Real-Time Vision-based Stop Sign Detection System on FPGA

Tam Phuong Cao Guang Deng
Department of Electronic Engineering, Latrobe University, Bundoora, Victoria, Australia
t.cao@latrobe.edu.au D.Deng@latrobe.edu.au

Abstract

Many fatal accident have happened due to drivers failed to stop at stop signs. A stop sign recognition system could be used to reduce the risk of accident by warning the driver when a vehicle approaches a stop sign at an unexpected speed. In this paper, we describe the implementation of a real-time vision-based stop sign recognition system on a Xilinx Virtex-4 Field Programmable Gate Array (FPGA) device. This system uses a variant of the Histogram of Oriented Gradient (HoG) feature set and the efficient integral map for processing. Simulation and in-vehicle test results show good potential for deploying the system in practice. The FPGA system is able to process 60 frames of 752×480 pixels per second .

1 Introduction

Road signs, including stop signs, are designed to control the traffic flow and to ensure the safety of people. If a driver fails to stop at a stop sign, accidents can happen and sometimes can be fatal. In the US, there are about 700,000 police reported crashes at stop signs every year [9]. In Australia, many lives have been lost due to accidents at stop sign sites [3]. A lot of such fatal accidents could have been avoided if a stop sign recognition system were implemented in vehicles.

In order to deploy a stop sign recognition system in automotive vehicles, there are some practical challenges to be overcome. The biggest one is the trade-off between cost and performance. On the one hand, cost is the major economic factor that drives the popularity of a car feature, such as the vision based stop sign recognition system. So a stop sign recognition system has to be produced at low cost. As a result, the available processing platforms are limited. On the other hand, the system must have good performance, which requires the system to run complicated algorithms in real-time. The system must also be robust against different lighting conditions and different appearance of stop signs due to normal wear, occlusions or fading of colour. These requirements normally result in the need for powerful and expensive systems.

FPGA is a promising technology to overcome the challenges. Due to its parallel architecture and small size, the FPGA platform is suitable for implementing various vision based safety systems on automotive vehicles at low cost. However, not all image processing algorithms are suitable for FPGA implementation due to overly expensive arithmetic calculations. In this paper, a real-time FPGA-based stop sign recognition system using a simplified Histogram of Oriented Gradients (HoG) feature [4] and the efficient integral map (IMap) [10] is proposed. The detection algorithm for this system has been optimized to achieve high performance while keeping it directly implementable on the FPGA platform. The rest of this paper is organized as follows. Section 2 briefly reviews related work. In section 3, the FPGA implementation of the algorithm is described. Section 4 presents some experimental results. This followed by conclusions and discussions of future work to improve the system.

2 Object Detection System and Related Work

In this section, we first describe major building blocks of general object detection system, which are relevant to our work. We then briefly review some previously proposed stop sign detection algorithms.

2.1 Object Detection System

A common way to detect an object is to scan a detection window across the image and classify the image patch inside the detection window into one of the predefined classes, stop sign or non-stop sign in this case. This classification process is normally performed in some feature space, such as the HoG [4] feature space.

2.1.1 Integral Map (IMap)

The Integral Map (IMap) was introduced by Viola and Jones [10] for fast calculation of Haar features. An entry in an

978-0-7695-3456-5/08 $25.00 © 2008 IEEE
DOI 10.1109/DICTA.2008.37

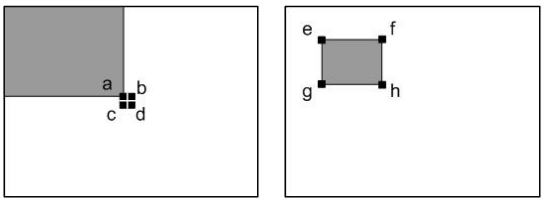
IMap is the sum of all pixel values to the left and above the current pixel's position in the original image. As shown on Fig.1(a), the entry to the IMap at point a is the sum of all pixels to the left and above (as shown by the shaded region) in the original image. To calculate the IMap more efficiently, neighbouring entries are used. Referring to Fig.1(a), the IMap entry at point d denoted as $M(d)$, is calculated as:

$$M(d) = M(b) + M(c) - M(a) + P(d) \quad (1)$$

where points a, b, c are direct neighbours of point d in the IMap, $M(a)$, $M(b)$, $M(c)$ are the computed IMap values at a,b,c in the IMap, and $P(d)$ is the pixel value at equivalent location d in the original image. When it is required to find the sum of all pixel values within a rectangular window bounded by any four points e, f, g and h as shown in Fig.1(b), then the sum S is calculated by:

$$S = M(h) - M(g) - M(f) + M(e) \quad (2)$$

where $M(e)$,$M(f)$, $M(g)$ and $M(h)$ are corresponding values of e,f,g and h in the IMap.



(a) (b)

Figure 1. IMap construction and use. (a). Construction of IMap from neighbouring entries. (b). Calculate the area using 4 corner points.

2.1.2 Histogram of Oriented Gradients

Our stop sign recognition system uses a powerful feature set adapted from the Histogram of Oriented Gradients (HoG) [4] method. In this method, the angle of the gradient for each pixel is calculated and quantized into one of nine different angular bins. Pixels within the detection window are divided into non-overlapping block of a specified size, such as 16×16 pixels. Each block is further divided into smaller cells, such as 8×8 pixels. From each cell, a 9-dimensional (corresponding to the 9 angular bins) vector is generated from the histogram of gradients (in terms of accumulated magnitude for each bin) of the pixels within that cell. All component vectors generated within a block, normally 4 per block, are then concatenated and normalized. Other overlapping blocks within the detection window are treated similarly to generate the final feature vector of a detection window. This feature vector is then used to classify the detection window into a predefined class. The HoG feature set has been used for road sign detection [1]. However the system is not capable of operating in real-time.

2.2 Stop Sign Detection Algorithms

A stop sign detection system has been proposed by Liu [6]. In this system, colour, shape and symmetry information are used to segment the image. The detection of the stop sign is carried out by using multi-layer neural networks. A major problem with this system is that the Hue-Saturation-Value (HSV) color segmentation process is not reliable in many difficult situations. For example the colour segmentation may fail to segment the stop sign when the image is too bright or too dark. Another traffic sign detection system was proposed by Gareth and Barnes [7]. This system is robust against rotation and has high detection rate. A major disadvantage of the system is the recursive (non-deterministic) computation which is not suitable for pipeline hardware implementation. Huang [5] described a method for processing colour images under various lighting conditions. However, calibrations are required for different types of cameras which have different colour outputs.

3 The Proposed System Implemented on FPGA

3.1 FPGA System Overview

An overview of the proposed system implemented on FPGA is shown in Fig.2. Images from the camera are acquired via a communication link to the camera control module. This module is also responsible for automatic exposure control which adjusts camera settings to record images with good contrast in different lighting conditions. The proposed system only buffers several lines of pixel to reduce the memory requirements of the system. As such, the system does not store the whole image. This is in contrast to several existing systems, such as [8], which store the whole image before processing it. The first step of the algorithm is to calculate the angle of the gradient for every pixel and quantize it into one of different angular bins. The next step is to calculate an entry to the IMap at the corresponding pixel location. The third and fourth steps are detection and verification of stop sign candidates within the image. Finally, the result is displayed on the Human Machine Interface (HMI). In order to reduce design time, various visual feed back devices, such as VGA or alphanumeric LCD display, have been used.

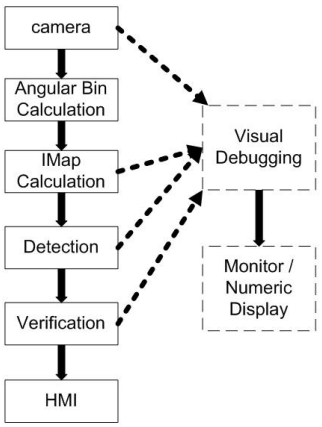


Figure 2. Overview of stop sign detection system implemented on FPGA. The Visual Debugging module is used in the development stage only. It's not required for the deployment of the system.

3.2 System Components

3.2.1 Angular Bin Calculation

The angle of a pixel's gradient is calculated from its vertical and horizontal gradients. There are several ways of computing a pixel gradients. This implementation follows the centered method of calculating pixel gradients as suggested in [4]. Normally the process of calculating angle of a pixel is calculated by:

$$g = tan^{-1}(\frac{g_y}{g_x}) \tag{3}$$

where g is the pixel's gradient angle, π is added to any negative angle; g_x, g_y are pixels gradients in the horizontal and vertical directions respectively. These gradients are found by convolving the image with the gradients mask: $[-1\ 0\ 1]$ and $[-1\ 0\ 1]^T$. The resulting angle from (3) is quantized into one of the angular bins. In our implementation, instead of the 9-bin configuration as originally proposed in [4], a 4-bin configuration is used. The angle spacing of the 4-bin HoG is shown in Fig.3.

The above outlined method of angle calculation is only suitable for processor implementation, but not for FPGA implementation because calculating the tan^{-1} function and division (in (3)) are very expensive to implement on FPGA. Even though there are many hardware friendly approximation algorithms available, they are generally iterative algorithms, which slows down the overall system's speed. A common approach that has been employed for similar cases is using look up tables (LUTs). However, using LUTs will require large amount of memory space, which eventually will increase system's cost. In our implementation, a cheaper (in terms of hardware usage level) alternative is used. We directly quantize of the pixel's angular bin from its corresponding gradients. For example, a pixel at location $P(x, y)$ belongs to bin 2 when its angle is in the range: $22.5^o < \theta < 67.5^o$ or $0.4142 < tan(\theta) < 2.4142$. This is equivalent to the condition $0.4142 < \frac{g_y}{g_x} < 2.412$. Division operation is prohibitive in term of hardware cost, it can be eliminated by multiplying both side to g_x:

$$\begin{cases} 0.4142 \times g_x < g_y < 2.412 \times g_x & \text{if } g_x > 0 \\ 0.412 \times (-g_x) < (-g_y) < 2.412 \times (-g_x) & \text{if } g_x < 0 \end{cases} \tag{4}$$

Because double precision multiplication is also expensive on FPGA and the task of classifying pixel's angle into one of the bins is a quantization process, (4) can be approximated by multiplying both side of (4) by a scaling factor, arbitrarily chosen to be 1024 in this case. Hence conditions for gradient angles fall into Bin 2 are as follow:

$$\begin{cases} 422 \times g_x < g_y < 2470 \times g_x & \text{if } g_x > 0 \\ 422 \times (-g_x) < (-g_y) < 2470 \times (-g_x) & \text{if } g_x < 0 \end{cases} \tag{5}$$

Integer multiplications can be performed in real-time using dedicated hardware multiplier. However, when dedicated hardware multiplier is not available, this calculation can still be carried out using shift and add operations. For example, a multiplication $p = a \times 422$ can be represented as $p = a \times (2^8 + 2^7 + 2^5 + 2^2 + 2^1)$ or $p = a << 8 + a << 7 + a << 5 + a << 2 + a << 1$ where the $a << 8$ is the shift operation which means shift the current value of a (in binary) to the left by 8 positions. This shift operation is the same as adding eight 0s to the left of a (in binary format) and this operation costs no hardware to perform on FPGA.

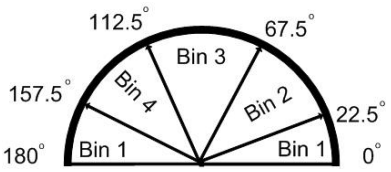


Figure 3. Angular spacing of 4-bin HoG with 'Bin 1' is divided into two non-continuous regions.

3.2.2 IMap Calculation

Calculating IMaps is a crucial step in the detection algorithm. There are four angular bins. Hence four IMaps, one for each angular bin, are constructed and stored in memory.

For fast implementation, dual-port block ram (RAMB) and flip flop buffers are used to stored IMaps. The dual-port RAMB allows stored data to be available on the data bus one clock cycle after the read command and read address are applied while at the same time allows data to be written to a different memory location.
The use of RAMB significantly speeds up the computation of the IMap compared to using traditional random access memory (RAM) devices (SRAM or DDR). Even though traditional RAM devices tend to have larger storage capacity, they normally require complicated interface and stricter timing. This means more hardware usage and sometimes significantly harder to pipeline the design. Moreover, additional devices connect to the FPGA may also increase the final product cost of the system.

3.2.3 Detection

The detection module is designed based on the moving detection window. At every pixel location in the image, selected HoG features within the detection window are extracted from the IMaps. Each HoG feature value is compared to its appropriate threshold. These features and their corresponding thresholds are determined during the training process which is described in Section 4.1. Computations of these features are performed in parallel. The results are then combined to determine the output. This detection process repeats for the next locations as the detection window scans the image from left to right and from top to bottom pixel by pixel.

3.2.4 Verification and HMI

Following the detection module, any potential stop sign candidate is recorded and processed by the verification module. Verification is an important step to remove false positive candidates from the detection module. This verification step looks at the correlation of the detection's outputs within several frames to decide if the output is legitimate or just random noise. The verification module assumes that the stop sign, if any, will remain being detected in multiple frames and relatively close to each other. This module can track multiple candidates and process them one by one. The amount of processing required for verification is small and not time critical. Therefore it is computed in a serial fashion to save hardware resources. In the current implementation, a candidate is said to be a stop sign if it appears within 4 pixels from each other and appears in at least 6 out of 10 consecutive frames.
The final output from verification module is fed into an HMI module, which warns the drivers about the existence of a stop sign ahead. With the current HMI implementation, the warning is displayed on a small monitor screen, shown on Fig.4.

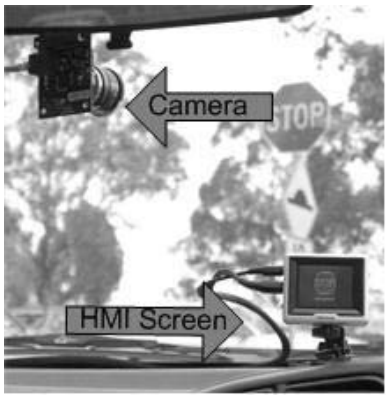


Figure 4. System configuration with camera attached to the windscreen and a small screen used as HMI

3.2.5 Human Visual Verification (HMV)

Although there are various debugging tools provided by hardware vendors, these tools are usually at signal levels and not giving enough information, at algorithmic level, to the designer. In this work, hardware drivers for VGA digital to analogue converter (DAC) (to connect to an external monitor) and alphanumeric LCD display (on the development board) have been built for debugging purposes. The VGA display can be effectively used to debug the implementation of IMaps because any mis-synchronization between reading and writing data will results in visible patterns or noisy image displayed on the monitor.

4 Experiments and Results

4.1 Training and Algorithm Adaptation for Hardware Implementation

To implement the stop sign recognition system on FPGA, a classifier needs to be trained and tested. The detection algorithm used for the system is adapted to suit the architecture and available resources on the FPGA. In order to accomplish this task, the detection algorithm is analysed to remove or modify any calculations that are prohibitive for FPGA implementation, such as division or square root.
After adapting the detection algorithm, a classifier is trained to detect stop signs. The classifier is trained using a training data set which consists of 225 positive and about 11.4 million negative samples. Positive samples are generated from 165 images of stop signs by adding random noise and some rotation to the original images, which make the classifier more robust. The random noise was derived from a uniform

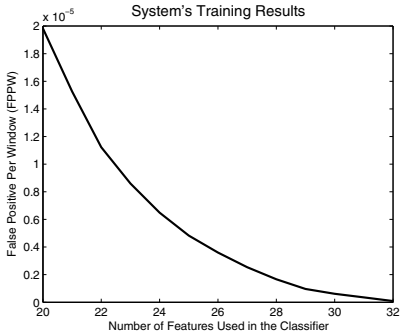


Figure 5. Training Results after 32 features are selected for final classifier. This results based of 100% detection rate targeted.

distribution with maximum magnitude of 20% of maximum pixel value. The size of the stop signs is 36×36 ±3 pixels across. About 11.4 million negative samples are generated by randomly sample 190 negative images which are at the resolution of 752×480. The negative images do not contain any stop sign, but may contain other road signs.

For the suitability of implementing on FPGA, we choose a training scheme that only involves simple comparison of image features, similar to simple the classifier from [10]. As such, the final trained classifier has the form: $H = \prod_{i=0}^{n} \alpha_i$ where:

$$\alpha_i = \begin{cases} 1 & \text{when } f_i \times p_i > t_i \times p_i \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

where f_i is a selected HoG feature; p_i is the parity of the comparison (either 1 or 0); t_i is the corresponding feature's threshold; n is the number of features selected. The final classification result is:

$$H = \begin{cases} 1 & \text{if there is a stop sign in detection window} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

Following this training scheme, the system is trained using Matlab. After training, the false positive rate (based on 100% detection rate) is shown on Fig.5. After the 30^{th} features used, the False Positive Per detection Window (FPPW) rate is about 6.14×10^{-7}. The FPPW rate indicates the ratio between the number of false alarm and the total number of detection windows processed. This rate achieved is significantly lower than the false positive rate of 10^{-3} or 10^{-4} at 80-90% detection rate reported in [1] or about 75% detection rate at similar 6×10^{-7} FPPW rate as in [2]. In addition, we only use about 30 features which is relatively small compare to about 200 Haar features used in [2] or 100 hidden neurons in [6] for a similar traffic sign recognition applications. The false positive rate is expected to be lower when more features are used in the system.

Algorithm	Liu[6]	Loy [7]	Huang[5]	Ours
Recognition	92.63%	95%	66.67%	97.64%
False Pos	N/A	0	N/a	7.79×10^{-7}
N^o of Images	540	15	36	1500
Image Size	N/A	320×240	256×240	752×480

Table 1. Performance comparison of different stop sign recognition systems. Result from [5] based on 36 frames that contains stop signs in 2 hours video recorded.

After training, a fixed point simulation system is built on Matlab that only use integer multiplications, adds and subtracts. These represent available resources on the FPGA. This system can be directly ported to FPGA without the need for modifying the algorithm. This system is then tested against a large test set that contains 1500 test images. These 1500 test images are randomly selected from 50 video sequences recorded which contain an average of 315 frames per sequence. After testing, the result obtained is 83 out of 85 valid stop signs (97.64%) were detected and the average false positive rate is 0.118 false positive per frame or equivalent to 7.79×10^{-7} FPPW. The size of detected stop signs is from 33 to 39 pixels across. These detection and false positive rates are based on frame by frame statistic, i.e., no correlation between consecutive frames. When tracking or verification step is used in a video sequence, the detection rate will be higher and the false positive rate will be lower. The performance comparison of different stop sign detection systems is shown on Table.4.1. The biggest source of false positive is from trees along side of the road. Some examples of detected signs as well as false positive is shown on Fig.6.

4.2 Hardware Implementation and Testing

Based on the simulated system, a stop sign detection system is implemented on a Xilinx Virtex-4 ML402 development board which is equipped with a Xilinx Virtex-4-Sx35 FPGA, as shown in Fig.7(a). The FPGA system developed is completely vendor independent. It can be easily port to FPGA from other vendors such as Altera. A Micron 752×480 pixel high frame rate (60 fps) grayscale camera is used for the system. An independent expansion board, shown on Fig.7(b), has been designed to provide the camera a communication link (via I^2C serial bus) and a power supply point for the whole system. The whole FPGA system

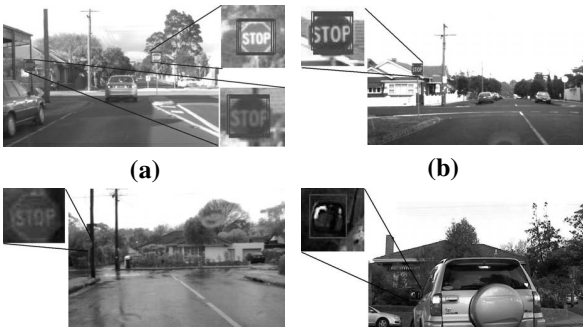


Figure 6. Examples of tested images. (a),(b). Double and single stop sign detected. (c). Stop sign missed due to bad weather and lighting conditions (image enhanced for display purposes). (d). A false positive.

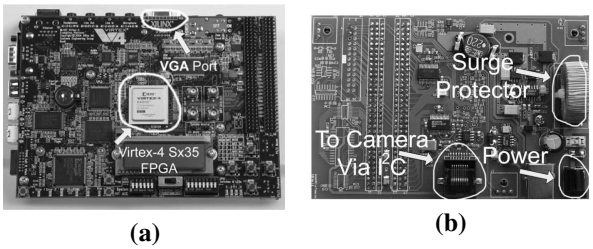


Figure 7. Processing system. (a). Xilinx Virtex-4 ML402 development board. (b). expansion board developed at La Trobe.

is in-vehicle tested in different conditions, which represent different conditions that the stop sign detection system is expected to function satisfactorily.

As the proof of concept, the detection module of the FPGA system implements only 15 out of 30 selected image features due to time constraint. The system's hardware resource utilization on the Xilinx ML-402 board including resources used to implement the visual debug module (VGA and LCD) is shown on Table.1. This table indicates that the system can be ported to a smaller and cheaper FPGA family. Moreover, the entire detection algorithm can be implemented on a single FPGA which could further reduce the system cost when deploying in vehicles. It is expected that when more features are implemented, the hardware utilization will not increase significantly because new features will share resource with existing ones.

The FPGA system was tested for a total of about 30 minutes (collectively) by driving on different test routes. There were total of 16 stop signs in these test routes. A stop sign is said to be detected in the experiment if the new sign indicator of the HMI, shown on Fig.4, is updated. This system is capable of operating in real-time at the frame rate of 60 fps. In fact, if the camera can operate at higher frame rate, the system can operate at higher frame rate. The system's detection rate is 81.25% (13 out of 16 signs detected) with the false positive rate of 1 false positive per 7200 frames (2 minute driving) or the equivalent false positive per detection window is on average about 4.96×10^{-10} when tracking and verification is used. The exact frame by frame test results are very hard to recorded on FPGA implementation. Like the simulated system, this FPGA system only targets stop signs in the image that have the size of about 36×36 ±3 pixels across.

Resources Type	Available	Used	Usage Level
Flip Flop	30720	4221	13%
4 Input LUTs	30720	8921	29%
Block RAM	192	88	45%
Multipliers	192	3	1%

Table 2. Hardware resource usage of the system on FPGA

5 Conclusion and Future Work

In this paper we have shown that a reliable stop sign recognition system can be built using a simplified 4-bin HoG and integral maps. Test results show that the FPGA simulated system has high detection rate and low false positive which is very important for practical implementation of the system. We have shown that this stop sign recognition system is suitable for FPGA implementation. The low level of hardware usage on a Virtex-4 FPGA indicates that the system can be implemented on a smaller (less resource available), hence cheaper, FPGA device. The stop sign recognition system can be used as a standalone system or can be integrated into an existing system to improve vehicle's safety. Because HoG and IMaps have been used for human detection task, this system could be further extended to detect pedestrians and other objects.

To improve the performance and reduce false positive rate, more HoG features will be added to our FPGA system. In addition, using multiple sizes of stop signs in the detection algorithm is expected to improve the system's reliability significantly. The verification system could be further improve by using certain statistical properties related to the expected position of the stop sign on the road, i.e. a stop sign is more likely to appear of the side than in the middle of the road.

6 Acknowledgments

This work has been supported by AutoCRC Australia and the Department of Electronic Engineering, La Trobe University, Australia, .

References

[1] B. Alefs, G. Eschemann, H. Ramoser, and C. Beleznai. Road sign detection from edge orientation histograms. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 993–998, Istanbul, Turkey, June 2007.

[2] C. Bahlmann, Y. Zhu, V. Ramesh, M. Pellkofer, and T. Koehler. A system for traffic sign detection, tracking, and recognition using color, shape, and motion information. In *Proc. IEEE Intelligent Vehicles Symposium*, pages 255–260, June 2005.

[3] M. R. J. Baldock and J. A. McLean. *Older drivers: Crash involvement rates and causes*. Centre for Automotive Safety Research, University of Adelaide, Adelaide, Australia, 2005.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proc. IEEE Computer Scociety conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, San Diego, USA, 2005.

[5] W.-C. Huang and C.-H. Wu. Adaptive color image processing and recognition for varying backgrounds and illumination conditions. *IEEE Trans. Ind. Electron.*, 45:351–357, 1998.

[6] H. X. Liu and B. Ran. Vision-based stop sign detection and recognition for intelligent vehicles. *Transportation Researc Record 1748*, 1:161–166, 2001.

[7] G. Loy and N. Barnes. Fast shape-based road sign detection for a driver assistance system. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems.*, volume 1, pages 70–75, Sendai, Japan, 2004.

[8] D. Nguyen and D. Halpuka. Real-time face detection and lip feature extraction using field-programmable gate arrays. *IEEE Trans. Syst., Man, Cybern. B*, 36:902–912, 2006.

[9] R. A. Retting, H. B. Weinstein, and M. G. Solomon. Analysis of motor-vehicle crashes at stop signs in four U.S. cities. *Journal of Safety Research*, 34:485–489, 2003.

[10] P. Viola and M. Jones. Rapid object detection using a boosted cascaded of simple features. In *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pages 511–518, 2001.