. Overett, L. Tychsen-Smith, L. Petersson, N. Pettersson, and L. Andersson, "Creating robust high-throughput traffic sign detectors using centre-surround hog statistics," *Machine Vision and Applications*, vol. 25, no. 3, pp. 713–726, Apr 2014.

[11] . Gonzalez, M. . Garrido, D. F. Llorca, M. Gavilan, J. P. Fernandez, P. F. Alcantarilla, I. Parra, F. Herranz, L. M. Bergasa, M. . Sotelo, and P. R. de Toro, "Automatic traffic signs and panels inspection system using computer vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 12, no. 2, pp. 485–499, June 2011.

[12] A. AbuBaker, R. Qahwaji, S. Ipson, and M. Saleh, "One scan connected component labeling technique," in *2007 IEEE International Conference on Signal Processing and Communications*, Nov 2007, pp. 1283–1286.

[13] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, June 2005, pp. 886–893 vol. 1.

[14] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[15] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The german traffic sign detection benchmark," *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2013.

[16] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 27:1–27:27, May 2011.

[17] 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, no. 1, pp. 79–93, Mar 2014.

[18] S. Waite, "Fpga-based traffic sign recognition for advanced driver assistance systems," vol. 03, pp. 1–16, 01 2013.

[19] N. Aguirre-Dobernack, H. Guzmn-Miranda, and M. A. Aguirre, "Implementation of a machine vision system for real-time traffic sign recognition on fpga," in *IECON 2013 - 39th Annual Conference of the IEEE Industrial Electronics Society*, Nov 2013, pp. 2285–2290.

[20] J. Park, J. Kwon, J. Oh, S. Lee, J. Y. Kim, and H. J. Yoo, "A 92-mw real-time traffic sign recognition system with robust illumination adaptation and support vector machine," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 11, pp. 2711–2723, Nov 2012.