A Pipeline Architecture for Traffic Sign Classification on an FPGA

Yuteng Zhou, Zhilu Chen, and Xinming Huang
Department of Electrical and Computer Engineering, Worcester Polytechnic Institute, MA 01609, USA

***Abstract*—This paper presents an efficient FPGA design that can classify 48 different traffic signs at real-time. The method is based on histogram of oriented gradients (HOG) feature extraction and support vector machine (SVM) for classification. A full-pipeline, resource efficient architecture is presented with detail design of each block. The FPGA implementation has a system clock of 241.7 MHz with an absolute response time of 6.5 s. Taking streaming pixel input, the system throughput is about 106 times faster than the same algorithm executed on a general purpose processor.**

***Index Terms*—Traffic signs, classification, HOG, SVM, FPGA**

I. Introduction

Traffic sign classification is an important task for driver assistance systems and self-driving vehicles [1]. Many accidents were occurred because the drivers did not observe the stop or yield signs at intersections. For accurate detection and classification of traffic signs, there remain many challenges. At first, there are many different types of traffic signs and some of them are very similar. Secondly, traffic signs may exist in various backgrounds, illumination conditions, and occlusions. Lastly, traffic sign detection and recognition must happen in real-time such that the system is practically useful.

In this paper, we focus on the classification problem. There is existing research on using histogram of oriented gradients (HOG) to extract features and then employ support vector machine (SVM) to perform classification. However, when implemented on a general purpose computer this method could take up to several seconds to process a single image frame [2] due to the complexity of the HOG operations. HOG are features descriptors widely used in computer vision for the purpose of object recognition such as motion detection, face recognition [3], and has also been used to extract features from traffic signs [4]. Some existing works have successfully implemented the HOG algorithm on an FPGA [5] or a platform with both FPGA and GPU [6]. But the HOG computation slows down the overall system performance significantly [7].

SVM belongs to a category of supervised learning methods. It constructs a set of high dimensional space that can separates the training data points multiple classes at the largest distance. Usually a SVM with nonlinear kernel results better performance but the linear SVM has lower complexity. Thus linear SVM is often chosen for the implementations on an FPGA [8] [9]. However, most of these SVM implementations are for two classes, or simply yes or no classification. In our case, we need to classify many different traffic signs simultaneously. Thus, we propose a parallel architecture with 48 one-vs-all SVM classifiers on the same FPGA.

Figure 1. A collection of 48 traffic signs that can be classified in our system

In this paper, we present an FPGA accelerator for traffic sign classification. The HOG algorithm has been implemented efficiently with minimum resources. The traffic candidates in 32-by-32 pixels are fed into the system as a stream with one pixel per clock cycles. The pipeline architecture can process these images continuously. A 48-way SVM is implemented to perform one-vs-all classifications for 48 traffic signs in the library.

The rest of the paper is organized as follows. Section 2 describes the algorithms of HOG as the feature descriptors and SVM as a linear classifier. Section 3 discusses the full pipeline architecture and detail design of each block. Section 4 provides the FPGA resource utilization and system performance evaluation. Section 5 gives the conclusions.

II. Methods for Traffic Sign Classification

The input to the FPGA system is a 32-by-32 pixel image that contains a candidate of traffic sign. We first obtain a total of 324 HOG feature descriptors for each input image. The HOG descriptors are connected to the SVM module for classification. The SVM is trained using BelgiumTS dataset [10] that contains a large number of positive and negative samples. More details about the training algorithm and its GPU implementation can be referred to our previous work [11]. Figure 1 shows a collection of all 48 traffic signs that we usually use. A few special types of signs are not included

978-1-4799-8391-9/15/$31.00 ©2015 IEEE

in our system because the dataset does not contain enough samples for training the SVM.

A. HOG Feature Descriptors

As an initial step of HOG, each input image is divided into blocks and cells. A block size is 16 * 16 pixels and a cell size is 8 * 8 pixels. When a block is sliding horizontally in a window, step is 8 pixels. The same step size applies when sliding vertically. Due to partial overlapping, the original image in the size of 32-by-32 pixels contains 9 blocks for HOG feature extraction.

For each block, 36 HOG descriptors are calculated. The HOG computation has three steps. They are weighted magnitude and bin class calculation, block histogram generation, and normalization.

For each pixel in the image, we first compute the its gradients in both *x* and *y* directions. The gradient for current pixel is determined by its neighboring pixels' luminance values using the equations below:

$$G_x(x,y) = |M_x(x+1,y) - M_x(x-1,y)| \tag{1}$$

$$G_y(x,y) = |M_y(x,y+1) - M_y(x,y-1)| \tag{2}$$

Then, the gradient magnitude and the gradient angle are given by:

$$G(x,y) = \sqrt{G_x(x,y) + G_y(x,y)} \tag{3}$$

$$\theta = \arctan \frac{G_y(x,y)}{G_x(x,y)} \tag{4}$$

According to the angle value, the gradient is assigned to 9 different bins. From 0 to 180 degrees every 20 degrees denote one bin. For each cell, a corresponding histogram is generated by summing up the weighted magnitude for each bin. In the step of computing block histogram, a total of 36 bin values are obtained from the histograms of four cells in one block. For each input image, all histograms from 9 blocks are concatenated together, generating 324 HOG descriptors.

Normalization is a necessary step to make the algorithm more robust to illumination and contrast changes. For the implementation on FPGA, the normalization equation can be simplified as,

$$b_{norm} = \sqrt{\frac{b}{sum(b)}} \tag{5}$$

B. Classification Using One-vs-All Linear SVM

The aim of SVM is to quickly separate hyperplanes between different classes and to efficiently map the inputs to high-dimensional feature spaces. Here, a linear SVM is employed which requires much less computation than nonlinear kernels. The linear SVM can be represented as,

$$Y = \alpha y^T + \gamma \tag{6}$$

where α is the support vector matrix, *y* is the vector of HOG feature descriptors, and γ is the offset. In order to classify 48 different signs, we train one SVM for each sign using the dataset. For classification, we can combine these 48 linear SVM together using matrix-vector operations. We refer it as a 48-way one-vs-all SVM.

Since each image produces 324 HOG descriptors, *y* is a 1-by-324 vector, α is a 48-by-324 matrix, and γ is a 1-by-48 vector. Result *Y* is a vector with 48 elements, and the traffic sign belongs to the class with the largest *Y* value.

III. FPGA Implementation

Both HOG feature extraction and SVM classifier are implemented on the same FPGA. This section presents pipeline architecture and the implementation details of each module.

A. Overall System Architecture

Figure 2 illustrates overall system architecture. The HOG algorithm is implemented into 5 modules in the form of a pipeline. Each module stands for an IP core. The input of an IP core is stored into a block RAM and an IP core sends output by streaming one pixel per clock cycle. The entire design is running in a single clock domain with its system clock frequency equal to the pixel rate.

The *Gradient Calculation* block takes in the input image one pixel at a time, and computes gradients G_x and G_y for each pixel. The *Magnitude & Bin Calculation* block computes magnitude for each pixel as in equation (3), and it also classifies the current pixel into 9 different bin classes. The *Address Encoder* block simply groups the magnitude and bin classes of every 8 pixels together and stores them into a block RAM. The *Cell Summation* block reads data from the previous block, calculates the histogram of each cell which contains 8 * 8 pixels, and then stores the results into another block RAM. Similarly, the *Normalization* block extracts 36 HOG features from each 16 * 16 pixels block, and stores its results into a block RAM. Finally, the *SVM* block performs the matrix multiplication row-by-row for each class and then provides the classification results by comparing the values of all 48 classes.

Figure 2. The pipeline architecture for the system design

B. Pipeline Structure for HOG

Although HOG algorithm has been previously implemented on FPGAs, the proposed architecture is a full-pipeline structure that can provide higher throughput. Block RAMs are employed to separate the HOG computation into multiple blocks without suspending data streaming. Thus this HOG architecture can provide streaming processing of image pixels. The detail design of each block is described as follows:

1) Gradient Calculation: Gradients are calculated for each pixel at every clock cycle. For each 32*32 image, pixels are scanned line by line. To compute vertical gradient G_y, at least one line needs to be buffered. Our implementation of the Gradient Calculation blocks results a pipeline latency of 33 clock cycles.

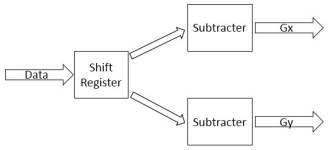

Figure 3. Design of the Gradient Calculation block

2) Magnitude and Bin Calculations: After computing gradients of pixels, magnitudes and bin classes are computed using equations (3-4). The structure of this IP core is shown in Figure 4. Magnitude is computed using 2 multipliers, 1 adder and 1 square root module. Bin classes calculation uses 4 fixed coefficient multipliers with fixed parameters {tan20, tan40, tan60, tan80}. Next, the multiplication results are compared to G_y. The bin class ranging from 1 to 9 can be obtained by passing through a simple classifier.

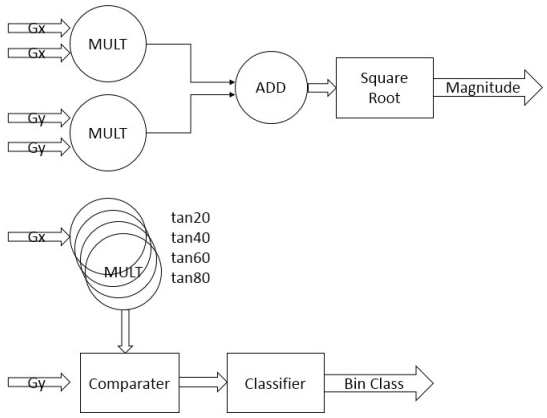

Figure 4. Design of the Magnitude&Bin Calculation block

3) Address Encoder: The Address Encoder IP core simply converts data streaming to a target memory. Each pixel is properly stored into a specific location of the block RAM. As in Figure 5, the 32*32 image is divided into 16 cells. Each cell has 8*8 pixels. These 8 pixels in a row are stored at one location of the block RAM. The 64 pixels in a cell are stored in eight consecutive memory locations of the same block RAM, which enables fast data access for the next stage.

4) Cell Summation: As in Figure 6, the Cell Summation block calculates the histogram of 64 pixels in a cell. This module reads the magnitude and bin class values of these 64 pixels, processes them through an adder tree, and then stores the histograms to another block RAM in a specific address. It requires the nine bin values of each cell to store at the same memory location of the target block RAM.

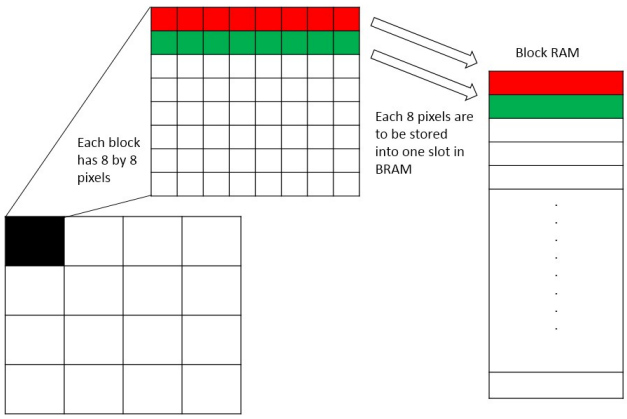

Figure 5. Mapping of the address encoder

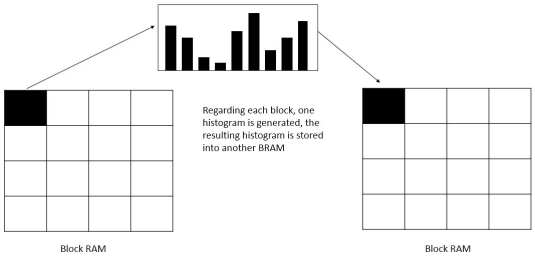

Figure 6. Design of the Cell Summation block

5) Normalization: The Normalization block takes the histograms from one block and sums up all 36 bin values. Only one divider is needed to compute the inverse of the summation. The result is then multiplied with 36 bin values in the parallel. The normalization results are obtained after the square root operation. Figure 7 shows the complete design of the normalizer.

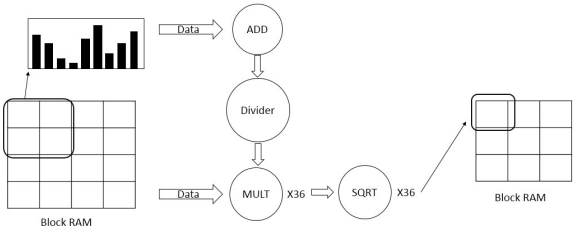

Figure 7. Design of the Normalization block

As shown in Figure 7, 36 bin values are read out and processed to get the normalized results. These 36 normalized values are stored to one location of the target memory. These normalized results make up the HOG features. The target memory contains nine memory locations, with each storing 36 HOG features. So a total of 324 HOG features extracted from a 32*32 image are obtained.

C. SVM calculation

Figure 8 shows the design of the 48-way linear SVM. It has a total of 15552 support vectors stored in a ROM.

Since at every clock cycle 36 HOG features are read out simultaneously, we need 36 multipliers in the design.

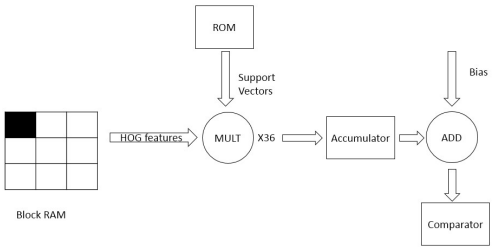

Figure 8. The design of SVM block

The SVM block continuously reads HOG features and multiplies by the support vectors. After all 324 HOG features has been read out, multiplied with corresponding support vectors, the accumulator passes its result to the adder to adjust with the bias. The result is fed into a comparator which retains the larger value. Upon completion of all 48 SVM computations, the comparator is able to output the largest value among all 48 classes and its associated class as the classification results.

IV. Real-Time Implementation Performance

The SVM classifier has been trained offline using the BelgiumTS dataset. We use 4,492 images for training and 2,520 images for testing. The classification rate is 93.77%. Our system is implemented on the Xilinx Zynq ZC706 FPGA board. The resource utilization is listed in Table I.

Table I
FPGA resource utilization of the system

	Used	Available	utilization
Slice Registers	11789	437200	2.7%
Slice LUTs	8168	218600	3.7%
DSP48E1s	37	900	4.1%
Block RAM	17	545	3.1%

Maximum frequency of the FPGA implementation is 241.7 MHz. The HOG module has the pipeline latency of 693 clock cycles and the SVM module has the latency of 864 clock cycles. Note that pipeline latency does not affect the system throughput, rather than the absolute time from the input of an image to the output of the classification result. For a total of 1557 clock cycles latency, the system response time is about 6.5 s when the system is operating at 241.7 MHz.

Since our system is a full-pipeline streaming architecture, every clock cycle it takes one pixel value as an input. The pixel rate is effectively the same as 241.7M pixels per second. Consider the image size of 32-by-32 pixels, the system can process 236,035 traffic sign images every second. If we consider high-definition image resolution of 1920-by-1080 pixels, the estimated frame rate is about 116 frames per second (fps).

We also implement the same classification algorithm on a Intel core-i5 processor at 2.67 GHz. The processor takes 0.451 ms to process one 32*32 image that is 2,217 images per second. Therefore, the throughput of our FPGA-based implementation is about 106 times higher than the processor. Even if we compare the absolute response time, the FPGA implementation is still 69 times faster than the processor.

V. Conclusion

In this paper, we present an FPGA-based design for real-time traffic sign classification. The main contributions include the full-pipeline streaming architecture for HOG feature extraction and an efficient design that combines 48 SVMs all together. The system can successfully classify 48 different traffic signs using the BelgiumTS dataset. Considering the traffic sign candidates as 32-by-32 pixels images, the systems can process 236,035 images per second with an absolute response time of 6.5 s. The system throughput is 106 times higher than a general purpose processor. As part of our future work, we will also implement the detection algorithm on the same FPGA with the input of 1080p videos directly from a camera.

References

[1] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 13, no. 4, pp. 1484–1497, 2012.

[2] X. Qingsong, S. Juan, and L. Tiantian, "A detection and recognition method for prohibition traffic signs," in *Image Analysis and Signal Processing (IASP), 2010 International Conference on*, April 2010, pp. 583–586.

[3] O. Déniz, G. Bueno, J. Salido, and F. D. la Torre, "Face recognition using histograms of oriented gradients," *Pattern Recognition Letters*, vol. 32, no. 12, pp. 1598 – 1603, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167865511000122

[4] I. Creusen, R. G. J. Wijnhoven, E. Herbschleb, and P. de With, "Color exploitation in hog-based traffic sign detection," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, Sept 2010, pp. 2669–2672.

[5] T. Groleat, M. Arzel, and S. Vaton, "Hardware acceleration of svm-based traffic classification on fpga," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International*, Aug 2012, pp. 443–449.

[6] S. Bauer, S. Kohler, K. Doll, and U. Brunsmann, "Fpga-gpu architecture for kernel svm pedestrian detection," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, June 2010, pp. 61–68.

[7] R. Kadota, H. Sugano, M. Hiromoto, H. Ochi, R. Miyamoto, and Y. Nakamura, "Hardware architecture for hog feature extraction," in *Intelligent Information Hiding and Multimedia Signal Processing, 2009. IIH-MSP '09. Fifth International Conference on*, Sept 2009, pp. 1330–1333.

[8] O. Pina-Ramirez, R. Valdes-Cristerna, and O. Yanez-Suarez, "An fpga implementation of linear kernel support vector machines," in *Reconfigurable Computing and FPGA's, 2006. ReConFig 2006. IEEE International Conference on*, Sept 2006, pp. 1–6.

[9] M. Papadonikolakis and C. Bouganis, "A novel fpga-based svm classifier," in *Field-Programmable Technology (FPT), 2010 International Conference on*, Dec 2010, pp. 283–286.

[10] "Belgiumts dataset, 2010 [online]. available: http://btsd.ethz.ch/shared-data/."

[11] Z. Chen, X. Huang, Z. Ni, and H. He, "A gpu-based real-time traffic sign detection and recognition system," *Proceeding of the IEEE SSCI Symposium*, 2014.